

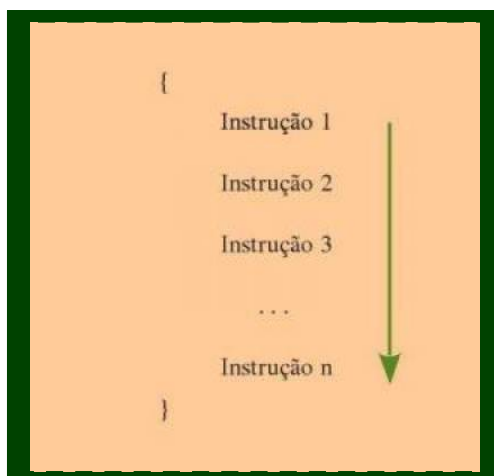
# Linguagem de Programação S Fundamentos e Aplicações em Recursos Florestais



## 5. Noções de Programação

### 5.3. Controle de Fluxo

O termo controle de fluxo se refer ao fato de que ao executar uma série de instruções, qualquer linguagem de programação (a linguagem S inclusive) procede na ordem seqüencial apresentada:



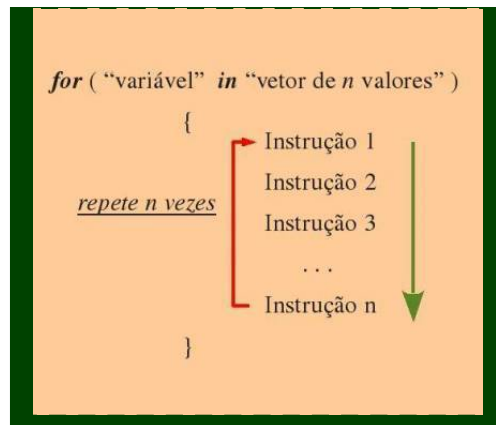
Assim, chama-se de controle de fluxo qualquer ação que visa controlar essa ordem seqüencial que as linguagens utilizam para executar as instruções.

#### 5.3.1. "For Loop"

Em linguagem de programação um loop é quando você força um programa a executar uma série de comandos repetidas vêzes. Num "for loop", isto é num "loop" do tipo "for", o número de repetições é definido no comando que estabelece o "loop".

A estrutura de loop no R é:





- A palavra **for** estabelece o “loop”.
- Dentro dos parênteses se define uma variável seguida da palavra **in** e um vetor de valores que a variável deverá assumir.
- Dentro das chaves se lista os comandos que devem ser repetidos a cada passo do “loop”.

Note que os comandos de “1” a “n” serão repetidos o número de vezes correspondente ao comprimento desse vetor.

Vejamos um exemplo: *Convergência da distribuição t de Student para distribuição Gaussiana (Normal) Padronizada:*

```
1 #####
2 ##### 5. NOÇÕES DE PROGRAMAÇÃO
3 ##### 5.3. Controle de Fluxo
4 #####
5 # 5.3.1. "For Loop"
6
7 converg.t2z = function( gl.max = 200)
8 {
9 #
10 # Convergência da distribuição t de Student
11 #     para distribuição Gaussiana Padronizada
12 #
13   curve(dnorm(x), from=-4, to=4, col="red", lwd=6)
14   for(gl in 1:gl.max)
15   {
16     curve(dt(x, gl), -4, 4, add=TRUE, col="green")
17     for(i in 1:100000) i
18   }
19 }
20
21 converg.t2z()
22
```

No exemplo acima temos a função **converg.t2z** com os seguintes elementos:

- **gl** é a variável definida para o “loop”;
- **1:gl.max** é o vetor de valores que a variável assumirá, logo, o “loop” será repetido **gl.max** vezes. O valor do argumento **gl.max** é definido por default como sendo 200.

## 5.3.2. Solução Vetorial x Loop

Sendo um ambiente vetorial os “loops” não são uma opção muito eficiente para computação dentro do R. Em geral, o R é mais eficiente se encontrarmos uma solução vetorial para problemas de computação que aparentemente exigem um loop. A solução vetorial, entretanto, costuma ser mais exigente em termos do tamanho de da memória RAM do computador.

Considere o problema o seguinte problema: temos a localização espacial de plantas num plano cartesiano com coordenadas  $(x, y)$ . Por exemplo:

```
23 # 5.3.2. Solução Vetorial x Loop
24
25 x = runif(100)
26 y = runif(100)
27 plot(x,y)
28
```

O objetivo é obter as distâncias entre as plantas duas-a-duas. Primeiro consideremos uma solução através de “loop”:

```
29 inter.edist = function(x, y)
30 {
31     n = length(x)
32     dist <- c()
33     for(i in 1:(n-1))
34     {
35         for(j in (i+1):n)
36         {
37             dist <- c(dist, sqrt( (x[i] - x[j])^2 + (y[i] -
y[j])^2 ))
38         }
39     }
40     dist
41 }
42
```

Consideremos agora uma solução vetorial:

```
43 inter.edist.v = function(x, y)
44 {
45     xd <- outer( x, x, "-" )
46     yd <- outer( y, y, "-" )
47     z <- sqrt( xd^2 + yd^2 )
48     dist <- z[ row(z) > col(z) ]
49     dist
50 }
51
```

Qual dessas soluções é mais eficiente em termos do uso do tempo?

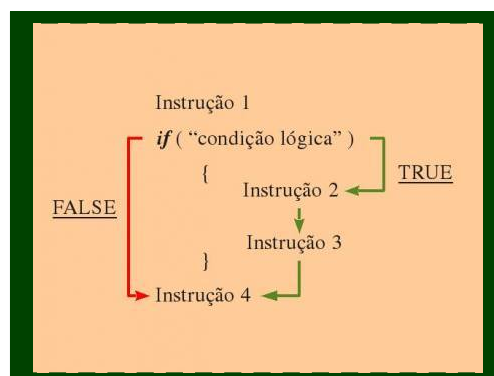
```
52 x = runif(400)
53 y = runif(400)
54 plot(x,y)
55
56 system.time( inter.edist(x,y) )
57 system.time( inter.edist.v(x,y) )
58
```

A função **system.time** retorna o tempo que a CPU necessita para executar a expressão da linguagem S que lhe fornecemos como argumento. Note que ela, realmente executa a expressão para cronometr -la.

- Por isso, n o tente rodar o exemplo acima com 1000 ou mais observa es, pois o tempo fica realmente longo para vers o em "loop".
- CONCLUS O: use apenas pequenos "loops" no R!

### 5.3.3. Controle de Fluxo: "If"

O controle **if** permite controlar se um conjunto de instru es   realizado no case de uma dada condi o l gica for verdadeira:



Um exemplo envolvendo o controle "if":

```
59 # 5.3.3. Controle de Fluxo: "If"
60
61 example.if = function(x)
62 {
63     y = 2*x
64     if( !is.null(names(y)) )
65     {
```

```

66     y.nm = names(y)
67     print(y.nm)
68   }
69   y
70 }
71

```

**Observação:** a função `is.null` retorna o valor **TRUE** se o seu argumento for nulo (**NULL**).

Um exemplo de aplicação da função `example.if`:

```

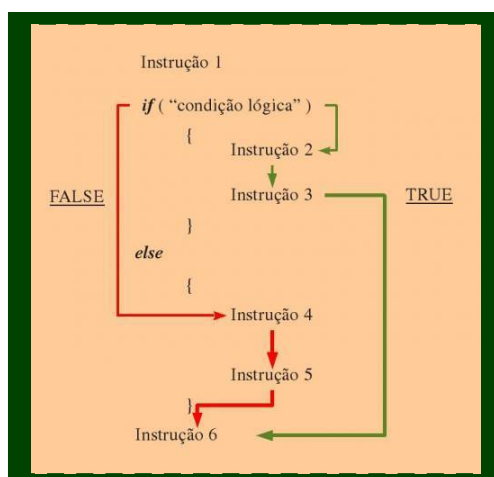
72 x = 1:3
73 x2 = x
74 names(x2) = c("a","b","c")
75
76 example.if(x)
77 example.if(x2)
78

```

### 5.3.4. Controle de Fluxo: "If Else"

O controle `if else` definir a seguinte situação:

- a condição lógica é verdadeira: executa-se o conjunto de instruções A (instruções 2 e 3, na figura abaixo);
- a condição lógica é falsa: executa-se o conjunto de instruções B (instruções 4 e 5, na figura abaixo).



Exemplo do controle `if-else`:

```

79 # 5.3.4. Controle de Fluxo: "If Else"
80
81 example.iffalse = function(x)

```

```
82 {
83     y = 2*x
84     if( !is.null(names(y)) )
85     {
86         print(names(y))
87     }
88     else
89     {
90         print(y)
91     }
92     print("Fim do controle IF-ELSE")
93 }
94
```

Aplicando a função:

```
95 x = 1:3
96 x2 = x
97 names(x2) = c("a","b","c")
98
99 example.ifelse(x)
100 example.ifelse(x2)
101
```

### 5.3.5. Controle de Fluxo: "Stop"

A função **stop** ela aborta a execução das instruções na linha em que ela for colocada, executando uma ação de erro. Vejamos um exemplo:

```
102 # 5.3.5. Controle de Fluxo: "Stop"
103
104 example.stop = function(x,y)
105 {
106     if( length(x) != length(y) )
107         stop("Os vetores 'x' e 'y' devem ter o mesmo
comprimento")
108     out = cbind(x,y)
109     out
110 }
111
```

Uma exemplo de aplicação:

```
112 x1 = 1:5
113 x2 = 6:10
```

```
114 x3 = 11:20
115
116 example.stop(x1,x2)
117 example.stop(x1,x3)
118
```

From:

<http://insilvaarbores.com.br/dokuwiki/> - **In Silva, Arbores ...**

Permanent link:

[http://insilvaarbores.com.br/dokuwiki/doku.php?id=pt:cursos\\_online:s\\_linguagem:5-nocoas-programacao:5-3-controle-fluxo](http://insilvaarbores.com.br/dokuwiki/doku.php?id=pt:cursos_online:s_linguagem:5-nocoas-programacao:5-3-controle-fluxo)

Last update: **2024/10/22 00:12**

