Introdução ao programa R

Ronald Targino, DEMA-UFC

Notas de aula

6. Programação - Controle de fluxo

6.1. if - else

```
# calcular as raízes do polinômio: a.x^2 + b.x + d
a = 5
b = 9
d = 1
delta = b^2 - 4 * a * d
if(delta >= 0){
  res = c('x1' = (-b - sqrt(delta)) / (2 * a), 'x2' = (-b + sqrt(delta)) / (2 * a))
  print(res)
} else {"delta negativo"}
```

```
##        x1        x2
## -1.681025 -0.118975
```

```
f = function(){
  a = scan(n = 1, what=double()) # what: logical, integer, numeric, complex, character, list.
  b = scan(n = 1, what=double())
  if(a < 0 | b < 0) return ("Entre com valores positivos") else return(c(a + b, b * a))
}
f()
resp = f()
resp
resp[1] # acessar os elementos do vetor resp
resp[2]
```

Atenção para a sintaxe

```
# desnecessário as chaves quando há uma única instrução
x = 5
if(x > 3) 10 else 40
```

```
## [1] 10
```

```
# desnecessário as chaves quando há uma única instrução
x = 5
y = 3
if(x != y) "valores diferentes" else "valores iguais"
```

```
## [1] "valores diferentes"
```

```
# desnecessário as chaves quando há uma única instrução
x = runif(1)
if(x > 0.5) y = 1 else y = 0
x;y
```

```
## [1] 0.8030777
```

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

```
# sintaxe anterior usando chaves({})
x = runif(1)
if(x > 0.5) {
  y = 1
} else {
    y = 0
  }
```

```
# encadeamento de condições
x = runif(1)
if(x < 0.25) {
  q = "q1"
} else {
  if(x < 0.50){
    q = "q2"
  } else {
    if(x < 0.75){
        q = "q3"
    } else {
      q = "q4"
    }
  }
}
x;q
```

```
## [1] 0.01742301
```

```
## [1] "q1"
```

6.2. for

```
# em cada iteração imprime o valor de 'i'
for(i in 1:4) {
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

```
# em cada iteração imprime o elemento da posição 'i' do vetor 'x'
x <- c("a", "b", "c", "d")
for(i in 1:4) {
  print(x[i])
}
```

```
## [1] "a"
## [1] "b"
## [1] "c"
## [1] "d"
```

```r
# imprimir os elementos de uma matriz
x <- matrix(1:6, 2, 3)
for(i in 1:nrow(x)) {
  for(j in 1:ncol(x)) {
    print(x[i, j])
  }
}
```

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

```r
# exercício: substituir 1:nrow(x) por  seq_len(nrow(x)) no código acima. Há alguma
# vantagem em usar seq_len(nrow(x))?
```

```r
# retornar a soma dos elementos de um vetor
x = 1:5
soma.x = 0 # inicializando o objeto
for(i in x) {
  soma.x = soma.x + i
}
x; soma.x
```

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

```
## [1] 15
```

```r
# retornar a soma acumulada dos valores de um vetor
x = 1:5
soma.x = 0
for(i in x) {
  soma.x = soma.x + i
  cat("Valor atual:", i, "\n")
  cat("Valor acumulado:", soma.x, "\n")
}
```

```
## Valor atual: 1
## Valor acumulado: 1
## Valor atual: 2
## Valor acumulado: 3
## Valor atual: 3
## Valor acumulado: 6
## Valor atual: 4
## Valor acumulado: 10
## Valor atual: 5
## Valor acumulado: 15
```

```r
# fatorial de um número inteiro
y = 4
fat = 1
if(y > 0 & (y-trunc(y)) == 0) {
  for(i in 1:y) fat = fat* i
  cat(" Valor:", y, "\n")
  cat(" Fatorial: ", fat)
```

```
} else {"Entre com um valor inteiro"}
```

```
##  Valor: 4
##  Fatorial:  24
```

```
# Sequência inversa e "sequência vazia"
n = 0
y = 0
for(i in 1:n) y[i] = i + 10
print(y)
```

```
## [1] 11
```

```
y = 0
for(i in seq_len(n)) y[i] = i + 10
print(y)
```

```
## [1] 0
```

6.3. while

```
# imprimir o valor atualizado de 'k' enquanto a condição 'k<10' for válida
k = 0
while(k < 10) {
  print(k)
  k = k + 1
}
```

```
## [1] 0
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
```

```
# usando mais de uma condição
z = 5
set.seed(1)
while(z >= 3 && z <= 10) {
  m <- rbinom(1, 1, 0.5)
  if(m == 1) {
    z = z + 1
    } else {
      z = z - 1
    }
}
z
```

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

6.4. ifelse

```
x = c(-2, -1, 0 , 1, 2)
ifelse(x >= 0, 'positivo', 'negativo')

ifelse(x >= 0, x^2, stop("Os valores devem ser positivos.", call. = FALSE))
```

```
x=runif(1); x; ifelse(x < 0.25, 1, ifelse(x < 0.5, 2, ifelse(x < 0.75, 3, 4)))
# reveja o último exemplo da Seção 6.1
```

6.5. Operadores lógicos

```
x=c(1,2,-2,3,4); x > 0
```

```
## [1]  TRUE  TRUE FALSE  TRUE  TRUE
```

```
# &: usa todos os elementos do vetor
# &&: usa apenas o 1o. elemento do vetor
y=c(1,2,4)
if(x>0 & y>0) print("verdadeiro") else print("falso")
```

```
## Warning in x > 0 & y > 0: comprimento do objeto maior não é múltiplo do
## comprimento do objeto menor
```

```
## Warning in if (x > 0 & y > 0) print("verdadeiro") else print("falso"): a
## condição tem comprimento > 1 e somente o primeiro elemento será usado
```

```
## [1] "verdadeiro"
```

```
if(x>0 && y>0) print("verdadeiro") else print("falso")
```

```
## [1] "verdadeiro"
```

```
if(all(x>0) &  all(y>0)) print("verdadeiro") else print("falso")
```

```
## [1] "falso"
```

```
if(all(x>0) && all(y>0)) print("verdadeiro") else print("falso")
```

```
## [1] "falso"
```

```
if(any(x>0) &  any(y>0)) print("verdadeiro") else print("falso")
```

```
## [1] "verdadeiro"
```

```
if(any(x>0) && any(y>0)) print("verdadeiro") else print("falso")
```

```
## [1] "verdadeiro"
```

```
x=c(1,2,-2,-3,-4)
y=c(5,2,1,6,7)

if(any(x>0 &  y<4)) print("verdadeiro") else print("falso")
```

```
## [1] "verdadeiro"
```

```
if(any(x>0 && y<4)) print("verdadeiro") else print("falso")
```

```
## [1] "falso"
```

```
# Exercício: Qual o 1o. número de Fibonacci maior que 100?
# F0=0, F1=1, F2=1, F3=2, Fn=F(n-1)+F(n-2), para n>=2
# Sequência de Fibonacci: 0,1,1,2,3,5,8,13,21,34,55,89,144,233, ...
# A sequência é iniciada no 0 ou no 1 ...
# A sequência de Fibonacci tem aplicações na análise de mercados
# financeiros, na ciência da computação e na teoria dos jogos.
# Também aparece em configurações biológicas, como, por exemplo,
# na disposição dos galhos das árvores ou das folhas em uma haste,
# no arranjo do cone da alcachofra, do abacaxi, ou no desenrolar
# da samambaia.
```