Revisão de programação no R

Estruturas de controle e funções

Prof. Walmes Zeviani
walmes@ufpr.br

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

Atualizado em 2018-08-01

Justificativas

- Um computador é uma máquina de propósito geral. A programação dá a ela um proprósito específico.
- O propósito de programar é automatizar processos.
- A linguagem R é considerada a lingua franca da Estatística.

Objetivos

- Relembrar os fundamentos de programação em R.
- Rever as estruturas de controle.
- Rever a definição e uso de funções.
- Resolver estudos de caso.

Estruturas de controle

Operadores relacionais

Relação	Operador	Emprego
Igual	==	5 == 3
Diferente	!=	5 != 3
Menor	<	5 < 3
Menor e igual	<=	5 <= 3
Maior	>	5 > 3
Maior e igual	>=	5 >=3

Operadores lógicos

Condição	Operador	Emprego
E	&&	x > 10 && y > 0
OU	||	x > 10 || y < 0
NÃO	!	!(x > 10)

Atenção!

- `&` e `|` são vetoriais.
- `%in%` é o operador pertence.
- Na dúvida sobre precedência de operações, use `()`.

Estruturas de controle

Estruturas condicionais (if)

```
# Apenas uma condição.
if (<condition>) {
    <statments if TRUE>
}
```

```
# Condição e seu complementar.
if (<condition>) {
    <statments if TRUE>
} else {
    <statments if FALSE>
}
```

```
# Condições em cadeia.
if (<condition>) {
    <statments if TRUE>
} else if (<condition>) {
    <statments if FALSE and TRUE>
} else {
    <statments if FALSE and FALSE>
}
```

- O R não tem o contrutor `elif`.
- Quando o código tem uma linha, não precisa de `{}`.

Estruturas de controle

Laço iterador (for)

```
for (<variable> in <set>) {
    <statments>
}
```

- A interação é sobre elementos de vetores e listas.
- `break` e `next` são usados para interromper e saltar ciclos.
- Jamais cresca uma vetor/lista durante um loop!

Estruturas de controle

Laço iterador com condional (while)

```
while (<condition>) {
    <statments while TRUE>
}
```

- O loop é executado enquanto a condição for `TRUE`.
- `break` e `next` podem ser usados.
- Cuidado com loops sem condições de parada (loops infinitos).

Estruturas de controle

Pergunta

1. Quando usar `for` e `while`?
2. Qual dos dois é mais geral?

Estruturas de controle

Laço de repetição com condional (repeat)

```
repeat {
    <statments before 1st condition>
    if (<1st condition>) break
    <statments before 2nd condition>
    if (<2nd condition>) break
    <statments after 2nd condition>
}
```

- O loop é executado enquanto as condiçõs forem TRUE.
- `break` e `next` podem ser usados.
- As condições podem ser várias e estar em diferentes posições dentro do código.

Estruturas de controle

Pergunta

1. Quando usar `while` e `repeat`?
2. Qual dos dois é mais geral?

Estruturas de controle

Estrutura de casos (switch)

```
switch(<element>,
       <value_1> = {
           <statments 1st condition>
       },
       <value_2>= {
           <statments 2nd condition>
       },
       {
           <statments last condition>
       })
```

- `switch()` pode representar estruturas de `if else if` planas.
- A condição é sempre a de igualdade.
- Geralmente `switch()` é usado para igualdade de valor de *strings*.

Alguns problemas

Qual o resultado desse código?

```
x <- 7
if (x > 3) {
    print("3")
    if (x < 5) {
        print("5")
        if (x == 7) {
            print("7")
        }
    }
}
```

Alguns problemas

Quais números são exibidos com a execução?

```
i <- 3
while (i >= 0) {
    print(i)
    i <- i - 1
}
```

Alguns problemas

Quais números são exibidos com a execução?

```
i <- 5
while (TRUE) {
    print(i)
    i <- i - 1
    if (i <= 2) {
        break
    }
}
```

Como ficaria se fosse usado `repeat`.

Alguns problemas

Qual o valor de `j`?

```
j <- 0
for (i in 1:10) {
    if (j %% 3 == 0) {
        j <- 1
    }
    j <- i
}
```

Funções

Protótipo da função

```
<funname> <- function(<arg_1>, <arg_2>, <arg_n>) {
    <statments>
    return(<values>)
}

<funname>(<arg_1> = <value_1>, <arg_2> = <value_2> , <arg_n> = <value_n>)
```

Funções

Exemplo: fórmula de Báskara

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```r
baskara <- function(a, b, c) {
    stopifnot(a != 0)
    delta <- b^2 - 4 * a * c
    if (delta >= 0) {
        den <- 2 * a
        sqrt_delta <- sqrt(delta)
        x1 <- (-b - sqrt_delta)/den
        x2 <- (-b + sqrt_delta)/den
        return(c(x1, x2))
    } else {
        return(NULL)
    }
}

baskara(-3, 2, 1)
# baskara(3, 2, 1)
# baskara(0, 2, 1)

## [1]  1.0000000 -0.3333333
```

Próximo assunto

- Representação de código.
- Práticas de implementação.
- Programação funcional.

Semana que vem

- Sabatina no Moodle.
- Aula com outro professor (talvez).