Estatística com Apoio Computacional

Introdução

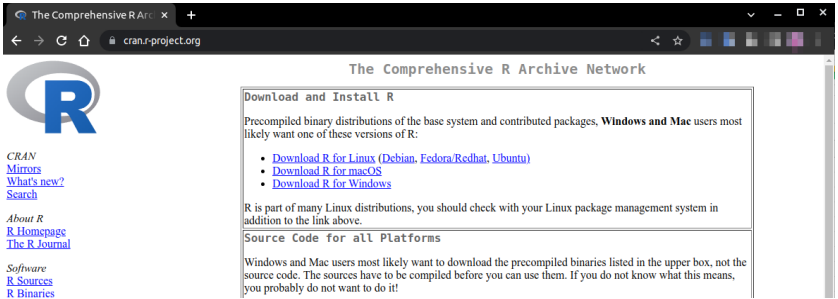
Universidade Estadual Vale do Acaraú – UVA

Paulo Regis Menezes Sousa

paulo_regis@uvanet.br

- R é um ambiente de software livre para computação estatística e gráficos.
- O ambiente possui uma linguagem, um ambiente de tempo de execução com gráficos, um depurador, acesso a funções do sistema e a capacidade de executar programas armazenados em arquivos de script.
- Existe uma enorme quantidade de procedimentos estatísticos em milhares de pacotes livremente disponíveis na internet e que podem ser carregados opcionalmente.

- CRAN (*Comprehensive R Archive Network*) é o repositório oficial do R.
- Seu site é http://cran.r-project.org/.



- Comentários: `# seu comentario aqui...`
- Operações aritméticas básicas: +, -, *, /
- Potenciação: `^` ou `**` (x^2 = `x^2` ou `x**2`)
- Raiz quadrada: `sqrt(x)` ou `x^(1/2)` ($\sqrt{x}$ = `sqrt(x)`)
- Raiz qualquer: `x^(1/n)` ($\sqrt[n]{x}$)
- Resto da divisão: %%
- Parte inteira da divisão: %/%

Input

```
> 2+3
> 6-3
> 4*3
> 5/2
> 2**3
> 2^3
> 25**(1/2)
> 27^(1/3)
> # comentário
> 5 %% 2
> 5 %/% 2
```

Output

```
[1] 5
[1] 3
[1] 12
[1] 2.5
[1] 8
[1] 8
[1] 5
[1] 3

[1] 1
[1] 2
```

Exercício 1

Qual o resultado da expressão abaixo?

$$2^{(4+8)} - 3^{10} \times \frac{\text{sen } (15)}{\sqrt{3}} + \lfloor 5/2 \rfloor = ?$$

- Operadores de comparação e lógicos

Operador	Descrição
==	Igual
!=	Diferente
>	Maior
<	Menor
>=	Maior ou igual
<=	Menor ou igual
&	E lógico
|	OU lógico

Input

```
> 5 == 7
> 4 != 4
> (5 == 7) & (4 != 4)
> (5 == 7) | (4 != 4)
> 7 > 9
> 5 < 10
> 45 >= 45
> 46 >= 45
> 6 <= 6
> 6 <= 7
```

Output

```
[1] FALSE
[1] TRUE
[1] FALSE
[1] TRUE
[1] FALSE
[1] TRUE
[1] TRUE
[1] FALSE
[1] TRUE
[1] FALSE
```

- A linguagem R possui tipagem dinâmica, então os tipos são associados aos objetos no momento da atribuição.
- Os principais tipos de objetos em R são:
  - `numeric`: números inteiros ou reais, como idade, renda, número de filhos.
  - `character`: texto ou caracteres.
  - `logical`: verdadeiro ou falso (TRUE/FALSE).
  - `complex`: números complexos.

- Atribuições de valores para objetos podem ser feitas por qualquer um destes três operadores
  1. Operador igual =
  2. Operador seta para esquerda <-
  3. Operador seta para direita ->

```
> x = 3
> y <- 5
> 7 -> idade
> primeiro_nome = 'Lucy'
> tipo2 <- "Risco Médio"
> num.de.cores <- 256
```

Exercício 2

Qual o resultado da expressão abaixo, para os valores de $x = 3$, $y = 9$, $z = 11$ e $k = 3$? Crie e atribua os valores às respectivas variáveis.

$$2^{(4+x)} - 3^k \times \frac{\text{sen } (y)}{\sqrt{3}} + \lfloor z/2 \rfloor = ?$$

- No ambiente de execução do R os dados podem ser organizados em diversos tipos de estruturas diferentes.
- As principais estruturas de dados usadas no R são:
  - `vector`: vetores com 1 ou mais elementos.
  - `matrix`: matrizes.
  - `array`: generalização de vetores para n dimensões.
  - `factor`: vetor representando dados categóricos.
  - `data.frame`: representação em memória de uma tabela de dados.
  - `list`: combina diferentes tipos no mesmo objeto.

- Criação de vetores: `vetor <- c(1,3,5,7,9)` (a função **c** cria um vetor com os valores especificados).
  - Recuperar um elemento `vetor[i]`
  - Forma simplificada de criação de uma sequência
    `vetor <- 1:10` (vetor1 será uma sequencia de números de 1 a 10).

Input

```
1 notas = c(9.8, 7.3, 8.0)
```

Output

```
1 [1] 9.8, 7.3, 8.0
```

Input

```
> v <- c(3,6,9,12,15)
> v[1]
> v[4]
> v[c(1,3,5)]
> v[2:4]
> 3:7
```

Output

```

[1] 3
[1] 12
[1]  3  9 15
[1]  6  9 12
[1] 3 4 5 6 7
```

- Criação de matrizes:

```
M <- matrix(c(2,4,6,8,10,12), nrow=2, ncol=3)
```

$$\begin{pmatrix} 2 & 6 & 10 \\ 4 & 8 & 12 \end{pmatrix}$$

  - Recuperar um elemento `M[i,j]`
  - Recuperar uma linha `M[i,]`
  - Recuperar uma coluna `M[,j]`

Input

```
1 > V = c(2,4,6,8,10,12)
2 > M = matrix(V, 2, 3)
3 > M
4 >
5 >
```

Output

```
1
2
3 [,1] [,2] [,3]
4 [1,]    2    6   10
5 [2,]    4    8   12
```

Input

```
> M



> M[1]
> M[1:4]
```

Output

```
     [,1] [,2] [,3]
[1,]    2    6   10
[2,]    4    8   12


[1] 2
[1] 2 4 6 8
```

Input

```
> M



> M[1, 2]
> M[2, ]
```

Output

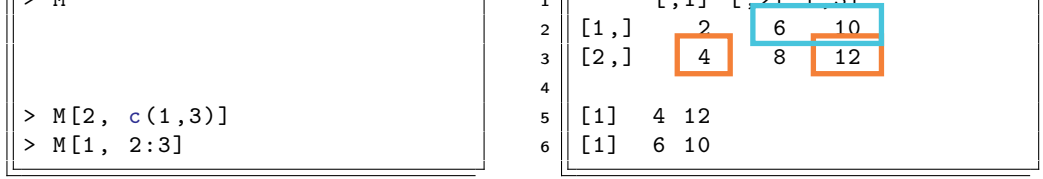
```
     [,1] [,2] [,3]
[1,]    2    6   10
[2,]    4    8   12

[1] 6
[1]  4  8 12
```

Input

```
> M



> M[2, c(1,3)]
> M[1, 2:3]
```

Output

```
     [,1] [,2] [,3]
[1,]    2    6   10
[2,]    4    8   12

[1]  4 12
[1]  6 10
```



Input

```
> M



> M[ , 2:3]
>
>
```

Output

```
     [,1] [,2] [,3]
[1,]    2    6   10
[2,]    4    8   12

[,1] [,2]
[1,]    6   10
[2,]    8   12
```

- Um script para o **R** é um arquivo que contém uma sequência de comandos. Cada linha do arquivo corresponde a uma linha digitada no console.
- Para executar uma linha de um script podemos selecioná-la, ou apenas manter o cursor sobre a linha, e pressionar `Ctrl+ENTER`.
- Para executar um conjunto de linhas as selecionamos e pressionamos `Ctrl+ENTER`.
- O comando `Ctrl+ENTER` pode ser substituído pela opção **Run** da IDE utilizada.

- Exemplo de script

```
1 a <- 2
2 b <- 2
3 c <- 3
4
5 if(a != b & b != c & c!= a){
6   print("É um triangulo escaleno")
7 }else{
8   print("Não é um triangulo escaleno")
9 }
```

```r
# for
for(i in 1:5){
  print(i^2)
}

# while
i <- 1
while(i <= 5){
  print(i^2)
  i <- i+1
}
```

```r
# repeat
i <- 1
repeat{
  print(i^2)

  if(i == 5){
    break
  }
  i <- i+1
}
```

- Sintaxe para criação de funções

  `nome_fun <- function(arg1, agr2,..., argn){ rotina da função }`

```r
# Área de um triangulo
area <- function(b, h){
  A <- (b*h)/2
  return(A)
}

area(3,6)
```

- É praticamente impossível memorizar todos os comandos do R, então a ajuda pode ser muito útil quando se deseja saber qual comando utilizar e como.
- Quando você não sabe **qual comando** utilizar, procure-o com base em uma **palavra-chave** (em inglês) usando duas interrogações.

```
> ??sequence
```

O R irá abrir seu navegador de internet e mostrar todos os comandos, dentro dos pacotes instalados, que contenham a palavra "*sequence*" na descrição.

- Caso você saiba o nome da função e deseja saber como usá-la digite no *prompt* do R, o nome do comando desejado precedido de uma interrogação. Por exemplo:

```
>?seq
```

- Data frames são objetos usados para guardar tabelas de dados.
- Esta é a classe de objetos mais utilizada.
- É o formato usado pelo **R** para ler as tabelas importadas de outros softwares.

- Criação de data frames:

```
1 tabela <- data.frame(Nome = c("André","João"), Idade = c(25,45))
```

Nome	Idade
André	25
João	45

- Seleção de uma coluna:
- `tabela$Nome` O retorno é um vetor
- `tabela[1]` O retorno é um data-frame

Nome
André
João

- O comando básico para a criação gráfica é o **plot()**. A função `plot(dados)` gera um gráfico simples, atribuindo pontos em coordenadas cartesianas.

```
> a <- 1:20
> b <- a^2
> plot(a, b) # grafico de pontos
> plot(a, b, type="l") # grafico continuo
```

- Os principais argumentos para controlar títulos e rótulos são: `main` (título principal), `sub` (subtítulo), `xlab` (rótulos do eixo x) e `ylab` (rótulos do eixo y).

```
> plot(a, b, type="l", main="Título", xlab="n (1-20)", ylab="n^2")
```

- Existem várias formas de fazer referência a cores no R. Uma lista completa dos nomes de cores conhecidos pelo R pode ser obtida com a função `colors()`.
- Podemos também criar cores personalizadas usando a função `rgb()`, que recebe como argumentos as quantidades de vermelho (red), verde (green) e azul (blue) e, opcionalmente, o grau de opacidade (alpha). Os valores devem ser números reais entre 0 e 1. Também é possível especificar vetores e, nesse caso, acrescentar um argumento com os nomes das cores resultantes.

```
1 > goiaba <- rgb(0.94, 0.41, 0.40)
2 > goiaba.semitrans <- rgb(0.94, 0.41, 0.40, alpha = 0.5)
3 > vitamina <- rgb(red = c(0.87, 0.70), green = c(0.83, 0.77),
4 +        blue= c(0.71, 0.30), names= c("leite", "abacate"))
```

- Outra forma de especificar cores é utilizando o código HTML correspondente, ou seja, o símbolo # seguido dos valores vermelho, verde, azul e (opcionalmente) opacidade, em notação hexadecimal, de 00 a FF.

```
> uva <- "#AD4DA3"
> salada <- c(vitamina, uva, goiaba, goiaba.semitrans)
```

- Uma terceira forma de fazer referência a uma cor é pelo seu índice na palheta de cores do R, cujos valores podem ser obtidos chamando-se a função `palette()` sem nenhum argumento.

```
> palette()
[1] "black"   "red"     "green3"  "blue"    "cyan"
[6] "magenta" "yellow"  "gray"
```

- Através dos comandos **lines()** e **points()** é possível adicionar, após dado um comando de **plot()**, linhas e pontos, respectivamente, a um gráfico já existente.

```
1 > a <- 1:20
2 > b <- a^2
3 > plot(a,b) # grafico de pontos
4 > lines(rev(a),b) #adição de linhas
5 > points(a, 400-b) #adição de pontos
```

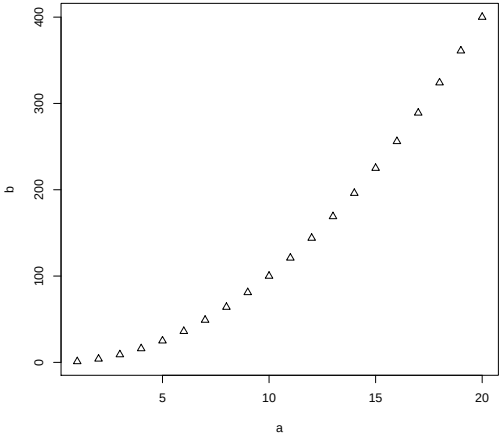
- Normalmente, pontos e textos são posicionados no centro das coordenadas fornecidas, mas, com o argumento `pos`, podemos definir que o texto ficará abaixo, à esquerda, acima ou à direita das coordenadas (cujos valores são, respectivamente, 1, 2, 3 e 4).

```
> text(5, 100, labels = c("esquerda", "direita"),
+      pos = c(2, 4), offset = 1, col = c("red", "blue"))
```

- O argumento `offset` define quantos caracteres o texto ficará distante das suas coordenadas, no caso de `pos` ter sido utilizado.

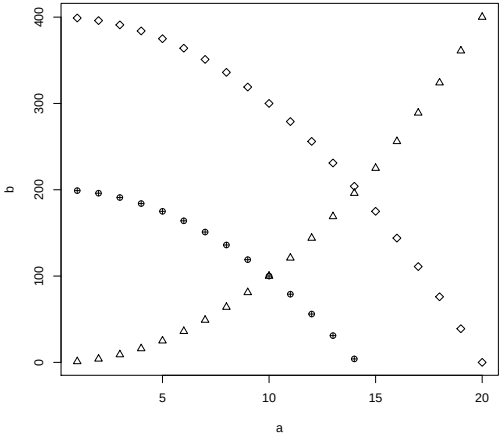
- O R permite que sejam feitas mudanças na representação dos indicadores gráficos (pontos) através do parâmetro **pch=** nos comandos **plot()** e **points()**.

```
1 > a <- 1:20
2 > b <- a^2
3 > plot(a,b,pch=2)
```



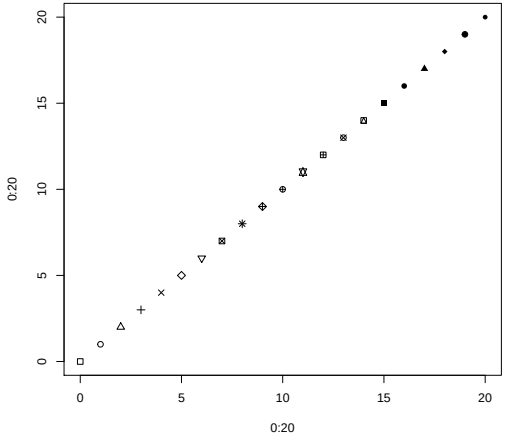
- O R permite que sejam feitas mudanças na representação dos indicadores gráficos (pontos) através do parâmetro **pch=** nos comandos **plot()** e **points()**.

```
1 > a <- 1:20
2 > b <- a^2
3 > plot(a,b,pch=2)
4 > points(a,400-b, pch=5)
5 > points(a,200-b, pch=10)
```



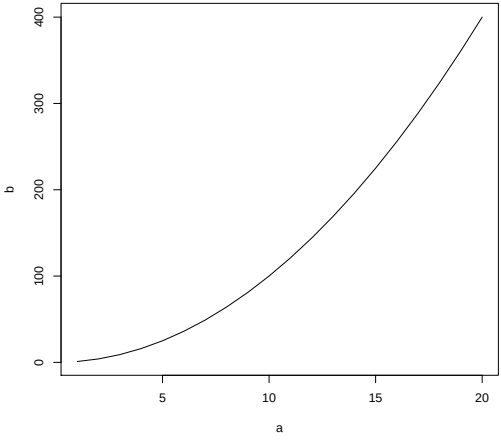
- O R permite que sejam feitas mudanças na representação dos indicadores gráficos (pontos) através do parâmetro **pch=** nos comandos **plot()** e **points()**.

```
1 plot(0:20,0:20, pch=0:20)
```



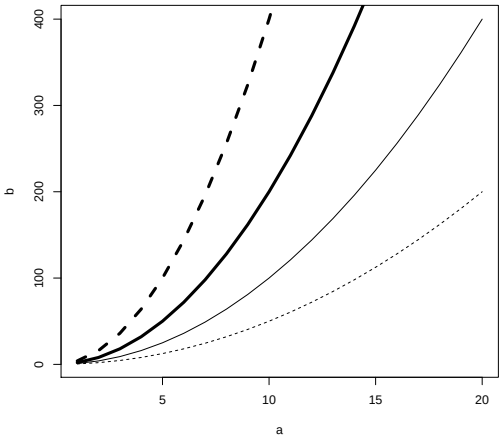
- Ainda, é possível realizar mudanças nas características das linhas. Para isso, basta utilizar os comandos **lwd=** e **lty=** que modificam, respectivamente, a largura e o estilo da linha.

```
1 > a <- 1:20
2 > b <- a^2
3 > plot(a,b,type="l")
```



- Ainda, é possível realizar mudanças nas características das linhas. Para isso, basta utilizar os comandos **lwd=** e **lty=** que modificam, respectivamente, a largura e o estilo da linha.

```
> a <- 1:20
> b <- a^2
> plot(a,b,type="l")
> lines(a,2*b,lwd=4)
> lines(a,0.5*b,lty=2)
> lines(a,4*b,lty=2,lwd=4)
```



- Para alterar a dimensão dos intervalos, pode-se primeiro plotar um gráfico em branco, ajustando os limites da abscissa e da ordenada e depois gerar o gráfico desejado.

```
> plot(c(-pi,pi),c(-1,1), type="n")
> x<-seq(-pi,pi,0.1)
> a <- sin(x)
> b <- sin(x-2/3*pi)
> c <- sin(x+2/3*pi)
> lines(x,a,col=2,lwd=1)
> lines(x,b,col=3,lwd=2)
> lines(x,c,col=4,lwd=3)
```