Estatística Descritiva com R

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, recomendamos instalar o `R` com versão pelo menos `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 próposito 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 *1996*, **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 Intermediário:** `R` for Data Science.
- **Nível Avançado:** Advanced `R`.

Em pt-br

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

O que você pode fazer quando estiver em apuros?

- consultar 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 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
```

Substração

```
2 - 1
```

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

Divisão

```
3 / 2
```

```
## [1] 1.5
```

Potenciação

```
2^3
```

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

Os dados no `R`

- **Tipo de dados:** caracter (`character`), número real (`double`), número inteiro (`integer`), número complexo (`complex`) e lógico (`logical`).
- **Estrutura de dados:** atomic `vector` (a estrutura de dados mais básicA no `R`), `matrix`, `array`, `list` e `data.frame` (`tibble` no `tidyverse`).
- **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 vetor: `c(...)` e `vector('<tipo de dados>', <comprimento do vetor>)`, `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 números reais

```
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 números inteiros

```
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 lógico

```
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 matriz: `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 array: `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 com vetores númericos (`double`, `integer` e `complex`).

- Operações básicas (operação, substração, multiplicação e divisão) realizada em cada elemento do vetor.
- *Slicing*: extrair 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 números)

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

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

Estrutura de dados homogênea

Substração (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 com matrizes números (`double`, `integer` e `complex`).

- Operações básicas (operação, substração, 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 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, onde: 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 <dbl> 1, 2
## $ variavel_2 <chr> "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`

Algumas funções úteis depois de aprender a carregar os dados no `R`.

Função	Descrição
`head()`	Mostra as primeiras linhas de um `tibble`
`tail()`	Mostra as últimas linhas de um `tibble`
`glimpse()`	Impressão de informações básicas dos dados
`add_case()` ou **`add_row()`**	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 elmento em uma lista

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

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

Acessando elementos em uma lista usando `$`

```
d <- list(elemento_1 = 1, elemento_2 = "docente")
d$elemento_2 # retorna o valor
```

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

Estrutura de dados heterogênea

Slicing uma lista com `["nome"]`

```
d <- list(elemento_1 = 1, elemento_2 = "docente", elemento_3 = list("olá"))
d["elemento_3"] # funciona como slicing
```

```
## $elemento_3
## $elemento_3[[1]]
## [1] "olá"
```

Obtendo os nomes dos elementos em um lista

```
d <- list(c(1, 2, 3), elemento_1 = 1, elemento_2 = "docente", elemento_3 = list("olá"))
names(d)
```

```
## [1] ""           "elemento_1" "elemento_2" "elemento_3"
```

Valores especiais em R

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

Parênteses 1: guia de estilo no `R`

- O nome de um objeto precisa ter um *significado*. O nome deve indicar e deixar claro o que este objeto é ou faz ~qualquer pessoa 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 já são usados (*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 (organização) consistente de diretórios em seus projetos.
- Sugestão de 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')`)
- Parêmetros 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')`).
- Parêmetros 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 arquivos no formato `ods`

- **Pacote:** `readODS` (instale com o comando `install.packages('readODS')`).
- Parêmetros 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, 16.5,…
## $ supp <chr> "VC", "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, 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

- **População**: todos os elementos ou indivíduos alvo do estudo.
- **Amostra**: parte da população.
- **Parâmetro:** característica numérica da população. Usamos letras gregas para denotar parâmetros populacionais.
- **Estatística:** característica numérica da amostra. Em geral, usamos uma estatística para estimar o parâmetro populacional.
- **Variável:** *característica mensurável comum a todos os elementos da população.* Usamos letras maiúsculas do alfabeto latino para representar uma variável e letras minúsculas do alfabeto latino para representar o valor observado da variável em um elemento da amostra.

Exemplo:

- **População**: Todos os residentes da cidade de Salvador com 25 anos ou mais.
- **Amostra**: 5 residentes da cidade de Salvador com 25 anos ou mais *selecionados segundo um plano de amostragem probabilística*.
- **Variável:** salário em R$ (denotado pela letra X).
- **Parâmetro:** *salário médio* da população de residentes da cidade de Salvador com 25 anos ou mais (denotado pela letra grega μ).
- **Estatística:** *salário médio* da amostra de 20 residentes da cidade de Salvador com 25 anos ou mais.

Exemplo (continuação):

Suponha que foi selecionada uma amostra de $n = 5$ residentes da cidade de Salvador com 25 anos ou mais para os quais foi observada a variável salário em R$.

Salário em R$ de uma amostra de 5 residentes da cidade de Salvador com 25 anos ou mais.

Elemento da amostra	Salário
1	843.95
2	876.98
3	1055.87
4	907.05
5	912.93

Exemplo (continuação):

Para este exemplo, temos que:

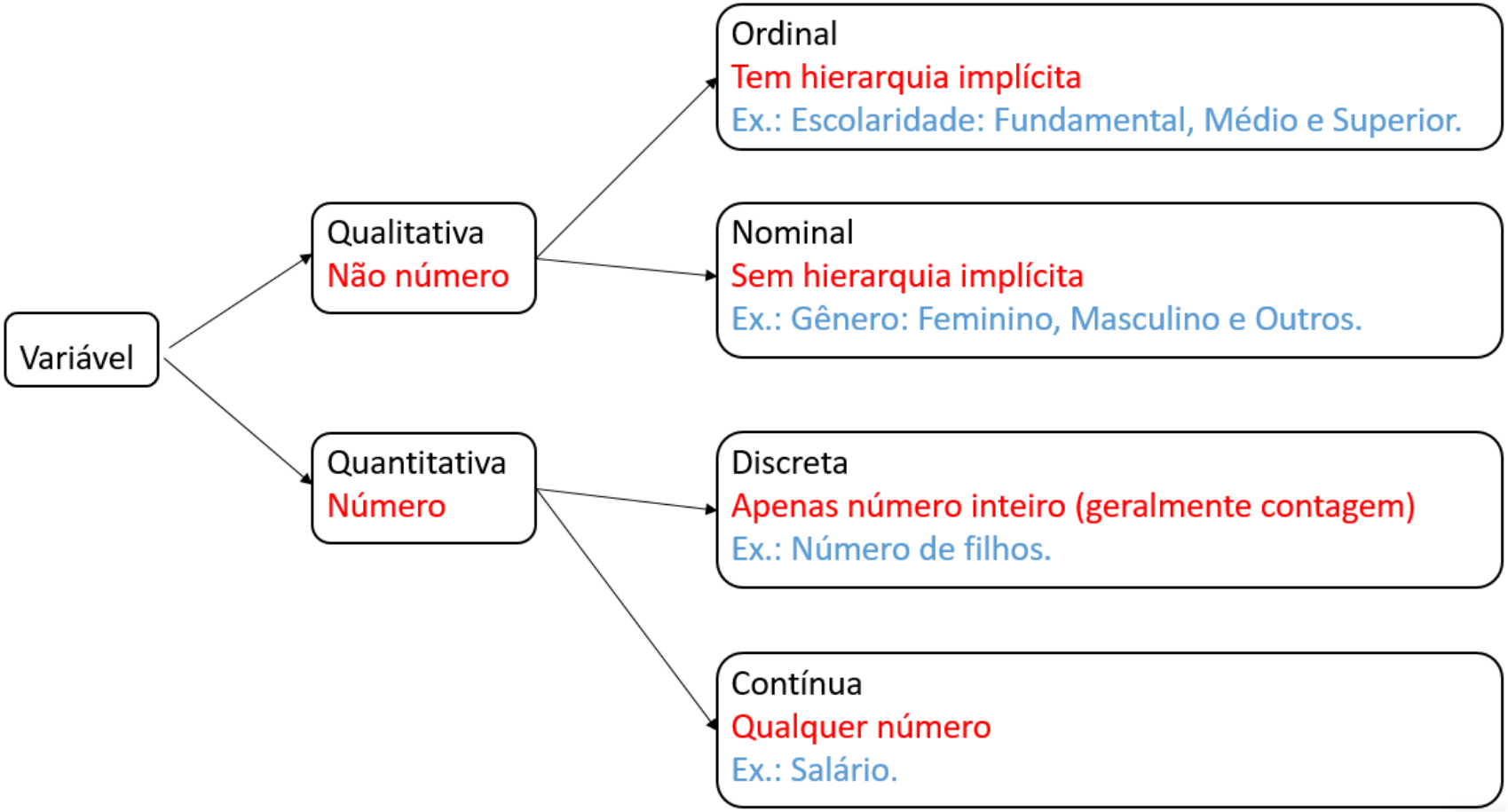
- Variável: X: sálario em R$.
- Valores observados de X: x_i: valor observado da variável no i-ésimo elemento da amostra, $i = 1, 2, 3, 4, 5$: 843.95; 876.98; 1055.87; 907.05; 912.93
- Parâmetro: μ: salário médio dos residentes da cidade de Salvador com 25 anos ou mais.
- Estatística: média amostral: $\overline{x} = \frac{x_1 + x_2 + x_3 + x_4 + x_5}{n}$.

Exemplo (continuação):

Média amostral:

$$\begin{aligned}\overline{x} &= \frac{x_1 + x_2 + x_3 + x_4 + x_5}{n} \\ &= \frac{843.95 + 876.98 + 1055.87 + 907.05 + 912.93}{5} \\ &= 919.356.\end{aligned}$$

Classificação de variáveis



Classificação de variáveis.

Tabela 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 frequência – Variável qualitativa

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

- **Pacote:** `tabyl` do `janitor` (instale com o comando `install.packages('janitor')`).
- Parêmetros da função `tabyl`:
  - `dat`: *data frame* ou vetor com os valores da variável que desejamos tabular.
  - `var1`: nome da primeira variável.
  - `var2`: nome da segunda variável (opcional).
- Para mais detalhes, consulte a documentação oficial do *janitor*: documentação de `tabyl`.

```r
library(tidyverse)
library(readxl)
library(janitor)
library(dplyr)

df_empresa <- read_xlsx("../data/raw/empresa.xlsx")

tab <- tabyl(df_empresa, escolaridade)
tab <- as_tibble(tab)
tab <- tab |> rename(frequencia = n,
                     freq_relativa = percent)
tab <-  tab |>
  add_column(porcentagem = tab$freq_relativa*100)

tab <-  tab |>
  add_case(escolaridade = "Total",
           frequencia = sum(tab$frequencia),
           freq_relativa = sum(tab$freq_relativa),
           porcentagem = sum(tab$porcentagem))
```

escolaridade	frequencia	freq_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 = "Escolaridade", y = "Porcentagem") +
  theme_gdocs() +
  coord_flip()
```

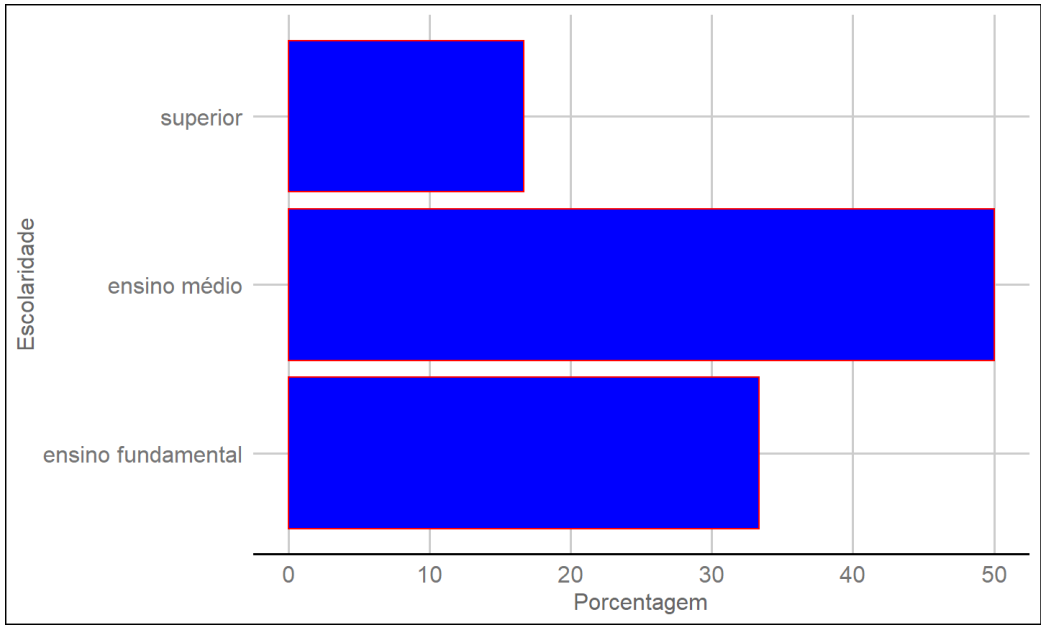


Gráfico no `R` com o *simplevis*

- **Pacote:** `gg_bar` do `simplevis` (instale com o comando `install.packages('simplevis')`).
- Parêmetros da função `gg_bar`:
  - `data`: *data frame* da variável tabulada.
  - `x_var`: variável do eixo `x`.
  - `y_var`: variável (numérica) do eixo `y`.
- Para mais detalhes, consulte a documentação oficial do *simplevis*: documentação de `gg_bar`.

```
tab <- tabyl(df_empresa, escolaridade)
tab <- as_tibble(tab)
tab <- tab |> rename(frequencia = n, freq_relativa = percent)
tab <-  tab |>   add_column(porcentagem = tab$freq_relativa*100)
```

```
library(simplevis)

gg_bar(tab, x_var = escolaridade, y_var = porcentagem)
```

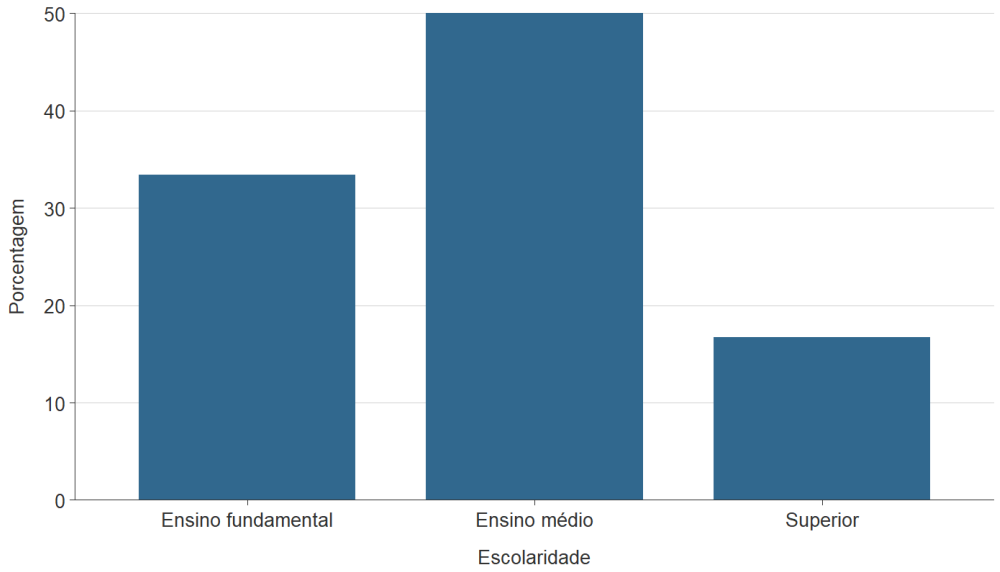


Tabela de frequência – Variável quantitativa discreta

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 frequência – Variável quantitativa discreta

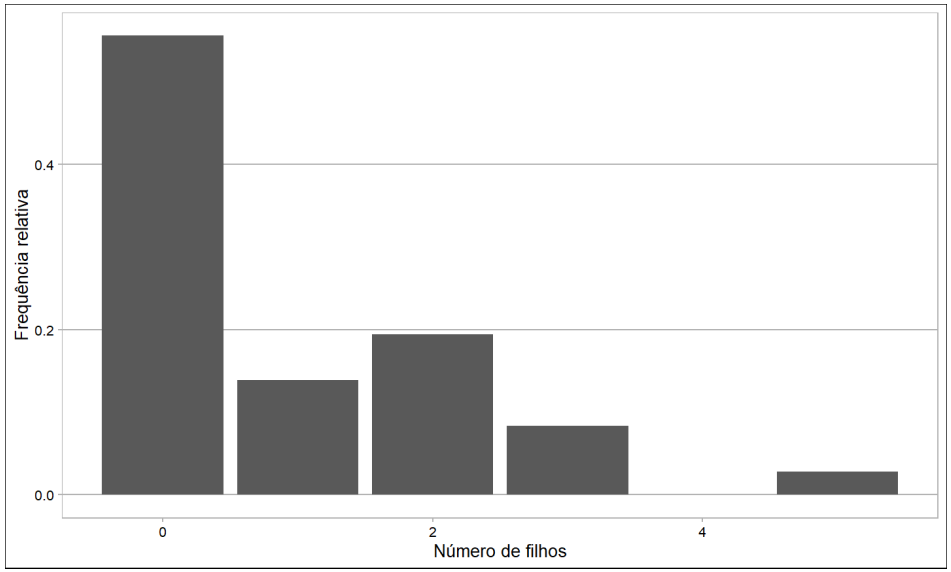
A primeira coisa que podemos fazer é construir a tabela de frequência.

```
df_empresa <- read_xlsx("../data/raw/empresa.xlsx")

tab <- tabyl(df_empresa, numero_filhos)
tab <- as_tibble(tab)
tab <- tab |> rename(frequencia = n,
                     freq_relativa = percent)
tab <-  tab |>
  add_column(porcentagem = tab$freq_relativa*100)

tab <- tab |> mutate(numero_filhos = as.character(numero_filhos))

tab <-  tab |>
  add_case(numero_filhos = "Total",
           frequencia = sum(tab$frequencia),
           freq_relativa = sum(tab$freq_relativa),
           porcentagem = sum(tab$porcentagem))
```

numero_filhos	frequencia	freq_relativa	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()
```



```
tab <- tabyl(df_empresa, numero_filhos)
tab <- as_tibble(tab)
tab <- tab |> rename(frequencia = n, freq_relativa = percent)
tab <-  tab |> add_column(porcentagem = tab$freq_relativa*100)
tab <- tab |> mutate(numero_filhos = as.character(numero_filhos))
```

```
library(simplevis)

gg_bar(tab, x_var = numero_filhos, y_var = porcentagem)
```

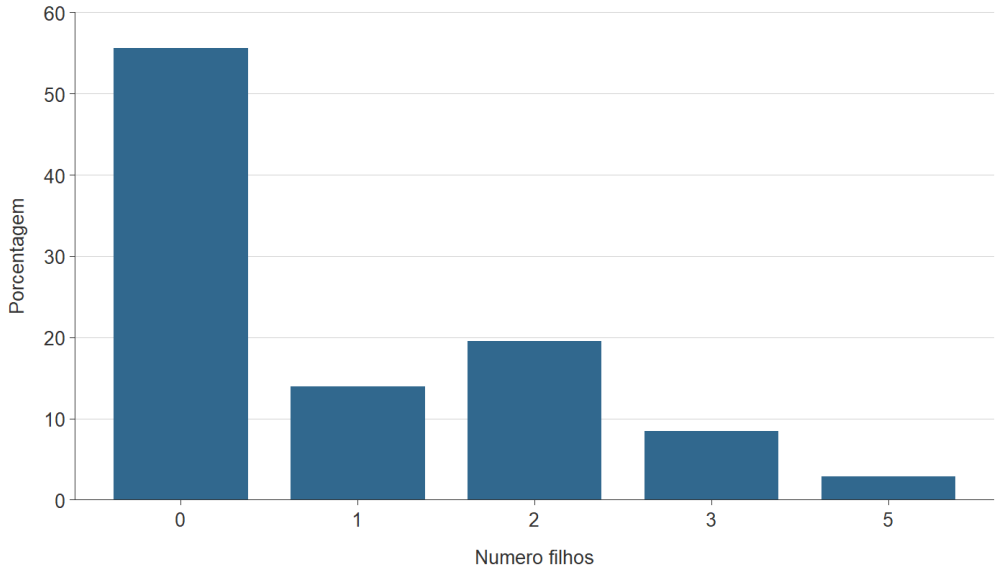


Tabela de frequência – Variável quantitativa contínua

- x: variável quantitativa contínua

Tabela de frequências para a variável quantitativa contínua.

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

Em que $\mathbf{min} = l_0 \leq l_1 \leq \cdots \leq l_{k-1} \leq l_k = \mathbf{max}$ ($\mathbf{min}$ é o menor valor do suporte da variável x e $\mathbf{max}$ 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 $l_0, l_1, \ldots, l_k$ quebram o suporte da variável x (*breakpoints*).

$l_0, l_1, \cdots, l_k$ são escolhidos de acordo com a teoria por trás da análise de dados (ou pelo regulador).

Recomendação: use $l_0, l_1, \cdots, l_k$ 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 frequência – Variável quantitativa contínua

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

k <- round(1 + log2(nrow(df_iris)))

sep_len_int <- cut(df_iris$Sepal.Length,
    breaks = k, include.lowest = T, right = F)

tab <- tabyl(sep_len_int)
tab <- as_tibble(tab)
tab <- tab |> rename(frequencia = n,
                     freq_relativa = percent)
tab <-  tab |>
  add_column(porcentagem = tab$freq_relativa*100)

tab <-  tab |>
  add_case(sep_len_int = "Total",
           frequencia = sum(tab$frequencia),
           freq_relativa = sum(tab$freq_relativa),
           porcentagem = sum(tab$porcentagem))
```

sep_len_int	frequencia	freq_relativa	porcentagem
[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

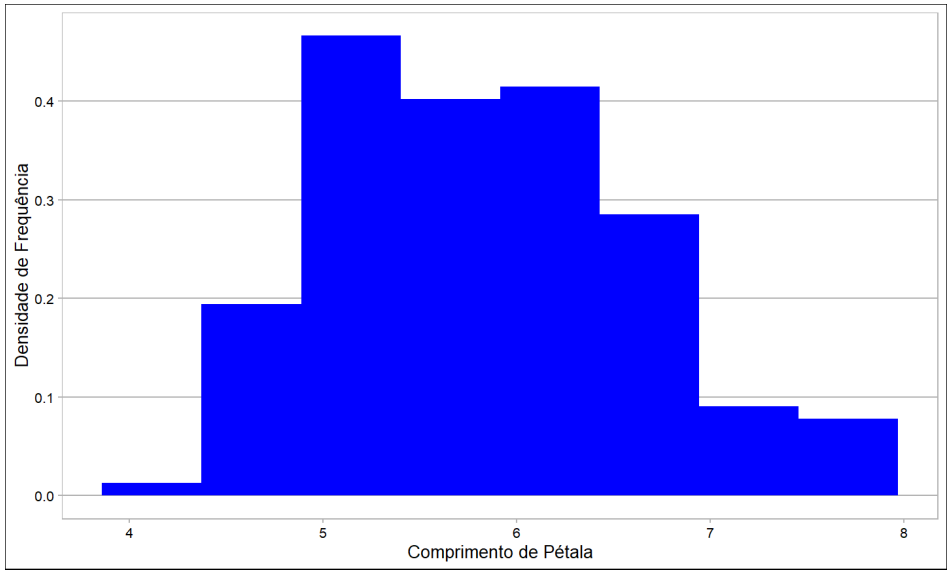
Histograma

Para variávieis quantitativas contínuas, geralmente não construímos gráficos de barras, e sim 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 $[l_{i-1}, l_i), i = 1, \ldots, n,$ será representada por um barra com área $f_i, i = 1, \ldots, n$.
- 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 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:** $\overline{x} = \dfrac{x_1 + \cdots + x_n}{n}$.
- **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:** $s^2 = \dfrac{(x_1 - \overline{X})^2 + \cdots + (x_n - \overline{X})^2}{n-1}$.
- **Desvio padrão:** $s = \sqrt{s^2}$ (mesma unidade dos dados).
- **Coeficiente de variação** $cv = \dfrac{s}{\overline{x}} \cdot 100\%$ (adimensional, ou seja, “sem unidade”).

Medidas resumo: exemplo

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

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

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

Medidas resumo: exemplo

Podemos usar a função `group_by` para calcular medidas resumo por categorias de uma variável qualitativa.

```
df_empresa <- read_xlsx("../data/raw/empresa.xlsx")

df_empresa |>
  group_by(escolaridade) |>
  summarise(media = mean(salario), md = median(salario), dp = sd(salario), cv = dp / media) |>
  gt() |>
  tab_header(
    title = "Medidas resumo por escolaridade.",
    subtitle = "Média, mediana, desvio padrão e coeficiente de variação em salários mínimos."
  ) |>
  cols_label(
    escolaridade = "Escolaridade",
    media = "Média salarial",
    md = "Salário mediano",
    dp = "Desvio padrão de salário",
    cv = "Coeficiente de variação"
  ) |>
  fmt_number(
    columns = c(media, md, dp, cv),
    decimals = 2,
    dec_mark = ",",
    sep_mark = "."
  )
```

Medidas resumo por escolaridade.

Média, mediana, desvio padrão e coeficiente de variação em salários mínimos.

Escolaridade	Média salarial	Salário mediano	Desvio padrão de salário	Coeficiente de variação
ensino fundamental	7,84	7,12	2,96	0,38
ensino médio	11,53	10,91	3,72	0,32
superior	16,48	16,74	4,50	0,27

```r
df_empresa <- read_xlsx("../data/raw/empresa.xlsx")

df_empresa |>
  group_by(escolaridade) |>
  summarise(media = mean(salario), md = median(salario), dp = sd(salario), cv = dp / media) |>
  gt() |>
  tab_header(
    title = "Medidas resumo da variável salário por categoria de escolaridade."
    #subtitle = "Média, desvio padrão, mediana e coeficiente de variação em salários mínimos."
  ) |>
  cols_label(
    escolaridade = "Escolaridade",
    media = "Média",
    md = "Mediana",
    dp = "Desvio padrão",
    cv = "Coeficiente de variação"
  ) |>
  fmt_number(
    columns = c(media, md, dp, cv),
    decimals = 2,
    dec_mark = ",",
    sep_mark = "."
  )
```

Medidas resumo da variável salário por categoria de escolaridade.

Escolaridade	Média	Mediana	Desvio padrão	Coeficiente de variação
ensino fundamental	7,84	7,12	2,96	0,38
ensino médio	11,53	10,91	3,72	0,32
superior	16,48	16,74	4,50	0,27