

Introdução a Linguagem R

Estruturas de controle

Delermundo Branquinho Filho

Introdução

Estruturas de controle em R permitem controlar o fluxo de execução do programa, dependendo das condições de tempo de execução. Estruturas comuns são

- **if,else**: testar uma condição
- **for**: executa um loop um número fixo de vezes
- **while**: executar um loop while uma condição é verdadeira
- **repeat**: executar um loop infinito
- **break**: interromper a execução de um loop
- **next**: ignora uma interação de um loop
- **return**: sair de uma função

A condição *if*

```
if(<condition>) {  
    ## do something  
} else {  
    ## do something else  
}  
if(<condition1>) {  
    ## do something  
} else if(<condition2>) {  
    ## do something different  
} else {  
    ## do something different  
}
```

Exemplos

```
x <- 3  
if(x > 3) {  
    y <- 10  
} else {  
    y <- 0  
}  
print(x,y)
```

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

Mais um ...

```
y <- if(x > 3) {
  10
} else {
  0
}
print(x,y)
```

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

Naturalmente, a cláusula else não é necessária.

```
if(<condition1>) {

}

if(<condition2>) {

}
```

for

For loops pegar uma variável interagir e atribuir-lhe sucessivos valores de uma sequência ou vetor. Para os loops são mais comumente usados para iterar sobre os elementos de um objeto (lista, vetor, etc.)

```
for(i in 1:10) {
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

Esses três loops têm o mesmo comportamento.

```
x <- c("a", "b", "c", "d")

for(i in 1:4) {
  print(x[i])
}
```

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

```
for(i in seq_along(x)) {
  print(x[i])
}
```

```
## [1] "a"
```

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

for(letter in x) {
  print(letter)
}
```

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

for(i in 1:4) print(x[i])
```

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

Aninhamento

for Loops podem ser aninhados.

```
x <- matrix(1:6, 2, 3)

for(i in seq_len(nrow(x))) {
  for(j in seq_len(ncol(x))) {
    print(x[i, j])
  }
}
```

Tenha cuidado com o aninhamento embora, aninhar além 2-3 níveis é muitas vezes muito difícil de ler / entender.

while

Enquanto os loops começam testando uma condição. Se for verdade, então eles executam o corpo do loop. Uma vez que o corpo do loop é executado, a condição é testada novamente, e assim por diante.

```
count <- 0
while(count < 10) {
  print(count)
  count <- count + 1
}
```

```
## [1] 0
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
```

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

Os loops *While* podem potencialmente resultar em loops infinitos se não escritos corretamente. Use com cuidado!

Às vezes, haverá mais de uma condição no teste.

```
z <- 5

while(z >= 3 && z <= 10) {
  print(z)
  coin <- rbinom(1, 1, 0.5)

  if(coin == 1) { ## random walk
    z <- z + 1
  } else {
    z <- z - 1
  }
}
```

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

As condições são sempre avaliadas da esquerda para a direita.

repeat

repeat inicia um loop infinito; Estes não são comumente utilizados em aplicações estatísticas, mas eles têm seus usos. A única maneira de sair de um loop **repeat** é chamar **break**.

```
x0 <- 1
tol <- 1e-8

repeat {
  x1 <- computeEstimate()

  if(abs(x1 - x0) < tol) {
    break
  } else {
    x0 <- x1
  }
}
```

repeat

O loop acima é um pouco perigoso porque não há garantia de que ele vai parar. É melhor definir um limite rígido no número de iterações (por exemplo, usando um loop for) e, em seguida, informar se a convergência foi alcançada ou não.

next, return

`next` É usado para ignorar uma iteração de um loop

```
for(i in 1:100) {  
  if(i <= 20) {  
    ## Skip the first 20 iterations  
    next  
  }  
  ## Do something here  
}
```

`return` Sinaliza que uma função deve sair e retornar um determinado valor

Estruturas de Controle

Resumo

- Estruturas de controle como `if`, `while` e `for` permitem controlar o fluxo de um programa R
- Loops infinitos geralmente devem ser evitados, mesmo que eles sejam teoricamente corretos.
- As estruturas de controle mencionadas aqui são principalmente úteis para escrever programas; Para o trabalho interativo de linha de comando, as funções `* apply` são mais úteis.

The Scientist