Aula 2.4 Estruturas de controle

Elder Sodre

Tuesday, February 24, 2015

Estruturas de controle controlam a execução do programa, de acordo com certas condições.

Normalmente, estruturas de controle não são utilizadas em sessões interativas. Em vez disso, elas são comumente usadas para escrever funções e expressões mais longas.

Neste curso, abordaremso as duas estruturas de controle mais comuns: if/else e for.

#### if/else

if/else permite que você teste uma condição é verdadeira. O R faz alguma coisa se a condição for verdadeira, e outra coisa se a condição for falsa.

Estrutura básica:

if(condição) {

Faz alguma coisa

} else {

Faz outra coisa

}

if/else também pode ser usado para testar mais de uma condição ao mesmo tempo:

if(condição1) {

Faz alguma coisa

} else if (condição2){

Faz alguma coisa diferente

} else {

Faz alguma coisa diferente

}

Por outro lado, o else é opcional; você pode fazer algo se a condição for verdadeira e não fazer absolutamente nada se a função for falsa:

if(condição) {

Faz alguma coisa

}

Vamos olhar as estruturas abaixo. Perceba que ambas as formas são válidas e fazem a mesma coisa:

x <- sample(1:10,1)  
if (x>3) {  
 y <- 10  
} else {  
 y <- 0  
}  
  
  
y <- if (x>3) {  
 10  
} else {  
 0  
}

Uma alternativa à estrutura if/else é a função ifelse(), que funciona assim:

ifelse(teste,sim,não)

ifelse(x>3,10,0)

## [1] 10

#### for

A estrutura for é uma das formas de fazer um loop, ou seja, fazer alguma coisa com sucessivos valores de uma sequência ou vetor.

Sua estrutura básica é:

for (i in1:n){ Para cada valor sucessivo de i até n

Alguma coisa

}

Vejamos mais alguns exemplos:

for (i in 1:10) { #Isto significa que, para cada número do 1 ao 10,  
 print(i) #o R vai retornar aquele valor.  
}

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

A estrutura acima pode não ser muito útil, mas serve como exemplo.

OBS: perceba que dentro da estrutura for, o R só retorna o que for explicitamente chamado dentro da função print(). Por exemplo, se eu escrever:

for (i in 1:10) {  
 log(i)  
}

Parece que o R não fez nada, mas na verdade ele calculou os valores de log e não nos mostrou. Uma forma mais útil de fazer isso seria com a função print() ou, melhor ainda, criando um objeto para guardar os resultados.

for (i in 1:10) {  
 print (log(i)) #Usando a função print  
}

## [1] 0  
## [1] 0.6931  
## [1] 1.099  
## [1] 1.386  
## [1] 1.609  
## [1] 1.792  
## [1] 1.946  
## [1] 2.079  
## [1] 2.197  
## [1] 2.303

resultado <- numeric(10) #Criei um vetor vazio para guardar os resultados  
for (i in 1:10) {  
 resultado[i] <- log(i) #Vou atribuir cada valor do log de i para a posição i do vetor  
}  
resultado #Viu só?

## [1] 0.0000 0.6931 1.0986 1.3863 1.6094 1.7918 1.9459 2.0794 2.1972 2.3026

Mais alguns exemplos. Observe que todos os casos abaixo são igualmente válidos.

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 1:length(x)) { #Útil para quando você não souber o comprimento do vetor x.  
 print(x[i])  
}

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

for (i in seq\_along(x)) { #A função seq\_along() cria automaticamente uma sequência de números inteiros de 1 até o comprimento do vetor.  
 print(x[i])  
}

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

for (i in 1:4) print(x[i]) #Quando a xpressão for pequena, pode-se omitir as chaves e colocar a expressão inteira na mesma linha.

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

Loops com for também podem ser aninhados, ou seja, você pode criar um for dentro do outro. Isto é frequentemente necessário para trabalhar com objetos bidimensionais, como matrizes.

Vejamos um exemplo:

y <- matrix(1:6,2,3)  
  
for (i in seq\_len(nrow(y))) { #obs: seq\_len(nrow(y)) cria uma sequência de números inteiros de 1 até o número de linhas de y; é a mesma coisa que escrever 1:nrow(y)  
 for (j in seq\_len(ncol(y))) { #Por convenção, i representa as linhas e j as colunas  
 print(y[i,j])  
 }  
}

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

Abaixo, outra forma de escrever a mesma coisa:

for (i in 1:nrow(y)) {  
 for (j in 1:ncol(y)) {  
 print(y[i,j])  
 }  
}

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