

## Aplicações Computacionais em Exploração e Análise de Dados

### Introdução ao R

---

---

---

---

---

---

---

---

### Primeiro Passo

- Copiem a pasta cursoR que está em Professor para a Área de Trabalho.
- Nesta pasta, está todo o material que vamos utilizar para a introdução ao R.
  - Slides
  - Banco de dados
  - etc...

---

---

---

---

---

---

---

---

### Introdução

- O R é um programa gratuito, de código aberto e livremente distribuído que proporciona análises estatísticas de alta qualidade.
- Pelo fato de ser usada uma linguagem de programação pode existir certa dificuldade até que se familiarize com os comandos mais comuns.
- Detalhes sobre o projeto, colaboradores, documentação e diversas outras informações podem ser encontradas na página oficial do projeto em <http://www.r-project.org>.

---

---

---

---

---

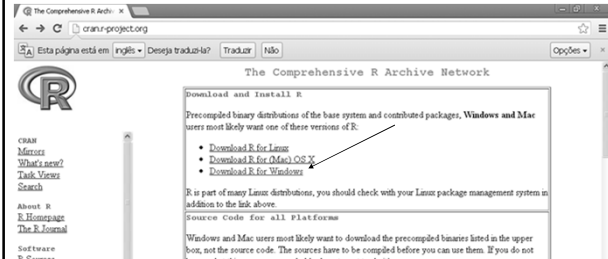
---

---

---

# Download do Programa

- <http://cran.r-project.org>



---

---

---

---

---

---

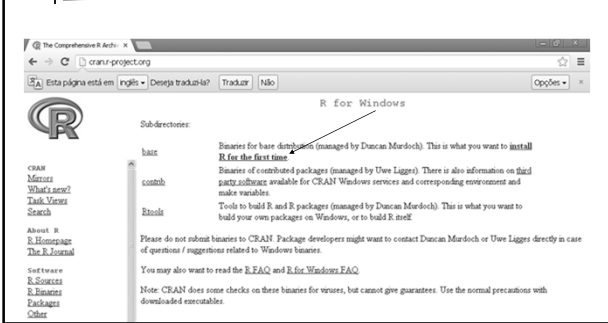
---

---

---

---

# Download do Programa



---

---

---

---

---

---

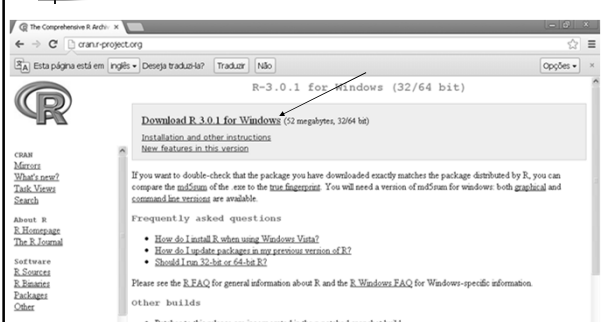
---

---

---

---

# Download do Programa



---

---

---

---

---

---

---

---

---

---

## Introdução

- Abram o R no seu computador!

---

---

---

---

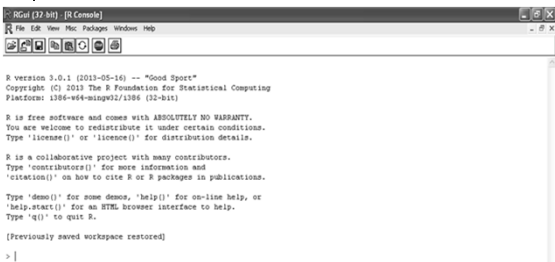
---

---

---

---

## R



```
R version 3.0.1 (2013-05-16) -- "Good Sport"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: i386-w64-mingw32/x86_64

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> |
```

---

---

---

---

---

---

---

---

## Introdução

- Para que o iniciante no R se torne apto a colocar em prática o que foi apresentado durante as aulas de Elementos de Estatística, é preciso uma pequena introdução de como usar o R.

---

---

---

---

---

---

---

---

## Considerações Iniciais

- Há uma distinção entre minúsculas e MAIÚSCULAS.
- Lembre que R usa um ponto "." em vez de vírgula "," quando há números com casas decimais.
- Se precisar importar dados que usam vírgulas em vez de pontos, troque na planilha as vírgulas por pontos usando Localizar e Substituir, se não, os dados não serão reconhecidos como números.

---

---

---

---

---

---

---

---

## Help

- Help no R é bastante completo e auto-suficiente.

*Comandos de ajuda do R*

```
> help.start() inicia documentação na forma de arquivos html visualizados no seu browser
> help (tópico) inicia uma janela de ajuda sobre tópico
> ?(tópico) a mesma coisa
```

---

---

---

---

---

---

---

---

## Operações básicas

- No R podemos resolver todas as operações básicas vistas freqüentemente.

- Adição/Subtração [+/-]
- Multiplicação [\*]
- Divisão [/]
- Potência [\*\* ou ^]
- Raiz quadrada [sqrt()]
- Exponencial [exp(), log()]

obs.: A função log() no R, é equivalente ao log na base e. Caso queria calcular na base 10, basta colocar log10().

---

---

---

---

---

---

---

---

## Comandos

- Para solicitar uma tarefa do R podemos digitar uma linha de comando no console.

---



---



---



---



---



---



---

## Exemplos

```
> 5+4      #adição      > sqrt(121) #raiz
[1] 9
[1] 11

> 6-2      #subtração   > exp(0)    #exponencial
[1] 4
[1] 1

> 7*3      #multiplicação > log(1)   #log na base e
[1] 21
[1] 0

> 81/9     #divisão    > log10(1) #log na base 10
[1] 9
[1] 0

> 2^2      #potência
[1] 4
```

---



---



---



---



---



---



---

## Sintaxe - Alguns Pontos Importantes

|     | Uso  |
|-----|--|
| ( ) | Para funções: como em $f(x)$<br>Priorização operações: como em $3*(4+2)$ |
| [ ] | Para indexação: Como em <code>vetor[3]</code>                            |

---



---



---



---



---



---



---

## Comando úteis

- Trabalho 'limpo'.
  - Para limpar o console usa-se CTRL + L.
- Apaga tudo!
  - `rm(list=ls(all=TRUE))`.

---

---

---

---

---

---

---

---

## Armazenando dados

- Tipos de dados:
  - Numéricos
  - Caracteres: compostos por letras ou palavras.
- Quando os dados são armazenados, eles são chamados de objeto.
- Para armazenar um objeto basta utilizar o símbolo "<-" ou "=".

---

---

---

---

---

---

---

---

## Exemplo

```
> x<-4      #O valor 4 é armazenado no objeto x
> x         #Exibe o valor do objeto
[1] 4
```

---

---

---

---

---

---

---

---

## Armazenando dados

- Algumas regras devem ser seguidas para dar nome a um objeto.
- O nome do objeto precisa começar com uma letra, não pode ser um número, não pode conter símbolos referentes a funções ou nome de funções e a "seta" deve estar sempre apontada para o nome do objeto.

---

---

---

---

---

---

---

---

## Exemplos não válidos

```
24e<-4 #O nome do objeto começa com números
12<-2 #O nome do objeto é um número
e*2<- #O nome do objeto contém o símbolo da multiplicação
x->5 #A seta está para o lado errado
```

---

---

---

---

---

---

---

---

## Apagar Objeto

- Comando `rm(objeto)`.
- Cuidado! Não tem undo.

---

---

---

---

---

---

---

---

## Armazenando dados

- As formas mais utilizadas para armazenar dados são vetores, matrizes e data-frames.

---

---

---

---

---

---

---

---

## Vetor

- Pode ser um vetor numérico ou de caractere.
- A função `c()` é utilizada na criação do vetor.
- Sempre que armazenar um vetor de caractere é necessário colocar aspas em palavra do vetor.

---

---

---

---

---

---

---

---

## Exemplos

```
> x<-c(1,2,3,4)      #Vetor numérico de nome x
> x                  #Exibe o valor do vetor armazenado no objeto x
[1] 1 2 3 4
> w<-c("A","B","A","A") #Vetor de caracteres de nome w
> w                  #Exibe o valor do vetor armazenado no objeto w
[1] "A" "B" "A" "A"
```

---

---

---

---

---

---

---

---



## Duas funções importantes

- Função seq().
- Função rep().
  
- A função seq() lista a seqüência de números que quiser, no intervalo que quiser. Já a função rep() lista números repetidos, quantos números quiser com quantas repetições quiser.

---



---



---



---



---



---



---

## Exemplos

```
> seq(1,100,1) #Sequência de números de 1 até 100, com intervalo de 1 número
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
[19] 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
[37] 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
[55] 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
[73] 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
[91] 91 92 93 94 95 96 97 98 99 100

> seq(1,100,2) #Sequência de números de 1 até 100, com intervalo de 2 números
[1] 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
[26] 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

---



---



---



---



---



---



---

## Exemplos

```
> seq(1,100,10) #Sequência de números de 1 até 100, com intervalo de 10
[1] 1 11 21 31 41 51 61 71 81 91

> seq(1,20,1) #Sequência crescente de 1 até 20
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

> seq(20,1,-1) #Sequência decrescente de 20 até 1
[1] 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
```

---



---



---



---



---



---



---

## Exemplos

```
> rep(1,10) #Número 1 repetido 10 vezes
[1] 1 1 1 1 1 1 1 1 1 1

> rep(1:5,2) #Sequência de 1 até 5, repetida 2 vezes
[1] 1 2 3 4 5 1 2 3 4 5

> rep(c(1,4),3) #Números 1 e 4 repetidos 3 vezes, alternadamente
[1] 1 4 1 4 1 4

> c(rep(1,3),rep(4,3)) #Números 1 e 4 repetidos 3 vezes, sem alternar
[1] 1 1 1 4 4 4
```

## Ferramenta útil

- Seleção de elementos dentro de um vetor.
- Para isso são utilizados os colchetes e indicados as posições ou os próprios elementos.

## Operadores lógicos

| Símbolo | Função                         |
|---------|--------------------------------|
| <       | Menor que                      |
| >       | Maior que                      |
| <=      | Menor que ou igual a           |
| >=      | Maior que ou igual a           |
| = =     | Igual a                        |
| ! =     | Não igual a                    |
| &       | E (para combinar expressões)   |
|         | Ou (para combinar expressões)  |
| !       | Não (para combinar expressões) |

## Exemplos

```
> x<-c(45,23,89,43)

> x[1] #Seleciona o primeiro elemento do vetor x
[1] 45

> x[4] #Seleciona o quarto elemento do vetor x
[1] 43

> x[x==23] #Seleciona o elemento que tem valor 3 dentro do vetor x
[1] 23
```

---



---



---



---



---



---



---



---

## Exemplos

- $x[x < 40]$
- $x[x > 40]$
- $x[x \geq 43]$
- $x[x > 43 \ \& \ x < 50]$

---



---



---



---



---



---



---



---

## Exemplos

```
> x[c(1,2,3)]
[1] 45 23 89
```

**Seleciona os elementos que estão na primeira, segunda e terceira posição no vetor x.**

```
> x[-c(2,3)]
[1] 45 43
```

**Seleciona todos os elementos menos o da segunda e terceira posição.**

```
> x[x==max(x)]
[1] 89
```

**Seleciona o maior elemento do vetor x.**

---



---



---



---



---



---



---



---

## Comando sort()

- Ordena os elementos do vetor.
- Ordenação crescente (default):
  - `sort(x)`
- Ordenação decrescente:
  - `sort(x,decreasing=T)`

---

---

---

---

---

---

---

---

## Exemplos

- `sort(x)`  
[1] 23 43 45 89
- `sort(x,decreasing=T)`  
[1] 89 45 43 23
- `Z<-c("B","A","D")`
  - `sort(Z)`  
[1] "A" "B" "D"

---

---

---

---

---

---

---

---

## Operações básicas nos vetores

- Ao somar, subtrair, dividir ou multiplicar dois vetores (numéricos), observamos que cada coordenada de um vetor é somada, subtraída, dividida ou multiplicada pela coordenada correspondente do outro vetor.

---

---

---

---

---

---

---

---

## Exemplos

```
> x+y #Soma dos vetores x,y
[1] 7 9 6 3 0 8 11

> x+y=(1+6,2+7,3+3,4+(-1),-5+5,6+2,7+4)

> sum(x,y) #Somatório dos vetores x,y
[1] 44

> sum(x,y)=(1+2+3+4+(-5)+6+7+6+7+3+(-1)+5+2+4)
```

---



---



---



---



---



---



---



---

## Matriz

- Matriz é um conjunto de vetores numéricos.
- Comando `matrix()` cria uma matriz a partir de uma seqüência de números, sendo esta seqüência atribuída às colunas da matriz a ser formada (preenchimento da matriz é feito da primeira coluna até a última).

---



---



---



---



---



---



---



---

## Exemplos

- Vamos criar uma matriz a partir da seqüência de números de 1 a 9, com 3 linhas e 3 colunas.

```
> matrix(c(1:9),nrow=3)
      [,1] [,2] [,3]
[1,]  1   4   7
[2,]  2   5   8
[3,]  3   6   9

> matrix(c(1:9),ncol=3)
      [,1] [,2] [,3]
[1,]  1   4   7
[2,]  2   5   8
[3,]  3   6   9
```

---



---



---



---



---



---



---



---

## Exemplos

```
> matrix(c(1:9), ncol=3, byrow=T)
      [,1] [,2] [,3]
[1,]  1   2   3
[2,]  4   5   6
[3,]  7   8   9

> matrix(c(1:9), nrow=3, byrow=T)
      [,1] [,2] [,3]
[1,]  1   2   3
[2,]  4   5   6
[3,]  7   8   9
```

A matriz foi preenchida da primeira linha até a última.

---

---

---

---

---

---

---

---

## Exemplos

```
> x<-c(1,2,3,4,-5,6,7,8)

> x
[1] 1 2 3 4 -5 6 7 8

> matrix(x, byrow=T, nrow=3) #Matriz formada pelo vetor x, por linhas
      [,1] [,2] [,3]
[1,]  1   2   3
[2,]  4  -5   6
[3,]  7   8   1

> matrix(x, byrow=F, nrow=3) #Matriz formada pelo vetor x, por colunas
      [,1] [,2] [,3]
[1,]  1   4   7
[2,]  2  -5   8
[3,]  3   6   1
```

---

---

---

---

---

---

---

---

## Matriz

- Outros comandos que também podem ser usados para criação de matrizes são os comandos cbind ou rbind.
- cbind: cria matrizes por colunas.
- rbind: cria matrizes por linhas.

---

---

---

---

---

---

---

---

## Exemplo

```
> x<-c(1,2,3,4,5,6)
> y<-c(7,8,9,10,11,12)

> x
[1] 1 2 3 4 5 6

> y
[1] 7 8 9 10 11 12

> e<-cbind(x,y) #Matriz formada pelos vetores x e y

> e
  x y
[1,] 1 7
[2,] 2 8
[3,] 3 9
[4,] 4 10
[5,] 5 11
[6,] 6 12

> class(e) #Com esse comando podemos confirmar que é matriz
[1] "matrix"
```

---

---

---

---

---

---

---

---

## Ferramenta útil

```
> e[1,] #Retirar linha 1, para outras linhas é só mudar o número
x y
1 7

> e[,1] #Retirar coluna 1, para outras colunas só mudar o número
[1] 1 2 3 4 5 6
```

---

---

---

---

---

---

---

---

## Data-frame

- Data-frame é muito parecido com a matriz, a diferença é que enquanto na matriz só podem haver números, no data-frame podem ter colunas de caracteres além das colunas numéricas.
- Uma das formas de armazenar dados, onde cada coluna é uma variável e cada linha é o indivíduo (ou unidade).

---

---

---

---

---

---

---

---

## Exemplo

```
> idade<-c(25,29,35,33,31,36)
> status<-c("solteiro","casado","casado","solteiro","casado","casado")
> nfil<-c(1,1,2,1,1,3)
> idade
[1] 25 29 35 33 31 36
> status
[1] "solteiro" "casado" "casado" "solteiro" "casado" "casado"
> nfil
[1] 1 1 2 1 1 3
```

## Exemplo

```
> data.frame(idade,status,nfil)
  idade status nfil
1    25 solteiro  1
2    29  casado  1
3    35  casado  2
4    33 solteiro  1
5    31  casado  1
6    36  casado  3
```

## Data-frame

- Uma maneira diferente de criar um data-frame é utilizando a função `edit(data.frame())`.
- Ao usar esta função, uma janela, parecida com uma planilha do Excel, é aberta para digitar as informações do banco de dados.
- Para colocar o nome da variável, basta clicar em cima do "var1", que uma janela é aberta para escrever o nome da variável e dizer se é numérica ou de caractere.



## Exemplo

```
edit(data.frame())
```

R Editor de dados

Arquivo Editar Ajuda

|   | var1 | var2 | var3 | var4 | var5 |
|---|------|------|------|------|------|
| 1 |      |      |      |      |      |
| 2 |      |      |      |      |      |
| 3 |      |      |      |      |      |

---

---

---

---

---

---

---

---

---

---

## Banco de dados no R

- Há ainda um outro caminho para se ter um banco de dados no R.
- A função `read.csv2(file="local do arquivo/nome do arquivo.csv")` pode ler arquivos com a extensão `.csv` que estiverem salvos no computador.

---

---

---

---

---

---

---

---

---

---

## Banco de dados no R

- Caso tenha digitado um banco de dados no Excel, basta salvar com a extensão `.csv` que ele poderá ser lido no R.
- E para você utilizar os dados que foram lidos, basta usar a função `attach(nome do objeto)`.

---

---

---

---

---

---

---

---

---

---

## Exemplo

- `quest <- read.csv2(file="C:/Users/ESTATÍSTICA/Desktop/cursoR/quest.csv")`
- `attach(quest)`
- Turma
- Sexo
- Idade

---

---

---

---

---

---

---

---

## Técnicas Estatísticas

- Agora que já foi feita uma introdução do R, já é possível utilizá-lo para colocar em prática o que foi estudado em Elementos de Estatística.

---

---

---

---

---

---

---

---

## Funções mais utilizadas

- Medidas de posição.
- Medidas de dispersão.
- Medidas de assimetria.
  
- Usando o R é possível realizar cálculos de forma rápida e prática, facilitando a observação dos resultados do banco de dados.

---

---

---

---

---

---

---

---

| Função        | Utilização                     |
|---------------|--------------------------------|
| Sum(x)        | Somatório dos elementos de x   |
| prod(x)       | Produtório dos elementos de x  |
| Max(x)        | Elemento máximo em x           |
| Min(x)        | Elemento mínimo em x           |
| range(x)      | Elemento máximo e mínimo em x  |
| length(x)     | Número total de elementos em x |
| mean(x)       | Média de x                     |
| median(x)     | Mediana de x                   |
| Var(x)        | Variância de x                 |
| sd(x)         | Desvio padrão de x             |
| cor(x,y)      | Correlação entre x e y         |
| quantile(x,p) | Quantil                        |

---

---

---

---

---

---

---

---

---

---

## Exemplos

- Para exemplificar cada uma das funções usaremos um banco de dados que mostra o sexo, o peso e altura de uma determinada turma de alunos.
- `dados <- read.csv2(file="C:/Users/ESTATÍSTICA/Desktop/corsoR/turma.csv")`
- `attach(dados)`
- Altura
- Peso
- Sexo

---

---

---

---

---

---

---

---

---

---

## Função length(x)

- Usamos esta função para calcular o tamanho da amostra.
- Utilizada no banco de dados, podemos descobrir o número de variáveis.

---

---

---

---

---

---

---

---

---

---

## Exemplos

```
> length(dados)      #Número de variáveis observadas
[1] 3

> length(Peso)       #Número de pesos observados, que é o total da amostra
[1] 20
```

---



---



---



---



---



---



---

## Funções min(x), max(x) e range(x)

- Usamos a função min(x) para achar o menor valor da variável quantitativa, a função max(x) para achar o maior valor da variável quantitativa e range(x) para achar, ao mesmo tempo, o menor e o maior valor da variável quantitativa observada.

---



---



---



---



---



---



---

## Exemplos

```
> min(Peso)          #Menor peso observado
[1] 68

> max(Peso)          #Maior peso observado
[1] 73

> range(Peso)        #Menor e maior valores observados
[1] 68 73
```

---



---



---



---



---



---



---

## Funções sum(x) e prod(x)

- Essas funções, basicamente, somam e multiplicam, respectivamente, os valores da variável observada.

---

---

---

---

---

---

---

---

## Exemplo

```
> sum(Peso)           #Soma de todos os valores da variável peso  
[1] 1396
```

---

---

---

---

---

---

---

---

## Exemplo - cálculo da média

```
> sum(Peso)/length(Peso)  #Média  
[1] 69.8
```

---

---

---

---

---

---

---

---

## Função mean(x)

- Usamos a função mean(x) para achar a média da variável observada.

---

---

---

---

---

---

---

---

## Exemplo

```
> mean(Peso)      #Média do peso  
[1] 69.8  
  
> mean(Altura)   #Média da altura  
[1] 160.05
```

---

---

---

---

---

---

---

---

## Função mean(x)

- Também podemos encontrar a média somente para alguns valores da variável.
- Neste caso, temos uma forma de especificar nossa escolha.

---

---

---

---

---

---

---

---

## Exemplo

Média do peso dos alunos de sexo feminino.

```
> mean(Peso[Sexo=="F"])      #Média do peso dos alunos do sexo feminino
[1] 70
```

---

---

---

---

---

---

---

---

## Função median(x)

- Usamos a função median(x) para achar a mediana da variável observada.

---

---

---

---

---

---

---

---

## Exemplos

```
> median(Peso)      #Mediana do peso
[1] 69.5

> median(Altura)    #Mediana da altura
[1] 160
```

Mediana do peso dos alunos do sexo masculino.

```
> median(Peso[Sexo=="M"])  #Mediana do peso dos alunos do sexo masculino
[1] 69.5
```

---

---

---

---

---

---

---

---

## Funções var(x) e sd(x)

- Usamos as funções var(x) e sd(x) para achar, respectivamente, a variância e o desvio padrão da variável observada.

---

---

---

---

---

---

---

---

## Exemplos

```
> var(Peso)           #Variância do peso
[1] 1.957895

> sd(Altura)         #Desvio padrão da altura
[1] 1.959457

> var(Peso[Sexo=="F"]) #Variância do peso do alunos do sexo feminino
[1] 2.181818

> sd(Altura[Sexo=="m"]) #Desvio padrão da altura dos alunos do sexo masculino
[1] 2.604940
```

---

---

---

---

---

---

---

---

## Função quantile(x,p)

- Usamos a função quantile(x,p) para calcular percentil, onde x é a variável observada e p é uma probabilidade que vai de 0 a 1.

---

---

---

---

---

---

---

---



## Exemplo

```
> quantile(Peso,0.7)      #Percentil 70
70%
70
```

Comparem

➤ `quantile(Peso,0.5)`

➤ `Median(Peso)`

---

---

---

---

---

---

---

---

## Função `cor(x,y)`

- Usamos essa função quando queremos observar a relação entre duas variáveis quantitativas.

---

---

---

---

---

---

---

---

## Exemplos

```
> cor(Peso,Altura)      #Correlação entre peso e altura
[1] 0.2149977
```

---

---

---

---

---

---

---

---

## Comentários

- `cor(x,y,method='pearson')` : default
- `cor(x,y,method='spearman')`
- `cor.test(x,y)`
- `cor.test(x,y,method='pearson')` : default
- `cor.test(x,y,method='spearman')`

---

---

---

---

---

---

---

---

## Gráfico de dispersão

- Para realizar o gráfico de dispersão no R, basta utilizarmos a função `plot(x,y)`, onde x e y são os vetores onde estão armazenados os valores das variáveis quantitativas.

---

---

---

---

---

---

---

---

## Exemplo

```
> plot(Peso,Altura) #Gráfico de dispersão
```

---

---

---

---

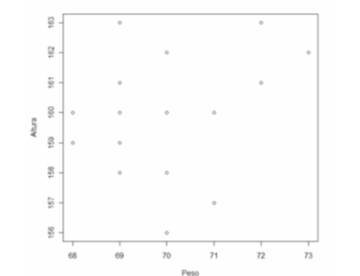
---

---

---

---

## Gráfico de dispersão




---

---

---

---

---

---

---

---

---

---

## Função summary()

- Usada para variáveis quantitativas. Essa função dá um resumo de algumas funções que foram aprendidas anteriormente: `min(x)`, `max(x)`, `median(x)`, `mean(x)`, `quantile(x,0.25)` e `quantile(x,0.75)`.

---

---

---

---

---

---

---

---

---

---

## Exemplos

```
> summary(Peso)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 68.00  69.00   69.50   69.80  70.25   73.00

> summary(Altura)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 156.0  159.0   160.0   160.1  161.2   163.0
```

---

---

---

---

---

---

---

---

---

---

## Tabelas

- Utilizamos as tabelas para observar de forma mais rápida o comportamento de certa variável.
- Deve-se ter atenção para a diferença entre as tabelas de variáveis qualitativas e quantitativas discretas para as quantitativas contínuas.
- Para as variáveis qualitativas e quantitativas discretas basta usar a função `table(x)`.

---

---

---

---

---

---

---

---

## Exemplos

```
> table(Sexo)      #Tabela para variáveis qualitativas e quantitativas discretas
Sexo
 f m
12 8
```

---

---

---

---

---

---

---

---

## Tabelas

- Podemos fazer também a tabela de frequência relativa.
- Neste caso, usamos `prop.table(x)`.

---

---

---

---

---

---

---

---

## Exemplo

```
> prop.table(table(Sexo)) #Tabela de frequência relativa
Sexo
 f  m
0.6 0.4
```

## Tabelas

- Para fazer tabela de frequências para variáveis quantitativas contínuas é um pouco mais complicado.
- Devemos observar seu menor valor, seu maior valor e a amplitude de cada intervalo.
- Para isso usamos a função `range(x)` que já foi utilizada anteriormente, e usamos a regra da raiz do tamanho da amostra, no caso é raiz de `length(x)`.

## Bloco de Notas

- Podemos digitar uma seqüência de comandos em um editor de textos (p.ex. o Bloco de Notas, e depois colar no console para execução das tarefas solicitadas
- Abram o arquivo `tabelacontinua.txt`

## Passos

- `range(Altura)`: obtemos o mínimo e máximo.
- `maximo = max(Altura)`.
- `minimo = min(Altura)`.
- `R = maximo - minimo`.
- `k = sqrt(length(Altura))`: iremos considerar no máximo k intervalos.
- `h = R/k` (tamanho do intervalo)

---

---

---

---

---

---

---

---

## Passos

- `minimo = 156`.
- `maximo = 163`.
- `R = 163 - 156 = 7`.
- `k = 4,472136` (no máximo 5 intervalos).
- `h = R/k = 1,565248` (arredondamos para 2).

---

---

---

---

---

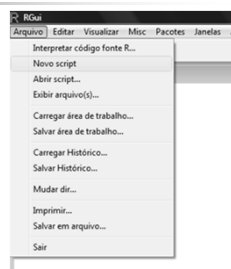
---

---

---

## Uso do Script

- Facilita para:
  - Correção ou expansão de comandos.
  - Repetição de comandos.
  - Armazenamento de resultados.




---

---

---

---

---

---

---

---

