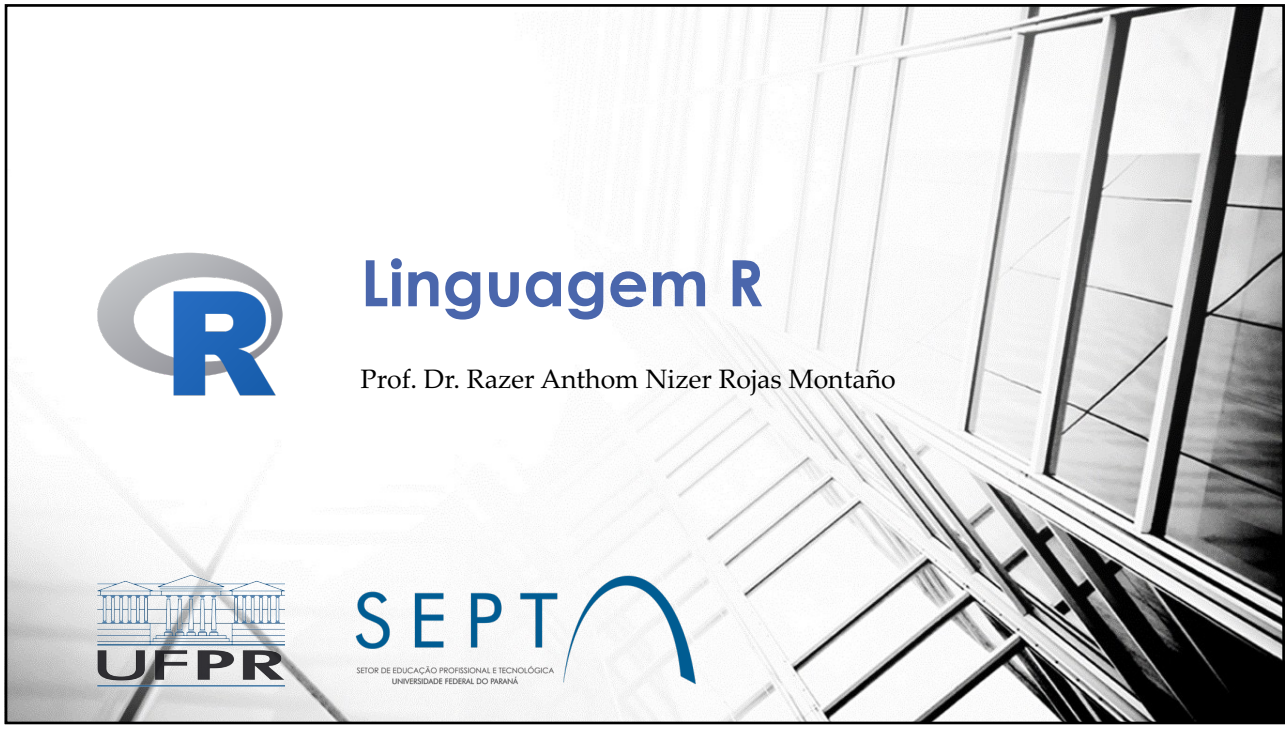
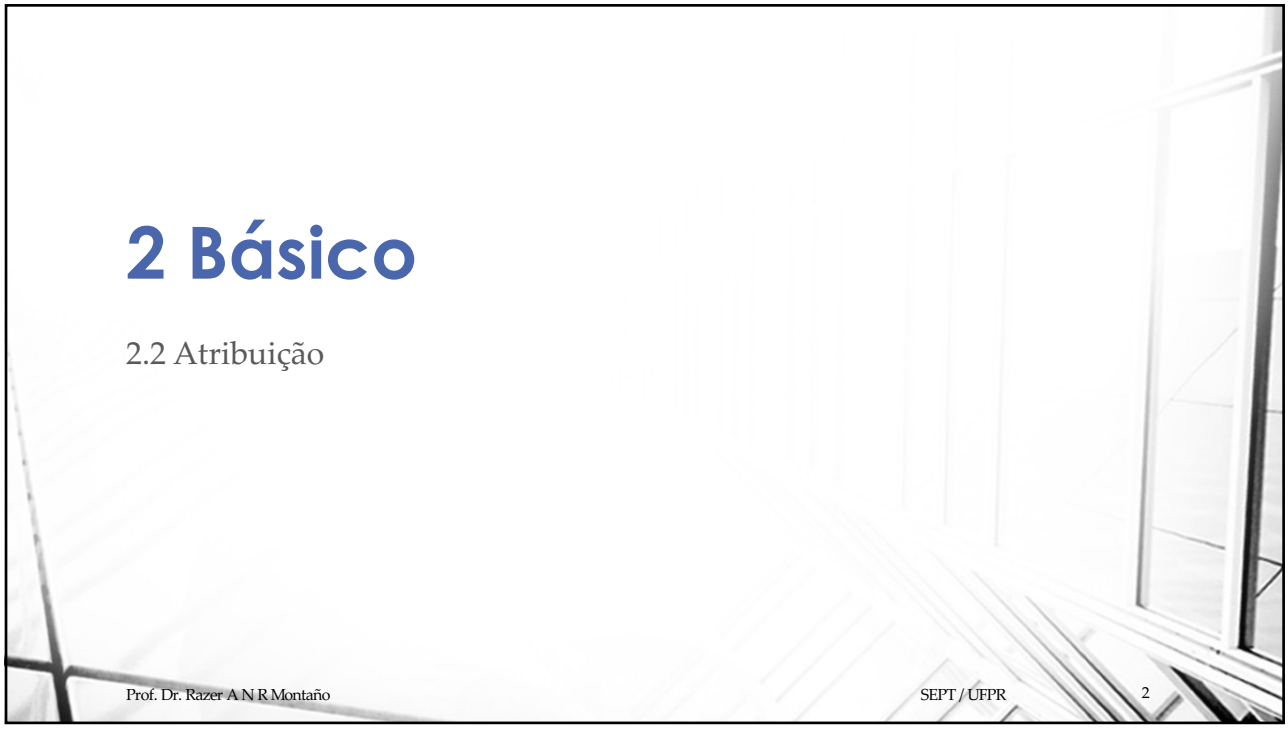

Linguagem R
Prof. Dr. Razer Anthom Nizer Rojas Montaño
UFPR
SEPT
SETOR DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
UNIVERSIDADE FEDERAL DO PARANÁ

1


2 Básico
2.2 Atribuição
Prof. Dr. Razer A N R Montaño
SEPT / UFPR
2

2

Atribuição, Objetos e Tipos

- Três operadores =, <- e <<-
- A diferença é o escopo, = e <- possuem escopo local e <<- escopo global

```
> a <- 10
> a
[1] 10
```

- Para atribuição de variáveis, recomenda-se <-, use = para setar parâmetros de funções

Operador	Escopo	Objetivo
=	Local	Atribuição de valores a objetos Passagem de parâmetros nomeados
<-	Local	Atribuição de valores a objetos
<<-	Global	Atribuição de valores a objetos

3

Atribuição, Objetos e Tipos

- R é *case-sensitive*

```
> a <- 10
> A
Erro: objeto 'A' não encontrado
```

- A função `mean()`, tira a média de um vetor
  - `1:10` gera um vetor de 1 até 10
  - Assim, `mean(1:10)` tira a média dos números entre 1 e 10

```
> mean(1:10)
[1] 5.5
> Mean(1:10)
Error in Mean(1:10) : não foi possível encontrar a função "Mean"
```


4

Atribuição, Objetos e Tipos

- Usando = para nomear um parâmetro de função

```
> mean(x=1:10)
[1] 5.5
> x
Erro: objeto 'x' não encontrado
```

- Neste caso o = está sendo usado para passar um parâmetro nomeado para a função `mean()`
- Usando <- e atribuindo dentro da função

```
> mean(x<-1:10)
[1] 5.5
> x
 [1]  1  2  3  4  5  6  7  8  9 10
```

- Neste caso, além de passar o parâmetro, o novo símbolo `x` foi criado

5

Atribuição, Objetos e Tipos

- Tudo em R é um Objeto

```
> a <- 1
```

- Neste caso, `a` é um objeto
- Possui 5 classes básicas ou atômicas
  - `character` : usada para representar strings : `"R é legal"`
  - `numeric` : representa valores decimais : `10.5`
  - `integer` : representa valores inteiros : `8`
  - `complex` : representa números complexos: `1 + 2i`
  - `logical` : representa valores lógicos: `TRUE` e `FALSE`

6

Atribuição, Objetos e Tipos

- O tipo é dado por:
  - `class()` : é a classe a qual o objeto faz parte, é um atributo dos objetos indicando sua classe (como se fosse uma classe da O.O.)
  - `typeof()` : determina seu tipo básico, interno
- Exemplo

```
> x <- 10
> class(x)
[1] "numeric"
> typeof(x)
[1] "double"
```


7

Atribuição, Objetos e Tipos

- Para vetores, a classe é igual ao tipo básico
- Exemplo

```
> x <- 1:10
> class(x)
[1] "integer"
> typeof(x)
[1] "integer"
```


8

Atribuição, Objetos e Tipos

- Para objetos complexos `class` denota uma classe, como da O.O.
- Exemplo

```
> x <- 1:10
> y <- x/5 + rnorm(10)
> modelo <- lm(y ~ x)
> class(modelo)
[1] "lm"
> typeof(modelo)
[1] "list"
```

- O objeto modelo é da classe `"lm"`, sob o ponto de vista O.O.
- Mas é implementado internamente como um `"list"`, sob o ponto de vista de implementação no R

9

Atribuição, Objetos e Tipos

- R possui 5 classes básicas ou atômicas
  - `character` : usada para representar strings : `"R é legal"`
  - `numeric` : representa valores decimais : `10.5`
  - `integer` : representa valores inteiros : `8`
  - `complex` : representa números complexos: `1 + 2i`
  - `logical` : representa valores lógicos: `TRUE` e `FALSE`

10

Atribuição, Objetos e Tipos

- Executar

```
> a <- "R é legal"
> a
[1] "R é legal"
> b <- 10.5
> b
[1] 10.5
> c <- 8
> c
[1] 8
> d <- 1 + 2i
> d
[1] 1+2i
> e <- b>c
> e
[1] TRUE
```


11

Atribuição, Objetos e Tipos

- Executar

```
> class(a)
[1] "character"
> class(b)
[1] "numeric"
> class(c)
[1] "numeric"
> class(d)
[1] "complex"
> class(e)
[1] "logical"
```


12

Atribuição, Objetos e Tipos

- Mesmo atribuindo 8 ao objeto c , ele continua numeric
- Execute

```
> class(c)
[1] "numeric"
> typeof(c)
[1] "double"
```

- A função `typeof()` indica qual o tipo, mais básico, sob o ponto de vista de implementação no R

13

Atribuição, Objetos e Tipos.

- Para criar um inteiro, usa-se `as.integer()` ou o sufixo `L`
- Execute

```
> f <- 10L
> class(f)
[1] "integer"
> typeof(f)
[1] "integer"
> g <- as.integer(20)
> class(g)
[1] "integer"
> typeof(g)
[1] "integer"
```


14

Coerção de Objetos

- Conversão de tipos
- Funções

```
as.character()
as.complex()
as.numeric()
as.integer()
as.logical()
```


15

Coerção de Objetos.

- Exemplo

```
> x <- 10
> class(x)
[1] "integer"
> y <- as.character(x)
> class(y)
[1] "character"
```


16

Verificação de Tipo

- Funções is.xxx:
  - `is.logical()` : verifica se um objeto é do tipo lógico
  - `is.integer()` : verifica se um objeto é do tipo inteiro
  - `is.double()` : verifica se um objeto é do tipo double
  - `is.complex()` : verifica se um objeto é do tipo complexo
  - `is.character()` : verifica se o objeto é uma String

17

Verificação de Tipo

- Exemplos: `is.logical()`

```
> is.logical(T)
[1] TRUE
> is.logical(TRUE)
[1] TRUE
> is.logical(10)
[1] FALSE
```


18

Verificação de Tipo

- Exemplos: `is.integer()`

```
> is.integer(TRUE)
[1] FALSE
> is.integer(10)
[1] FALSE
> is.integer(10L)
[1] TRUE
```


19

Verificação de Tipo

- Exemplos: `is.double()`

```
> is.double(10)
[1] TRUE
> is.double(10L)
[1] FALSE
> is.double(10+2i)
[1] FALSE
```


20

Verificação de Tipo

- Exemplos: `is.complex()`

```
> is.complex(10)
[1] FALSE
> is.complex(10L)
[1] FALSE
> is.complex(10+2i)
[1] TRUE
```


21

Verificação de Tipo.

- Exemplos: `is.character()`

```
> is.character(10)
[1] FALSE
> is.character(10+2i)
[1] FALSE
> is.character("oi mundo")
[1] TRUE
```


22

Valores Especiais

- Para representar dados faltantes, indefinições, infinito ou `null`:
  - **NA** (*Not Available*) significa dado faltante/indisponível.
    - É da classe **logical**.
    - É o que deve ser representado quando um dado está faltando
    - Verifica-se com **`is.na(x)`**
  - **NaN** (*Not a Number*) representa indefinições matemáticas.
    - É da classe **numeric**.
    - Um NaN é um NA, mas a recíproca não é verdadeira.
    - Por exemplo, `0/0` e `log(-1)`
    - Verifica-se com **`is.nan(x)`**
  - **Inf** / **-Inf** (*Infinito*) é um número muito grande ou o limite matemático
    - É da classe **numeric**.
    - Por exemplo, `1/0` e `10^310`.
    - Verifica-se com **`is.infinite(x)`**
  - **NULL** (*Nulo*) representa a ausência de informação.
    - É um objeto da classe **NULL**
    - NULL não é o mesmo que NA
    - Usado como valor retornado por funções e expressões com valores indefinidos
    - Verifica-se com **`is.null(x)`**

Valores Especiais.

- Exemplos:

```
> is.na(10)
[1] FALSE
> is.na(NA)
[1] TRUE
> is.nan(0/0)
[1] TRUE
> is.infinite(1/0)
[1] TRUE
> is.null(10)
[1] FALSE
> is.null(NULL)
[1] TRUE
```