Linguagem de Programação

Cristiano de Carvalho Santos
professor.pacotes.estatisticos@gmail.com
Grupo Google: PacotesEstatisticos2016

Departamento de Estatística,
Universidade Federal de Minas Gerais (UFMG)

Agrupando comandos

- É possível atribuir os mesmos valores a vários objetos de uma só vez utilizando atribuições múltiplas de valores
  a <- b <- 10;
- Um grupo de comandos podem ser agrupado com "{ }" e separados por ";" para digitação em uma mesma linha. Em certas situações, como no "prompt" do R as chaves são opcionais.

Execução condicional

Execuções condicionais são controladas por funções especiais que verificam se uma condição é satisfeita para permitir a execução de um comando. As seguintes funções e operadores podem ser usadas para controlar execução condicional.

- if() (opcionalmente) acompanhado de else
- ifelse()
- switch()
- &, |, && e ||

if

- A estrutura if() else é comumente usada, em especial dentro de funções;
- Quando a expressão que segue o if() e/ou else tem uma única linha ela pode ser escrita diretamente, entretanto, caso sigam-se mais de duas linhas deve-se usar chaves;

Exemplo 1:

```
if(condição){
   ações a serem realizadas se
   condição for verdadeira
}
```

Exemplo 2:

```
if(condição){
   ações a serem realizadas se
   a condição for verdadeira
} else{
   ações a serem realizadas se
   a condição for falsa
}
```

else if é usado para especificar um segunda condição para o caso em que a primeira não foi atendida.

Exemplo 3:

```
if(condição 1){
   ações a serem realizadas se
   a condição for verdadeira
} else if(condição 2){
   ações a serem realizadas se
   a condição for falsa
}
```

ifelse

Uso:

`ifelse(teste, yes, no)`

- "teste" é a condição a ser testada;
- "yes" é o valor retornado se a condição for verdadeira;
- "no" é o valor retornado se a condição for falsa.

switch

- Permite várias comparações sem a necessidade de vários comandos de if;
- É mais rápido que o if.

Exemplo:

```
switch(tipo,
mean = 1,
median = 2,
trimmed = 3)
```

- "tipo" neste caso é uma variável (com números ou caracteres) que será testada;
- mean, median e trimmed são possíveis respostas para tipo. Se tipo for igual a mean a função retornará 1 e assim por diante.

comandos lógicos

- & e && são usados para fazer interseção de condições. No caso vetorial eles se diferem, sendo que & faz a interseção para para cada entrada do vetor e && faz usando apenas a primeira entrada do vetor;
- | e || são usados para fazer união de condições e diferem da mesma forma que o comando de interseção;
- ! pode ser usado para negar uma condição.

Loops no R

- O controle de fluxo no R é implementado pelas funções for(), while() e repeat().
- A escolha de qual usar vai depender do contexto e objetivo do código e em geral não existe solução única, sendo que uma mesma tarefa pode ser feita por uma ou outra.

Estruturas

for

```
for(i in conjunto de indices){
    Ações a serem executadas para i igual a todos os ind
}
```

while

```
i <- 1
while(condição){
    Ações a serem executadas enquanto a condição for ver
    i <- i + 1
}
```

repeat

```
i <- 1
repeat{
    Ações a serem executadas
    if(condição){
        break
    }
    i <- i+1
}
```

Tempo computacional

- **proc.time** determina o tempo real e tempo de CPU (em segundos) que o atual processo do R está em execução. Para descobrir o tempo de execução de uma função podemos subtrair o tempo depois e antes de rodar a função;
- **system.time** retorna o tempo de CPU e outros que o comando avaliado usou;
- Função microbenchmark compara o tempo computacional de execução de duas funções.

Família apply

O R é uma linguagem vetorial e "loops"podem e devem ser substituídos por outras formas de cálculo sempre que possível. Usualmente usamos as funções apply(), sapply(), tapply() e lapply() para implementar cálculos de forma mais eficiente.

- replicate() repete o uso de alguma função
- apply() para uso em matrizes, arrays ou data-frames
- tapply() pode ser usada para calcular o resultado de uma operação sobre dados, para cada um dos níveis de uma segunda variável

- sapply() para uso em vetores, simplificando a estrutura de dados do resultado se possível (para vetor ou matriz)

- mapply() para uso em vetores, versão multivariada de sapply()
- lapply() para ser aplicado em listas
- vapply() é similar ao sapply, mas tem um tipo predeterminado de valor retornado. Pode ser mais seguro e rápido
- rapply() para a versão recursiva do lapply

Lista de Exercícios 4

- Forma de entrega: Mandar por email um arquivo ".txt" ou ".R" com os comandos utilizados na resolução da lista de exercícios.
- Salvar arquivo com nome Lista4-nomes dos autores-incompleta ou Lista4-nomes dos autores-final.

Exercícios:

1. Usando o for(), crie uma função que calcule o fatorial de n. Compare os resultados e o tempo computacional de sua implementação com a função factorial() e com a função recursiva fatorial implementada em sala de aula.

2. Crie uma função que encontre o máximo de um vetor (use for() na sua função). Compare os resultados e a performance de sua implementação com a função max() do R. Sua função é quantas vezes mais lenta?

3. Crie duas funções que calculem a soma de um vetor usando os comandos while e repeat. Compare os resultados e o tempo computacional com a função sum().

4. Crie uma função que calcule a soma acumulada de um vetor. Compare os resultados e a performance de sua implementação com a função cumsum() do R.

5. Descubra o que fazem as funções "split" e "aggregate" e crie um exemplo para ilustrar.

6. Usando os comandos if e else, construa uma função que calcule, para um vetor de dados, uma medida de tendência central escolhida como parâmetro. Esta função deve ser capaz de calcular média, mediana, média geométrica, média harmônica e média aparada. Esta função deve ter como parâmetros o vetor e a medida escolhida.

7. Refaça o exercício anterior usando o comando switch.

8. Gere um vetor de 100 observações da distribuição Normal padrão e categorize os dados em três categorias: 0 para observações menores que -1, 1 para valores entre -1 e 1, 2 para valores maiores que 1. Construa dois códigos para fazer a categorização, o primeiro usando if e segundo usando o ifelse (É necessário usar essa função duas vezes).

9. Pesquise e use a função cut() para resolver o exercício anterior.

10. Use a função replicate() para calcular a média de 100 amostras com tamanho 50 de uma distribuição Log-Normal padrão.

11. Considere que Y é mínimo de uma amostra $z_1, \ldots, z_N$ de tamanho N da distribuição Exponencial com parâmetro igual a 2. Use o sapply para gerar 20 valores de Y após informar um vetor com 20 valores de N. Inicialmente é necessário criar um função para gerar um valore de Y para um N especificado.

12. Usando o mesmo problema do exercício anterior, construa uma função que para um determinado valor de N retorne valores de Y e $z_1, \ldots, z_N$ em uma lista com duas posições. Gere uma amostra com 10 valores de N a partir de uma distribuição uniforme discreta entre 1 e 5. Usando o lapply, aplique a função construída a todos os valores de N gerados.