



1



2

Scripts

- Criar um arquivo texto contendo um script para execução no R
 - Mais fácil de manter
 - Manter scripts armazenados e não só no workspace
- Convencionou-se a extensão .R

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

3

3

Scripts: teste.R

```
# isso é um comentário  
x <- 1:10  
  
# Para mostrar, usar print  
print(x)  
  
# Mostra a soma dos elementos do vetor  
print(sum(x))
```

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

4

4

Scripts: Execução do teste.R .

\$ Rscript teste.R

```
[razer@Neumann:~$ Rscript teste.R  
[1] 1 2 3 4 5 6 7 8 9 10  
[1] 55  
razer@Neumann:~$
```

Comandos Condicionais

- Comando `if else`

```
if (condição) {  
    comandos  
}  
  
else {  
    comandos  
}
```

Comandos Condicionais

- Exemplo

```
> x <- 1
> if(x < 0) {
+   sinal <- "negativo"
+ } else if(x == 0) {
+   sinal <- "neutro"
+ } else if(x > 0) {
+   sinal <- "positivo"
+ }
> sinal
[1] "positivo"
```

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

7

7

Comandos Condicionais

- Função ifelse()

- Vetorizada

```
ifelse(vetor, valor_se_TRUE, valor_se_FALSE)
```

- Exemplo

```
> ifelse(c(TRUE, FALSE, FALSE, TRUE), 1, -1)
[1] 1 -1 -1 1
```

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

8

8

Comandos Condicionais.

- Um pouco mais complexo, com vetor e condição
 - Saber se os números em um vetor são pares ou ímpares

```
> vet <- 1:10
> ifelse(vet %% 2 == 0, "par", "impar")
[1] "impar" "par"  "impar" "par"  "impar" "par"  "impar"
"par"    "impar"
[10] "par"
```

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

9

9

Laços

- Comando FOR

```
for (variável in conjunto) {
  comandos
}
```

- Exemplo

```
for (i in 1:5) {
  print(i)
}
```

```
for (i in letters[1:5]) {
  print(i)
}
```

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

10

10

Laços

- Exemplo

```
for (i in seq(10, 20, 2)) {  
  print(i)  
}  
  
for (i in seq(1, 6, 0.5)) {  
  print(i)  
}  
  
for (i in seq(10, 1, -1)) {  
  print(i)  
}
```

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

11

11

Laços

- Exemplo

```
vet <- c(10, 15, 20)  
for (i in vet) {  
  print(i)  
}  
  
lista <- as.list(1:10)  
for (i in lista) {  
  print(i)  
}
```

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

12

12

Laços

- Pode-se varrer uma lista com elementos diversos
 - A cada volta do laço a variável de interação recebe um dos elementos da lista

```
lista <- list(prim="oi mundo", seg=list(a=10, b=20), ter=c(111,222))
for (x in lista) {
  cat("-----\n");
  print(x);
}
```

```
-----
[1] "oi mundo"
```

```
$a
```

```
[1] 10
```

```
$b
```

```
[1] 20
```

```
-----
[1] 111 222
```

Prof. Dr. Razer A N R Montañó

SEPT/UFPR

13

13

Laços

- Para imprimir todos os elementos das sub-listas/vetores individualmente, aninha-se um laço

```
lista <- list(prim="oi mundo", seg=list(a=10, b=20), ter=c(111,222))
for (x in lista) {
  for (valor in x) {
    cat("-----\n");
    print(valor);
  }
}
```

```
-----
[1] "oi mundo"
```

```
-----
[1] 10
```

```
-----
[1] 20
```

```
-----
[1] 111
```

```
-----
[1] 222
```

Prof. Dr. Razer A N R Montañó

SEPT/UFPR

14

14

Laços

- Pode-se varrer uma matriz
 - Usa-se laços aninhados para varrer linha e coluna

```
matriz <- matrix(data=1:12, nrow=6, ncol=2)
for (l in 1:nrow(matriz)) {
  for (c in 1:ncol(matriz)) {
    print( matriz[l, c]*10 )
  }
}
```

Prof. Dr. Razer A N R Montão

SEPT/UFPR

15

15

Laços

- Deve-se sempre lembrar que R é vetorizado
- Então um laço como este

```
> x <- 10:20
> for (i in seq_along(x))
+ x[i] <- x[i]/10
```

```
> x
```

```
[1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0
```

- Pode ser trocado por um comando como este

```
> x <- 10:20
```

```
> x <- x/10
```

```
> x
```

```
[1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0
```

Cria um vetor com os índices (início em 1) para poder percorrer o vetor e indexá-lo

Prof. Dr. Razer A N R Montão

SEPT/UFPR

16

16

Laços

- Comando WHILE

```
while (expressão) {  
    comandos  
}
```

- Exemplo

```
i <- 1  
while (i <= 10) {  
    print(i)  
    i <- i+1  
}
```

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

17

17

Laços

- Comando REPEAT

```
repeat {  
    comandos  
}
```

- Não há condição de saída

- O programador deve usar o comando **break** para sair

- Exemplo

```
i <- 1  
repeat {  
    print(i)  
    i <- i+1  
    if (i > 10) { break }  
}
```

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

18

18

Laços: Controle de Fluxo.

- Comando BREAK e NEXT

- BREAK: para a execução do laço e sai

```
i <- 1
repeat {
  print(i)
  i <- i+1
  if (i > 10) {
    break
  }
}
```

- NEXT: para a execução do laço e vai para a próxima iteração

```
i <- 1
while (i <= 10) {
  if (i %% 2 == 0) {
    i <- i+1
    next
  }
  print(i)
  i <- i+1
}
```

Prof. Dr. Razer A N R Montão

SEPT / UFPR

19

19



Exercícios

- Escreva um laço que varre os números de 1 a 7 e imprime seus quadrados, usando o comando **print()**.
- Usando laços, varra uma lista de números aleatórios gerados por **rnorm()**, mas pare se o número encontrado for mais que 1.
- Usando laços, varra uma lista de números aleatórios gerados por **rnorm()**, mas use o comando **next** para pular os números negativos.
- Use laços aninhados para criar a matriz abaixo. Faça a alocação prévia da matriz com valores NA.

```
0  1  2  3  4
1  0  1  2  3
2  1  0  1  2
3  2  1  0  1
4  3  2  1  0
```

Prof. Dr. Razer A N R Montão

SEPT / UFPR

20

20



Exercícios

- Crie o seguinte data frame:

```
student.df <- data.frame( name = c("Sue", "Eva", "Henry", "Jan"),
                          sex = c("f", "f", "m", "m"),
                          years = c(21, 31, 29, 19));
```

- Usando um comando **ifelse()**, crie uma coluna chamada **teen**, booleana, que indica se a pessoa possui menos de 20 anos.

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

21

21



Exercícios

- Crie o seguinte data frame:

```
a = c(3, 7, NA, 9)
b = c(2, NA, 9, 3)
f = c(5, 2, 5, 6)
d = c(NA, 3, 4, NA)
```

```
mydf = data.frame(a=a, b=b, f=f, d=d)
```

- Adicione uma quinta coluna usando as seguintes regras:

1. A 5a coluna tem o valor da coluna 2 se a coluna 1 é NA
2. A 5a coluna tem o valor da coluna 4 se a coluna 2 é NA
3. A 5a coluna contém o valor da coluna 3 em qualquer outro caso

- O resultado deve ser:

	a	b	f	d	V5
1	3	2	5	NA	5
2	7	NA	2	3	3
3	NA	9	5	4	9
4	9	3	6	NA	6

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

22

22



Exercícios.

- Crie uma matriz com 10 colunas contendo 100.000 números, sendo os números de 1:100000. Faça um laço `for` que calcula a soma de cada linha desta matriz.
- Crie o seguinte data frame

```
vector1 <- 1:10

vector2 <- c("Odd", "Loop", letters[1:8])

vector3 <- rnorm(10, sd = 10)

df1 <- data.frame(vector1, vector2, vector3, stringsAsFactors = FALSE)
```

- Faça um laço genérico sobre as colunas deste data frame efetuando o seguinte cálculo:
 - Se a coluna for numérica, calcula sua média
 - Se a coluna for de texto calcula a soma dos caracteres na coluna (`nchar()`)

Funções Customizadas (UDF)

- UDF – *User-Defined Functions*
- Sintaxe Básica

```
nome <- function (lista de argumentos) { corpo da função }
```

- Para o retorno usa-se

```
return ()
```

- Exemplo:

```
minha.funcao <- function( argumento1, ...) {
  comando 1
  comando 2
  ...
  return(resultado)
}
```

Funções Customizadas (UDF)

- Exemplo:

```
> potencia <- function(x, n) { return (x^n) }  
> potencia  
function(x, n) { return (x^n) }  
> potencia(2, 3)  
[1] 8
```

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

25

25

Funções Customizadas (UDF)

- Exemplo: média de um vetor de valores

```
> media <- function(valores) {  
+ n <- length(valores)  
+ med <- sum(valores) / n  
+ return (med)  
+ }  
  
> dados <- c(10, 20, 30, 40)  
> media(dados)  
[1] 25
```

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

26

26

Funções Customizadas (UDF)

- Parâmetros
 - São passados por cópia
 - Se forem alterados dentro da função, não altera o parâmetro passado fora da função

Prof. Dr. Razer A N R Montão

SEPT/UFPR

27

27

Funções Customizadas (UDF)

- Exemplo: Passagem de parâmetros

```

> funcao <- function(x) {
+   x[["oi"]] <- "tchau"
+   return (x)
+ }
> a <- list(teste="teste")
> a
$teste
[1] "teste"
> y <- funcao(a)

```

```

> y
$teste
[1] "teste"
$oi
[1] "tchau"
> a
$teste
[1] "teste"

```

Prof. Dr. Razer A N R Montão

SEPT/UFPR

28

28

Funções Customizadas (UDF)

- Parâmetros podem ser opcionais e ter valores default
 - Usa-se o símbolo = na definição do parâmetro

```
potencia <- function(x, y=2) {
  return (x^y)
}
res <- potencia(10)
print(res)
res <- potencia(10, 3)
print(res)
```

```
> potencia <- function(x, y=2) {
+   return (x^y)
+ }
> res <- potencia(10)
> print(res)
[1] 100
> res <- potencia(10, 3)
> print(res)
[1] 1000
```

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

29

29

Funções Customizadas (UDF).

- Para verificar se um parâmetro foi passado, usa-se `missing()`

```
potencia <- function(x, y=2) {
  if (missing(x)) {
    x <- 50
  }
  return (x^y)
}

res <- potencia(10)
print(res)
res <- potencia(10, 3)
print(res)
res <- potencia()
print(res)
```

```
> potencia <- function(x, y=2) {
+   if (missing(x)) {
+     x <- 50
+   }
+   return (x^y)
+ }
>
> res <- potencia(10)
> print(res)
[1] 100
> res <- potencia(10, 3)
> print(res)
[1] 1000
> res <- potencia()
> print(res)
[1] 2500
```

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

30

30



Exercícios.

- Crie um script R (chamado funcoes.R) e escreva nele as seguintes funções:
 - Para calcular o quadrado de um número
 - Para receber duas matrizes e retornar a multiplicação
 - Para receber um data frame contendo uma coluna nome e uma coluna idade, e retornar a média das idades
 - Para receber um data frame contendo uma coluna nome e uma coluna idade, e retornar uma lista com dois dados, o nome e a idade, da pessoa que contém a maior idade
- Carregue o script no R e execute as funções