

Introdução ao R  
2. Noções básicas  
1/12

Fúlvio Nedel  
SPB/UFSC

Noções básicas  
Workspace  
Vetor  
Função  
Exercício

# Introdução ao uso do R em Ciências da Saúde

## 2. Noções básicas para o trabalho em R



Fúlvio Borges Nedel  
Departamento de Saúde Pública – SPB  
Centro de Ciências da Saúde – CCS  
Universidade Federal de Santa Catarina – UFSC

*Grups de Recerca d'Amèrica i Àfrica Llatines – GRAAL*  
<http://graal.uab.cat>

17 de outubro de 2017

## Introdução ao R

### 2. Noções básicas

2/12

Fábio Nedel

SPB/UFSC

#### Noções básicas

Workspace

Vetor

Função

Exercício

### 1 Noções básicas

- Workspace
- Vetor
- Função

### 2 Exercício

## Espaço de trabalho

- ★ O R trabalha com **objetos** em **espaços de trabalho**
- ★ Os objetos podem ser de diferentes **classes**
- ★ O *workspace* contém ainda os **pacotes**, chamados automaticamente na inicialização do programa ou por demanda do usuário
- ★ Um pacote contém as **funções**, comandos para a execução das ações desejadas
- ★ O R trabalha com os objetos e funções **em memória – não sobre o arquivo de dados mas uma cópia**, que permanece ativa durante a sessão de trabalho

## Workspace

- ★ Permite, entre outras coisas, trabalhar com diferentes bancos de dados ao mesmo tempo
- ★ As ações – modificação e criação de objetos – realizadas numa *sessão de trabalho* são descartadas ao encerrar a sessão, a menos que se grave um arquivo com os elementos desejados do espaço de trabalho.
- ★ É recomendável que o diretório de trabalho seja o mesmo com os arquivos de dados e outros relacionados à análise

# Noções básicas

## ■ O sinal do console:

- “>” indica a primeira linha do comando
- “+” indica sua continuação, quando houver

## ■ Pacote / biblioteca (library)

base, foreign, Hmisc, epicalc, Epi, ...

## ■ Objeto

- É a forma com que são armazenados em memória os elementos (criados ou carregados) na sessão de trabalho
- Banco de dados, vetor, matriz, tabela, função, resultado da função, fórmula, ...
- São criados pelo sinal de destinação (“<-”)

> x <- 3

> x

[1] 3

- Vários objetos podem ser salvos em um único arquivo

## Introdução ao R

### 2. Noções básicas

6/12

Fáulio Nedel  
SPB/UFSC

Noções básicas

Workspace

Vetor

Função

Exercício

# Noções básicas

## ■ Vetor

- Sequência de valores de um mesmo tipo de dados, em uma ordem específica
- É o objeto mais simples do R
- Podem ser **numéricos** (e sofrer operações matemáticas) ou **alfanuméricos** (e ser combinados)

```
> x <- 3:7      > z <- x + y
> x              > z
[1] 3 4 5 6 7    [1] 10 10 10 10 10
> y <- 7:3      > w <- z + sqrt(4)
> y              > w
[1] 7 6 5 4 3    [1] 12 12 12 12 12
```

```
> x <- 'banana'
> y <- 'laranja'
> z <- x + y
Erro em x + y :
  argumento não-numérico
  para operador binário
> z <- c(x,y)
> z
[1] "banana"   "laranja"
```

## Introdução ao R

### 2. Noções básicas

7/12

Fábio Nedel  
SPB/UFSC

Noções básicas

Workspace

Vetor

Função

Exercício

# Noções básicas

## ■ Vetor

### Seleção de valores específicos de um objeto

```
> animais <- c('cachorro', 'galinha', 'cachoro', 'gato',
  'esquilo', 'tagtaguga', 'gralha', 'sangue-de-boi',
  'cachorro')

> animais
[1] "cachorro"      "galinha"       "cachoro"
[4] "gato"          "esquilo"        "tagtaguga"
[7] "gralha"         "sangue-de-boi"   "cachorro"

> animais[c(3,6)] <- c('cachorro', 'tartaruga') # arrumo o 3º
  e o 6º registro

> animais[3:6] # observo os valores do 3º ao 6º registro
[1] "cachorro" "gato"    "esquilo"  "tartaruga"
```



# Noções básicas

## ■ Função `função(x, ...)`

- `x = objeto`
  - `... demais argumentos`
- É um comando direcionado a um *objeto* qualquer do espaço de trabalho (*workspace*)
  - Tem *argumentos* com as opções de execução
    - Podem ser obrigatórios ou não
    - O nome do objeto geralmente é obrigatório
  - Eventualmente a função pode não exigir a definição de nenhum argumento, como em
    - `ls()` # lista os objetos do espaço de trabalho
    - `help()` # abre a ajuda sobre a função '`help()`'
    - `par()` # apresenta os parâmetros ativos de construção de gráficos

# Noções básicas

- **Função** `read.table()`, `help()`, `table()`, ...

```
> library(Hmisc) # carrega o pacote 'Hmisc'  
[...] Attaching package: 'Hmisc' [...]
```

- “**Case-sensitive**”, diferencia maiúsculas e minúsculas

```
> library(hmisc)  
Erro em library(hmisc) : there is no package called 'hmisc'  
  
> which(animais=='esquilo')  
[1] 5  
  
> animais[5]  
[1] "esquilo"  
  
> which(animais=='Esquilo')  
integer(0)
```

# Noções básicas

## ■ Funções úteis:

- ◆ `c(a, b, ...)`:
- ◆ combina os elementos em uma linha
  - ◆ usada anteriormente para combinar em um novo vetor os objetos `x` ("banana") e `y` ("laranja"):  
`z <- c(x,y)`
  - ◆ e para criar um vetor ("animais") com o nome de alguns animais
  - ◆ `animais <- c('cachorro', 'galinha', 'cachoro', 'gato', 'esquilo', 'tagtaguga', 'gralha', 'sangue-de-boi', 'cachorro')`

# Noções básicas

## ■ Funções úteis:

- **cbind(a, b, ...):**

- Combina os elementos em coluna, formando uma matriz.

- **rbind(a, b, ...):**

- Combina os elementos em linha formando uma matriz

```
> x <- 3:7  
> y <- 7:3  
> z <- x + y
```

```
> rbind(x, y, z)  
 [,1] [,2] [,3] [,4] [,5]  
 x     3     4     5     6     7  
 y     7     6     5     4     3  
 z    10    10    10    10    10
```

```
> cbind(x, y, z)  
      x y z  
 [1,] 3 7 10  
 [2,] 4 6 10  
 [3,] 5 5 10  
 [4,] 6 4 10  
 [5,] 7 3 10
```

# Noções básicas

## ■ Funções úteis:

- t(): transposição da matriz

```
> rbind(x, y, z)
 [,1] [,2] [,3] [,4] [,5]
 x     3     4     5     6     7
 y     7     6     5     4     3
 z    10    10    10    10    10
```

```
> t(cbind(x, y, z))
 [,1] [,2] [,3] [,4] [,5]
 x     3     4     5     6     7
 y     7     6     5     4     3
 z    10    10    10    10    10
```

```
> x <- 3:7
> y <- 7:3
> z <- x + y
```

```
> cbind(x, y, z)
      x  y  z
[1,] 3 7 10
[2,] 4 6 10
[3,] 5 5 10
[4,] 6 4 10
[5,] 7 3 10
```

```
> t(rbind(x, y, z))
      x  y  z
[1,] 3 7 10
[2,] 4 6 10
[3,] 5 5 10
[4,] 6 4 10
[5,] 7 3 10
```

# Noções básicas: funções úteis

Introdução ao R  
2. Noções básicas  
8/12

Fáulio Nedel  
SPB/UFSC

Noções básicas  
Workspace  
Vetor  
**Função**

Exercício

- `data.frame()`: cria uma base de dados
- `str()`: apresenta a estrutura do objeto
  - veja também a função `class()` (digite `?class`)

```
(banco <- data.frame(cbind(x,y,z)))
```

	x	y	z
1	3	7	10
2	4	6	10
3	5	5	10
4	6	4	10
5	7	3	10

# Noções básicas: funções úteis

Introdução ao R  
2. Noções básicas  
8/12

Fáulio Nedel  
SPB/UFSC

Noções básicas  
Workspace  
Vetor  
**Função**  
Exercício

- `data.frame()`: cria uma base de dados
- `str()`: apresenta a estrutura do objeto
  - veja também a função `class()` (digite `?class`)

```
(banco <- data.frame(cbind(x,y,z)))
```

	x	y	z
1	3	7	10
2	4	6	10
3	5	5	10
4	6	4	10
5	7	3	10

Veja uma forma abreviada da  
função `print()`: basta colocar  
o comando entre parênteses.

# Noções básicas: funções úteis

Introdução ao R  
2. Noções básicas  
8/12

Fáulio Nedel  
SPB/UFSC

Noções básicas  
Workspace  
Vetor  
**Função**  
Exercício

- `data.frame()`: cria uma base de dados
- `str()`: apresenta a estrutura do objeto
  - veja também a função `class()` (digite `?class`)

```
(banco <- data.frame(cbind(x,y,z)))
```

	x	y	z
1	3	7	10
2	4	6	10
3	5	5	10
4	6	4	10
5	7	3	10

Veja uma forma abreviada da  
função `print()`: basta colocar  
o comando entre parênteses.

# Noções básicas: funções úteis

Introdução ao R  
2. Noções básicas  
8/12

Fáulio Nedel  
SPB/UFSC

Noções básicas  
Workspace  
Vetor  
**Função**  
Exercício

- `data.frame()`: cria uma base de dados
- `str()`: apresenta a estrutura do objeto
  - veja também a função `class()` (digite `?class`)

```
(banco <- data.frame(cbind(x,y,z)))  
  
  x y z      str(banco)  
1 3 7 10    'data.frame': 5 obs. of  3 variables:  
2 4 6 10    $ x: int  3 4 5 6 7  
3 5 5 10    $ y: int  7 6 5 4 3  
4 6 4 10    $ z: int  10 10 10 10 10  
5 7 3 10  
  
Note o modo de chamar a variável:  
banco$y  <nome do data frame>$<nome da variável>  
[1] 7 6 5 4 3
```

- `data.frame()`: cria uma base de dados
- `str()`: apresenta a estrutura do objeto
  - veja também a função `class()` (digite `?class`)

```
(banco <- data.frame(cbind(x,y,z)))
```

	x	y	z
1	3	7	10
2	4	6	10
3	5	5	10
4	6	4	10
5	7	3	10

```
str(banco)
'data.frame': 5 obs. of 3 variables:
 $ x: int 3 4 5 6 7
 $ y: int 7 6 5 4 3
 $ z: int 10 10 10 10 10
```

Note o modo de chamar a variável:

```
banco$y <nome do data frame>$<nome da variável>
[1] 7 6 5 4 3
```

**Selecionar o 4º registro da variável 'y':**

```
banco[4,2]
[1] 4
```

```
banco[4,"y"]
[1] 4
```

```
banco$y[4]
[1] 4
```

- `data.frame()`: cria uma base de dados
- `str()`: apresenta a estrutura do objeto
  - veja também a função `class()` (digite `?class`)

```
(banco <- data.frame(cbind(x,y,z)))
```

	x	y	z
1	3	7	10
2	4	6	10
3	5	5	10
4	6	4	10
5	7	3	10

```
str(banco)
'data.frame': 5 obs. of 3 variables:
 $ x: int 3 4 5 6 7
 $ y: int 7 6 5 4 3
 $ z: int 10 10 10 10 10
```

Note o modo de chamar a variável:

```
banco$y <nome do data frame>$<nome da variável>
[1] 7 6 5 4 3
```

**Selecionar o 4º registro da variável 'y':**

```
banco[4,2]
[1] 4
```

```
banco[4,"y"]
[1] 4
```

```
banco$y[4]
[1] 4
```

# Noções básicas: Funções úteis

Introdução ao R  
2. Noções básicas  
9/12

Fáulio Nedel  
SPB/UFSC

Noções básicas  
Workspace  
Vetor  
Função

Exercício

**Calcular o desvio-padrão:**  $\sigma = \sqrt{\frac{\sum(x-\mu)^2}{N}}$ ;  $s = \sqrt{\frac{\sum(x-\bar{x})^2}{n-1}}$

```
head(obs <- round(rnorm(n=30, mean=20, sd=5)))
[1] 17 20 20 18 26 21
(m <- mean(obs))
[1] 20.03333
head(obs - m, 4)
[1] -3.03333333 -0.03333333 -0.03333333 -2.03333333
head((obs-m)^2, 4)
[1] 9.20111111 0.00111111 0.00111111 4.134444444
sum((obs-m)^2)
[1] 674.9667
(n <- length(obs))
[1] 30
```

$\sigma$

```
sqrt(sum((obs-m)^2)/n)
[1] 4.743299
```

$s$

```
sqrt(sum((obs-m)^2)/(n-1))
[1] 4.824387
```

A função `sd`

```
sd(obs)
[1] 4.824387
```

## Introdução ao R 2. Noções básicas 10/12

Fáulio Nedel  
SPB/UFSC

Noções básicas  
Workspace  
Vetor  
**Função**  
Exercício

O usuário pode criar funções para automatizar cálculos e outras ações, segundo sua conveniência

```
dppop <- function(x){  
  # Função para o cálculo do desvio-padrão de uma população,  
  # tomando todos seus elementos ( $N$ , ao invés de  $N-1$ ).  
  x <- na.omit(x)  
  m <- mean(x)  
  sqrt(sum((x-m)^2)/length(x))  
}  
dppop(obs)  
[1] 4.743299
```

Salvar a função em arquivo

```
save(dppop, file="dppop.r")
```

Para ativar a função, em futuras sessões de trabalho neste diretório, digite:

```
load('dppop.r')
```

## Noções básicas

### ■ Funções úteis:

```
> demo()  
?demo  
> demo(graphics)  
> demo(lm.glm)
```

## Introdução ao R 2. Noções básicas 12/12

Fáulio Nedel  
SPB/UFSC

Noções básicas  
Workspace  
Vetor  
Função  
  
Exercício

### 1 Noções básicas

- Workspace
- Vetor
- Função

### 2 Exercício

# Exercício - entregar no Moodle

No R e supondo uma população infinita com distribuição normal de peso (média=70 Kg, DP=20 Kg) e altura (média=1,7 m, DP=25 cm),

- 1)Crie um vetor com 10 valores aleatórios\* com um decimal, para representar uma medida do peso;
- 2)Crie um vetor com 10 valores aleatórios\* com duas casas decimais, para representar uma medida da altura;
- 3)Crie um vetor com o resultado do cálculo do Índice de Massa Corporal (IMC, Kg/m<sup>2</sup>);
- 4)Combine os vetores criados em um banco de dados (*data frame*), com cada vetor em uma coluna e cada registro em uma linha;
- 5)Se ainda não o fez, nomeie as variáveis: peso, altura, imc;
- 6)Salve a sintaxe num arquivo “.R” e faça seu ‘upload’ na página do curso.

# Exercício: funções utilizadas

- \* Para gerar os números aleatórios, utilize a função `rnorm()` com os parâmetros definidos de peso e altura. Para ajuda, digite `?rnorm` no console.

O arredondamento pode ser feito de forma automática com a função `round()`.

Use também a função `set.seed()`, para permitir a reprodutibilidade dos resultados. `set.seed()` deve ser usada antes de `rnorm()`.

**`set.seed(<seu nº de ordem na lista de inscrição no curso>)`**

- \* As funções podem ser combinadas, como em

```
x <- data.frame(cbind(a,b,c,...))
```

# Exercício - exemplo

Por exemplo, com a “semente” (v. ?set.seed) 47, o banco de dados resultado do exercício é o seguinte:

	x	peso	altura	imc
1	109,9	1,47	50,85844	
2	84,2	1,71	28,79518	
3	73,7	1,82	22,24973	
4	64,4	1,24	41,88345	
5	72,2	1,72	24,40508	
6	48,3	1,87	13,81223	
7	50,3	1,68	17,82171	
8	70,3	2,02	17,22870	
9	65,0	1,52	28,13366	
10	40,7	1,69	14,25020	