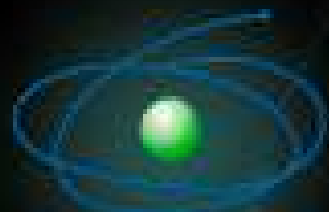




PPGI



CAPES
Recomendado



Introdução à Ciência de Dados utilizando a Linguagem R

Introdução à Ciência de Dados Utilizando a Linguagem R

Curso Preparado pelo Prof. José Carmino
Ministrado pelo
E-Mail:

Estrutura dos Slides

Entrada de Arquivos Externos

```
rm(list = ls())
setwd('C:\\users\\jose carmino\\Documents\\Rdados')
getwd()
dir()
carregar <- read.table("A1" " dec=".")
names(carregar); dim
carregar[1,3]; carregar
dados_vitimas <-
acidentes
read.table("http://w
=".")
```

No caso de haver um código associada ao slide, este terá o nome no canto inferior esquerdo

```
=";",dec=".")
<-
,header=T,sep=";",dec
```

Agenda

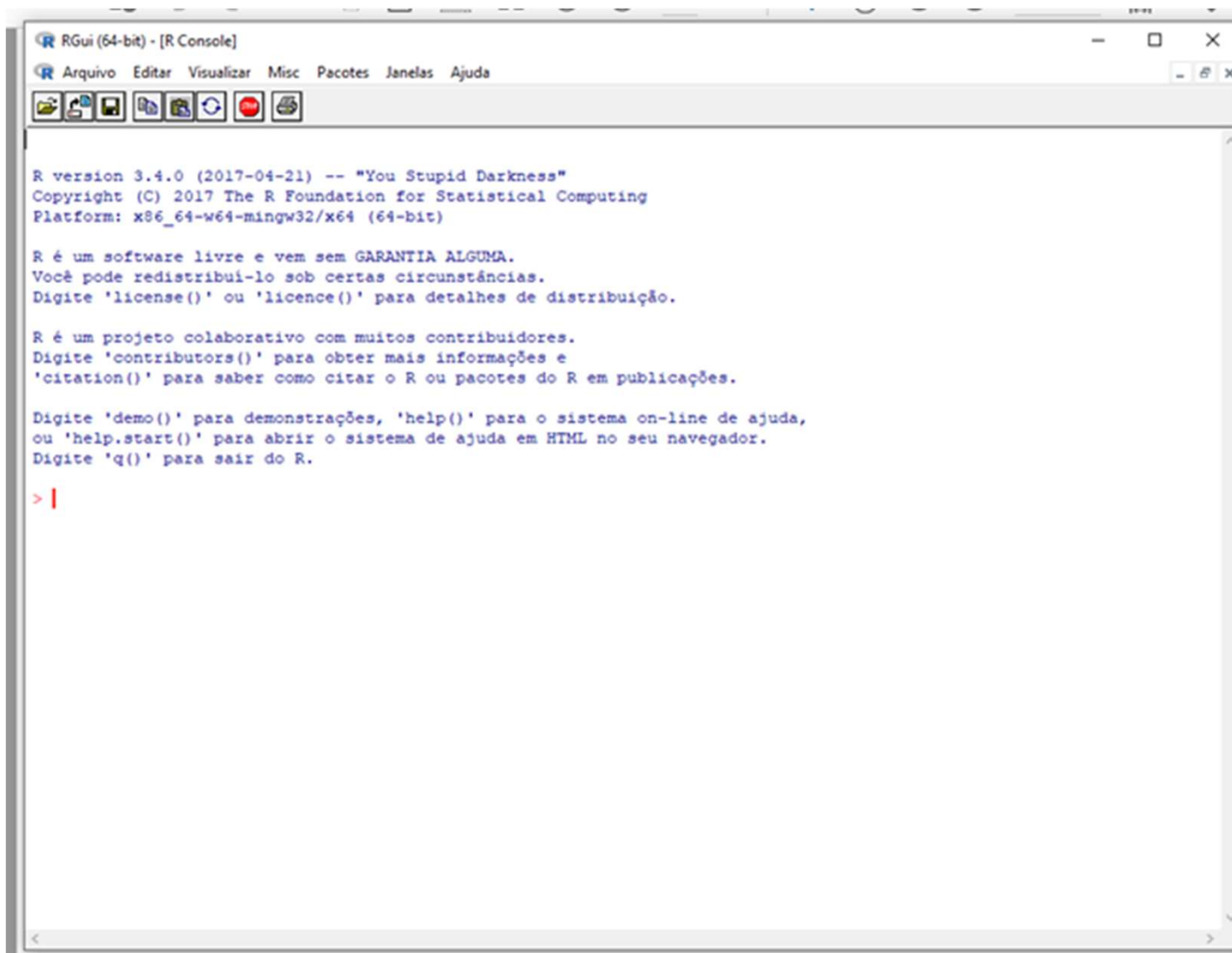
Tópico	Descrição
Apresentação do R	Criação interface, sintaxe, tipos de dados, comandos básicos, vetores e matrizes e entrada de arquivos
Programação em R	Estruturas de repetição, decisão e instruções iterativas
Gráficos	Gráficos de análise descritiva
Estatística descritiva	Medidas de posição e medidas de dispersão
Probabilidade	Definição e axiomas
Variáveis aleatórias	Variáveis aleatórias discretas e contínuas
Inferência estatística	Teste de hipótese de uma e duas amostras
Regressão e correlação linear simples	Determinando a equação linear, coeficiente de correlação e coeficiente de determinação

Apresentação do R

O R é resultado de um esforço colaborativo, sendo originalmente por Ross Ihaka e por Robert Gentleman na universidade de Auckland, Nova Zelândia.

O R é ao mesmo tempo uma linguagem de programação e um ambiente para computação estatística e gráfica, o que torna o R uma linguagem especializada em computação de dados.

Interfaces



The image shows a screenshot of the RGui (64-bit) - [R Console] window. The window has a menu bar with 'Arquivo', 'Editar', 'Visualizar', 'Misc', 'Pacotes', 'Janelas', and 'Ajuda'. Below the menu bar is a toolbar with icons for file operations and execution. The main text area displays the R startup message, which includes the version number (3.4.0), copyright information (© 2017 The R Foundation for Statistical Computing), and platform details (x86_64-w64-mingw32/x64 (64-bit)). It also provides instructions on how to use the software, including how to get help and how to cite R in publications. The prompt '>' is visible at the bottom of the console.

```
RGui (64-bit) - [R Console]
Arquivo  Editar  Visualizar  Misc  Pacotes  Janelas  Ajuda

R version 3.4.0 (2017-04-21) -- "You Stupid Darkness"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R é um software livre e vem sem GARANTIA ALGUMA.
Você pode redistribuí-lo sob certas circunstâncias.
Digite 'license()' ou 'licence()' para detalhes de distribuição.

R é um projeto colaborativo com muitos contribuidores.
Digite 'contributors()' para obter mais informações e
'citation()' para saber como citar o R ou pacotes do R em publicações.

Digite 'demo()' para demonstrações, 'help()' para o sistema on-line de ajuda,
ou 'help.start()' para abrir o sistema de ajuda em HTML no seu navegador.
Digite 'q()' para sair do R.

> |
```

Interfaces

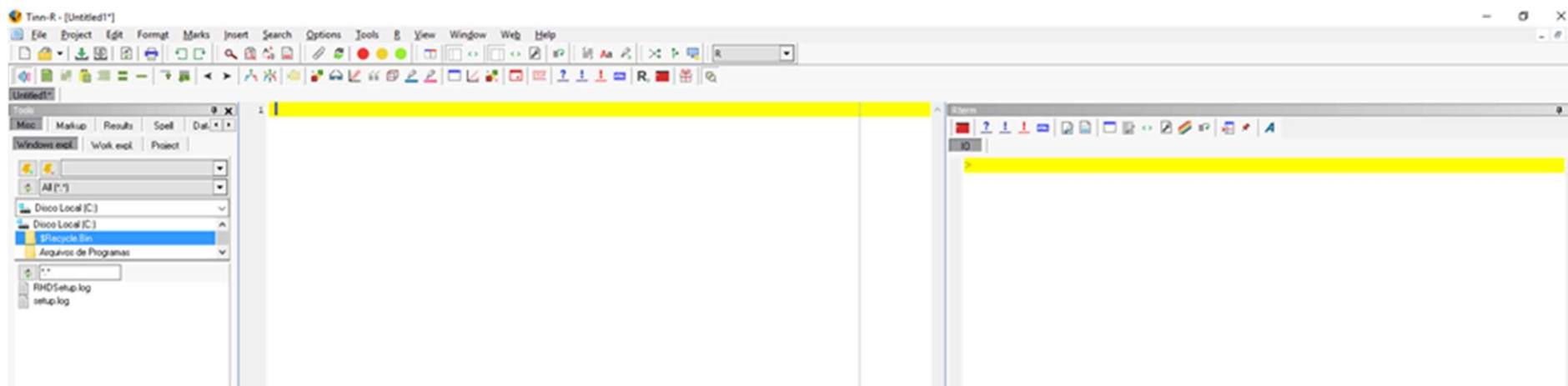


Figura 2 - Tinn-R

Interfaces

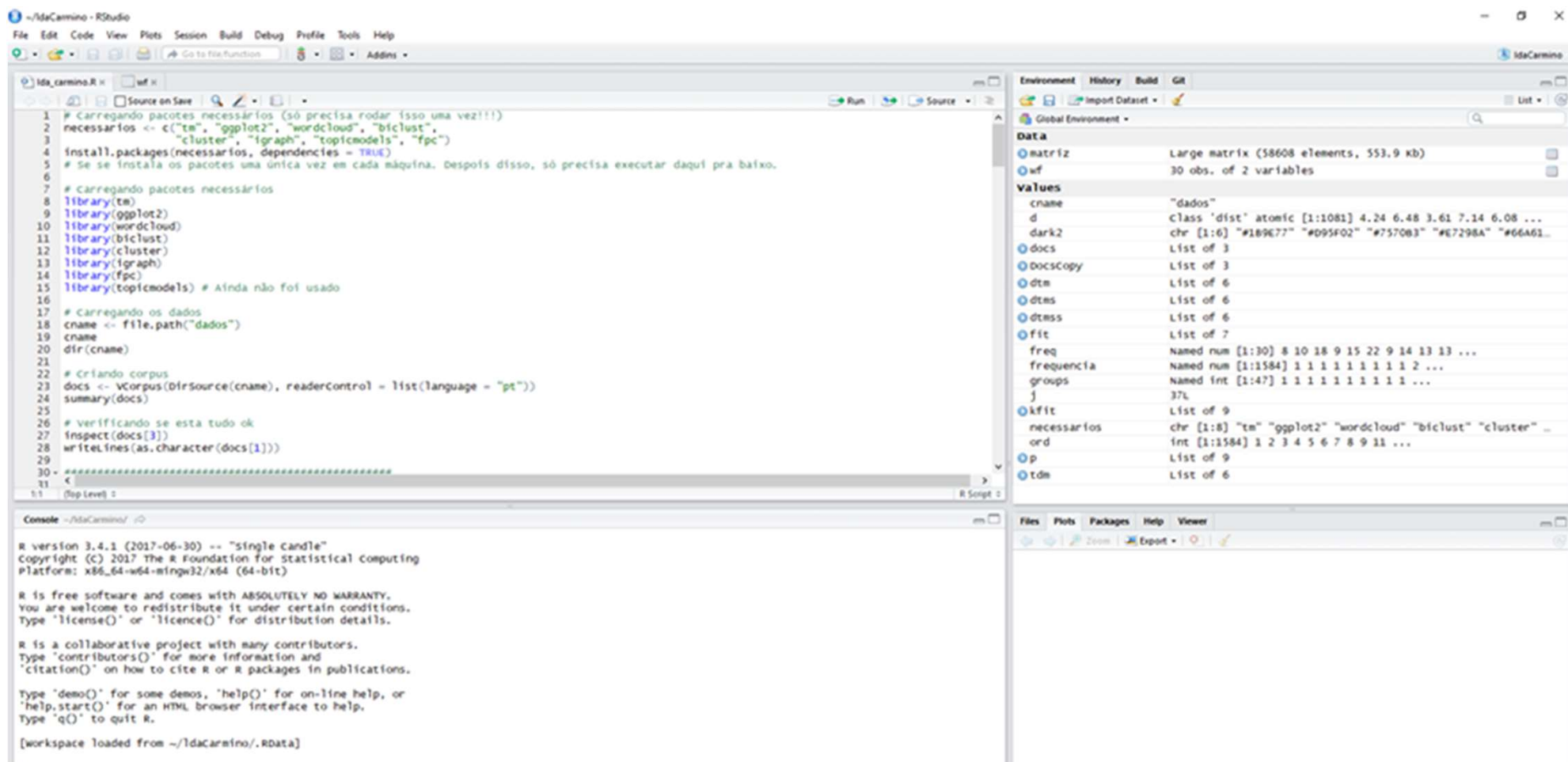


Figura 3 - Rstudio

Comandos Auxiliares

Função	Descrição
ls() ou objects()	lista curta de variáveis definidas
ls.str()	lista detalhada de variáveis definidas
str(x)	ver informações detalhadas de x
ls.str(ab)	ver informações detalhadas sobre todas as variáveis com “ab” em seu nome
rm(x)	deletar variável x
rm(x, y)	deletar as variáveis x e y
rm(list = ls())	deletar todas as variáveis (limpar a workspace)
class(x)	ver que tipo de objeto é x
q()	sair do R com a opção de salvar a workspace em um arquivo (“Name.RData”) e o histórico de comandos em outro arquivo (“Name.RHistory”)
ctrl + L	no teclado, pressione “ctrl+L” para limpar a tela da console

Sintaxe do R

R é uma linguagem de expressões com uma sintaxe simples e objetiva, sendo sensitive case. Seus comandos elementares são expressões ou atribuições.

O comando é uma expressão, o que implica que ao visualizar um valor, este é calculado e em seguida descartado. Já uma atribuição, ao contrário, calcula a expressão e atribui o resultado

Os comandos são separados por ponto e vírgula (“;”) ou são inseridos em nova linha. Podendo ser agrupados dentro de chaves (“{...}”) vários comandos elementares numa expressão mais complexa.

Sintaxe do R

```
rm(list = ls())
```

```
#Exemplo: entre com o vetor {48, 49, 51, 50, 49} e,  
#em seguida, acrescente os valores 60 e 63.
```

```
#
```

```
x<-c(48, 49, 51, 50, 49)# entrada do dados
```

```
x # apresentação dos dados
```

```
x=edit(x)# comando de edição dos dados
```

```
x# reapresentação dos dados
```

Tipos de Dados

De forma geral há quatro tipos de dados no R: numéricos, caracteres, lógicos e números complexos. Sendo que cada objeto possui dois atributos: tipo (mode) e o tamanho (length).

Essas informações são bastante importantes durante a manipulação de dados. Por exemplo, vetores devem possuir obrigatoriamente todos os elementos do mesmo tipo (exceto numéricos e complexos, que podem ser agrupados).

Tipos de Dados

```
rm(list = ls())
```

```
#Numérico
```

```
valor <- 605; valor
```

```
#Caracteres
```

```
string <- "Olá, mundo!"
```

```
string
```

```
#Lógicos
```

```
2 < 6
```

```
#Números complexos
```

```
nc <- 2 + 3i; nc
```

```
mode(valor); length(valor)
```

```
mode(string); length(string)
```

```
mode(2<4); length(2<4)
```

```
mode(nc); length(nc)
```

Atribuição de Valores

É comum em linguagens de programação a necessidade de atribuir valores para algumas variáveis, antes de utilizá-las, em R é possível fazer atribuição de valores e varias formas, conforme o exemplo abaixo.

```
rm(list = ls())
```

```
x <- 10 #x é a variável que recebe o valor 10;
```

```
0.56 -> x #x é a variável que recebe o valor 0.56;
```

```
x = -8 #x é a variável que recebe o valor -8;
```

```
assign("x", 2i) #x é a variável que recebe o imaginário 2i;
```

```
temp <- .Last.value;
```

```
temp
```

Atribuição de Valores

```
rm(list = ls())
```

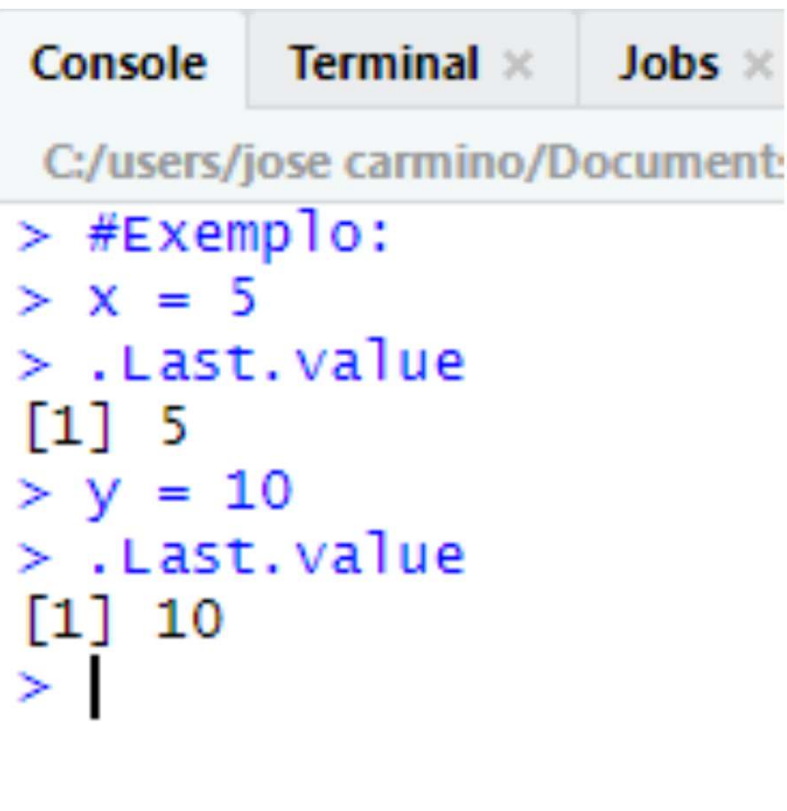
```
#Exemplo:
```

```
x = 5
```

```
.Last.value
```

```
y = 10
```

```
.Last.value
```



The screenshot shows an R console window with three tabs: 'Console', 'Terminal', and 'Jobs'. The 'Console' tab is active, displaying the following text: 'C:/users/jose carmino/Document:'. Below this, the code from the slide is entered and executed line by line. The prompt is '>'. The first line is '#Exemplo:', followed by 'x = 5', then '.Last.value' which returns '[1] 5'. The next line is 'y = 10', followed by '.Last.value' which returns '[1] 10'. The prompt '>' is shown again with a vertical cursor.

```
> #Exemplo:  
> x = 5  
> .Last.value  
[1] 5  
> y = 10  
> .Last.value  
[1] 10  
> |
```

Operações matemáticas simples

Precedência	Símbolo	Função
1	^	Potenciação
2	/	Divisão
2	*	Multiplicação
3	+	Adição
3	-	Subtração

$$h = 4 * \sqrt{3 * x} + \frac{15}{y - z} - x^2$$

#Exemplo:

```
rm(list = ls());
```

```
x=3; y=5; z=10;
```

```
h = 4*sqrt(3*x) + 15/(y-z) - x^2;
```

```
h
```

```

Console Terminal x Jobs x
C:/users/jose carmino/Documents/Rdados/ ↗
> source('~\Curso-R/Aula_001_005.R', echo=TRUE)

> #Exemplo:
> rm(list = ls());

> x=3; y=5; z=10;

> h = 4*sqrt(3*x) + 15/(y-z) - x^2;

> h
[1] 0
> |

```


Operadores relacionais

Símbolo	Descrição
<	Menor
<=	Menor do que ou igual a
>	Maior
>=	Maior do que ou igual a
==	Igual (comparação)
!=	Não igual (diferente)
&	E
	Ou
!	Negação
True ou 1	Valor booleano para verdadeiro
False ou 0	Valor booleano para falso

Funções matemáticas simples

Função	Descrição
<code>abs(x)</code>	valor absoluto de x
<code>log(x, b)</code>	logaritmo de x com base b
<code>log(x)</code>	logaritmo natural de x
<code>log10(x)</code>	logaritmo de x com base 10
<code>exp(x)</code>	exponencial elevado a x
<code>sin(x)</code>	seno de x
<code>cos(x)</code>	cosseno de x
<code>tan(x)</code>	tangente de x
<code>round(x, digits = n)</code>	arredonda x com n decimais
<code>ceiling(x)</code>	arredondamento de x para o maior valor
<code>floor(x)</code>	arredondamento de x para o menor valor
<code>length(x)</code>	número de elementos do vetor x
<code>sum(x)</code>	soma dos elementos do vetor x
<code>prod(x)</code>	produto dos elementos do vetor x
<code>max(x)</code>	seleciona o maior elemento do vetor x
<code>min(x)</code>	seleciona o menor elemento do vetor x
<code>range(x)</code>	retorna o menor e o maior elemento do vetor x

Vetores e Matrizes

Definição

Vetor é um conjunto de dados unidimensional, sua principal utilidade é armazenar dados na forma de lista, o que possibilita a aplicação de funções e operações sobre todos os dados pertencentes a determinado vetor com apenas poucos comandos.

#sintaxe: $x = c(a_1, a_2, a_3, \dots, a_{n-1}, a_n)$

#Exemplos:

```
vec <- c(1, 4, 10.5, 54.48, 9, 10); vec;
```

```
vec2 <- (1:10); vec2;
```

```
vec3 <- c((1:3),(3:1)); vec3;
```

```
vec4 <- c(0, vec3, 0); vec4;
```

```
max(vec4); min(vec4); range(vec4)
```

Vetores e Matrizes

```

Console Terminal x Jobs x
C:/users/jose carmino/Documents/Rdados/
> source('~/.Curso-R/Aula_001_006.R', encoding = 'UTF-8', echo=TRUE)

> #sintaxe: x = c(a1, a2, a3, . . . , an-1, an)
> #Exemplo:
> rm(list = ls());

> vec <- c(1, 4, 10.5, 54.48, 9, 10); vec;
[1] 1.00 4.00 10.50 54.48 9.00 10.00

> vec2 <- (1:10); vec2;
[1] 1 2 3 4 5 6 7 8 9 10

> vec3 <- c((1:3),(3:1)); vec3;
[1] 1 2 3 3 2 1

> vec4 <- c(0, vec3, 0); vec4;
[1] 0 1 2 3 3 2 1 0

> max(vec4); min(vec4);
[1] 3
[1] 0

> range(vec4)
[1] 0 3
> |

```

Vetores e Matrizes

Outra forma de declarar vetores

```
rm(list = ls());  
#vetor de "a" até "z" seq(from= a, to= z);  
#vetor de "a" até "z" com passo "n" seq(from= a, to= z, by= n );  
#vetor de "a" até "z" com "n" elementos seq(from= a, to= z,  
length.out= n);  
#Exemplos:  
seq(from=1, to=5);  
seq(from=1, to=5, by=0.5);  
seq(from=0, to=10, length.out= 4)
```

Vetores e Matrizes

```

Console Terminal x Jobs x
C:/users/jose carmino/Documents/Rdados/ ↗
> source('~\Curso-R/Aula_001_007.R', encoding = 'UTF-8', echo=TRUE)

> rm(list = ls());

> #vetor de "a" até "z" seq(from= a, to= z);
> #vetor de "a" até "z" com passo "n" seq(from= a, to= z, by= n );
> #vetor de "a" até "z" com "n" elemen .... [TRUNCATED]
[1] 1 2 3 4 5

> seq(from=1, to=5, by=0.5);
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0

> seq(from=0, to=10, length.out= 4)
[1] 0.000000 3.333333 6.666667 10.000000
> |

```

Arrays e Matrizes

Array ou matriz é definida como um conjunto de elementos de dados, que em geral é do mesmo tamanho e tipo de dado. Os seus elementos são acessados por sua posição no array, sendo que esta posição é dada por um índice, sendo que o índice é uma sequência de números naturais. Arrays podem ser de qualquer tipo, em particular trataremos de arrays numéricos. Os arrays unidimensionais são vetores e multidimensionais as matrizes.

Arrays e Matrizes

#sintaxe: x <- array(dados , dim= vetor_dimensão)

#Exemplo:

```
x <- array(c(1:10), dim = c(2,5))
```

x

```
max.col(x); max.col(x)
```

```
max.col(x,"first")
```

```
max.col(x, "last")
```

```
(mm <- rbind(x = round(2*stats::runif(12)), y =
round(5*stats::runif(12)), z = round(8*stats::runif(12))))
```

```
max.col(mm) ; max.col(mm)
```

```
max.col(mm, "first")
```

```
max.col(mm, "last")
```


Arrays e Matrizes

```
> rm(list = ls());

> #sintaxe: x <- array( dados , dim= vetor_dimensão )
> #Exemplo:
>
> x <- array(c(1:10), dim = c(2,5))

> x
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10

> max.col(x); max.col(x)
[1] 5 5
[1] 5 5

> max.col(x,"first")
[1] 5 5

> max.col(x, "last")
[1] 5 5

> (mm <- rbind(x = round(2*stats::runif(12)), y = round(5*stats::runif(12)), z = round(8*stats::runif(12))))
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
x         2    0    1    0    1    0    1    2    2    2    1    1
y         2    1    1    2    2    5    3    2    1    4    4    1
z         1    6    5    1    0    1    3    1    2    2    2    1

> max.col(mm) ; max.col(mm)
[1] 10 6 2
[1] 10 6 2

> max.col(mm, "first")
[1] 1 6 2

> max.col(mm, "last")
[1] 10 6 2
~ |
```

Arrays e Matrizes

```
rm(list = ls())  
A <- c(1:10) #vetor  
A  
rm(A)  
x <- matrix(c(10:20),2,5,1) #matriz 5x2  
x  
A <- matrix(c(1:10),2,5,1) #matriz 5x2  
A  
rm(A)  
A <- matrix(c(1:10),2,5,0) #matriz 5x2  
A  
A[2,4]  
A[2,4] - x[1,5]  
A[2,]  
A[,2:4]  
A[,]
```

Arrays e Matrizes

```
> x <- matrix(c(10:20),2,5,1) #matriz 5x2
> x
      [,1] [,2] [,3] [,4] [,5]
[1,]   10   11   12   13   14
[2,]   15   16   17   18   19
> A <- matrix(c(1:10),2,5,1) #matriz 5x2
> A
      [,1] [,2] [,3] [,4] [,5]
[1,]     1     2     3     4     5
[2,]     6     7     8     9    10
> rm(A)
> A <- matrix(c(1:10),2,5,0) #matriz 5x2
> A
      [,1] [,2] [,3] [,4] [,5]
[1,]     1     3     5     7     9
[2,]     2     4     6     8    10
> A[2,4]
[1] 8
> A[2,4] - x[1,5]
[1] -6
> A[2,]
[1] 2 4 6 8 10
> A[,2:4]
      [,1] [,2] [,3]
[1,]     3     5     7
[2,]     4     6     8
> A[,]
      [,1] [,2] [,3] [,4] [,5]
[1,]     1     3     5     7     9
[2,]     2     4     6     8    10
> |
```

A <- c(1:10) ; A; rm(A)

x <- matrix(c(10:20),2,5,1); x

A <- matrix(c(1:10),2,5,1); A; rm(A)

A <- matrix(c(1:10),2,5,0)

A

A[2,4]

A[2,4] - x[1,5]

A[2,]

A[,2:4]

A[,]

Operações e funções com Matrizes

Função	Descrição
$A * B$	produto elemento a elemento de A e B
$A \% * \% B$	produto matricial de A por B
$B = \text{aperm}(A)$	matriz transposta: $B = A_t$
$B = t(A)$	matriz transposta: $B = A_t$
$B = \text{solve}(A)$	matriz inversa: $B = A^{-1}$
$x = \text{solve}(A, b)$	resolve o sistema linear $Ax = b$
$\text{det}(A)$	retorna o determinante de A
$\text{diag}(v)$	retorna uma matriz diagonal onde o vetor v é a diagonal
$\text{diag}(A)$	retorna um vetor que é a diagonal da matriz A
$\text{diag}(n)$	sendo n um inteiro, retorna uma matriz identidade de ordem n
$\text{eigen}(A)$	retorna os autovalores e autovetores de A
$\text{eigen}(A)\$values$	retorna os autovalores de A
$\text{eigen}(A)\$vectors$	retorna os autovetores de A

Operações e funções com Matrizes

#Exemplos:

```
rm(list = ls())
```

```
x <- array(c(1:10), dim = c(2,5))
```

```
A <- matrix(c(1:10),2,5,0)
```

```
B = t(A); B
```

```
b= array(c(0,1,5),dim=c(3,1));
```

```
C= matrix(c(c(1,1,0),c(0,1,4),c(0:2)),3,3,1);
```

```
y = solve(C,b); y
```

```
invC = solve(C)
```

```
invC%*%b
```

Operações e funções com Matrizes

```
> #Exemplos:
> rm(list = ls())

> x <- array(c(1:10), dim = c(2,5))

> A <- matrix(c(1:10),2,5,0)

> B = t(A); B
      [,1] [,2]
[1,]     1     2
[2,]     3     4
[3,]     5     6
[4,]     7     8
[5,]     9    10

> b= array(c(0,1,5),dim=c(3,1));

> C= matrix(c(c(1,1,0),c(0,1,4),c(0:2)),3,3,1);

> y = solve(C,b); y
      [,1]
[1,]    -9
[2,]     9
[3,]    -2

> invC = solve(C)

> invC%%b
      [,1]
[1,]    -9
[2,]     9
[3,]    -2
> |
```

Entrada de Arquivos Externos

Os dados salvos em arquivos, na forma de planilhas, tabelas, etc., devem ser lidos pelo R o qual os transforma em um objeto. Para que o R reconheça o conjunto de dados do arquivo é necessário que as colunas sejam separadas. No caso desta separação não existir, o R não conseguirá interpretar os dados, e será emitida uma mensagem de erro. Uma maneira fácil de resolver este problema é usar o arquivo com formato (.csv) que utiliza virgula (,) como elemento separador das colunas.

Entrada de Arquivos Externos

```
rm(list = ls())
setwd('C:\\users\\jose carmino\\Documents\\Rdados')
getwd()
dir()
carregar <- read.table("Alunos_001.csv",header=T,sep=";",dec=".")
names(carregar)
dim(carregar)
carregar[1,3]; carregar[1:3,]; carregar[1:4,3]
dados_vitimas <- read.table("Dados_acidentes.txt",header=T,sep=";",dec=".")
acidentes <-
read.table("http://www.standclass.com.br/dados_r/acidentes.txt",header=T,sep=";",dec
=".")
```

Os dados no arquivo Dados acidentes e acidentes são os dados estatísticos dos acidentes de trânsito com vítimas ocorridos no município de São Paulo em 2017, disponíveis em
<http://www.cetsp.com.br/media/646657/relatorioanualacidentestransito-2017.pdf> (acesso 16/04/2019)

~\IdaCarmino - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

Aula_001_009.R x Aula_001_010.R* x

Source on Save Run Source

```

1 rm(list = ls())
2 setwd('C:\\users\\jose carmino\\Documents\\Rdados')
3 getwd()
4 dir()
5 carregar <- read.table("Alunos_001.csv",header=T,sep=";",dec=".")
6 names(carregar)
7 dim(carregar)
8 carregar[1,3]; carregar[1:3,]; carregar[1:4,3]
9 dados_vitimas <- read.table("Dados_acidentes.txt",header=T,sep=";",dec=".")
10 acidentes <- read.table("http://www.standclass.com.br/dados_r/acidente
11 |

```

11:1 (Top Level) R Script

Environment History Connections Build Git

Global Environment

Data

acidentes	53 obs. of 14 variables
carregar	50 obs. of 4 variables
dados_vit...	53 obs. of 14 variables

Console Terminal x Jobs x

C:/users/jose carmino/Documents/Rdados/

```

> view(acidentes)
> acidentes <- read.table("http://www.standclass.com.br/acidentes.txt",header=T,sep=";",dec=".")
Error in file(file, "rt") : cannot open the connection
> acidentes <- read.table("http://www.standclass.com.br/acidentes.txt",header=T,sep=";",dec=".")
Error in file(file, "rt") : cannot open the connection
> acidentes <- read.table("http://www.standclass.com.br/dados_r/acidentes.txt",header=T,sep=";",dec=".")
> view(acidentes)
> view(dados_vitimas)
> view(acidentes)
> |

```

Files Plots Packages Help Viewer

runif

R: The Uniform Distribution Find in Topic

THE UNIFORM DISTRIBUTION

Description

These functions provide information about the uniform distribution on the interval from min to max. `dunif` gives the density, `punif` gives the distribution function `qunif` gives the quantile function and `runif` generates random deviates.

R ~/IdaCarmino - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

Aula_001_009.R Aula_001_010.R* acidentes

Filter

	..Rua	X2005	X2006	X2007	X2008	X2009
1	Marginal Tiet�	43	49	48	58	50
2	Av. Sen. Teot�nio Vilela	27	25	16	20	17
3	Marginal Pinheiros	28	37	33	29	30
4	Estr. de Itapeberica	9	9	14	14	10
5	Av. Dona Belmira Marin	6	8	13	10	5
6	Av. Jacu-P�ssegos/Nova Trabal.	6	10	14	7	17
7	Av. Sapopemba	15	14	14	16	15
8	Av. Marechal Tito	10	13	9	7	9

Showing 1 to 10 of 53 entries, 14 total columns

Console Terminal Jobs

```

C:/users/jose carmino/Documents/Rdados/
> acidentes <- read.table("http://www.standclass.com.br/acidentes.txt",header=T,sep=";",dec=".")
Error in file(file, "rt") : cannot open the connection
> acidentes <- read.table("http://www.standclass.com.br/acidentes.txt",header=T,sep=";",dec=".")
Error in file(file, "rt") : cannot open the connection
> acidentes <- read.table("http://www.standclass.com.br/dados_r/acidentes.txt",header=T,sep=";",dec=".")
> view(acidentes)
> view(dados_vitimas)
> view(acidentes)
> view(acidentes)
>

```

Environment History Connections Build Git

Global Environment

Data

- acidentes 53 obs. of 14 variables
- carregar 50 obs. of 4 variables
- dados_vit... 53 obs. of 14 variables

Files Plots Packages Help Viewer

R: The Uniform Distribution Find in Topic

The Uniform Distribution

Description

These functions provide information about the uniform distribution on the interval from min to max. `dunif` gives the density, `punif` gives the distribution function, `qunif` gives the quantile function and `runif` generates random deviates.

Linguagem R

Prof. José Carmino

E-Mail: prof.carmino@gmail.com

Agenda

Tópico	Descrição
Apresentação do R	Criação interface, sintaxe, tipos de dados, comandos básicos, vetores e matrizes e entrada de arquivos
Programação em R	Estruturas de repetição, decisão
Gráficos	Gráficos de análise descritiva
Estatística descritiva	Medidas de posição e medidas de dispersão
Probabilidade	Definição e axiomas
Variáveis aleatórias	Variáveis aleatórias discretas e contínuas
Inferência estatística	Teste de hipótese de uma e duas amostras
Regressão e correlação linear simples	Determinando a equação linear, coeficiente de correlação e coeficiente de determinação

Programação em R

O R possui diversas funções que têm como objetivo de ler dados introduzidos pelo usuário via teclado. A função **print()**, por exemplo, pode ser usada para escrever o conteúdo de qualquer objeto. Porém, para objetos muito grandes é mais conveniente utilizar a função **str()**. Outra função é a **cat()** que também pode ser usada para escrever objetos ou constantes. Funciona com um número qualquer de argumentos, sendo que sua função é transformar os seus argumentos em “strings” e depois escrevê-los no Console.

```
#Exemplo:
```

```
ano=2019
```

```
cat("Estamos no ano ",ano,"\n\tComo passa rápido!\n")
```


Programação em R

O R interpreta os caracteres entre aspas como uma mensagem introduzida pelo programador. Já os caracteres `"\t"` e `"\n"` representam, respectivamente, os comandos de tabulação e de nova linha.

A leitura de dados pode ser feita com o comando `scan()`. Neste comando o R fará a leitura de uma quantidade finita (até que sejam dados dois ENTER's seguidos) de elementos introduzidos pelo usuário. Tendo a necessidade de especificar o número de elementos que o R fará a leitura, é necessário acrescentar o argumento, número de elementos.

```
#Sintaxe: scan(n=número de elementos)
```

```
#Exemplo:
```

```
x<-scan(n=2)
```

Programação em R

O comando `scan()` também pode ser usado para a leitura de strings e valores numéricos

```
scan(what=character())
```

```
> scan(what=character())  
1: Curso da linguagem R  
5:  
Read 4 items  
[1] "Curso"      "da"          "linguagem"  "R"  
>
```

Condicionais

As estruturas de decisão permitem ao programador explicitar diferentes alternativas a serem executadas em função de uma condição a ser testada em dado ponto do código, usando um bloco de instruções. Um bloco de instruções, em linguagem R, é indicado organizar essas instruções em linhas separadas e seguidas, delimitadas por chaves.

A instrução *if* é uma estrutura de decisão onde o *else* é opcional. Podemos, ainda, usar várias instruções *if* aninhadas,

#Exemplo:

```
rm(list = ls())
x<-2 #x é positivo
z<-1
if (x > 0)
{
  cat("x é positivo!\n")
  y <- z / x
} else
{
  cat("x não é positivo!\n")
  y <- z
}
```


Condicionais

```
> #Exemplo:
> rm(list = ls())

> x<-2 #x é positivo

> z<-1

> if (x > 0)
+ {
+   cat("x é positivo!\n") #comando para escrever texto
+   y <- z / x
+ }else
+ {
+   cat("x não é positivo!\n")
+   y <- z
+ }
x é positivo!
> |
```

Condicionais

#Exemplo:

```
rm(list = ls())
```

```
idade = 20
```

```
if (idade < 18) {
```

```
  cat("Idade menor que 18\n")
```

```
} else if (idade < 35) {
```

```
  cat("Idade menor que 35\n")
```

```
} else if (idade < 60) {
```

```
  cat("Idade menor que 65\n")
```

```
} else {
```

```
  cat("Idade maior ou igual a 60.
```

```
BEM-VINDO À TERCEIRA
```

```
IDADE!\n")
```

```
}
```

```
> #Exemplo:
```

```
> rm(list = ls())
```

```
> idade = 60
```

```
> if (idade < 18) {
```

```
+   cat("Idade menor que 18\n")
```

```
+ } else if (idade < 35) {
```

```
+   cat("Idade menor que 35\n")
```

```
+ } else if (idade < 60) {
```

```
+   cat("Ida ..." ... [TRUNCATED]
```

```
Idade maior ou igual a 60. BEM-VINDO À TERCEIRA IDADE!
```

```
> |
```

Condicionais

Outra instrução condicional é o *switch()* o qual é usado para escolher uma entre várias alternativas.

```
#Sintaxe switch(valor,lista de componentes)
```

```
#Exemplo:
```

```
rm(list = ls())
```

```
semaforo <- "verde" #situação do semáforo entre aspas (string)
```

```
a<-switch(semaforo, verde = "continua", amarelo = "acelera", vermelho = "pára")
```

```
a
```

```
> #Exemplo:
```

```
> rm(list = ls())
```

```
> semaforo <- "verde" #situação do semáforo entre aspas (string)
```

```
> a<-switch(semaforo, verde = "continua", amarelo = "acelera", vermelho = "pára")
```

```
> a
```

```
[1] "continua"
```

```
> |
```

Instruções Iterativas

O R tem várias instruções iterativas (ciclos estrutura de repetição) que nos permitem repetir blocos de instruções. Para o controle destes ciclos o R possui as instruções como **repeat**, **for** e **while**.

#Sintaxe while (condição) {bloco de instruções}

Instruções Iterativas

```
rm(list = ls())  
x <- rnorm(1)  
while (x < 1) { cat("x=", x, "\t"); x <- rnorm(1)  
  if (x >= 1) { cat('\n'); } }
```

```
> x <- rnorm(1)  
  
> while (x < 1)  
+ {  
+   cat("x=", x, "\t")  
+   x <- rnorm(1)  
+   if (x >= 1)  
+   {  
+     cat("\n")  
+   }  
+ }  
x= -0.9572006   x= 0.3372781   x= 0.7912818   x= -0.6783721   x= 0.5030373   x= 0.448444  
> |
```

Instruções Iterativas

A instrução `repeat` permite que o R execute um bloco de instruções uma ou mais vezes.

#sintaxe `repeat {bloco de instruções}`

Observe que não há uma condição de saída do bloco, como em `while`. Logo, o R usa outra função para parar a iteração, esta função é o **break**. O seu papel é fazer com que o R termine a execução do bloco. No entanto vale ressaltar que, para evitar *ciclos infinitos* é necessário garantir que a instrução **break** ocorra dentro do bloco de instrução.

Instruções Iterativas

A instrução `repeat` permite que o R execute um bloco de instruções uma ou mais vezes.

```
texto <- c()
repeat
{
  cat("Introduza uma frase (frase vazia termina): "); fr <- readLines(n=1)
  if (fr == "")
  {
    cat("Frase vazia inserida!"); cat("\nFIM\n"); break
  }else
  {
    texto <- c(texto,fr); cat("Frases inseridas:\n",texto,"\n")
  }
}
```

Instruções Iterativas

Além da instrução `break`, temos a instrução **`next`** que avança para a próxima iteração.

```
ans = 0
repeat
{
  cat("Introduza um número (zero termina): \n");  nro <- scan(n = 1)
  if (nro < 0 || nro > 0)
  {
    ans <- c(ans, nro) ;    next
  }
  if (nro == 0)
  {
    cat("Os números introduzidos foram:\n", ans, "\n");  print("FIM")
    cat("\n");  break
  }
}
```


Instruções Iterativas

A ultima estrutura de iteração é a **for**, a qual permite o controle do número de vezes que um ciclo é executado. Isto é possível, por meio de uma variável de controle a qual assume uma serie de valores determinados pelo programador.

#Sintaxe

```
for(<variável de controle> in <conjunto>)  
{bloco de instruções}
```

Instruções Iterativas

```
x <- rnorm(10)
soma <- 0
for (v in x)
{
  if (v > 0) { y <- v }
  else      { y <- 0 }
  soma <- soma + y
}
x
```

```
> x <- rnorm(10)
> soma <- 0
> for (v in x)
+ {
+   if (v > 0)
+   {
+     y <- v
+   }
+   else
+   {
+     y <- 0
+   }
+   soma <- soma + y
+ }
> x
[1]  0.16127471 -0.81250154  0.07962572  0.3230
[8] -1.83707964  1.14941295 -0.27715120
> |
```

Instruções Iterativas

```
rm(list = ls())  
x <- rnorm(1e+05)  
t <- Sys.time()  
soma <- 0  
for (v in x)  
{  
  if (v > 0) { y <- v }  
  else { y <- 0 }  
  soma <- soma + y  
}  
Sys.time() - t; t <- Sys.time()  
soma <- sum(x[x > 0])  
Sys.time() - t
```

```
> Sys.time() - t  
Time difference of 0.02598405 secs  
  
> t <- Sys.time()  
  
> soma <- sum(x[x > 0])  
  
> Sys.time() - t  
Time difference of 0.003998995 secs  
> |
```

Agenda

Tópico	Descrição
Apresentação do R	Criação interface, sintaxe, tipos de dados, comandos básicos, vetores e matrizes e entrada de arquivos
Programação em R	Estruturas de repetição, decisão
Gráficos	Gráficos de análise descritiva
Estatística descritiva	Medidas de posição e medidas de dispersão
Probabilidade	Definição e axiomas
Variáveis aleatórias	Variáveis aleatórias discretas e contínuas
Inferência estatística	Teste de hipótese de uma e duas amostras
Regressão e correlação linear simples	Determinando a equação linear, coeficiente de correlação e coeficiente de determinação

Gráficos

Em R a representação gráfica é uma componente importante e versátil do ambiente. É possível construir gráficos bidimensionais simples e tridimensionais mais complexos, isto com a aplicação de comandos simples. O que faz com que o R sempre seja lembrado quanto a criação de gráficos estatísticos, tais como histogramas, curvas de distribuições, gráfico de barras dentre outros.

Um exemplo simples:

```
a <- 1:20; b <- a^2; plot(a,b)
```

Gráficos

Um exemplo simples:

```
a <- 1:20; b <- a^2; plot(a,b)
```

Quais são os valores atribuídos a *a* e *b*

Observe ainda que no comando plot o primeiro parâmetro representa o eixo x e o segundo o eixo y

Já o comando plot(b, a, type = "l") com o parâmetro type="l", informa que o gráfico será contínuo.

Gráficos

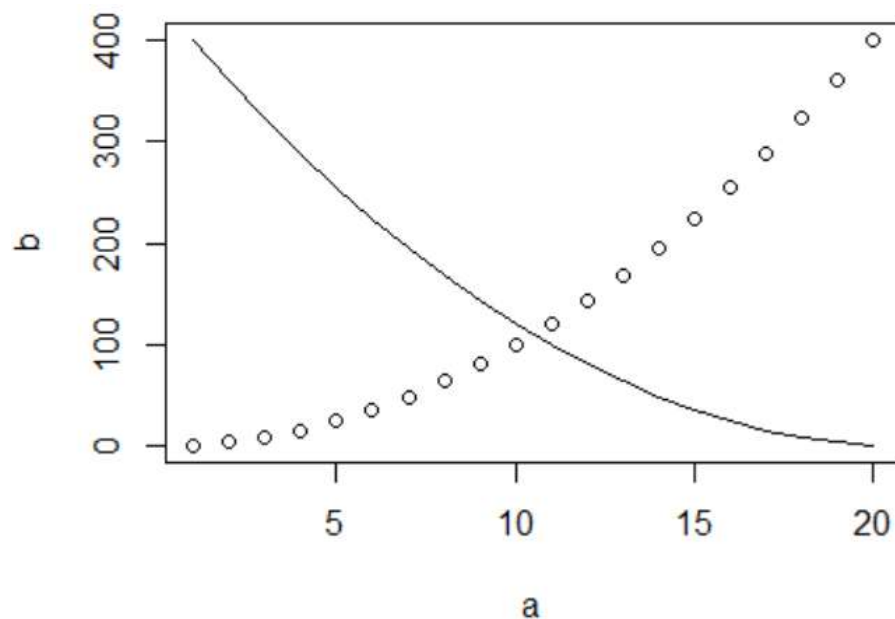
Experimente:

```
a <- 1:20; b <- a^2; plot(a,b)
```

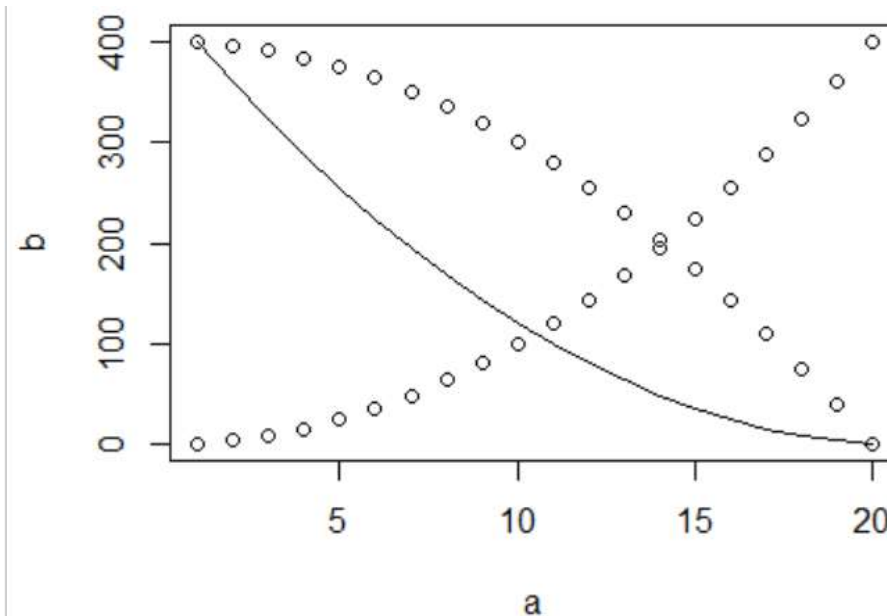
```
lines(rev(a),b)
```

```
points(a, 400-b)
```

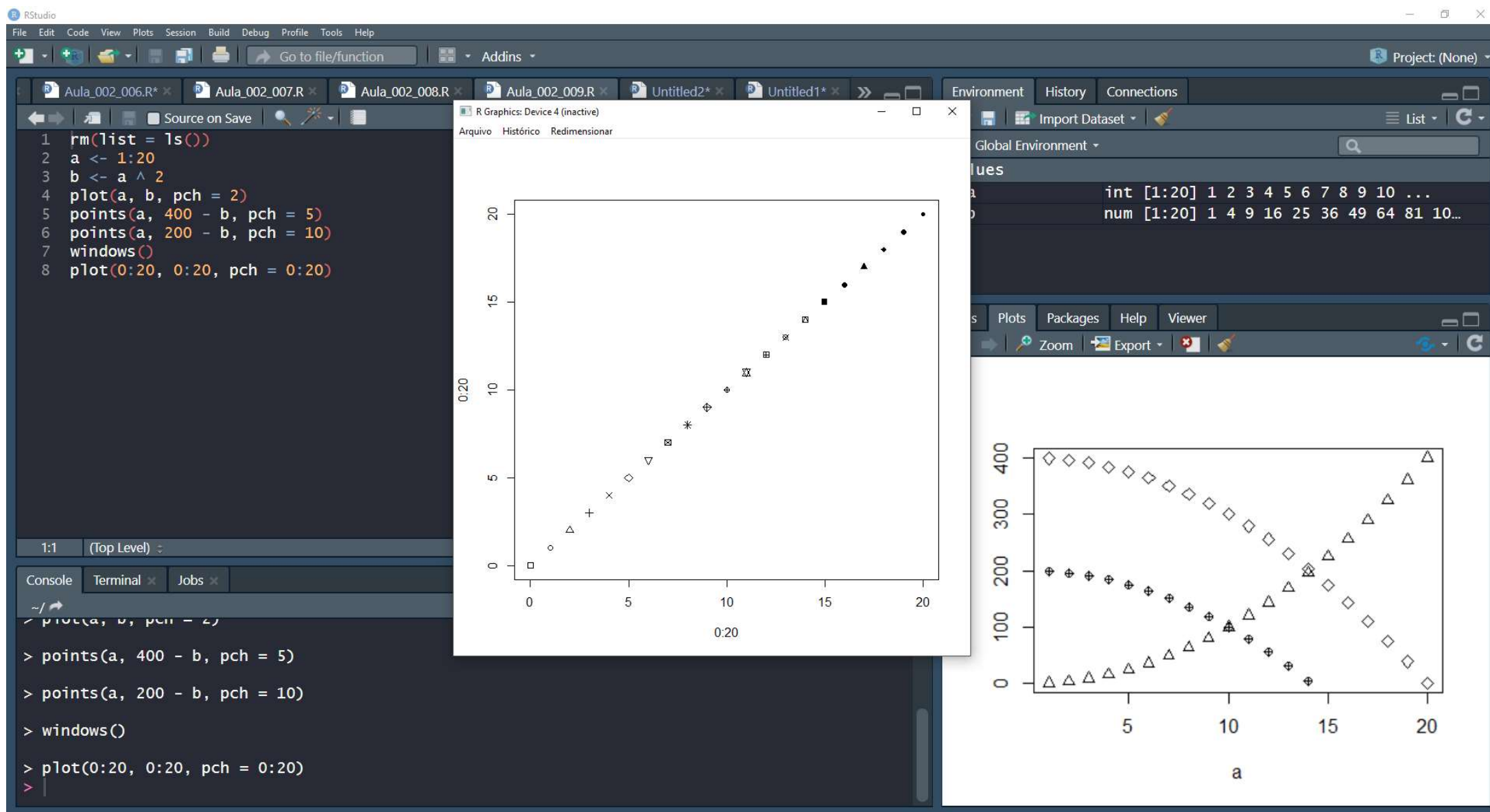
Ao executar lines(rev(a),b)



Ao executar points(a, 400-b)



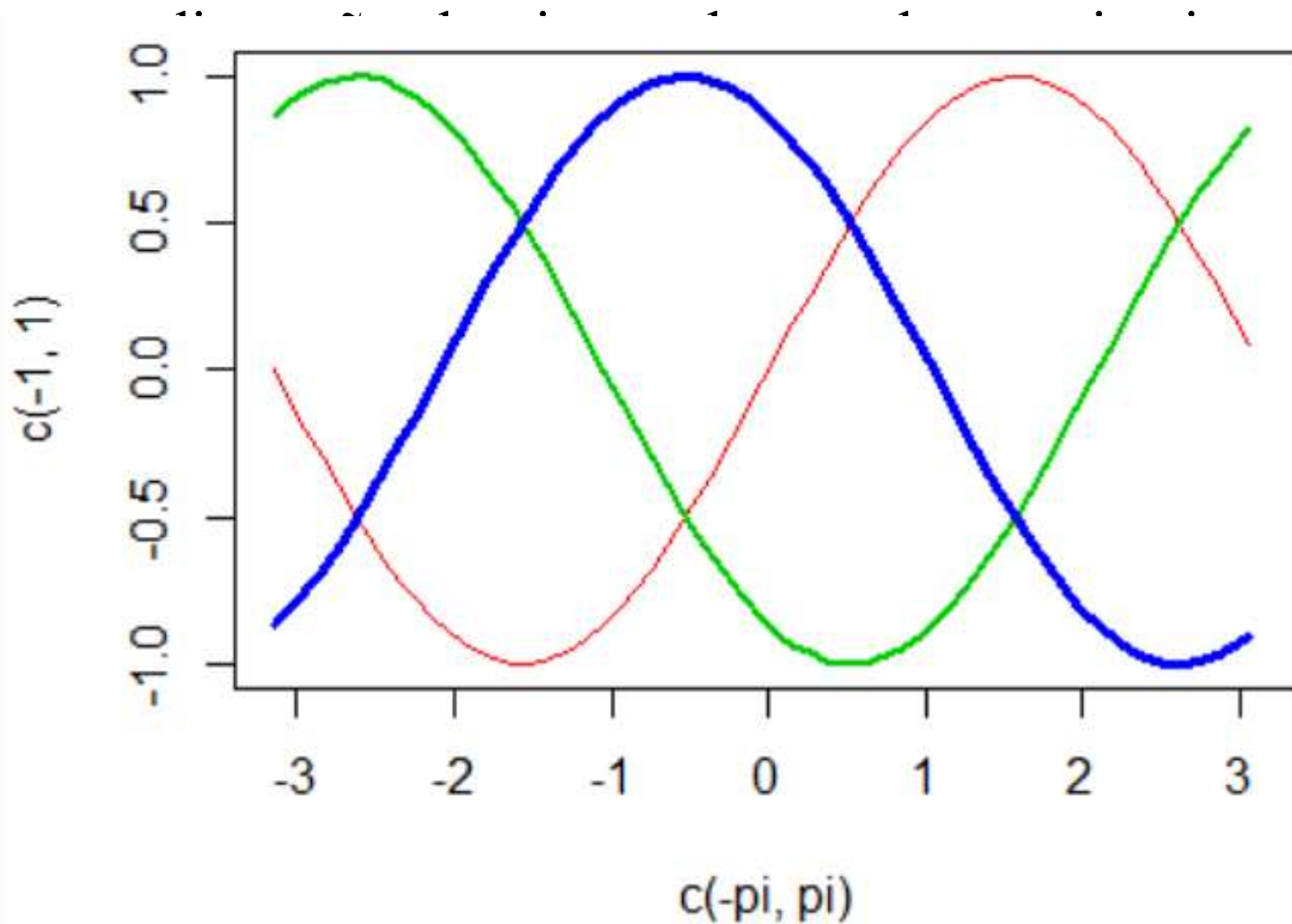
Gráficos



Gráficos

Para alterar
gráfico e
depois p

```
rm(list =  
plot(c(-pi, pi)  
x <- seq  
a <- sin(  
b <- sin(  
c <- sin(  
lines(x, a  
lines(x, b  
lines(x, c
```

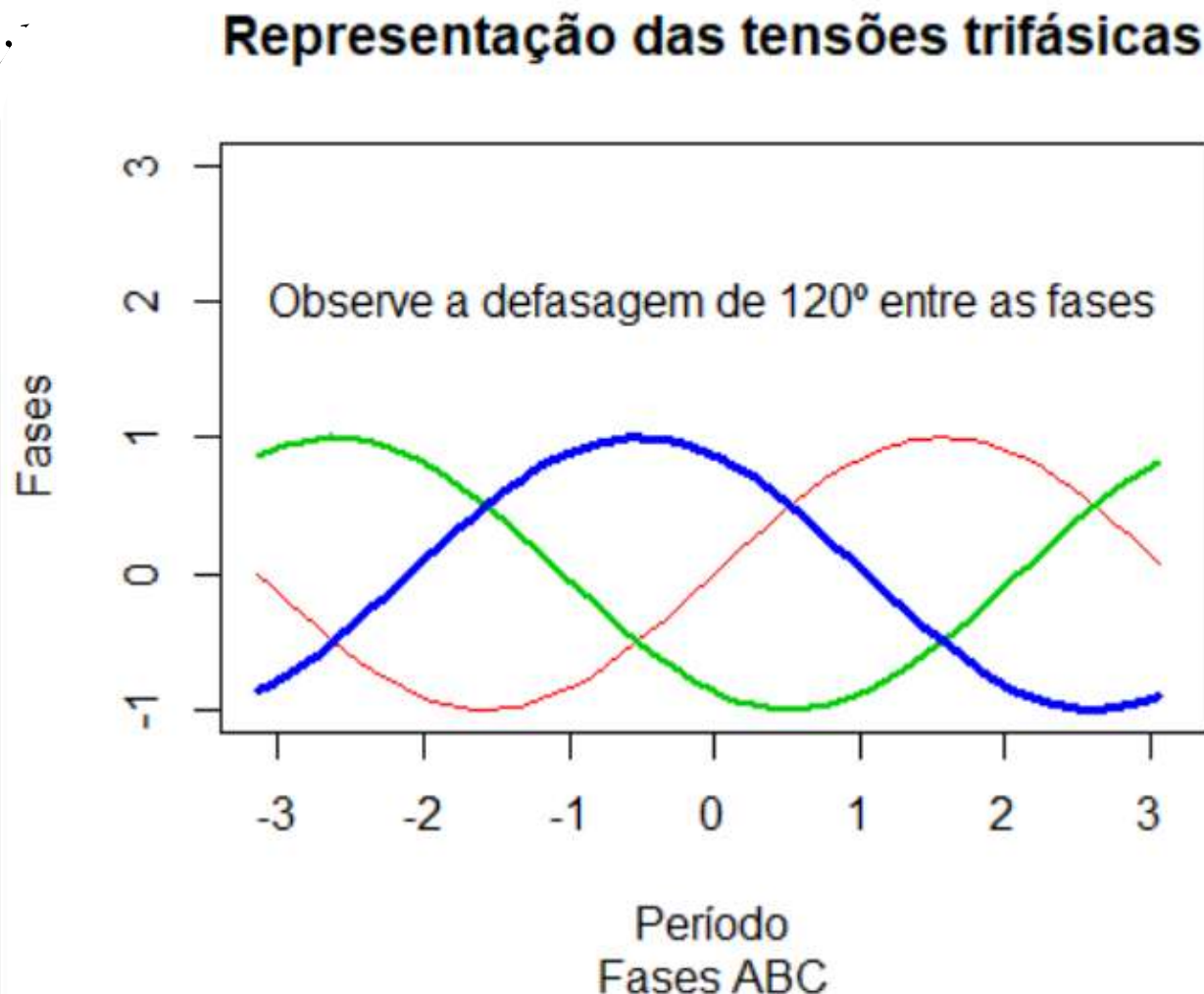


um
da e

Gráficos

Por fim para acrescentar os nomes dos eixos, temas, parâmetros
“xlab=” e “ylab=”

```
rm(list = ls())
plot(c(-pi, pi), c(-1, 1),
x <- seq(-pi, pi, 0.1),
c <- sin(x + pi/3),
title("Representação das tensões trifásicas"),
lines(x, a, c, "a"),
lines(x, b, c, "b"),
lines(x, c, c, "c"),
text(0, 2, "Observe a defasagem de 120° entre as fases"))
```

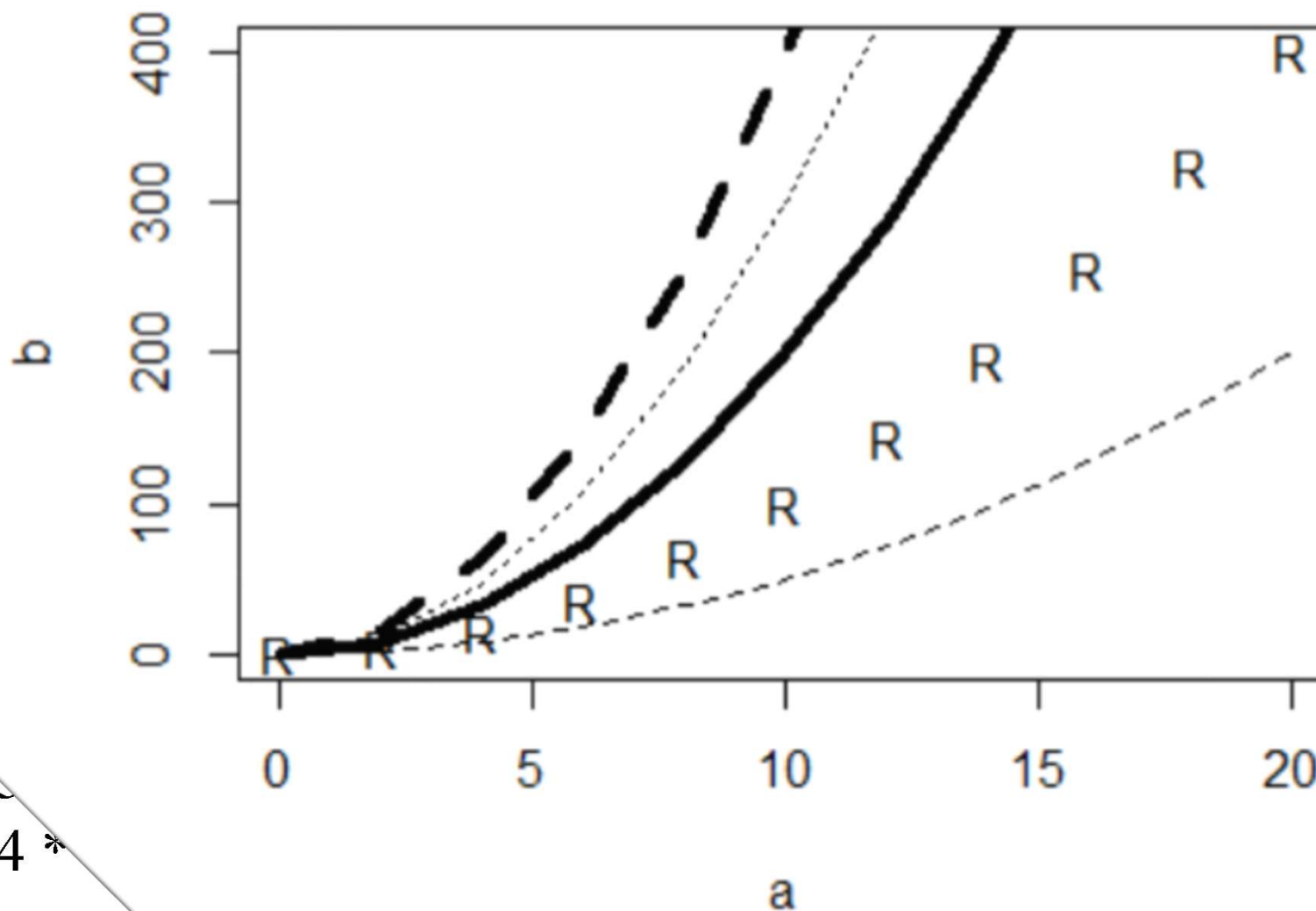


= "n")

Gráficos

Pode-se
um sím
entende

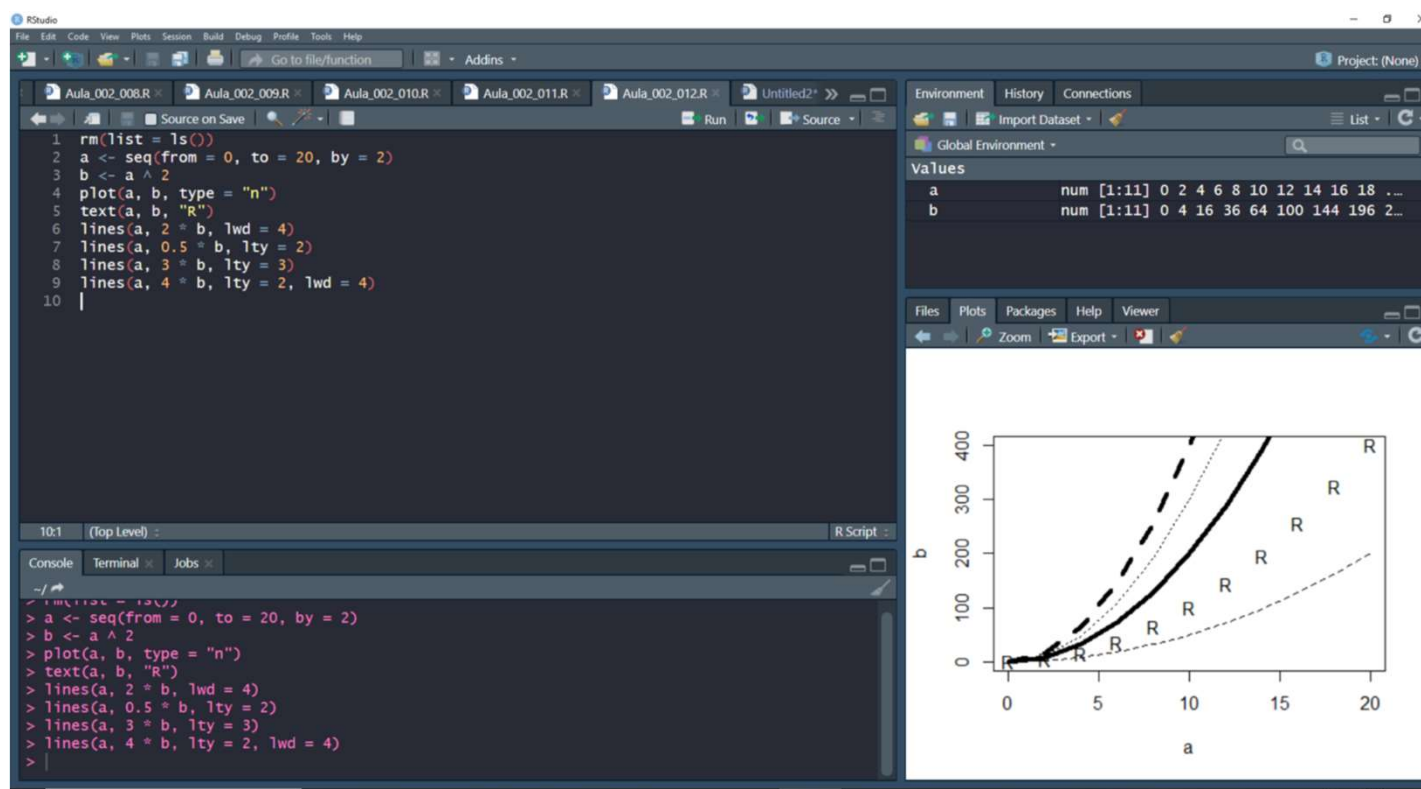
```
rm(list = ls())
a <- seq(0, 20, by = 1)
b <- a^2
plot(a, b, type = "n", las = 1)
text(a, b, "R", col = "red", size = 12)
lines(a, b, col = "black", lty = 1)
lines(a, b, col = "black", lty = 2)
lines(a, b, col = "black", lty = 3)
lines(a, 4 * a, col = "black", lty = 4)
```



atribuir
para

Gráficos

Ao criar sucessivos gráficos por meio do comando `plot()`, estes são sobrepostos na mesma janela gráfica chamada de device ACTIVE. Para evitar este problema, podemos proceder de duas formas, conforme a conveniência:



Gráficos

Os gráficos podem ser salvos imediatamente ao serem gerados. Existem vários formatos em que o R pode salvar imagens gráficas. Alguns deles são: JPEG, BMP, PDF, TIFF, PNG. Faremos abaixo um exemplo utilizando o formato JPEG, mas tenha em mente que a sintaxe para qualquer formato segue idêntica.

```
rm(list = ls())  
a <- seq(from = 0, to = 20, by = 2); b <- a ^ 2  
plot(a, b, type = "n"); text(a, b, "R")  
lines(a, 2 * b, lwd = 4) ; lines(a, 0.5 * b, lty = 2)  
lines(a, 3 * b, lty = 3); lines(a, 4 * b, lty = 2, lwd = 4)  
jpeg(file="MUDALIN.JPG") ; plot(rnorm(10))  
dev.off()
```

Gráficos

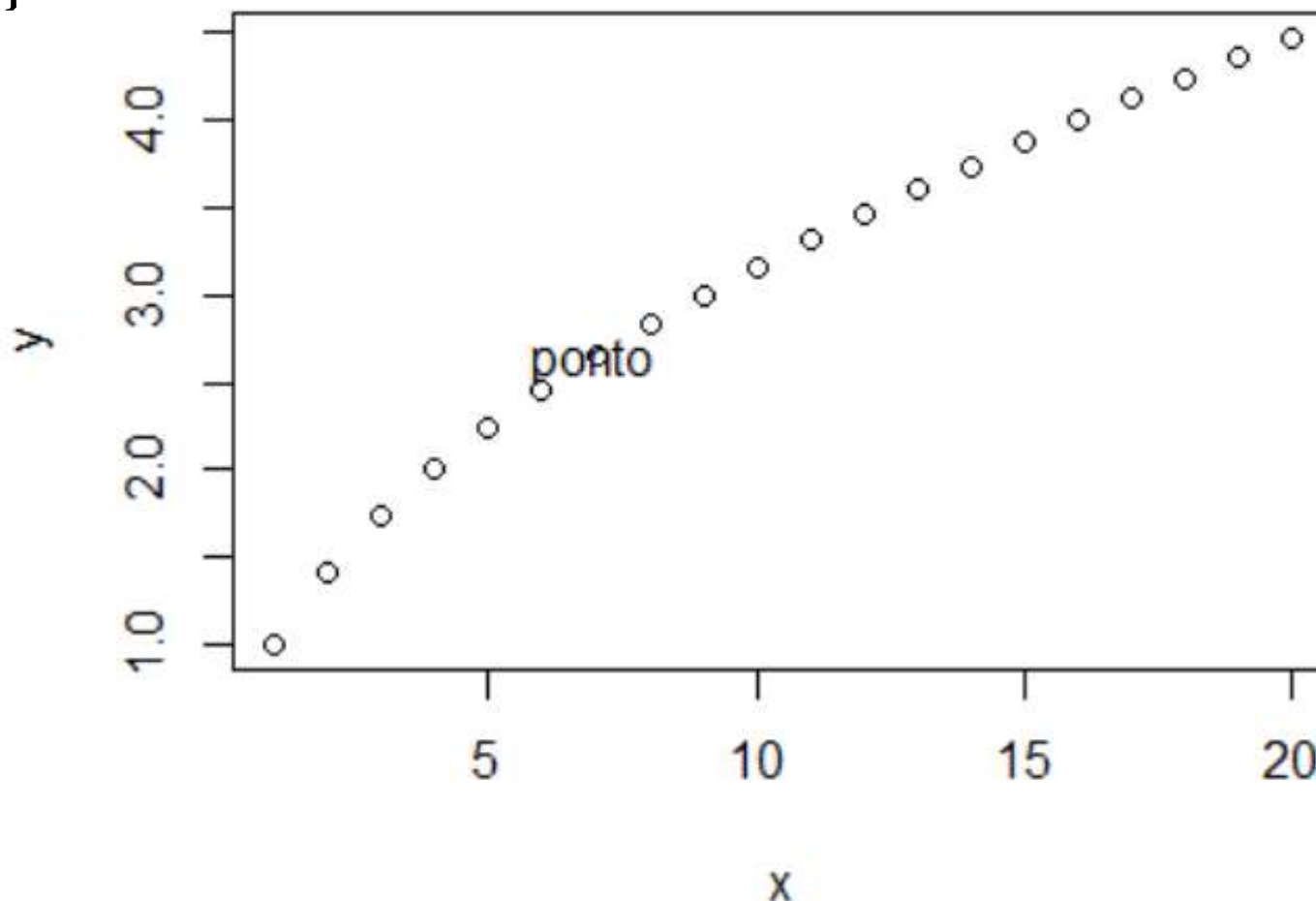
Existem outras funcionalidades que estão ligadas há gráficos, quando utilizando o R. Uma delas é a identificação de um ponto ou um conjunto de pontos nos dados. R então disponibiliza dois identificadores que são:

locator(): permite selecionar regiões do gráfico utilizando o botão esquerdo do mouse até que se tenha um número n de pontos selecionados ou até pressionar o botão direito do mouse.

Gráficos

Veja o exemplo

```
x=1:20
y=sqrt(x)
plot(x,y)
text(locator(2),
#onde for c
#ou de out
plot(x,y)
locator(2)
#localiza d
x
```



Gráficos

identify():

identificação

Observe

```
x <- c(2, 3, 4, 5, 6, 7, 8, 9)
```

```
y <- c(15, 25, 45, 5, 75, 10, 60, 30)
```

```
nomes <- c("a", "b", "c", "d", "e", "f", "g", "h")
```

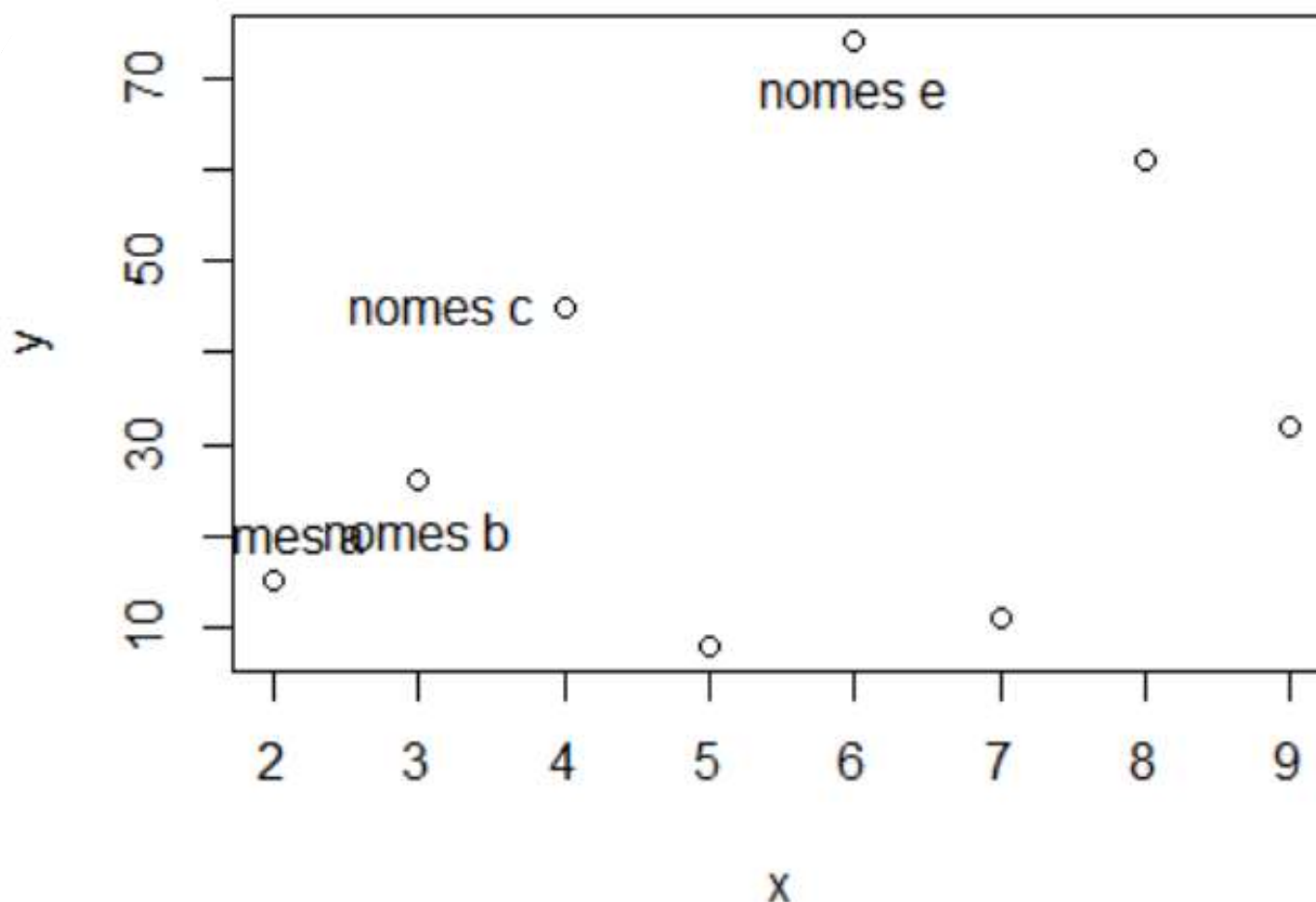
```
alunos <- data.frame(x, y, nomes)
```

```
alunos
```

```
plot(alunos)
```

```
identify()
```

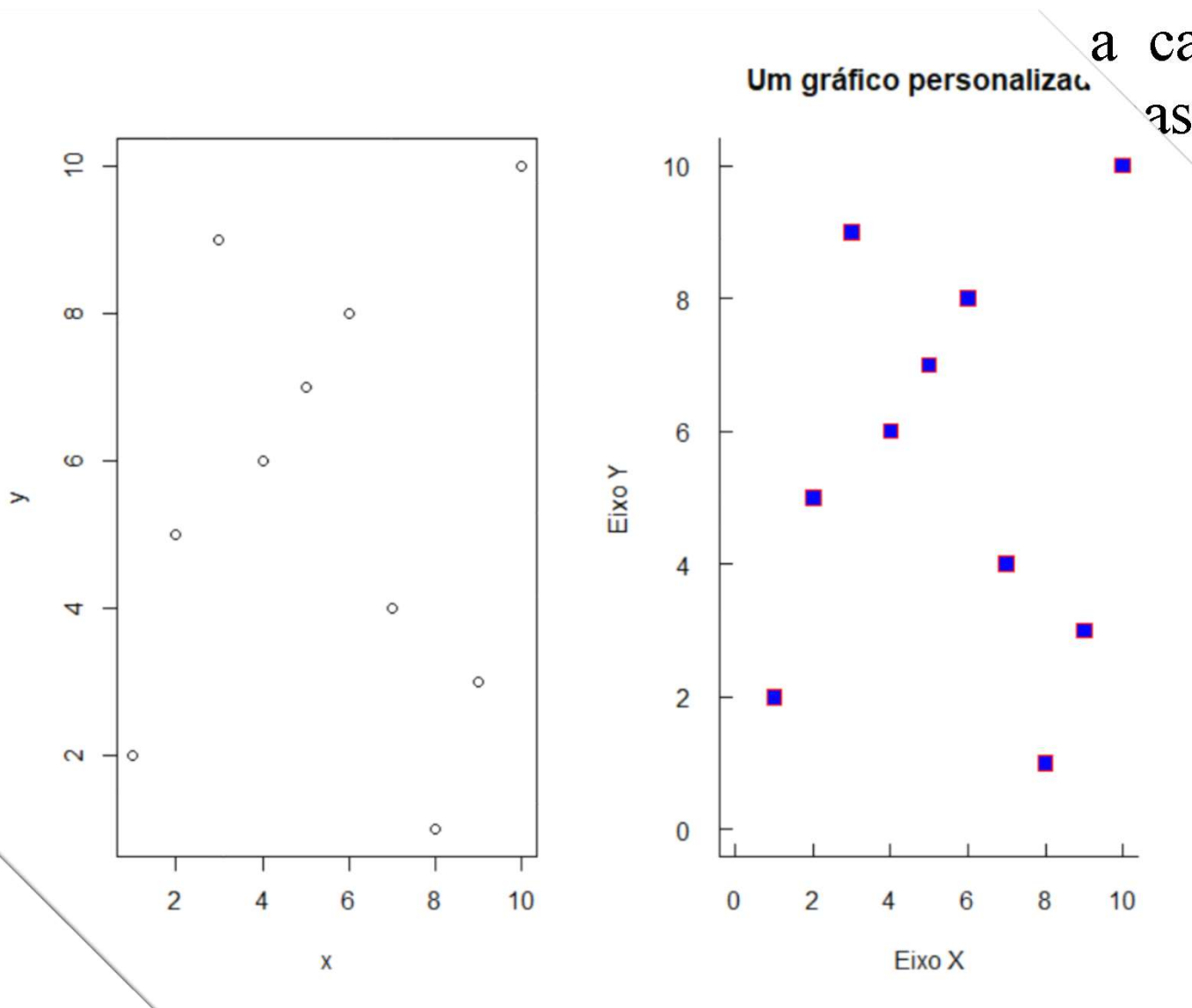
Índice de
posição.



Gráficos

identify
identific
Observe

```
par(mfrow = c(1, 2))
x; y; plot(x, y)
plot(x, y, main = "Um gráfico personalizado",
      xlim = c(0, 10), ylim = c(0, 10),
      col = "blue", bg = "red", las = 1,
      xlab = "Eixo X", ylab = "Eixo Y")
)
```



Gráficos

Utilize o script abaixo para plotar o gráfico, e adicione o nome dos alunos a cada um dos pontos.

```
x <- c(2,3,4,5,6,7,8,9)
y <- c(15,26,45,8,74,11,61,32)
nomes <- paste("nomes", letters[1:8], sep= " ")
alunos <- data.frame(x,y,row.names=nomes)
alunos
plot(alunos)
```

Gráficos de Análise Descritiva

Os três gráficos considerados fundamentais para a análise descritiva são: o histograma, gráfico de barras e gráfico de caixas. Estes são implementados no R com os nome `hist`, `barplot` e `boxplot`.

O histograma divide uma série de dados em distintas classes igualmente espaçadas e mostra a frequência de valores em cada classe. Em um gráfico, o histograma mostra diferentes barras, com bases iguais e amplitudes relativas às frequências dos dados em cada classe. O eixo das ordenadas, portanto, mostra a frequência relativa de cada classe e o eixo das abcissas os valores e intervalos das classes.

Histograma

Tendo os dados, montar o histograma é uma tarefa simples em R

96	96	102	102	102	104	104	108
126	126	128	128	140	156	160	160
164	170	115	121	118	142	145	145
149	112	152	144	122	121	133	134
109	108	107	148	162	96		

```
rm(list = ls())
```

```
rest <- c(96,96,102,102,102,104,104,108, 126,126,128,128,140,156,  
160,160,164,170,115,121,118,142,145,145,149,112,152,144,122, 121,  
133,134,109,108,107,148,162,96)
```

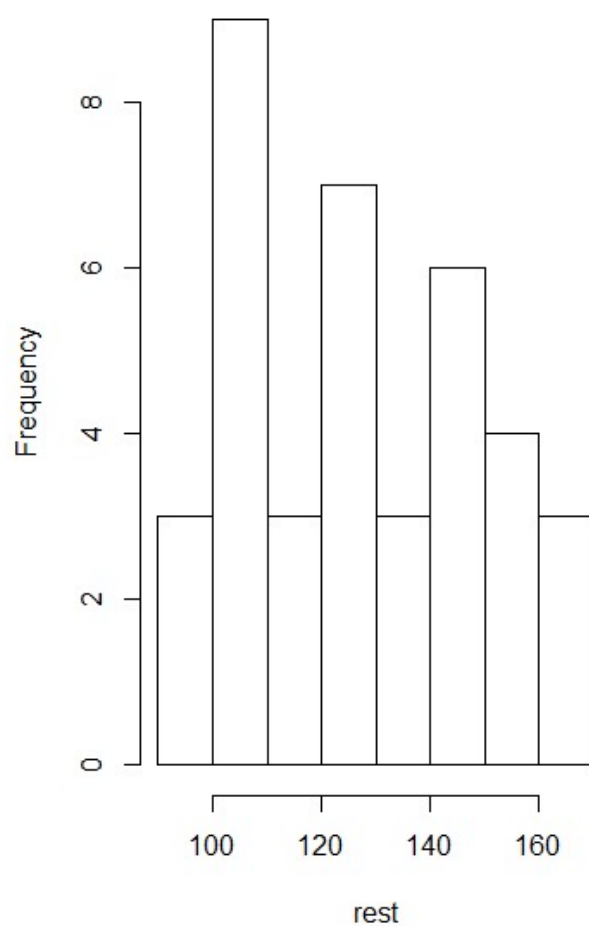
```
hist(rest,nclass=15); hist(rest,nclass=12)
```

```
par(mfrow=c(1,2)); hist(rest,nclass=6)
```

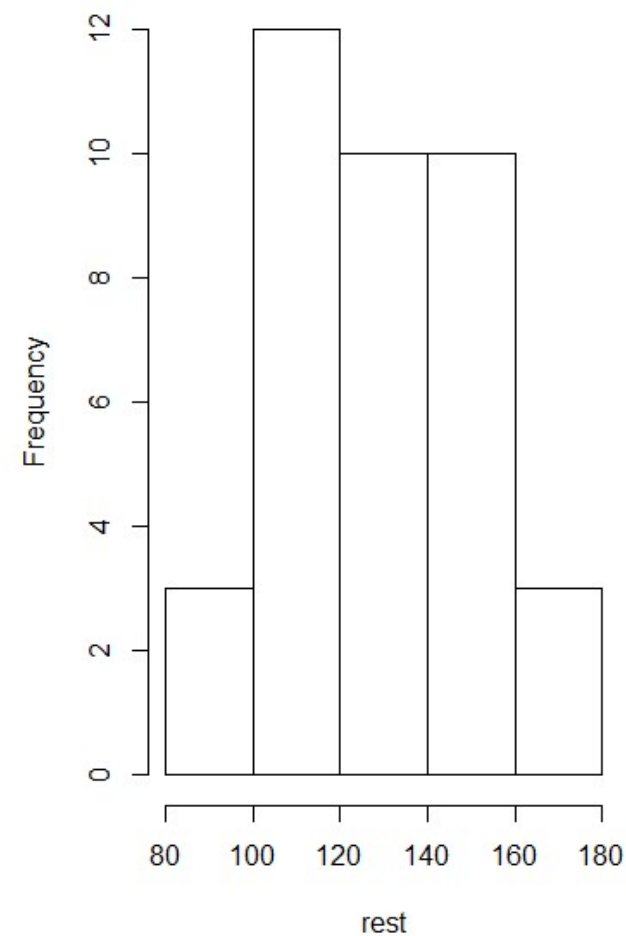
```
hist(rest,nclass=3)
```

Histograma

Histograma 6 classes



Histograma 3 classes



Barplot

A função `barplot()` produz gráfico de barras, onde cada barra representa a medida de cada elemento de um vetor, ou seja, as barras são proporcionais com a “dimensão” do elemento.

#Sintaxe:

```
barplot(x, col=" ", legend.text=" ", xlab=" ", ylab=" ")
```

x - é o vetor ou arquivo de dados;

col="" - define-se a cor de exibição do gráfico de barras;

legend.text="" - legenda do gráfico (o que representa a altura dos gráficos);

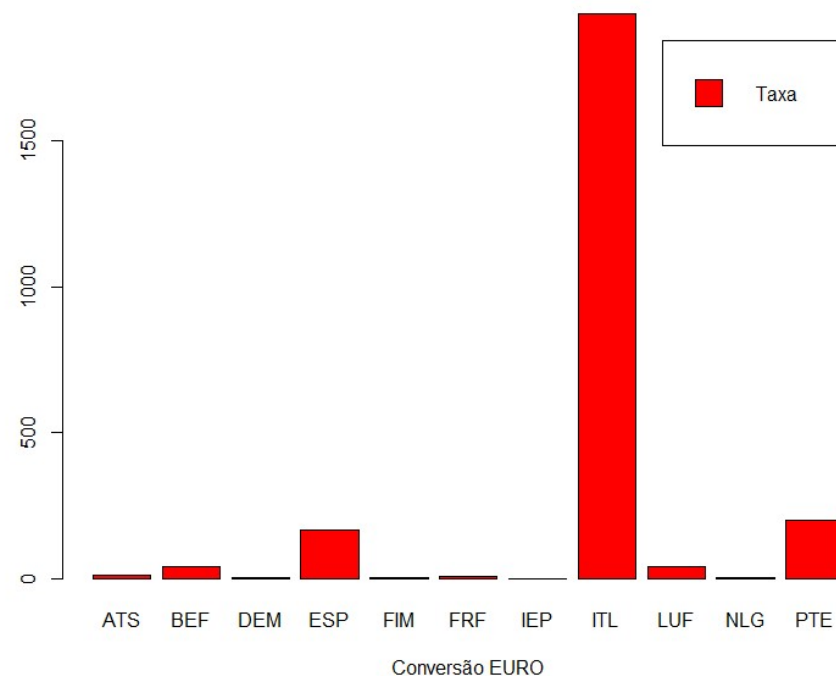
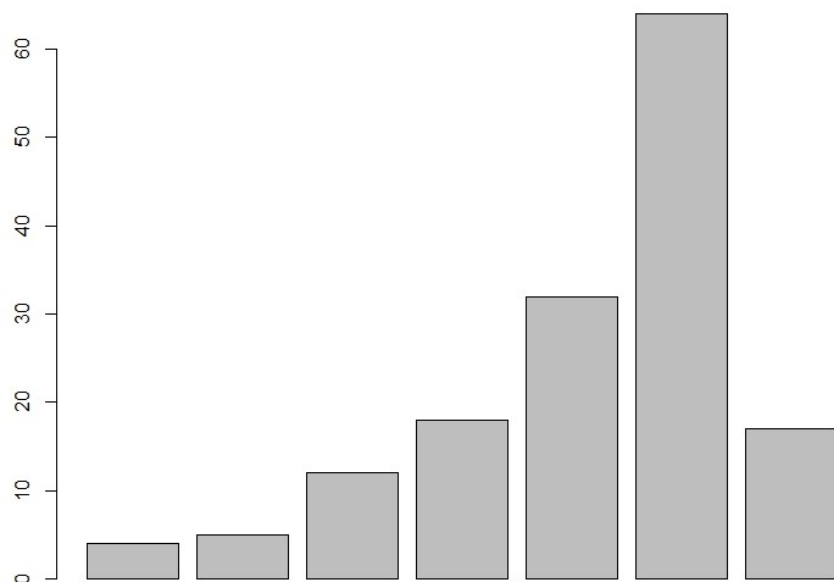
xlab="" e **ylab**="" - nome das grandezas expressas nos eixos x e y, respectivamente.

Barplot

```
x <- c(4, 5, 12, 18, 32, 64, 17)
```

```
barplot(x)
```

```
barplot(euro,xlab="Conversão EURO",col="red", legend.text="Taxa")
```



Boxplot

Boxplot é um gráfico que representa a distribuição de um conjunto de dados com base em parâmetros descritivos (mediana e os quartis). Desta forma o boxplot é útil para revelar o centro, a dispersão e a distribuição dos dados. Em síntese permite avaliar a simetria dos dados, particularmente recomendado para a comparação de dois ou mais conjuntos de dados correspondentes às categorias de uma variável qualitativa.

```
x = c(5,5,5,13,7,11,11,9,8,9)
```

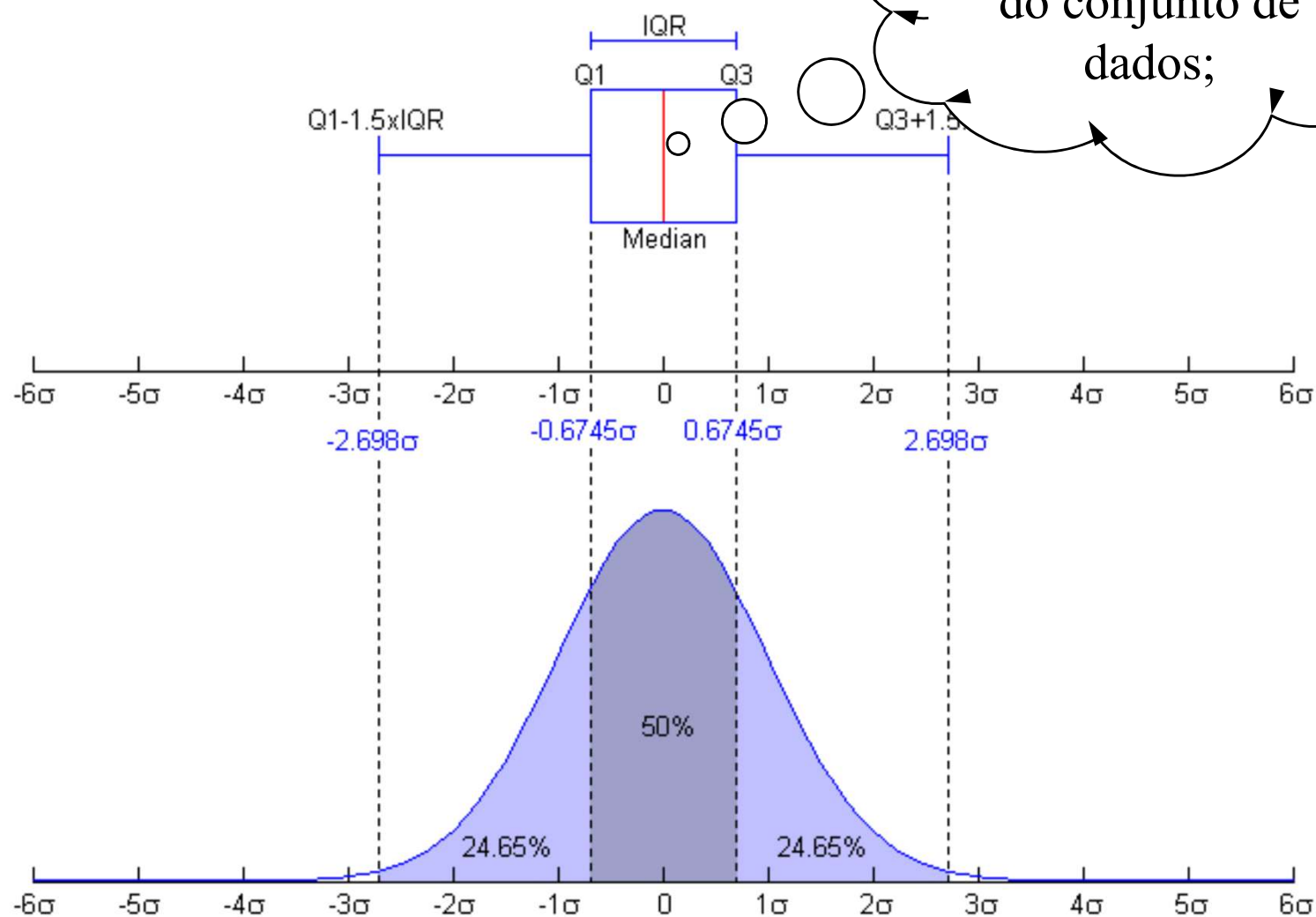
```
y = c(11,8,4,5,9,5,10,5,4,10)
```

```
boxplot(x,y) #plota no mesmo gráfico (comparação)
```

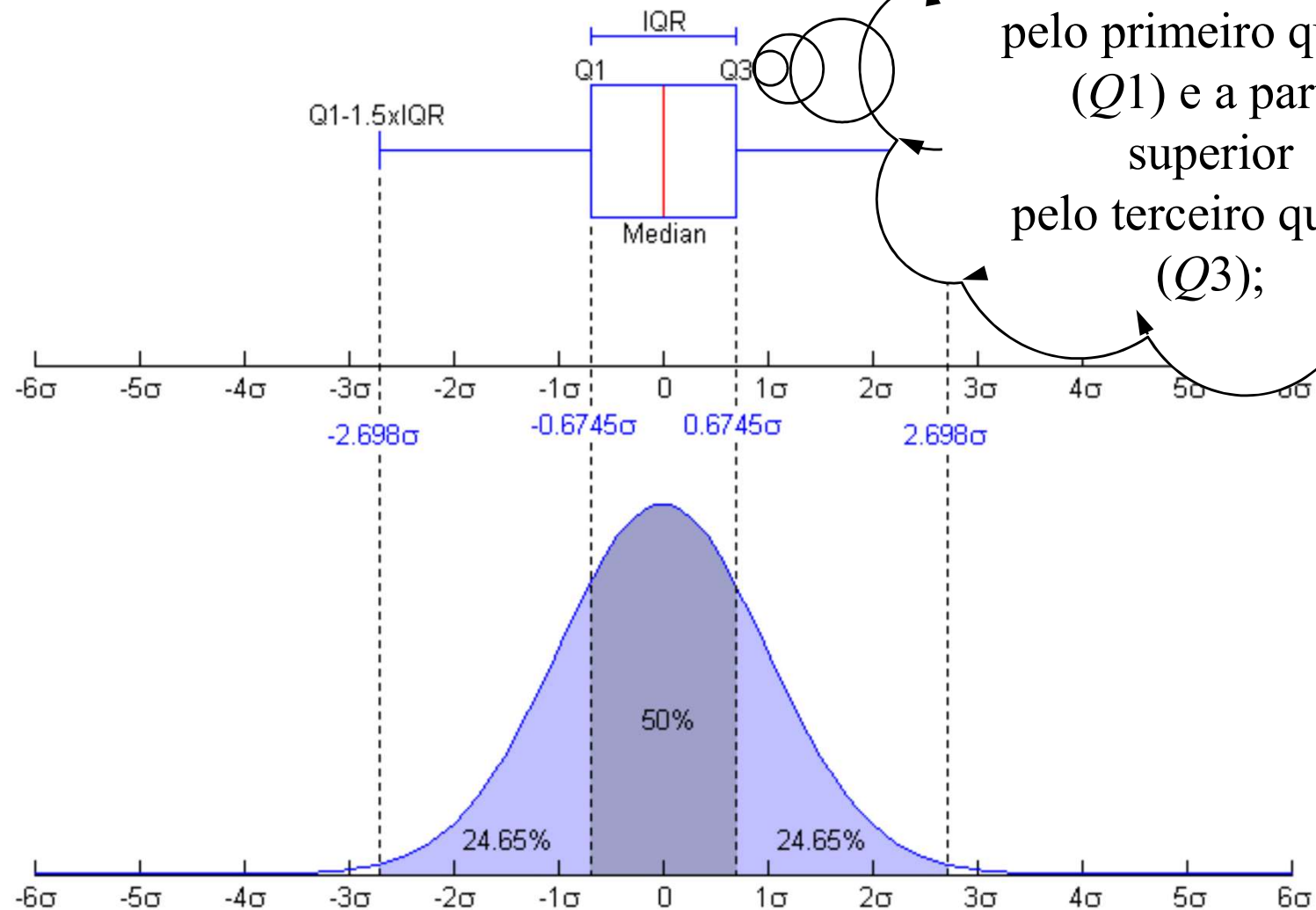
```
boxplot(x); boxplot(y) #plota em gráficos distintos
```


Boxplot

A linha central
marca a mediana
do conjunto de
dados;

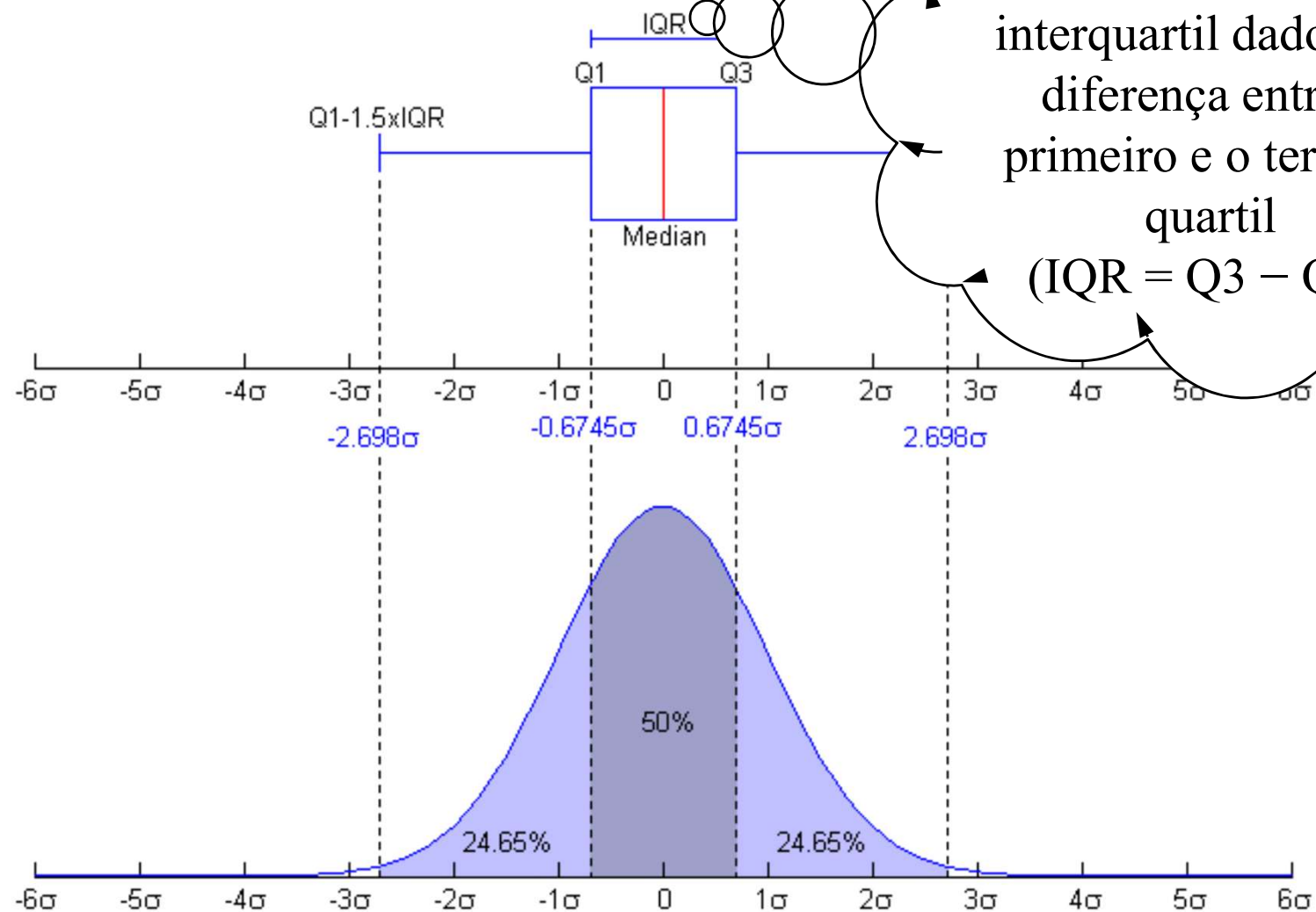


Boxplot



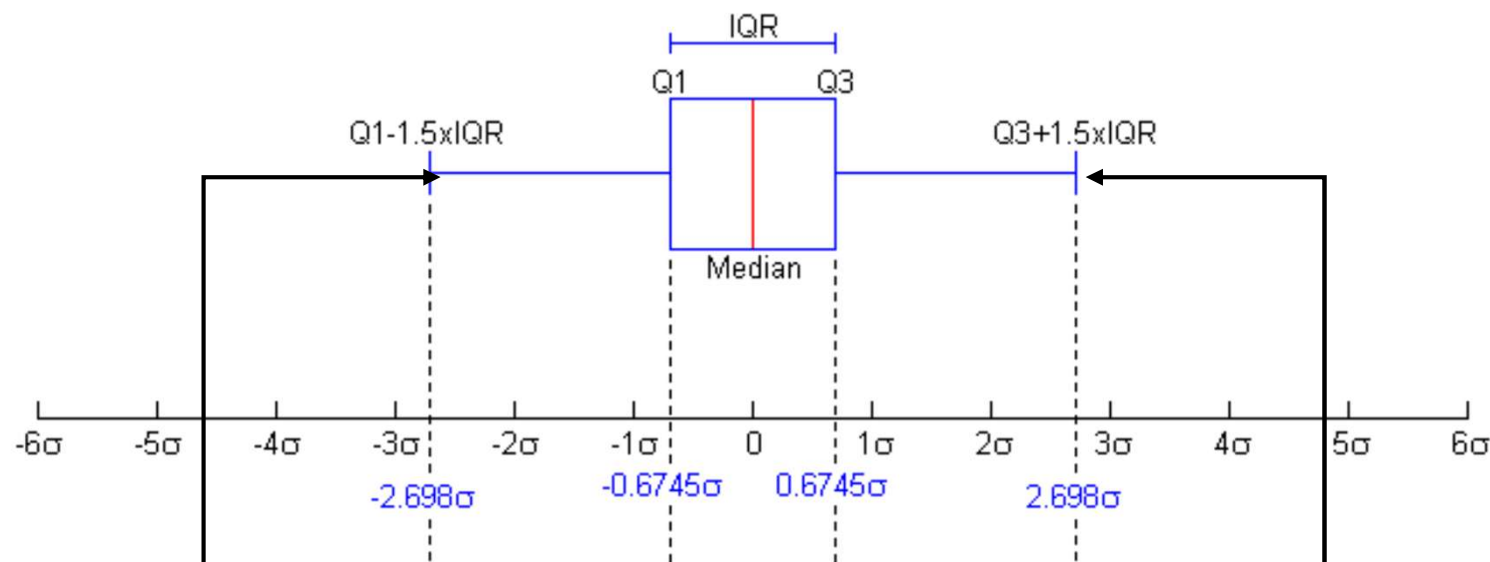
A parte inferior da caixa é delimitada pelo primeiro quartil ($Q1$) e a parte superior pelo terceiro quartil ($Q3$);

Boxplot



O que possibilita verificar o intervalo interquartil dado pela diferença entre o primeiro e o terceiro quartil ($IQR = Q3 - Q1$);

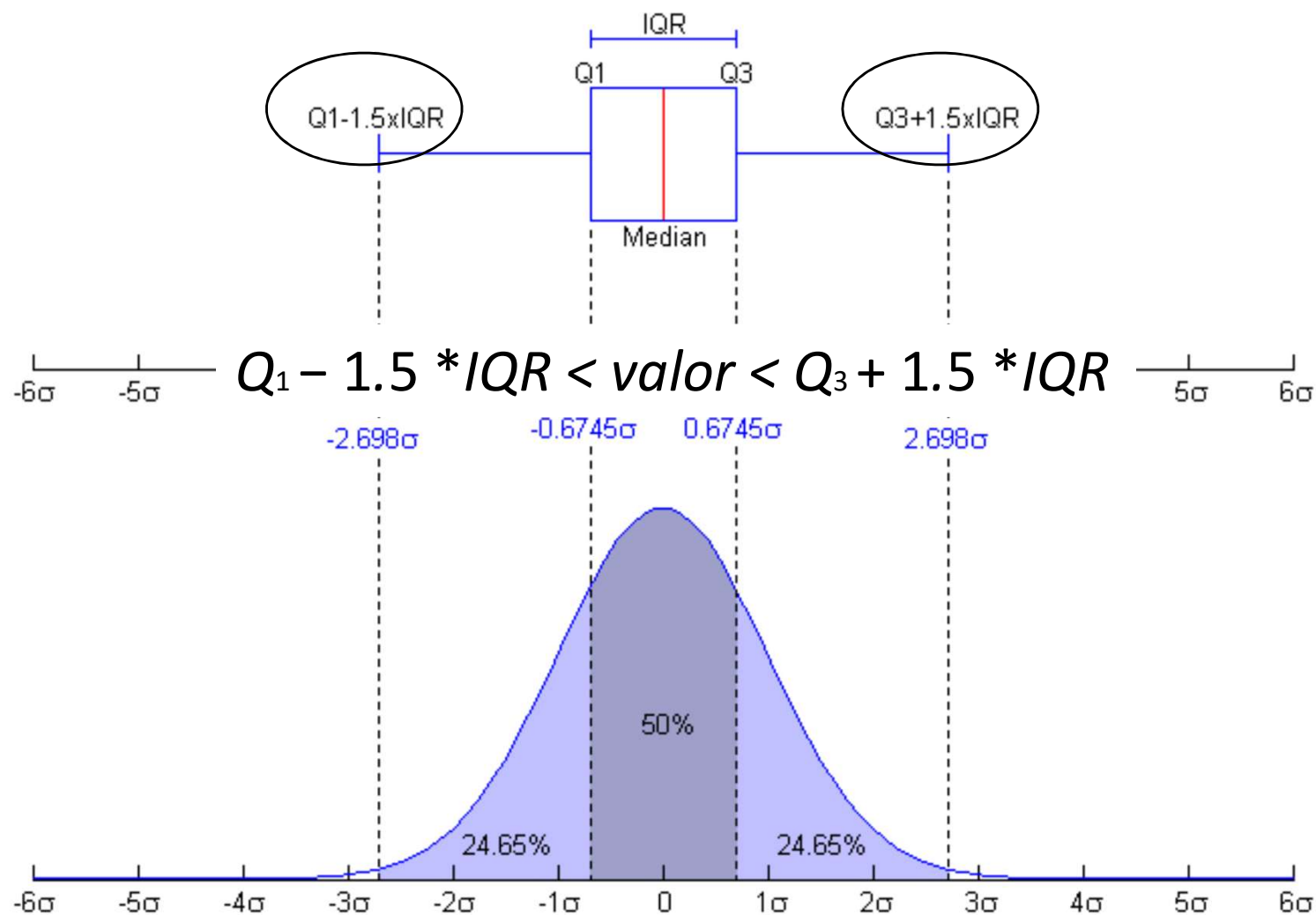
Boxplot



As hastes inferiores e superiores se estendem, respectivamente, do quartil inferior até o menor valor não inferior a $Q_1 - 1.5 \ IQR$ e do quartil superior até o maior valor não superior a $Q_3 + 1.5 \ IQR$;



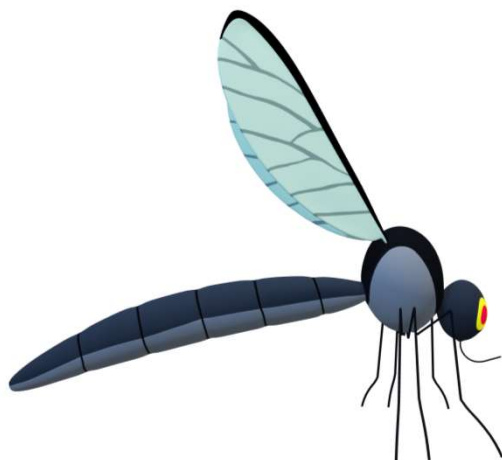
Boxplot



Boxplot - aplicação

Os *boxplots* são gráficos extremamente úteis, especialmente quando temos uma variável categórica associada aos dados.

InsectSprays contido
no pacote datasets



Console			Terminal ×	Jobs ×
~/				
> InsectSprays				
	count	spray		
1	10	A		
2	7	A		
3	20	A		
4	14	A		
5	14	A		
6	12	A		
7	10	A		
8	23	A		
9	17	A		
10	20	A		

Boxplot - aplicação

Os *boxplots* são gráficos extremamente úteis, especialmente quando temos uma variável categórica associada aos dados.

InsectSprays contido
no pacote datasets



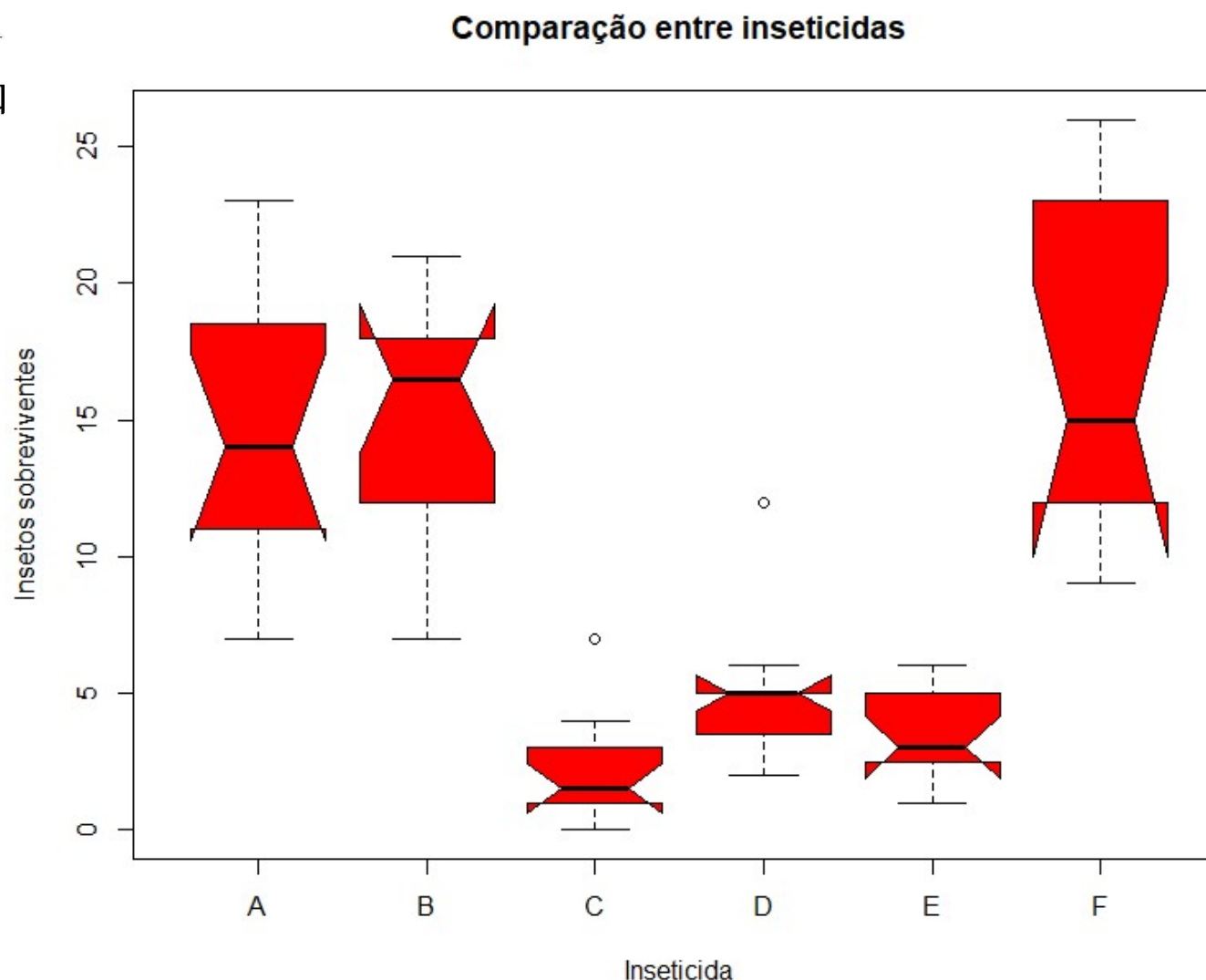
```
boxplot(  
  count ~ spray,  
  data = InsectSprays,  
  xlab = "Inseticida",  
  ylab = "Insetos sobreviventes",  
  main = "Comparação entre inseticidas",  
  col = 2  
)
```

Boxplot - aplicação



A
fu
m
E
c

```
boxplot(
  count~spray
  data = Insect
  xlab = "Inset
  ylab = "Inset
  main = "Con
  col = 2
)
```

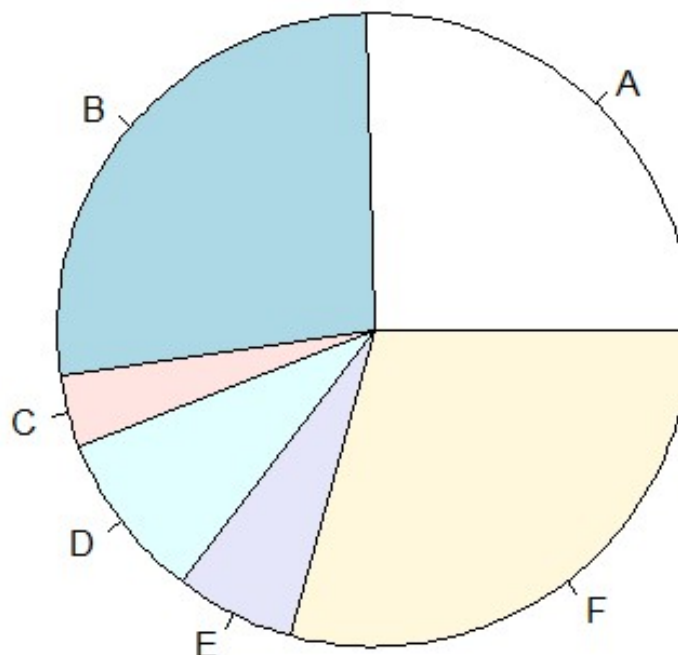


antes é
quanto
eriam.
spray"

Gráfico de barras

Para exibir
o gráfico de
síntese é a r

```
a<-c(0.12, 0.15, 0.18, 0.22, 0.25, 0.28)
names(a)<-c("A", "B", "C", "D", "E", "F")
pie(a,col = c("white", "lightblue", "lightcoral", "lightcyan", "lightblue", "lightyellow"))
mm<-aggregate(a, by = names(a), FUN = sum)
pie(mm[,2], col = c("white", "lightblue", "lightcoral", "lightcyan", "lightblue", "lightyellow"))
```



), é utilizado
e grupos, em

"))
m)

Linguagem R

Prof. José Carmino

E-Mail: prof.carmino@gmail.com

Agenda

Tópico	Descrição
Apresentação do R	Criação interface, sintaxe, tipos de dados, comandos básicos, vetores e matrizes e entrada de arquivos
Programação em R	Estruturas de repetição, decisão
Gráficos	Gráficos de análise descritiva
Estatística descritiva	Medidas de posição e medidas de dispersão
Probabilidade	Definição e axiomas
Variáveis aleatórias	Variáveis aleatórias discretas e contínuas
Inferência estatística	Teste de hipótese de uma e duas amostras
Regressão e correlação linear simples	Determinando a equação linear, coeficiente de correlação e coeficiente de determinação

Estatística descritiva

Os objetivos da estatística descritiva é a organização, apresentação e sintetização dos dados. Desta forma a estatística descritiva descreve e avalia certo grupo de dados, seja uma população, seja uma amostra.

Em se tratando de amostras, o simples uso de estatísticas descritivas não permite tirar quaisquer conclusões ou inferências sobre um grupo maior. Para estabelecimento de inferências ou conclusões sobre um grupo maior (a população) é necessário ir além da Estatística Descritiva.

Há na Estatística Descritiva dois métodos que podem ser usados para a apresentação dos dados: métodos: **gráficos** (envolvendo apresentação gráfica e tabular); e **numéricos** (envolvendo apresentações de medidas de posição e dispersão, entre outras).

Medidas de posição

São as representações estatísticas de uma série de dados orientando-nos quanto à posição da distribuição em relação ao eixo horizontal (eixo "x") do gráfico da curva de frequência.

As medidas de posição mais importantes são as **medidas de tendência central**, no qual se verifica uma tendência dos dados observados a se agruparem em torno dos valores centrais.

Medidas de posição – média aritmética

$$\bar{X} = \sum_{i=1}^n \frac{X_i}{n}$$

\bar{X} é igual ao quociente entre a soma dos valores do conjunto e o número total dos valores

```
x <- c(10, 14, 13, 15, 16, 18, 12)  
mean(x)
```

Medidas de posição - mediana

A mediana de um conjunto de valores, em ordem crescente ou decrescente, é um valor situado de tal maneira que separa o conjunto em dois subconjuntos.

$$Md = \frac{n + 1}{2}$$

Esta formula é para á serie que tiver um número ímpar de termos

$$Md = \frac{\left[\frac{n}{2} + \left(\frac{n}{2} \right) + 1 \right]}{2}$$

Esta formula é para á serie que tiver um número par de termos

```
k <- c(1,3,0,0,2,4,1,2,5)
```

```
g <- c(1,3,0,0,2,4,1,3,5,6)
```

```
median(k) #media aritmética para um conjunto impar de elementos
```

```
median(g) #media aritmética para um conjunto par de elementos
```

Medidas de posição - mediana

Exemplo: Calcule a mediana da série {1, 3, 0, 0, 2, 4, 1, 2, 5}

1. Ordenar a série: {0, 0, 1, 1, 2, 2, 3, 4, 5};
2. $n = 9$ elementos.
3. Pela fórmula $(n + 1) / 2$ é dado por $(9+1) / 2 = 5$;
4. Logo, o quinto elemento da série ordenada será a mediana. Este elemento é o número 2.

Em R temos:

```
impar <- c(1,3,0,0,2,4,1,2,5)  
median(impar)
```


Medidas de posição - mediana

Exemplo: Calcule a mediana da série {1, 3, 0, 0, 2, 4, 1, 3, 5, 6}.

1. Ordenar a série {0, 0, 1, 1, 2, 3, 3, 4, 5, 6};
2. $n = 10$ elementos.
3. Pela fórmula $[(10/2) + (10/2 + 1)]/2$ resultará na realidade (5º termo + 6º termo)/2. Estes termos são 2 e 3, respectivamente.
4. Logo a mediana será $(2+3)/2$, ou seja, $Md = 2,5$

Em R temos:

```
par <- c(1,3,0,0,2,4,1,3,5,6)  
median(par)
```

Medidas de posição - moda

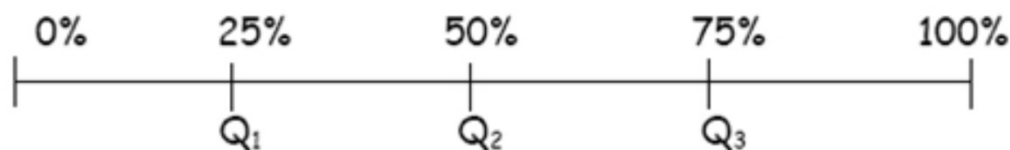
Moda é o valor que ocorre com maior frequência em uma série de valores, vale ressaltar que existem séries que não existe valor modal, tal série é dita amodal.

`table()` – retorna uma lista de elementos e sua contagem

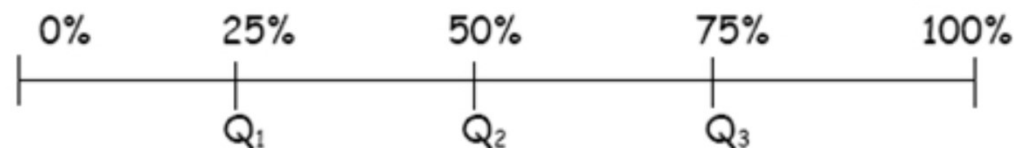
`subset(table(y),table(y)==max(table(y)))` – retorna somente o elemento que está em função do segundo parâmetro.

Medidas de posição - moda

A denominação quartil de uma série, são os valores que dividem a série em quatro partes iguais. O que leva a dizer que são necessários três quartis (Q_1 , Q_2 e Q_3), o que divide a série em quatro partes iguais.



Medidas de posição - moda



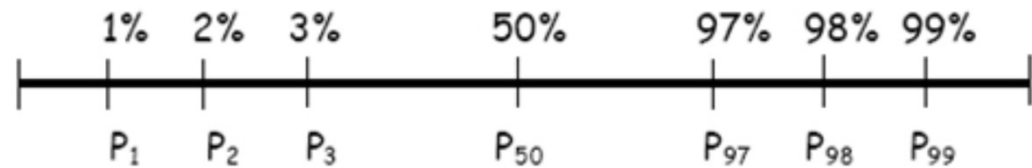
Exemplo: Calcule os quartis da série: {5, 2, 6, 9, 10, 13, 15}

1. Ordene o vetor em crescente, que resulta: {2, 5, 6, 9, 10, 13, 15}.
2. O valor que divide a série em duas partes iguais é o elemento 9, logo a mediana e o quartil 2 (Q_2) é 9.
3. Temos agora {2, 5, 6, 9} e {9, 10, 13, 15} como sendo os dois grupos contendo 50% das informações sobre os dados da série. Para o cálculo do primeiro e do terceiro quartis, basta calcular as medianas dos dois grupos resultantes.
4. Logo em {2, 5, 6, 9} a mediana é 5.5, ou seja, o quartil Q_1 é 5.5 e em {9, 10, 13, 15} a mediana é 11.5, ou seja, o quartil Q_3 é 11.5.

Em R temos: `summary(vetor)`

Medidas de posição – percentis/decis

Percentis são as medidas que dividem a amostra em 100 partes iguais.



```
quartis <- c(48,49,51,50,49,50,50)
cempartes = seq(.01,0.99,.01)
quantile(quartis, cempartes)
```

Decis são as medidas que dividem a amostra em 10 partes iguais.

```
decis <- c(48,49,51,50,49,50,50)
dezpartes = seq(.10,.90,.10)
quantile(decis, dezpartes)
```

Medidas de dispersão amostral

As medidas de dispersão descrevem a variabilidade que ocorre em um conjunto de dados analisado, complementando assim as medidas posição. O que torna as medidas de dispersão elementos fundamentais na caracterização de uma amostra (MELLO; PETERNELLI, 2013).

A primeira medida é a variância que é dada por:

$$s^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1} = \frac{SQD_X}{n - 1}$$

Em R temos:

```
x <- c(1, 2, 3, 4, 5)  
var(x)
```

Medidas de dispersão amostral

Desvio padrão

Definido como a raiz quadrada positiva da variância, o desvio padrão tem grande vantagem sobre a variância, uma vez que é apresentado na mesma unidade de medida dos dados brutos.

Em R temos:

```
x<-c(1,2,3,4,5)
```

```
sd(x)
```

```
sqrt(var(x))
```

Medidas de dispersão amostral

Amplitude Total

É a diferença entre o maior (máximo) e o menor (mínimo) valor de um conjunto de dados. Tem a vantagem de ser calculada de forma rápida e fácil; porém, fornece um número "grosseiro" da variabilidade dos dados, por levar em conta apenas dois valores.

```
x<-c (2,4,5,6,10)
```

```
range(x)
```

```
range(x)[2]-range(x)[1]
```

```
max(x)-min(x)
```


Medidas de dispersão amostral

Variancia (σ^2)

A variância é a medida de dispersão mais empregada geralmente, isto porque leva em consideração a totalidade dos valores da variável estudada, tendo como base os desvios em torno da média aritmética, sendo então um indicador de variabilidade.

$$\sigma^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2 F_i}{n - 1}$$

Em R temos:

```
v <- c(10,11,9,10,10,9,11)
var(v)
```

Medidas de dispersão amostral

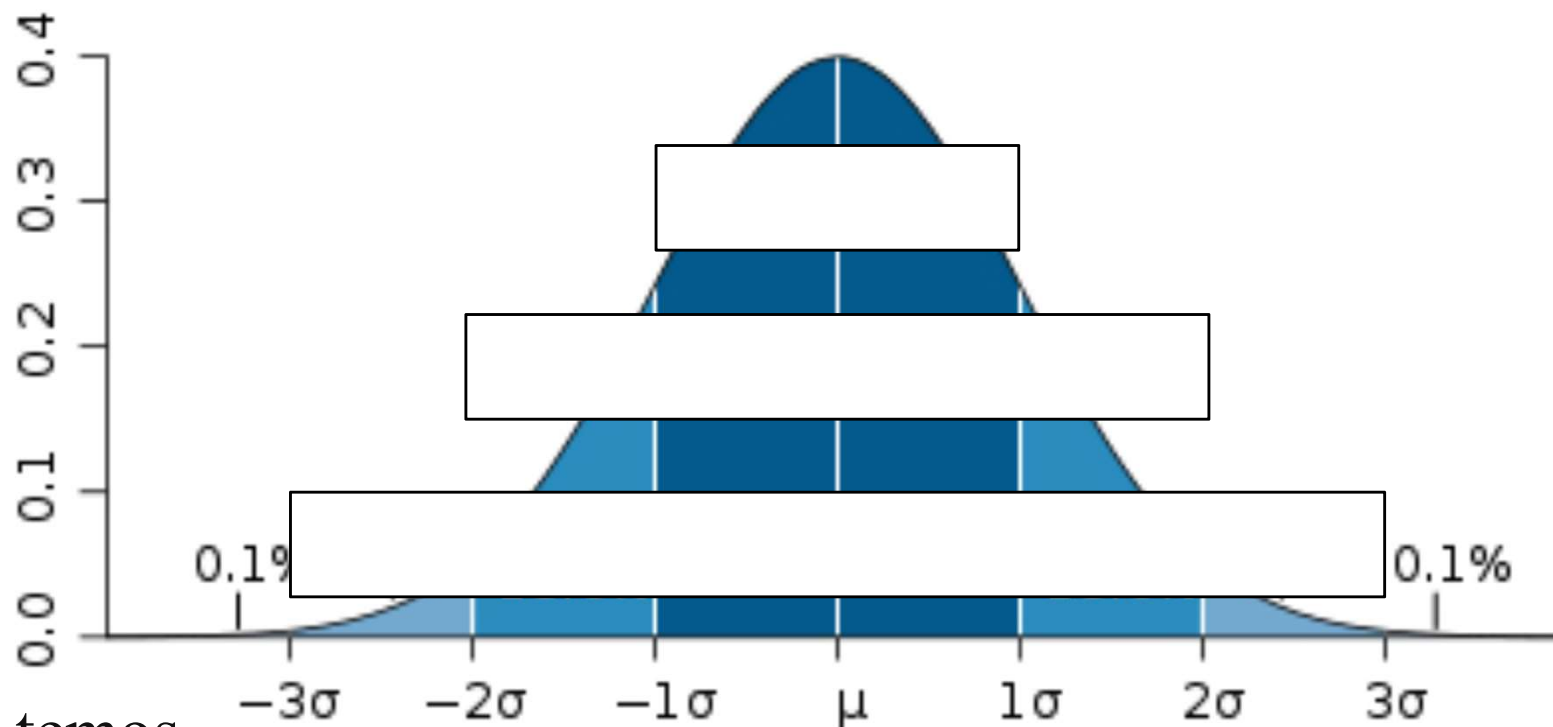
Desvio padrão (σ)

O desvio padrão segue o mesmo raciocínio do cálculo da variância, porém é necessário agora, aproximar a medida de dispersão da variável original. Para isso, calculamos o desvio padrão, que é a raiz quadrada da variância.

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2 F_i}{n - 1}}$$

Medidas de dispersão amostral

O desvio padrão (σ), pode ser representado pela distribuição normal como representado abaixo.



Em R temos
`x<-c(1,2,3,4,5)`
`sd(x); sqrt(var(x))`

Medidas de dispersão amostral

Coeficiente de Variação (CV) é uma medida relativa de dispersão, útil para a comparação em termos relativos do grau de concentração em torno da média de séries distintas.

$$CV = \frac{\sigma}{\bar{x}} 100\%$$

A importância de se estudar o coeficiente de variação é justificada, uma vez que o desvio-padrão é relativo à média, e como duas distribuições podem ter médias diferentes, o desvio destas distribuições não é comparável. Portanto, o coeficiente de variação é uma forma de comparar amostras.

Em R temos

```
v <-c(10,11,9,10,10,9,11); CV <- 100*sd(v)/mean(v)
```

Medidas de dispersão amostral

Exemplo de aplicação adaptado de (MELLO; PETERNELLI, 2013)

Um psicólogo deseja obter informações sobre o grau de dispersão de dados referentes a idade dos frequentadores de um grupo de alcoólicos anônimos. Coletou, portanto, os seguintes dados:

33 17 39 78 29 32 54 22 38 18

Ele deseja saber a variância, o desvio padrão, a amplitude total, o erro padrão da média e o coeficiente de variação de seu conjunto de dados.

Em R temos:

```
x <- c(33, 17, 39, 78, 29, 32, 54, 22, 38, 18)
var(x); sd(x); max(x) - min(x)
sd(x) / sqrt(length(x)) ; sd(x) / mean(x) * 100
```

Medidas de dispersão amostral

Exemplo de aplicação adaptado de (MELLO; PETERNELLI, 2013)

Um psicólogo deseja obter informações sobre o grau de dispersão de dados referentes a idade dos frequentadores de um grupo de alcoólicos anônimos. Coletou, portanto, os seguintes dados:

33 17 39 78 29 32 54 22 38 18

Ele deseja saber a variância, o desvio padrão, a amplitude total, o erro padrão da média e o coeficiente de variação de seu conjunto de dados.

Em R temos:

```
x <- c(33, 17, 39, 78, 29, 32, 54, 22, 38, 18)
var(x); sd(x); max(x) - min(x)
sd(x) / sqrt(length(x)) ; sd(x) / mean(x) * 100
```