Universidade Federal do Amazonas
Departamento de Estatística

Curso: Introdução à Ciência de Dados
Professor: Leonardo Nascimento
14/12/2023
Versão 1.0

1 Controle de Fluxo

- O controle de fluxo em R refere-se à capacidade de direcionar a execução do código com base em condições específicas.
- Existem várias estruturas de controle de fluxo em R que permitem que você tome decisões, repita operações e controle o fluxo de execução do seu programa:
- `if`, `else if`, `for`, `while`, `repeat` e `switch`.

1.1 if, else e else if

- `if` é usado para executar um bloco de código se uma condição for verdadeira.
- O `else` funciona como uma extensão do `if`, apresenta outra alternativa para o caso do teste executado em `if` seja falso.

```
if (condition) true_action
```

```
if (condition) true_action else false_action
```

- If `condition` is `TRUE`, `true_action` é avaliada;
- if `condition` is `FALSE`, a opção `false_action` é avaliada.

```
x = "Leonardo"
if(class(x)=="character"){
  cat(" O objeto x=",x,"é da classe character",sep=" ")
}
#>  O objeto x= Leonardo é da classe character
```

```r
x = 1

if(class(x)=="character"){
  paste(" O objeto x=",x,"é da classe character",sep=" ")
} else{
  paste(" O objeto x=",x,"não é da classe character",sep=" ")
}
#> [1] " O objeto x= 1 não é da classe character"
```

```r
x = 70
 if (x >= 90 & x <= 100) {
    "A"
  } else if (x >= 80 & x < 90) {
    "B"
  } else {
    "C"
  }
#> [1] "C"
```

```r
x = 80
 if (x >= 100 | x <= 70) {
    "Extremo"
  } else {
    "Não é extremo"
  }
#> [1] "Não é extremo"
```

```r
x = 1:10
ifelse(x %% 2 == 0, "par", "ímpar")
#>  [1] "ímpar" "par"   "ímpar" "par"   "ímpar" "par"   "ímpar" "par"   "ímpar"
#> [10] "par"
```

1.2 for

- Utilizado para iterar sobre uma sequência de elementos.

```r
for (variavel in sequencia) {
  # código a ser executado em cada iteração
}
```

- Ao executar este código, você verá a saída que mostra os quadrados dos números de 1 a 5.

```r
for (i in 1:5) {
  quadrado <- i^2
  cat("O quadrado de", i, "é", quadrado, "\n")
}
#> O quadrado de 1 é 1
#> O quadrado de 2 é 4
```

```
#> O quadrado de 3 é 9
#> O quadrado de 4 é 16
#> O quadrado de 5 é 25
```

- Ao executar este código, você obterá a saída mostrando a soma dos primeiros 10 números naturais.

```
soma <- 0
for (i in 1:10) {
  soma <- soma + i
}
cat("A soma dos primeiros 10 números naturais é:", soma, "\n")
#> A soma dos primeiros 10 números naturais é: 55

cumsum(1:10) # forma alternativa
#>  [1]  1  3  6 10 15 21 28 36 45 55
```

```
n = 10 # linhas
m=3 # colunas
matrix_normal = matrix(0,nrow = n,ncol = m) # criando a matriz

set.seed(2) # gerar sempre as mesmas amostras
for(k in 1:m){
  x = rnorm(n,mean = 0,sd = 1) # gerando amostra da distribuição normal
  matrix_normal[,k] <- x # guardando a amostra gerada na coluna
}
matrix_normal
#>              [,1]        [,2]         [,3]
#>  [1,] -0.89691455  0.41765075  2.090819205
#>  [2,]  0.18484918  0.98175278 -1.199925820
#>  [3,]  1.58784533 -0.39269536  1.589638200
#>  [4,] -1.13037567 -1.03966898  1.954651642
#>  [5,] -0.08025176  1.78222896  0.004937777
#>  [6,]  0.13242028 -2.31106908 -2.451706388
#>  [7,]  0.70795473  0.87860458  0.477237303
#>  [8,] -0.23969802  0.03580672 -0.596558169
#>  [9,]  1.98447394  1.01282869  0.792203270
#> [10,] -0.13878701  0.43226515  0.289636710

# forma alternativa
set.seed(2)
replicate(m,rnorm(n,mean = 0,sd = 1))
#>              [,1]        [,2]         [,3]
#>  [1,] -0.89691455  0.41765075  2.090819205
#>  [2,]  0.18484918  0.98175278 -1.199925820
#>  [3,]  1.58784533 -0.39269536  1.589638200
#>  [4,] -1.13037567 -1.03966898  1.954651642
#>  [5,] -0.08025176  1.78222896  0.004937777
#>  [6,]  0.13242028 -2.31106908 -2.451706388
#>  [7,]  0.70795473  0.87860458  0.477237303
#>  [8,] -0.23969802  0.03580672 -0.596558169
```

```
#>  [9,]  1.98447394  1.01282869  0.792203270
#> [10,] -0.13878701  0.43226515  0.289636710
```

Utilizando dois `for`

```
# Criar uma matriz 3x3
matriz <- matrix(0, nrow = 3, ncol = 3)

# Preencher a matriz com números sequenciais usando um loop for
contador <- 1

for (i in 1:3) {
  for (j in 1:3) { # fixa o i e varia o j
    matriz[i, j] <- contador
    contador <- contador + 1
  }
}

print(matriz)
#>      [,1] [,2] [,3]
#> [1,]    1    2    3
#> [2,]    4    5    6
#> [3,]    7    8    9
```

1.3 while

- Também é utilizado para iterar sobre uma sequência de elementos.

```
while (condition) {
  # código executado enquanto a condição for verdade
}
```

- O código a seguir irá imprimir o valor de `i` enquanto este objeto for menor que 5.
- Quando a condição não for mais respeitada, o processo será interrompido.

```
i = 0
while(i<5){
  cat(i,"é menor que 5","\n")
  i = i+1
}
#> 0 é menor que 5
#> 1 é menor que 5
#> 2 é menor que 5
#> 3 é menor que 5
#> 4 é menor que 5
```

- Lançamento de um dado até que um determinado número seja obtido.

```
numero_escolhido <- 6
resultado_dado = 1
```

```r
while (resultado_dado!=numero_escolhido) {
  resultado_dado = sample(1:6,1,replace = T,prob = NULL) # sorteando número de 1 a 6
  cat("Número do dado é",resultado_dado,"\n")
}
#> Número do dado é 1
#> Número do dado é 2
#> Número do dado é 4
#> Número do dado é 5
#> Número do dado é 6
```

1.4 repeat

- A ideia é repetir um loop e parar quando a condição for satisfeita, para isso será utiliza a função `break`

```r
repeat {
  # código será executado repetidamente
  if (condition) {
    break  # se a condição for satisfeita, para a repetição
  }
}
```

- Rode esse código no seu computador
- O programa irá gerar um número e o usuário vai ficar tentanto acertar o número gerado

```r
# Gera um número aleatório entre 1 e 10
numero_correto <- sample(1:10, 1,replace = T)

# Inicializa a variável para armazenar a tentativa do usuário
tentativa_usuario <- NULL

# Use um loop repeat para continuar o jogo até que o número correto seja adivinhado
repeat {
  # Solicita que o usuário insira uma tentativa
  tentativa_usuario <- as.integer(readline("Tente adivinhar o número (entre 1 e 10): "))

  # Verifica se a tentativa do usuário é correta
  if (tentativa_usuario == numero_correto) {
    cat("Parabéns! Você adivinhou corretamente.\n")
    break  # Sai do loop quando adivinha corretamente
  } else {
    cat("Tente novamente. Dica: ", ifelse(tentativa_usuario < numero_correto, "Tente um número
        maior.", "Tente um número menor."), "\n")
  }
}
```

- Rode esse código no seu computador
- Avaliar como o tamanho da amostra afeta a média amostral

```r

media = 30
```

```
desvio_padrao = 5
erro = 10^(-3) # escolha com cuidado, pode travar o computador
n = 1 # tamanho da amostra inicial
crescimento_do_n = 3 # se a condição não for satisfeita, o n anterior será multiplicado por 3
# escolha com cuidado, pode travar o computador

repeat{
  x = rnorm(n,mean = media,sd = desvio_padrao) # gerando amostra
  media_x = mean(x) # média da amostra

  if(abs(media_x-media)<erro){
    cat("n=",n,"media_x"=",media_x=",media_x,"\n")
    break
  }else{
    n = n*crescimento_do_n
     cat("n=",n,"media_x"=",media_x",media_x,"\n")
  }
}
```

1.5 switch

- A função `switch` uma construção condicional que permite escolher um valor com base na correspondência de casos.
- Ela é frequentemente utilizada quando você tem várias opções e deseja realizar diferentes ações dependendo do valor de uma expressão.
- É útil quando você tem uma série de casos possíveis e deseja evitar a aninhamento excessivo de instruções `if-else`.
- Torna o código mais limpo e legível, especialmente em situações em que há muitas opções possíveis.
- Esse exemplo faz uma correspondência entre o número e o dia da semana

```
numero = 1
dia <- switch(numero,
                "1"="Domingo",
                "2"="Segunda-feira",
                "3"="Terça-feira",
                "4"="Quarta-feira",
                "5"="Quinta-feira",
                "6"="Sexta-feira",
                "7"="Sábado"
               )
cat("O dia correspondente ao número", numero, "é", dia, "\n")
#> O dia correspondente ao número 1 é Domingo
```

- Esse exemplo faz uma correspondência entre o número percentual (character) e o número decimal (numeric)

```
PROP = "25%"
prop <- switch (PROP,
    "25%" = 0.25,
```

```
    "50%" = 0.5,
    "75%" = 0.75
  )
prop
#> [1] 0.25
```

2 Funções

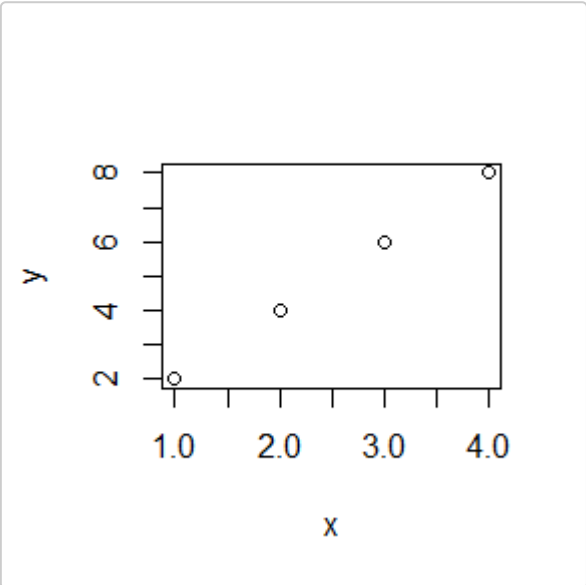
- Funções são blocos de código que podem ser chamados para realizar uma tarefa específica
- Elas facilitam a organização do código, promovem a reutilização e tornam o código mais organizados.
- Dica: você deve considerar escrever uma função sempre que copiar e colar um bloco de código mais de duas vezes
- Para definir uma função em R, você usa a palavra-chave `function` seguida pelos **parâmetros/argumentos** da função e o **corpo da função**

```
name <- function(arguments) {
  body
  return(objeto que guarda o resultado final)
}
```

- $f(x) = 2x$

```
f_x = function(x){
  f_x = 2*x
  return(f_x)
}

x = c(1,2,3,4)
y = f_x(x)
plot(x,y)
```



- Média de um vetor numérico

```r
media = function(x){
  media_x = sum(x)/length(x)
  return(media_x)
}
y = sample(1:100,20,replace=T)
media(y)
#> [1] 48.35
```

- Função para fazer uma correspondência entre o número e o dia da semana

```r
dia_semana <- function(numero_dia) {
  dia <- switch(
    numero,
    "1" = "Domingo",
    "2" = "Segunda-feira",
    "3" = "Terça-feira",
    "4" = "Quarta-feira",
    "5" = "Quinta-feira",
    "6" = "Sexta-feira",
    "7" = "Sábado"
  )
  return(cat("O dia correspondente ao número", numero, "é", dia, "\n"))
}
dia_semana(2)
#> O dia correspondente ao número 1 é Domingo
```

3 Referências

- Advanced R
- R for Data Science
- Hands-On Programming with R