



Estatística descritiva usando R bem-vind@ ao tidyverse

Curso livre de R

Profa Carolina e Prof Gilberto
Parte 1

sobre o curso

Preparando o ambiente

- Você precisa de um computador para acompanhar as aulas
- Usaremos nas aulas: colab.research.google.com/#language=r
- No seu dia-a-dia, recomendo instalar o R com versão pelo 4.1: cran.r-project.org
- IDE recomendadas: [RStudio](#) e [VSCode](#)
 - Caso você queira usar o [VSCode](#), instale a extensão da linguagem R: `ikuyadeu.r`
- Neste curso, usaremos o *framework* [tidyverse](#):
 - Instale o framework a partir do repositório CRAN: `install.packages("tidyverse")`
- Outras linguagens interessantes: [python](#) e [julia](#)
 - [python](#): linguagem interpretada de propósito geral, contemporânea do R, simples e fácil de aprender
 - [julia](#): linguagem interpretada para análise de dados, lançada em 2012, promete simplicidade e velocidade



a linguagem **R**

uma introdução

O começo de tudo

O precursor do R: S

- R é uma linguagem derivada do S
- S foi desenvolvido em **fortran** por **John Chambers** em **1976** no **Bell Labs**
- S foi desenvolvido para ser um ambiente de análise estatística
- Filosofia do S: permitir que usuários possam analisar dados usando estatística com pouco conhecimento de programação

História do R

- Em **1991**, **Ross Ihaka** e **Robert Gentleman** criaram o R na **Nova Zelândia**
- Em **1995**, **Ross** e **Robert** liberam o R sob a licença “GNU General License”, o que tornou o R um software livre
- Em **1997**, **The Core Group** é criado para melhorar e controlar o código fonte do R



Porque usar R

- Constante melhoramento e atualização
- Portabilidade (roda em praticamente todos os sistemas operacionais)
- Grande comunidade de desenvolvedores que adicionam novas capacidades ao R através de pacotes
- Gráficos de maneira relativamente simples
- Interatividade
- Um grande comunidade de usuários (especialmente útil para resolução de problemas)



Onde estudar fora de aula?

Livros

- Nível *cheguei agora aqui*: [zen do R](#)
- Nível Iniciante: [R Tutorial na W3Schools](#)
- Nível Iniciante: [Hands-On Programming with R](#)
- Nível Intermediária: [R for Data Science](#)
- Nível Avançado: [Advanced R](#)

Em pt-br

- Curso-R: material.curso-r.com



O que você fazer quando estiver em apuros?

- check a documentação do R:

```
help(mean)  
?mean
```

- Peça ajuda a um programador mais experiente
- Consulte o pt.stackoverflow.com
- Use ferramentas de busca como o [google](https://www.google.com) e duckduckgo.com

```
log("G")
```

- Na ferramenta de busca, pesquise por `Error in log("G"): non-numeric argument to mathematical function`



Operações básicas

Soma

$1 + 1$

[1] 2

Subtração

$2 - 1$

[1] 1

Divisão

$3 / 2$

[1] 1.5



Operações básicas

Potenciação

```
2^3
```

```
## [1] 8
```

Resto da divisão e parte inteira da divisão

```
5 %% 3
```

```
## [1] 2
```

Parte inteira da divisão

```
5 %/% 3
```

```
## [1] 1
```



$\begin{array}{r} 5 \\ -3 \\ \hline 2 \end{array}$	$\begin{array}{ l} 3 \\ \hline 1 = 5 \% \% 3 \end{array}$
$2 = 5 \% / \% 3$	

Estrutura de dados no R

- **Estrutura de dados:** atomic vector (a estrutura de dados mais básico no R), matrix, array, list e data.frame (tibble no tidyverse)
- **Tipo de dados:** character (character), número real (double), número inteiro (integer), número complexo (complex) e lógico (logical)
- **Estrutura de dados Homogênea:** vector, matrix e array
- **Estrutura de dados Heterôgenea:** list e data.frame (tibble no tidyverse)



Tipo de dados no R

Número inteiro

```
typeof(1L)
```

```
## [1] "integer"
```

Número real

```
typeof(1.2)
```

```
## [1] "double"
```

Número complexo

```
typeof(1 + 1i)
```

```
## [1] "complex"
```



Tipo de dados no R

Número lógico

```
typeof(TRUE)
```

```
## [1] "logical"
```

Caracter

```
typeof("Gilberto")
```

```
## [1] "character"
```



Estrutura de Dados Homogênea

Vetor

- Agrupamento de valores de mesmo tipo em um único objeto
- Criação de vetores: `c(...)` e `vector('<tipo de dados>', <comprimento do vetor>)`, `vector()` é bastante usado em *laços de repetição*, que veremos na semana 4, o operador `:` e `seq(from = a, to = b, by = c)`

Vetor de caracteres

```
a <- c("Gilberto", "Sassi")
```

```
a
```

```
## [1] "Gilberto" "Sassi"
```

```
b <- vector("character", 3)
```

```
b
```

```
## [1] "" "" ""
```



Estrutura de Dados Homogênea

Vetor de número real

```
a <- c(0.2, 1.35)
```

```
a
```

```
## [1] 0.20 1.35
```

```
b <- vector("double", 3)
```

```
b
```

```
## [1] 0 0 0
```

```
d <- seq(from = 1, to = 3.5, by = 0.5)
```

```
d
```

```
## [1] 1.0 1.5 2.0 2.5 3.0 3.5
```



Estrutura de Dados Homogênea

Vetor de número inteiro

```
a <- c(1L, 2L)
```

a

```
## [1] 1 2
```

```
b <- vector("integer", 3)
```

b

```
## [1] 0 0 0
```

```
d <- 1:4
```

d

```
## [1] 1 2 3 4
```



Estrutura de Dados Homogênea

Vetor de número inteiro

```
a <- c(TRUE, FALSE)
```

a

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

```
b <- vector("logical", 3)
```

b

```
## [1] FALSE FALSE FALSE
```



Estrutura de Dados Homogênea

Matriz

- Agrupamento de valores de mesmo tipo em um único objeto de dimensão 2
- Criação de vetores: `matrix(..., nrow = <integer>, ncol = <integer>)` ou `diag(<vector>)`

Matriz de caracteres

```
a <- matrix(c("a", "b", "c", "d"), nrow = 2)
a
```

```
##      [,1] [,2]
## [1,] "a"  "c"
## [2,] "b"  "d"
```

Matriz de números reais

```
a <- matrix(seq(from = 0, to = 1.5, by = 0.5), nrow = 2)
a
```

```
##      [,1] [,2]
## [1,] 0.0  1.0
## [2,] 0.5  1.5
```



Estrutura de Dados Homogênea

Matriz de inteiros

```
a <- matrix(1L:4L, nrow = 2)
```

```
a
```

```
##      [,1] [,2]  
## [1,]    1    3  
## [2,]    2    4
```

Matriz de valores lógicos

```
a <- matrix(c(TRUE, F, F, T), nrow = 2)
```

```
a
```

```
##      [,1] [,2]  
## [1,] TRUE FALSE  
## [2,] FALSE TRUE
```



Estrutura de Dados Homogênea

Array

- Agrupamento de valores de mesmo tipo em um único objeto em duas ou mais dimensões
- Criação de vetores: `array(..., dim = <vector of integers>)`

```
dados_matriz_1 <- 10:13
dados_matriz_2 <- 14:17
resultado <- array(c(dados_matriz_1, dados_matriz_2), dim = c(2, 2, 2))
resultado
```

```
## , , 1
##
##      [,1] [,2]
## [1,]   10   12
## [2,]   11   13
##
## , , 2
##
##      [,1] [,2]
## [1,]   14   16
## [2,]   15   17
```



Estrutura de Dados Homogênea

Operações em Vetores numéricos (double, integer e complex)

- Operações básicas (operação, subtração, multiplicação e divisão) realizada em cada elemento do vetor
- *Slicing*: extrae parte de um vetor (não precisa ser vetor numérico)

Slicing

```
a <- c("a", "b", "c", "d", "e", "f", "g", "h", "i")  
a[1:5] # selecionado todos os elementos entre o primeiro e o quinta
```

```
## [1] "a" "b" "c" "d" "e"
```

Adição (vetores numéricos)

```
a <- 1:5  
b <- 6:10  
a + b
```

```
## [1] 7 9 11 13 15
```



Estrutura de Dados Homogênea

Subtração (vetores numéricos)

```
a <- 1:5  
b <- 6:10  
b - a
```

```
## [1] 5 5 5 5 5
```

Multiplicação (vetores numéricos)

```
a <- 1:5  
b <- 6:10  
b * a
```

```
## [1] 6 14 24 36 50
```

Divisão (vetores numéricos)

```
a <- 1:5  
b <- 6:10  
b / a
```

```
## [1] 6.000000 3.500000 2.666667 2.250000 2.000000
```



Estrutura de Dados Homogênea

Operações em Matrizes numéricas (`double`, `integer` e `complex`)

- Operações básicas (adição, subtração, multiplicação e divisão) realizada em cada elemento das matrizes
- Multiplicação de matrizes (vide [multiplicação de matrizes](#)), inversão de matrizes (vide [inversão de matrizes](#)), matriz transposta (vide [matriz transposta](#)), determinante (vide [determinante de uma matriz](#)) e solução de sistema de equações lineares (vide [sistema de equações lineares](#))



Estrutura de Dados Homogênea

Soma de matrizes

```
A <- matrix(c(1, 2, 3, 4), nrow = 2)
B <- matrix(5:8, ncol = 2)
C <- A + B
C
```

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

Soma de subtração

```
A <- matrix(c(1, 2, 3, 4), nrow = 2)
B <- matrix(5:8, ncol = 2)
C <- B - A
C
```

```
##      [,1] [,2]
## [1,]    4    4
## [2,]    4    4
```



Estrutura de Dados Homogênea

Multiplicação ponto-a-ponto ou produto de Hadamard

- Para detalhes vide [produto de Hadamard](#)

```
A <- matrix(c(1, 2, 3, 4), nrow = 2); B <- matrix(5:8, ncol = 2)
C <- A * B
C
```

```
##      [,1] [,2]
## [1,]    5  21
## [2,]   12  32
```

Multiplicação de matrizes

- Para detalhes vide [multiplicação de matrizes](#)

```
C <- A %*% B
C
```

```
##      [,1] [,2]
## [1,]   23  31
## [2,]   34  46
```



Estrutura de Dados Homogênea

Matriz inversa

- Para detalhes vide [matriz inversa](#)

```
A <- matrix(1:4, ncol = 2)
B <- solve(A)
B
```

```
##      [,1] [,2]
## [1,]  -2  1.5
## [2,]   1 -0.5
```

```
A %*% B
```

```
##      [,1] [,2]
## [1,]    1  0
## [2,]    0  1
```



Estrutura de Dados Homogênea

Matriz transposta

- Para detalhes vide [matriz transposta](#)

```
A <- matrix(1:4, ncol = 2)
B <- t(A)
B
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
```

Determinante de uma matriz

- Para detalhes vide [determinante](#)

```
A <- matrix(1:4, ncol = 2)
det(A)
```

```
## [1] -2
```



Estrutura de Dados Homogênea

Solução de sistema de equações lineares

- Para detalhes vide [sistema de equações lineares](#)

```
b <- c(1, 2); A <- matrix(1:4, nrow = 2)
x <- solve(A, b)
x
```

```
## [1] 1 0
```

Matriz inversa generalizada

- G é a matriz inversa generalizada de A se $A \cdot G \cdot A = A$. Para detalhes vide [matriz inversa generalizada](#)

```
library(MASS) # ginv é uma função do pacote MASS
A <- matrix(c(1, 1, 2, 3), nrow = 2)
ginv(A)
```

```
##      [,1] [,2]
## [1,]    3  -2
## [2,]   -1    1
```



Estrutura de Dados Homogênea

Outras operações com matrizes

Operador ou função	Descrição
A %% B	<u>produto diádico $A \cdot B^T$</u>
crossprod(A, B)	<u>$A \cdot B^T$</u>
crossprod(A)	<u>$A \cdot A^T$</u>
diag(x)	retorna uma matrix diagonal com diagonal igual a x (<code>class(x) == 'numeric'</code>)
diag(A)	retorna um vetor com a diagona de A (<code>class(A) == 'matrix'</code>)
diag(k)	retorna uma matriz diagona de ordem k (<code>class(k) == 'numeric'</code>)
rowMeans(A)	retorna um vetor com as médias das linhas
colMeans(A)	retorna um vetor com as médias das colunas



Estrutura de Dados Heterogênea

Lista

- Agrupamento de valores de tipos diversos e estrutura de dados
- Criação de listas: `list(...)` e `vector("list", <comprimento da lista>)`

```
a <- list(pedido_id = 8001406,  
         nome = "Fulano",  
         sobrenome = "de Tal",  
         cpf = "12345678900",  
         itens = list(list(descricao = "Ferrari",  
                           frete = 0,  
                           valor = 500000),  
                       list(descricao = "Dolly", frete = 1.5, valor = 3.90)))
```



Estrutura de Dados Heterogênea

- Agrupamento de dados em tabela em que: cada coluna é uma variável; cada linha é uma observação
- Criação de tibble: `tibble(...)` e `tribble(...)`

tibble (data frame)

```
library(tidyverse) # carregando o framework tidyverse
a <- tibble(variavel_1 = c(1, 2), variavel_2 = c("a", "b"))
glimpse(a)
```

```
## Rows: 2
## Columns: 2
## $ variavel_1 [3m [38;5;246m<dbl> [39m [23m 1, 2
## $ variavel_2 [3m [38;5;246m<chr> [39m [23m "a", "b"
```

```
a
```

```
## # A tibble: 2 × 2
##   variavel_1 variavel_2
##       <dbl> <chr>
## 1         1 a
## 2         2 b
```



Estrutura de Dados Heterogênea

Operações em um `tibble`

Vamos ver o uso dessas funções depois de aprender a carregar os dados no R.

Função	Descrição
<code>head()</code>	Mostra as primeiras linhas de um <code>tibble</code>
<code>tail()</code>	Mostra as últimas linhas de um <code>tibble</code>
<code>glimpse()</code>	Impressão de informações básicas dos dados
<code>add_case()</code> ou <code>add_row()</code>	Adiciona uma nova observação



Estrutura de Dados Heterogênea

Concatenação de listas

```
a <- list("a", "b")  
b <- list(1, 2)  
d <- c(a, b)  
d
```

```
## [[1]]  
## [1] "a"  
##  
## [[2]]  
## [1] "b"  
##  
## [[3]]  
## [1] 1  
##  
## [[4]]  
## [1] 2
```



Estrutura de Dados Heterogênea

Slicing a lista

```
d[1:2]
```

```
## [[1]]  
## [1] "a"  
##  
## [[2]]  
## [1] "b"
```

Acessando o valor de elemento em uma lista

```
d[[2]] # acessando o segundo elemento da lista d
```

```
## [1] "b"
```

Acessando o valor de elemento em uma lista pela chave

```
d <- list(chave_1 = 1, chave_2 = "docente")  
d$chave_2 # retorna o valor
```

```
## [1] "docente"
```



Estrutura de Dados Heterogênea

Slicing uma lista por chaves

```
d <- list(chave_1 = 1, chave_2 = "docente", chave_3 = list("olá"))  
d[c("chave_2", "chave_3")] # funciona como slicing
```

```
## $chave_2  
## [1] "docente"  
##  
## $chave_3  
## $chave_3[[1]]  
## [1] "olá"
```

Enumerando chaves em um lista

```
d <- list(c(1, 2, 3), chave_1 = 1, chave_2 = "docente", chave_3 = list("olá"))  
names(d)
```

```
## [1] ""      "chave_1" "chave_2" "chave_3"
```



Valores especiais em R

Valores especiais	Descrição	Função para identificar
NA (Not Available)	Valor faltante.	<code>is.na()</code>
NaN (Not a Number)	Resultado do cálculo indefinido.	<code>is.nan()</code>
Inf (Infinito)	Valor que excede o valor máximo que sua máquina aguenta.	<code>is.inf()</code>
NULL (Nulo)	Valor indefinido de expressões e funções (diferente de NaN e NA)	<code>is.null()</code>



Parênteses 1: guia de estilo no R

- Nome de um objeto precisa ter um *significado*. Esse significado precisa falar imediatamente o que este objeto é ou faz ~sua bisavó precisa entender o que este objeto é ou faz~
- Use a convenção do R:
 - Use apenas letras minúsculas, números e *underscore* (comece sempre com letras minúsculas)
 - Nomes de objetos precisam ser substantivos e precisam descrever o que este objeto é ou faz (seja conciso, direto e significativo)
 - Evite ao máximo os nomes que são usados por objetos que são *buit-in* do R
 - Coloque espaço depois da vírgula
 - Não coloque espaço antes nem depois de parênteses. Exceção: Coloque um espaço () antes e depois de `if`, `for` ou `while`, e coloque um espaço depois de ().
 - Coloque espaço entre operadores básicos: `+`, `-`, `*`, `==` e outros. Exceção: `^`.
- Para mais detalhes, consulte: [guia de estilo do tidyverse](#)



Parênteses 2: estrutura de diretórios

- Mantenha uma estrutura consistente de diretórios em seus projetos.
- Eu uso a seguinte estrutura:
 - **data**: diretório para armazenar seus conjuntos de dados
 - **raw**: dados brutos
 - **processed**: dados processados
 - **scripts**: código fonte do seu projeto
 - **figures**: figuras criadas no seu projeto
 - **output**: outros arquivos que não são figuras
 - **previous**: arquivos da versão anterior do projeto
 - **notes**: notas de reuniões e afins
 - **relatorio** (ou **artigos**): documento final de seu projeto
 - **documents**: livros, artigos e qualquer coisa que são referências em seu projeto
- Para mais detalhes, consulte esse guia do [curso-r: diretórios e .Rproj](#)



Lendo dados no R

Leitura de arquivos no formato `xlsx` ou `xls`

- **Pacote:** `readxl` do `tidyverse` (instale com o comando `install.packages('readxl')`)
- Parâmetros das funções `read_xls` (para ler arquivos `.xls`) e `read_xlsx` (para ler arquivos `.xlsx`):
 - `path`: caminho até o arquivo
 - `sheet`: especifica a planilha do arquivo que será lida
 - `range`: especifica uma área de uma planilha para leitura. Por exemplo: `B3:E15`
 - `col_names`: Argumento lógico com valor padrão igual a `TRUE`. Indica se a primeira linha tem o nome das variáveis
- Para mais detalhes, consulte a documentação oficial do `tidyverse`: [documentação de read_xl](#)



Lendo dados no R

Leitura de arquivos no formato `xlsx` ou `xls`

```
library(tidyverse)
library(readxl)
dados_iris <- read_xlsx("../data/raw/iris.xlsx")

head(dados_iris, n = 4)
```

```
## # A tibble: 4 × 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##   <dbl>         <dbl>         <dbl>         <dbl> <chr>
## 1         5.1         3.5         1.4         0.2 setosa
## 2         4.9         3         1.4         0.2 setosa
## 3         4.7         3.2         1.3         0.2 setosa
## 4         4.6         3.1         1.5         0.2 setosa
```



Lendo dados no R

Leitura de arquivos no formato csv

- **Pacote:** `readr` do `tidyverse` (instale com o comando `install.packages('readr')`)
- Parâmetros das funções `read_csv` e `read_csv2`:
 - `path`: caminho até o arquivo

Padrão imperial inglês versus o resto do planeta

- Se você mora ou está em um país que usa padrão *imperial inglês*:
 - colunas separadas por `,`
 - casa decimal indicada por `.`
- Se você mora ou está em um país que usa o sistema métrico:
 - colunas separadas por `;`
 - casa decimal por `,`

Preste atenção em como o seus dados estão armazenados!

Para mais detalhes, consulte a documentação oficial do *tidyverse*: [documentação de read_r](#)



Lendo dados no R

Leitura de arquivos no formato csv

```
library(tidyverse)
library(readr)
dados_mtcars <- read_csv2("../data/raw/mtcars.csv")
dados_mtcars
```

```
## # A tibble: 32 × 11
##       mpg   cyl  disp    hp  drat    wt   qsec    vs  am  gear  carb
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  21       6  160   110  3.9   2.62  16.5    0    1     4     4
## 2  21       6  160   110  3.9   2.88  17.0    0    1     4     4
## 3  22.8     4  108    93  3.85  2.32  18.6    1    1     4     1
## 4  21.4     6  258   110  3.08  3.22  19.4    1    0     3     1
## 5  18.7     8  360   175  3.15  3.44  17.0    0    0     3     2
## 6  18.1     6  225   105  2.76  3.46  20.2    1    0     3     1
## 7  14.3     8  360   245  3.21  3.57  15.8    0    0     3     4
## 8  24.4     4  147.    62  3.69  3.19  20      1    0     4     2
## 9  22.8     4  141.    95  3.92  3.15  22.9    1    0     4     2
## 10 19.2     6  168.   123  3.92  3.44  18.3    1    0     4     4
## # ... with 22 more rows
```



Lendo dados no R

Leitura de arquivo *formato com comprimento fixo (fixed width format: fwf)*

- Pacote: readr do tidyverse (instale com o comando `install.packages('readr')`)
- Parâmetros das funções `read_fwf`:
 - `file`: caminho até o arquivo
 - `col_positions`: use a função `fwf_widths()` fornece as delimitações e os nomes das colunas
 - `col_types`: texto (string) especificando o tipo de cada coluna: “c” para character, “d” para número real.



Exemplo

- **velocidade**: começa na coluna 1 e termina na coluna 4 com comprimento 4;
- **distancia**: começa na coluna 5 e termina na coluna 7 com comprimento 3.

```
dados_carros <- read_fwf(  
  "../data/raw/carros.txt",  
  col_positions = fwf_widths(c(4, 3), col_names = c("velocidade", "dist")),  
  col_types = "dd"  
)  
glimpse(dados_carros)
```

```
## Rows: 50
```

```
## Columns: 2
```

```
## $ velocidade <dbl> 4, 4, 7, 7, 8, 9, 10, 10, 10, 11, 11, 12, 12, 12, 12...
```

```
## $ dist      <dbl> 2, 10, 4, 22, 16, 10, 18, 26, 34, 17, 28, 14, 20, 24...
```



Lendo dados no R

Leitura de arquivos no formato ods

- **Pacote:** `readODS` (instale com o comando `install.packages('readODS')`)
- Parâmetros das funções `read_ods`:
 - `path`: caminho até o arquivo
 - `sheet`: especifica a planilha do arquivo que será lida
 - `range`: especifica uma área de uma planilha para leitura. Por exemplo: `B3:E15`
 - `col_names`: Argumento lógico com valor padrão igual a `TRUE`. Indica se a primeira linha tem o nome das variáveis
- Para mais detalhes, consulte a documentação do *readODS*: [documentação de readODS](#)



Lendo dados no R

Leitura de arquivos no formato ods

```
library(tidyverse)
library(readODS)
dados_toothgrowth <- read_ods("../data/raw/ToothGrowth.ods")

glimpse(dados_toothgrowth)
```

```
## Rows: 60
## Columns: 3
## $ len <dbl> 4.2, 11.5, 7.3, 5.8, 6.4, 10.0, 11.2, 11.2, 5.2, 7.0, 16.5...
## $ supp <chr> "VC", "VC", "VC", "VC", "VC", "VC", "VC", "VC", "VC", "VC"...
## $ dose <dbl> 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 1.0, 1.0...
```



Salvando dados no R

Salvar no formato **.csv** (sistema métrico)

```
library(readr)
write_csv2(dados_toothgrowth, file = "../data/processed/dados_csv2.csv")
```

Salvar no formato **.xlsx**

```
library(writexl)
write_xlsx(dados_toothgrowth, path = "../data/processed/dados_xlsx.xlsx")
```

Salvar no formato **ods**

```
library(readODS)
write_ods(dados_toothgrowth, path = "../data/processed/dados_ods.ods")
```



estatística descritiva no **R**

gráficos e tabelas

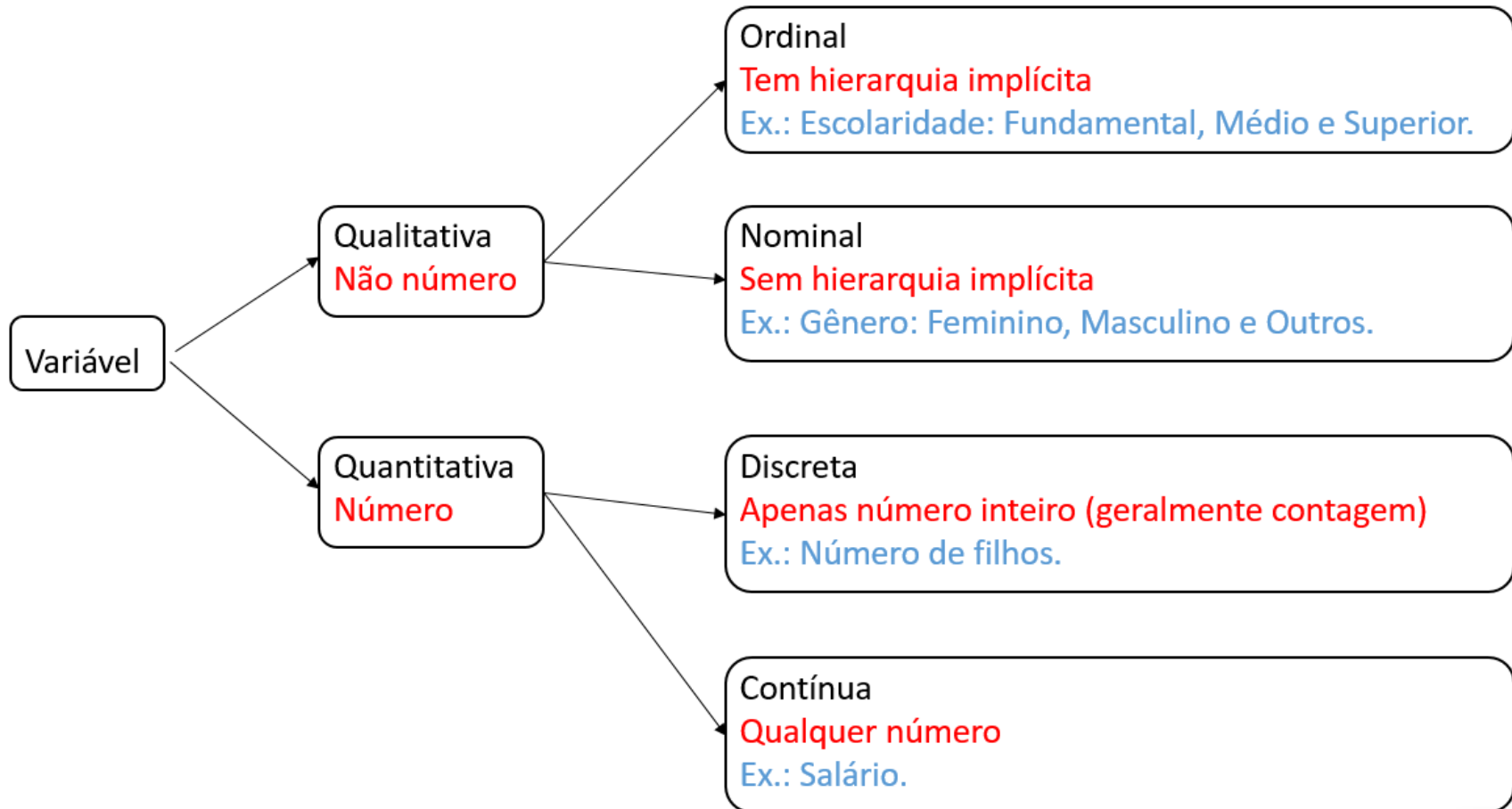
Alguns conceitos básicos

Começamos com alguns conceitos básicos, que usaremos durante todo esse curso.

- **População:** Todos os elementos ou indivíduos alvo do estudo;
- **Amostra:** Parte da população;
- **Parâmetro:** característica da população (grandeza);
- **Estimativa:** característica da amostra. Usamos a estimativa para aproximar o parâmetro;
- **Variável:** *característica de um elemento da população (mensurando ou analito).* Geralmente usamos uma letra maiúscula do alfabeto latino para representar uma variável (mensurando ou analito), e uma letra minúscula do alfabeto latino para representar o valor de uma variável para um elemento (indicação) da população. Por exemplo, podemos representar a variável “Teor de hidrocloro” por X e um indicação da amostra por $x = 25,1 \text{ mg/comprimido}$.



Classificação de variáveis



Classificação de variáveis.

Tabela de distribuição de frequência – Variável qualitativa

A primeira coisa que fazemos é contar!

X	frequência	frequência relativa	porcentagem
B_1	n_1	f_1	$100 \cdot f_1 \%$
B_2	n_2	f_2	$100 \cdot f_2 \%$
\vdots	\vdots	\vdots	\vdots
B_k	n_k	f_k	$100 \cdot f_k \%$
Total	n	1	100%

Em que n é o tamanho da amostra.



Tabela de distribuição de frequência – Variável qualitativa

A primeira coisa que podemos fazer é construir a tabela de distribuição de frequência.

```
df_empresa <- read_xlsx("../data/raw/empresa.xlsx")
tab_freq <- df_empresa |>
  group_by(escolaridade) |>
  summarise(frequencia = n()) |>
  mutate(frequencia_relativa = frequencia / sum(frequencia),
         porcentagem = 100 * frequencia_relativa)
tab_freq |>
  add_case(Escolaridade = "Total",
          frequencia = sum(tab_freq$frequencia),
          frequencia_relativa = sum(tab_freq$frequencia_relativa),
          porcentagem = sum(tab_freq$porcentagem))
```



escolaridade	frequencia	frequencia_relativa	porcentagem
ensino fundamental	12	0,33	33,33
ensino médio	18	0,50	50,00
superior	6	0,17	16,67
Total	36	1,00	100,00



Gráficos no R

- Pacote: `ggplot2`
- Permite gráficos personalizados com uma sintaxe simples e rápida, e iterativa *por camadas*
- Começamos com um camada com os dados `ggplot(dados)`, e vamos adicionando as camadas de anotações, e sumários estatísticos
- Usa a *gramática de gráficos* proposta por Leland Wilkinson: [Grammar of Graphics](#)
- Ideia desta gramática: delinear os atributos estéticos das figuras geométricas (incluindo transformações nos dados e mudança no sistema de coordenadas)
- Para mais detalhes, você pode consultar [ggplot2: elegant graphics for data analysis](#) e [documentação do ggplot2](#)



Gráficos no R

Estrutura básica de ggplot2

```
ggplot(data = <data possible tibble>) +  
  <Geom functions>(mapping = aes(<MAPPINGS>)) +  
  <outras camadas>
```

Você pode usar diversos temas e extensões que a comunidade cria e criou para melhorar a aparência e facilitar a construção de ggplot2.

- Lista com extensões do ggplot: [extensões do ggplots](#)

Indicação de extensões:

- Temas adicionais para o pacote ggplot2: [ggthemes](#)
- Gráfico de matriz de correlação: [ggcorrplot](#)
- Gráfico quantil-quantil: [qqplotr](#)



Gráficos no R

Gráfico de Barras no ggplot2

- **função:** `geom_bar()`. Para porcentagem: `geom_bar(x = <variável no eixo x>, y = ..prop.. * 100)`.
- Argumentos adicionais:
 - **fill:** mudar a cor do preenchimento das figuras geométricas
 - **color:** mudar a cor da figura geométrica

Rótulos dos eixos

- **Mudar os rótulos:** `labs(x = <rótulo do eixo x>, y = <rótulo do eixo y>)`
- **Trocar o eixo-x pelo eixo-y:** `coord_flip()`



```
library(ggthemes)
ggplot(df_empresa) +
  geom_bar(mapping = aes(x = escolaridade, y = ..prop.. * 100, group = 1),
    fill = "blue", color = "red") +
  labs(x = "Espécies", y = "Porcentagem") +
  theme_gdocs() +
  coord_flip()
```

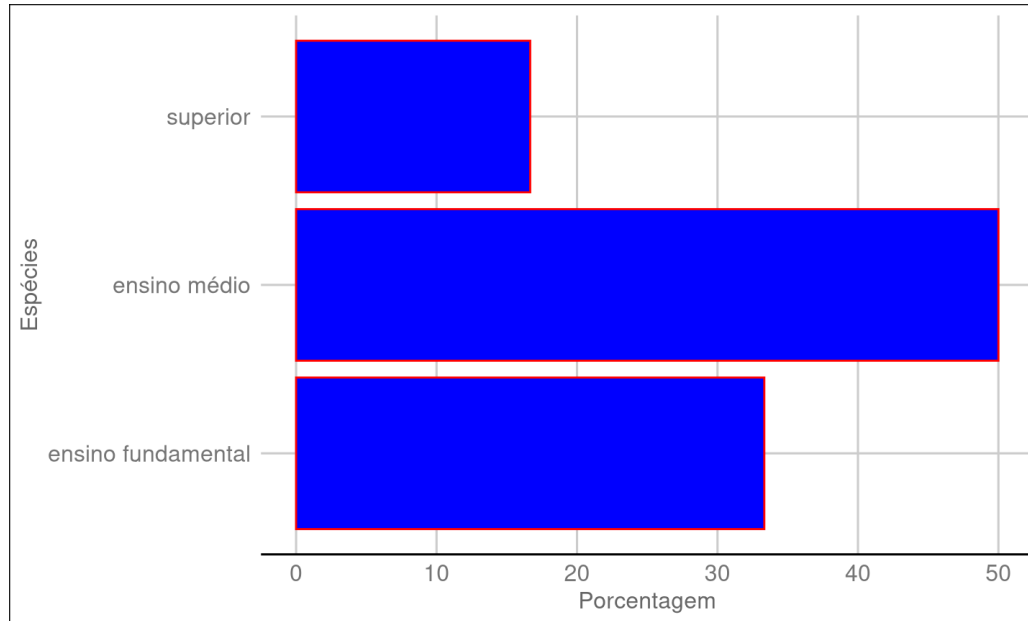


Tabela de distribuição de frequência – Variável quantitativa discreta

A primeira coisa que fazemos é contar!

X	frequência	frequência relativa	porcentagem
x_1	n_1	f_1	$100 \cdot f_1 \%$
x_2	n_2	f_2	$100 \cdot f_2 \%$
x_3	n_3	f_3	$100 \cdot f_3 \%$
\vdots	\vdots	\vdots	\vdots
x_k	n_k	f_k	$100 \cdot f_k \%$
Total	n	1	100%

Em que n é o tamanho da amostra.



Tabela de distribuição de frequência – Variável quantitativa discreta

A primeira coisa que podemos fazer é construir a tabela de distribuição de frequência.

```
tab <- df_empresa |>
  group_by(n_filhos = as.character(numero_filhos)) |>
  summarise(frequencia = n()) |>
  mutate(fr = frequencia / sum(frequencia),
         porcentagem = 100 * fr)
tab <- tab |>
  add_case(n_filhos = "Total",
          frequencia = sum(tab$frequencia),
          fr = sum(tab$fr),
          porcentagem = sum(tab$porcentagem))
tab
```



n_filhos	frequencia	fr	porcentagem
0	20	0,56	55,56
1	5	0,14	13,89
2	7	0,19	19,44
3	3	0,08	8,33
5	1	0,03	2,78
Total	36	1,00	100,00

Gráfico de barras no R

```
ggplot(df_empresa) +  
  geom_bar(aes(x = numero_filhos, y = ..prop.., group = 1)) +  
  labs(x = "Número de filhos", y = "Frequência relativa") +  
  theme_calc()
```

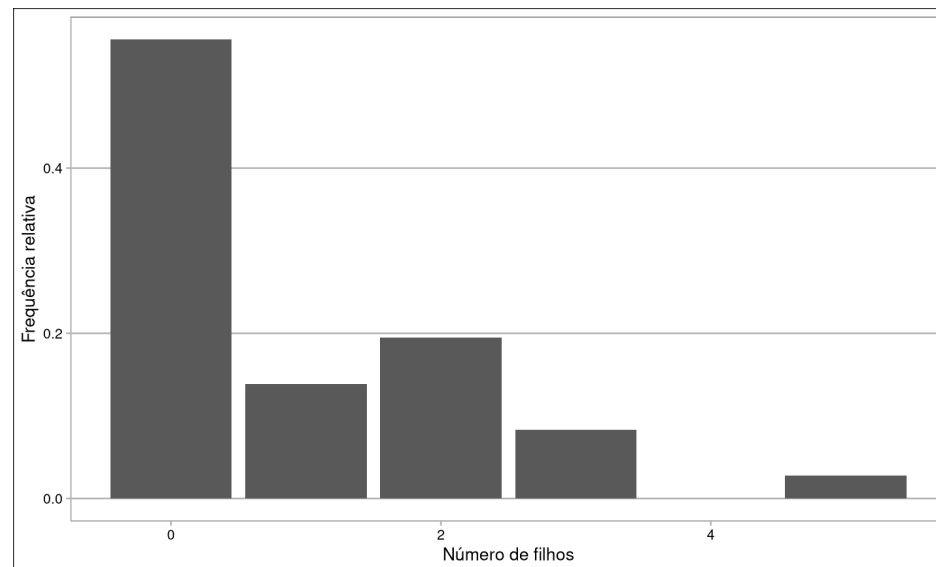


Tabela de distribuição de frequência – Variável quantitativa contínua

- X: variável quantitativa contínua

Tabela de Distribuição de Frequências para a variável quantitativa contínua.

x	Frequência	Frequência relativa	Porcentagem
$[l_0, l_1)$	n_1	$f_1 = \frac{n_1}{n_1 + \dots + n_k}$	$p_1 = f_1 \cdot 100$
$[l_1, l_2)$	n_2	$f_2 = \frac{n_2}{n_1 + \dots + n_k}$	$p_2 = f_2 \cdot 100$
\vdots	\vdots	\vdots	\vdots
$[l_{k-1}, l_k]$	n_k	$f_k = \frac{n_k}{n_1 + \dots + n_k}$	$p_k = f_k \cdot 100$

Em que $\min = l_0 \leq l_1 \leq \dots \leq l_{k-1} \leq l_k = \max$ (\min é o menor valor do suporte da variável X e \max é o maior valor do suporte da variável X), n_i é número de valores de X entre l_{i-1} e l_i , e l_0, l_1, \dots, l_k quebram o suporte da variável X (*breakpoints*).

l_0, l_1, \dots, l_k são escolhidos de acordo com a teoria por trás da análise de dados (ou pelo regulador). Se você está em uma nova área, use l_0, l_1, \dots, l_k igualmente espaçados, e use a [regra de Sturges](#) para determinar o valor de k : $k = 1 + \log_2(n)$ onde n é tamanho da amostra. Se $1 + \log_2(n)$ não é um número inteiro, usamos $k = \lceil 1 + \log_2(n) \rceil$.



Tabela de distribuição de frequência – Variável quantitativa contínua

```
df_iris <- read_xlsx("../data/raw/iris.xlsx")

k <- round(1 + log2(nrow(df_iris)))
tab <- df_iris |>
  group_by(sepal_length_intervalo = cut(Sepal.Length, breaks = k,
                                       include.lowest = T, right = F)) |>
  summarise(freq = n()) |>
  mutate(fr = freq / sum(freq), p = fr * 100)
tab <- tab |>
  add_case(
    sepal_length_intervalo = "Total",
    freq = sum(tab$freq),
    fr = sum(tab$fr),
    p = sum(tab$p)
  )
tab
```



sepal_length_intervalo	freq	fr	p
[4.3,4.75)	11	0,07	7,33
[4.75,5.2)	30	0,20	20,00
[5.2,5.65)	24	0,16	16,00
[5.65,6.1)	24	0,16	16,00
[6.1,6.55)	31	0,21	20,67
[6.55,7)	17	0,11	11,33
[7,7.45)	7	0,05	4,67
[7.45,7.9]	6	0,04	4,00
Total	150	1,00	100,00



Histograma

Para variáveis quantitativas contínuas, geralmente não construímos gráficos de barras, e usamos uma figura geométrica chamada de *histograma*.

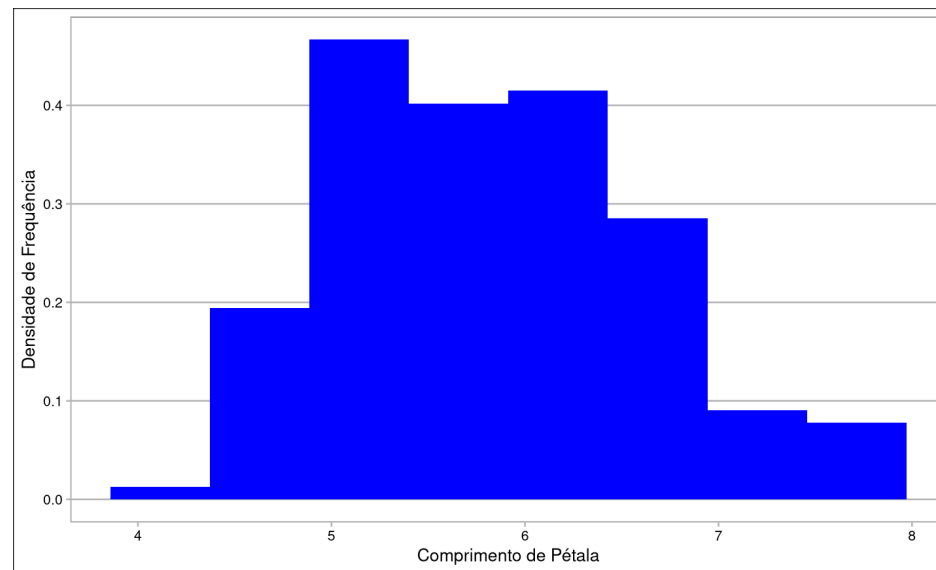
- O histograma é um gráfico de barras contíguas em que a área de cada barra é igual à frequência relativa.
- Cada faixa de valor $[l_{i-1}, l_i)$, $i = 1, \dots, n$, será representada por uma barra com área f_i , $i = 1, \dots, n$.
- Como cada barra terá área igual a f_i e base $l_i - l_{i-1}$, e a altura de cada barra será $\frac{f_i}{l_i - l_{i-1}}$.
- $\frac{f_i}{l_i - l_{i-1}}$ é denominada de densidade de frequência.



Histograma

```
df_iris <- read_xlsx("../data/raw/iris.xlsx")
k <- round(1 + log2(nrow(df_iris)))

ggplot(df_iris) +
  geom_histogram(aes(x = Sepal.Length, y = ..density..),
                 bins = k, fill = "blue") +
  theme_calc() +
  labs(x = "Comprimento de Pétala", y = "Densidade de Frequência")
```



Medidas de Resumo (variável quantitativa)

A ideia é encontrar um ou alguns valores que sintetizem todos os valores.

Medidas de posição (tendência central)

A ideia é encontrar um valor que representa *bem* todos os valores.

- **Média:** $\bar{x} = \frac{x_1 + \dots + x_n}{n}$
- **Mediana:** valor que divide a sequência ordenada de valores em duas partes iguais.

Medidas de dispersão

A ideia é medir a homogeneidade dos valores.

- **Variância:** $s^2 = \frac{(x_1 - \bar{X})^2 + \dots + (x_n - \bar{X})^2}{n - 1}$;
- **Desvio padrão:** $s = \sqrt{s^2}$ (mesma unidade dos dados);
- **coeficiente de variação** $cv = \frac{s}{\bar{x}} \cdot 100\%$ (adimensional, ou seja, “sem unidade”)



Medidas de Resumo: exemplo

Podemos usar a função `summarise` do pacote `dplyr` (incluso no pacote `tidyverse`).

```
df_empresa |>
  summarise(media = mean(salario), dp = sd(salario), mediana = median(salario),
            cv = dp / media)
```

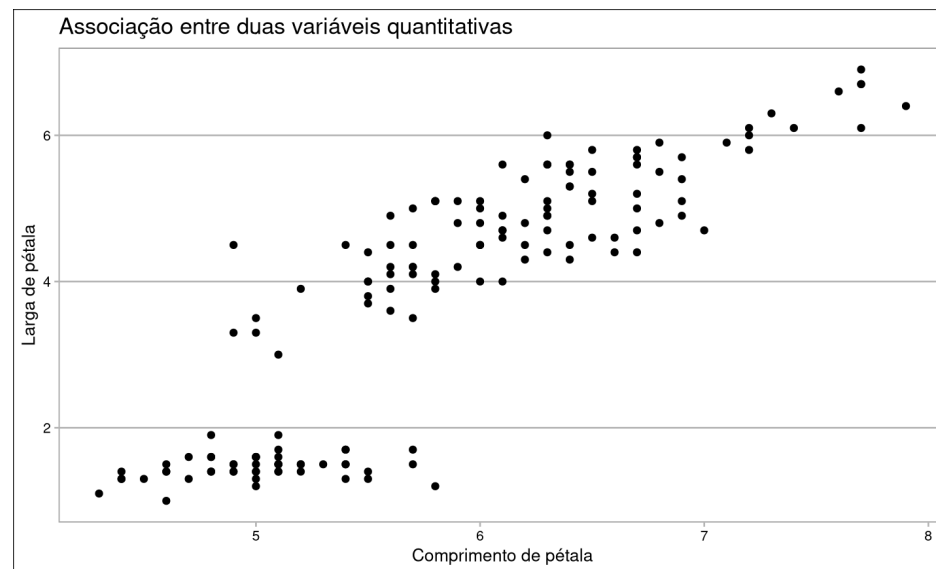
```
## # A tibble: 1 × 4
##   media    dp mediana    cv
##   <dbl> <dbl>   <dbl> <dbl>
## 1  11.1  4.59    10.2  0.412
```



Associação entre duas variáveis quantitativas

Para duas variáveis quantitativas, estudamos a associação entre as duas variáveis usando o gráfico de dispersão. Além disso, podemos calcular o coeficiente de correlação linear de Pearson.

```
ggplot(data = df_iris) +  
  geom_point(aes(x = Sepal.Length, y = Petal.Length)) +  
  labs(x = "Comprimento de pétala", y = "Larga de pétala",  
       title = "Associação entre duas variáveis quantitativas") +  
  theme_calc()
```



Associação entre duas variáveis quantitativas

Também podemos calcular o coeficiente de correlação linear de Pearson. Lembre que se X e Y são duas variáveis quantitativas com valores

Amostra de duas variáveis quantitativas X e Y .

X	x_1	x_2	\dots	x_n
Y	y_1	y_1	\dots	y_n

Então, o coeficiente de correlação linear é dado por

$$r = \left(\frac{(x_1 - \bar{x})}{s_x} \cdot \frac{(y_1 - \bar{y})}{s_y} \right) + \dots + \left(\frac{(x_n - \bar{x})}{s_x} \cdot \frac{(y_n - \bar{y})}{s_y} \right).$$

```
cor(df_iris$Sepal.Length, df_iris$Petal.Length)
```

```
## [1] 0.8717538
```

