Estatística Computacional

Patrícia de Siqueira Ramos

PPGEAB
UNIFAL-MG

21 de Março de 2018

Conteúdo programático

- Introdução ao R
- Geração de variáveis aleatórias uniformes
- Geração de variáveis aleatórias não uniformes
- Geração de variáveis aleatórias multidimensionais
- Algoritmos para médias, variâncias e covariâncias
- Aproximação de distribuições
- Métodos de reamostragem (*bootstrap*, *jackknife*)

Bibliografia

DALGAARD, P. **Introductory Statistics with R**. Springer, 2002.

R CORE TEAM. **R**: A Language and Environment for Statistical Computing. Vienna, Austria, 2018. Disponível em: <http://www.R-project.org/>.

JAMES, G. W.; HASTIE T.; TIBSHIRANI R. **An introduction to statistical learning**: with applications in R: Springer-Verlag New York, 2013.

VENABLES, W. N.; RIPLEY, B. D. **Modern Applied Statistics with S-Plus**. Springer Science & Business Media, 2013.

Avaliação

- Listas
- Trabalho prático
- Avaliação?

Estatística Computacional

- O uso de computadores em Matemática e Estatística possibilitou o estudo de técnicas para resolver problemas antes "intratáveis"
- Estatística Computacional e Computação Estatística - duas áreas na Estatística que podem ser descritas como abordagens computacionais, gráficas e numéricas para resolver problemas estatísticos
- Ambas envolvem diretamente ou indiretamente a aleatoriedade e simulação

Estatística Computacional

- Muitas vezes, as propriedades de um modelo estocástico podem ser obtidas experimentalmente, por meio do uso de um computador que simule muitas réplicas do modelo e da análise do resultado obtido
- Esses métodos são denominados "métodos de Monte Carlo" e é um dos principais tópicos em Estatística Computacional
- Além disso, em muitos problemas, as aproximações assintóticas são insatisfatórias ou intratáveis

1 Introdução ao R

1.1 Apresentação do R

Por que usar R?

- código aberto e livre
- qualidade comercial (escolhido por estatísticos e não estatísticos)
- popularidade crescente
- poderoso
- possibilidade de criação de pacotes
- novidades frequentes (alguns métodos estatísticos são disponibilizados primeiro no R)

Como usar o R

- Linguagem interpretada e não compilada

- Editores:
  - editor próprio do R
  - RStudio (ótima opção)
  - outros editores (como o Tinn-R)

- Link com dicas para baixar o R e o RStudio:
http://www.analisededadosempython.org/r.html

Uso do R

- Console: é por onde nos comunicamos com o R e o prompt é o sinal >
- Podemos escrever os comandos diretamente no console ou usar um script e ir executando os comandos com CTRL R ou CTRL ENTER
- O R é bom em aritmética. Exemplos:

```
265 + 343
123 * 45
375 / 12
73 ^ (1 / 3)
```

- Se você pedir ajuda, os resultados aparecerão na janela de informações. Exemplo:

```
> ?mean
```

Uso do R

- Atribuição: pode-se utilizar o sinal de '=' ou '<-' (vamos utilizar o '=' para facilitar a conversão para outras linguagens)
- Vários comandos em uma mesma linha: separar usando ;
- Limpar console: CTRL L
- Comentários: usar #
- Variáveis:

```
produto = 15.3 * 23.4 # salva resultado na variável
produto # mostra variável
x = 45; y = 5; z = x / y
z
# outra opção
(z = x / y) # armazena e já mostra o valor de z
```

Dicas

- O R é *case-sensitive* (diferencia maiúsculas e minúsculas)
- O RStudio te ajuda na edição dos comandos, já fechando parêntese, chave etc. que você abriu. Ele também cuida da indentação
- O uso de TAB pode agilizar! O RStudio vai mostrar uma lista de opções para completar o comando. Pressione TAB para escolher sua opção
- Se aparecer $+$ no lugar de $>$ significa que o R está esperando algo (ou você se esqueceu de fechar um parêntese ou cometeu algum erro de sintaxe). Pressione ESC para retornar tudo ao normal

Funções muito usadas no R

descrição	R
raiz quadrada	`sqrt`
logaritmo natural	`log`
exponecial e^x	`exp`
fatorial	`factorial`
números aleatórios uniformes	`runif`
números aleatórios normais	`rnorm`
distribuição normal	`pnorm, dnorm, qnorm`
ordenar	`sort`
variância, covariância	`var, cov`
desvio padrão	`sd`
correlação	`cor`
tabelas de frequência	`table`
valores faltantes	`NA, is.na`

Tipos de objetos

- vetor: estrutura de uma dimensão (um só tipo de dados)
- matriz: estrutura de duas dimensões (um só tipo de dados)
- *array*: estrutura de três dimensões (várias matrizes)
- lista: uma coleção ordenada de objetos que podem ser de diferentes tipos, são mais gerais que os data frames
- *data frame*: não é uma matriz, mas pode ser representada por uma tabela e pode conter diferentes tipos de dados

Obs.: como o R é uma linguagem vetorial é capaz de operar vetores e matrizes sem necessidade de *loops* (*for*, *while*, *repeat*)

Vetores

Vetores são o tipo básico e mais simples de objeto para armazenar dados no R:

```
(x = c(2, 3, 4, 5.1, 3.2))
1 / x   # operações diretas com vetores
range(x)
length(x)
(y = 1:6)
x1 = seq(from = 1, to = 10, by = 0.5)
# ou seq(1,10,0.5)
rep(x, each=5)  # repete cada elemento 5 vezes
x5 = rep(x, times=5)  # repete o vetor x 5 vezes
x5[1]  # acessa primeiro elemento de x5
x5[4] = 9  # alterar o valor de um elemento do vetor
x5
seq(length=19, from=1, by=0.5)  #tamanho, v.inicial, passo
seq(length=19, from=1, by=-0.5)
```

Exemplos de sintaxe para vetores e matrizes no R

descrição	R	exemplo
vetor de zeros	`numeric(n)`	`x = numeric(n)`
	`integer(n)`	`x = integer(n)`
	`rep(0, n)`	`x = rep(0, n)`
matriz de zeros	`matrix(0, n, p)`	`x = matrix(0, n, p)`
elem. *i* de um vetor **a**	`a[i]`	`a[i] = 0`
col. *j* de uma matriz **A**	`A[,j]`	`sum(A[,j])`
elem. *ij* da matriz **A**	`A[i,j]`	`x = A[i,j]`

Vetores

```
w = c(TRUE, TRUE, TRUE, FALSE, TRUE, FALSE) # vetor lógico
# ou w = c(T, T, T, F, T, F) # vetor lógico
(x = c(1, 3, 5, 8, 9))
y = x > 5
y
z = !y  # z é a negação de y
z
w = x[x > 2]  # vetor com elementos de x > 2
w
w1 = (x + 1)[x >= 3]  # soma 1 aos elementos de x >= 3
w1
which(x > 4)  # retorna posições dos elementos que satisfazem
x[-2]  # retira elementos dos índices com -
sample(1:100, 10, replace=T) # função sample
sample(c(2, 6, 5, 9, 3, 1, 7), 3, replace=T)
```

Matrizes

Matrizes: arranjos de duas dimensões (linhas e colunas):

- `A * B`: multiplicação elemento a elemento
- `A %*% B`: multiplicação matricial
- `t(A)`: transposta
- `diag(n)`: matriz identidade de dimensão n
- `solve(A)`: inversa de $\mathbf{A}$
- `solve(A, b)`: solução de $\mathbf{Ax} = \mathbf{b}$
- `rowMeans(A)`: devolve o vetor das médias por linha
- `rowSums(A)`: devolve o vetor das somas por linha
- `colMeans(A)`: devolve o vetor das médias por coluna
- `colSums(A)`: devolve o vetor das somas por coluna

Exemplos com matrizes

```
(A = matrix(0, 3, 2))  # valor, n. linhas, n. colunas
(B = matrix(c(1, 2, 3, 4), 2, 2))  # por coluna
(C = matrix(c(1, 2, 3, 4), 2, 2, byrow=T))  # por linha
```

Operações e funções com matrizes:

```
B * C  # multiplica elemento a elemento
A %*% B  # efetua a multiplicação de matrizes
ncol(A); nrow(A)  # retorna n. linhas e n. colunas, resp.
V = diag(A)  # retorna a diagonal de A
A1 = solve(A)  # inversa de A
# sistemas de equações
A = matrix(c(2,3,4,-2), 2, 2)
y = c(10, 5)
x = solve(A, y)
x
```

Exemplos de funções com matrizes

```
(m1 = matrix(1:6, nc=3))
# linhas
margin.table(m1, margin=1)
apply(m1, 1, sum)
rowSums(m1)
rowMeans(m1)
# colunas
margin.table(m1, margin=2)
apply(m1, 2, sum)
colSums(m1)
colMeans(m1)
# solução de sistema
mat = matrix(c(1, 5, 2, 3, -2, 1, -1, 1, -1), nc=3)
vec = c(10, 15, 7)
solve(mat, vec)
# inversa
solve(mat)
```

Data frames

- Vetores e matrizes forçam todos os elementos a serem do mesmo tipo (numérico, caracter)
- O *data frame* é uma estrutura semelhante à matriz, porém, com cada coluna sendo tratada separadamente (mas, dentro de uma mesma coluna, todos os elementos ainda serão forçados a serem do mesmo tipo)

```
d1 = data.frame(X = 1:10,
                Y = c(51, 54, 61, 67, 68, 75, 77, 75, 80, 82))
d1
names(d1)
d1$X
d1$Y
plot(d1)
plot(d1$X, d1$Y)
```

Data frames

```
# acesso aos elementos
# a) matriz
d1[2,2]
d1[,2]
d1[3,]
# b) lista
d1$X
d1$Y
# c) attach() - melhor não usar
attach(d1)
Y
detach(d1)
```

Exercícios

1. Mostrar comandos que podem ser usados para criar os objetos ou executar as instruções a seguir:

a) o vetor: [4 8 2]

b) selecionar o primeiro e terceiro elemento do vetor acima

c) o vetor com a sequência de valores: [-3 -2 -1 0 1 2 3]

d) o vetor com a sequência de valores:
[2.4 3.4 4.4 5.4 6.4 7.4 8.4 9.4 10.4]

e) o vetor:
[1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39]

f) o vetor: [1 3 5 7 9 11 14 17 20]

g) o vetor de sequência repetida: [1 1 1 2 2 2 3 3 3 4 4 4]

h) o vetor de sequência repetida: [4 4 4 3 3 3 2 2 2 1 1 1]

i) o vetor de elementos repetidos: [1 2 3 1 2 3 1 2 3 1 2 3]

Exercícios (cont.)

j) a sequência de valores: [1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99]
k) o vetor: [11 10 9 8 7 6 5 4 3 2 1]
l) o vetor alfanumérico: [“Parana” “Sao Paulo” “Minas Gerais”]

2. Construa um *dataframe* com três colunas: x, x^2 e $exp(x)$, com x variando de 0 a 50.

Exercícios (cont.)

3. A função *sum*(x) retorna a soma dos elementos do vetor x. A expressão `z = rep(x, 10)` faz o vetor z igual a uma sequência de 10 vetores x. Use estas e outras funções para calcular a soma dos 100 primeiros termos das séries:

a) $1 + 1/2 + 1/3 + 1/4 + \ldots$

b) $1 + 1/22 + 1/42 + 1/62 + 1/82 + \ldots$

c) $1/(1+1/1!)2 + 1/(1+1/2!)2 + 1/(1+1/3!)2 + \ldots$
Obs.: a função '!' no R é a `factorial`

Exercícios (cont.)

4. Crie o seguinte *data frame* no R:

```
      nome idade altura peso
1     Alex    25    177   57
2  Liliane    31    163   69
3   Marcos    23    190   83
4   Olivia    52    179   75
5    Marta    76    163   70
6    Lucas    49    183   83
7 Caroline    26    164   53
```

a) Usando a função `row.names` faça com que os nomes sejam os rótulos das linhas.
b) Inclua a coluna "sexo" no *data frame*: `"M", "F", "M", "F", "F", "M", "F"`.
c) Inclua também a coluna "trabalho": `"S", "N", "N", "S", "S", "S", "S"`.
d) Quantas linhas e colunas tem o novo conjunto de dados? Use a função apropriada para isso.

e) Qual o tipo de dados de cada coluna (variável)? Use a função `str`.