Probabilidade e Estatística com R

Fernando Náufel

(versão de 23/05/2023)

Sumário

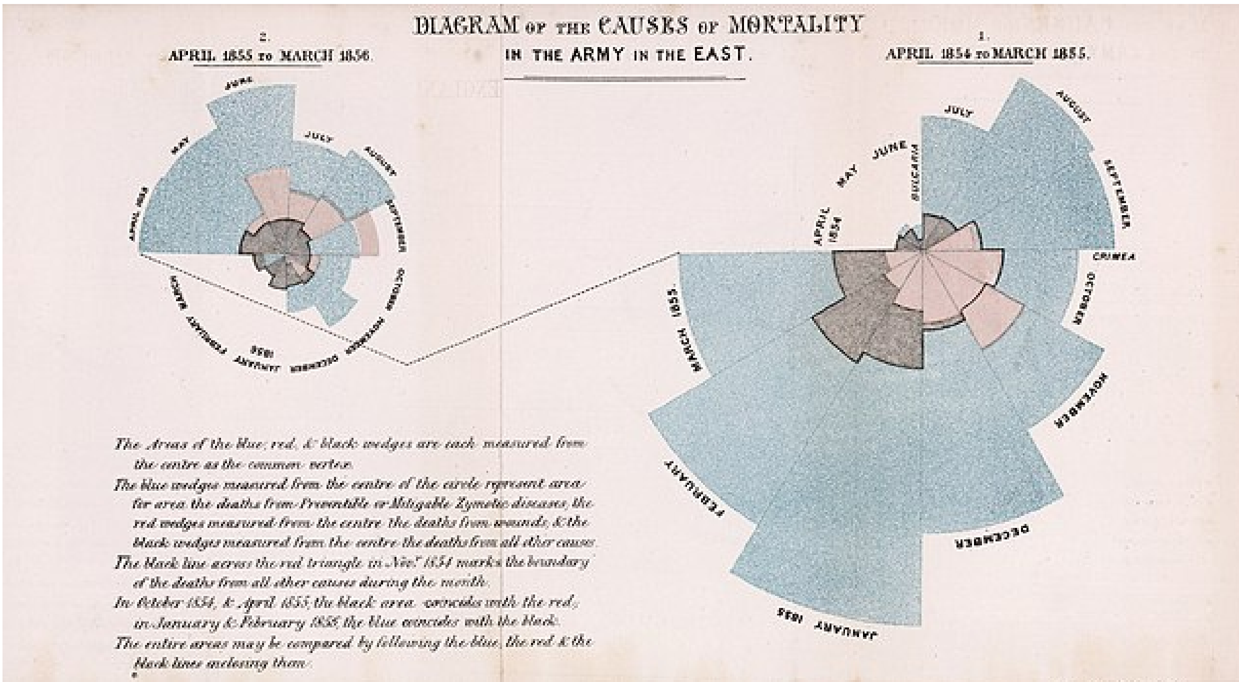
Apresentação

Atenção

Este material ainda está em construção.

Pode haver mudanças a qualquer momento.

Verifique, no rodapé da página *web* ou na capa do arquivo pdf, a data desta versão.



Este livro/*site* foi iniciado em 2020, durante a pandemia de COVID-19, quando a Universidade Federal Fluminense (UFF) funcionou em regime de ensino remoto durante mais de um ano.

Para atender os alunos do curso de Probabilidade e Estatística do curso de graduação em Ciência da Computação da UFF, decidi gravar aulas em vídeo e disponibilizar os arquivos usados nelas. Foram esses arquivos que deram origem a este livro/*site*.

Este livro/*site* foi construído para pessoas que já saibam programar, embora não necessariamente em R.

Para tirar o máximo proveito deste material, você deve fazer o seguinte:

1. Assistir aos vídeos contidos em cada capítulo. A *playlist* completa está em https://www.youtube.com/playlist?list=PL7SRLwLs7ocaV-Y1vrVU3W7mZnnS0qkWV.
2. Instalar o R no seu computador ou abrir uma conta no RStudio Cloud, para poder usar o R *online*. Você encontra instruções para fazer isto no capítulo de introdução a R.
3. Baixar, neste repositório do Github, o código-fonte deste livro/*site*, para poder rodar e alterar os exemplos.
4. Seguir os *links* para outras fontes *online* que abordam assuntos que não são cobertos em detalhes neste curso.
5. Fazer os exercícios. Ao longo do tempo, acrescentarei *links* para vídeos explicando as soluções.

Se você estiver lendo este material na *web*, você pode clicar nos comandos e funções que aparecem nos blocos de código em R para abrir páginas da documentação sobre eles.

Se você preferir ler este livro em pdf, ou se quiser imprimi-lo, faça o *download* do arquivo aqui.

Referências recomendadas

Em português

- Sillas Gonzaga, *Introdução a R para Visualização e Apresentação de Dados*, http://sillasgonzaga.com/material/curso_visualizacao/index.html
- Allan Vieira de Castro Quadros, *Introdução à Análise de Dados em R utilizando Tidyverse*, https://allanvc.github.io/book_IADR-T/
- Paulo Felipe de Oliveira, Saulo Guerra, Robert McDonnel, *Ciência de Dados com R – Introdução*, https://cdr.ibpad.com.br/index.html
- Curso R, *Ciência de Dados em R*, https://livro.curso-r.com/

Em inglês

- Garrett Grolemund, Hadley Wickham, *R for Data Science*, https://r4ds.had.co.nz/
- Chester Ismay, Albert Y. Kim, *A ModernDive into R and the Tidyverse*, https://moderndive.com/

Exercício

1. Pesquise sobre a imagem do início deste capítulo. Ela foi criada em 1858 por Florence Nightingale.

CAPÍTULO 1

O Que É Estatística?

1.1 Vídeo 1

https://youtu.be/6Q_XSoLCIpc

1.2 Exercícios

1. Você está interessado em estimar a altura de todos os homens da sua faculdade. Para isso, você decide medir as alturas de todos os homens da sua turma de Estatística.
   - Qual é a amostra?
   - Qual é a população?
2. Um instituto de pesquisa entrevista um grupo de 1000 pessoas, perguntando a cada uma se ela vai votar a favor do candidato A na próxima eleição. Dos entrevistados, 600 responderam que sim. A proporção $0{,}6$ (ou 60%) é uma estatística ou um parâmetro?
3. Você vê alguma diferença entre as cinco situações abaixo? Quais das situações são equivalentes em termos da probabilidade de conseguir 10 cartas do mesmo naipe?
   a. Usando um baralho normal, você retira 10 cartas e registra as cartas retiradas.

b. Usando um baralho normal, você repete a seguinte sequência de ações 10 vezes: retirar uma carta do baralho, registrar a carta retirada e repor a carta no baralho.

c. Usando uma caixa contendo todas as cartas de 1 milhão de baralhos reunidos, você retira 10 cartas e registra as cartas retiradas.

d. Usando uma caixa contendo todas as cartas de 1 milhão de baralhos reunidos, você repete a seguinte sequência de ações 10 vezes: retirar uma carta da caixa, registrar a carta retirada e repor a carta na caixa.

e. Usando um baralho *infinito*, você retira 10 cartas e registra as cartas retiradas.

f. Usando um baralho *infinito*, você repete a seguinte sequência de ações 10 vezes: retirar uma carta do baralho, registrar a carta retirada e repor a carta no baralho.

4. Qual a graça dos quadrinhos na Figura 1.1, que também aparecem no vídeo?

Figura 1.1: http://xkcd.com/552/

5. Qual a graça dos quadrinhos na Figura 1.2?

6. Veja este vídeo sobre o cavalo Hans:

https://youtu.be/G3VkCmdUfZE

Qual a relação entre esta história e a necessidade de duplo cegamento?

1.3 Vídeo 2

https://youtu.be/492VASxlDRo

LIMITAÇÕES DE ESTUDOS COM CEGAMENTO

Figura 1.2: http://xkcd.com/1462/

1.4 Exercícios

1. Por que não faz sentido calcular a média dos CEPs de um grupo de pessoas?
2. Uma temperatura de -40 graus Celsius é igual a uma temperatura de -40 graus Fahrenheit?
3. Uma temperatura de zero graus Celsius é igual a uma temperatura de zero graus Fahrenheit?
4. Uma variação de temperatura de 1 grau Celsius é igual a uma variação de temperatura de 1 grau Fahrenheit?
5. Um saldo bancário de zero reais é igual a um saldo bancário de zero dólares?
6. Um produto de 1 milhão de reais custa o mesmo que um produto de 1 milhão de dólares?
7. Meses representados por números de 1 a 12 são dados de que nível?

CAPÍTULO 2

Introdução a R

2.1 Vídeo 1

https://youtu.be/1kXQDNqm41c

2.2 Usando o RStudio

Leia uma introdução ao R e ao RStudio no livro Ciência de Dados com R.

Ali, você vai encontrar *exemplos* e **exercícios**.

2.3 RMarkdown

Para ver uma referência sobre a sintaxe do RMarkdown, vá ao menu `Help` do RStudio, escolha `Cheatsheets` e, a seguir, `R Markdown Cheat Sheet` ou `R Markdown Reference Guide`.

2.4 Apresentação

R é várias linguagens em uma:

- R base,
- OO (S3),
- OO (S4),
- Tidyverse (pacote usado para Ciência de Dados e gráficos).

2.5 Tudo é vetor

- Usamos a função `c()` (*concatenate*) para criar vetores:

```
vetor <- c(1, 2, 4, 7, 0, -1)
vetor
```

```
## [1]  1  2  4  7  0 -1
```

- Em um vetor, todos os elementos precisam ser do mesmo tipo.
- Mesmo que você use `c(c(...), c(...))`, o vetor criado vai ter um único nível. Não existem vetores aninhados.

```
v1 <- c(1, 2)
v2 <- c(3, 4, 5)
v3 <- c(v1, v2)
v3
```

```
## [1] 1 2 3 4 5
```

- Mais adiante, veremos como criar listas, que podem ter elementos de tipos diferentes e sublistas aninhadas.
- Outras maneiras de criar vetores:
  - O operador `:` constrói sequências:

```
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
10:1
```

```
##  [1] 10  9  8  7  6  5  4  3  2  1
```

```
0.5:10.5
```

```
##  [1]  0,5  1,5  2,5  3,5  4,5  5,5  6,5  7,5  8,5  9,5 10,5
```

```
0.5:10
```

```
##  [1] 0,5 1,5 2,5 3,5 4,5 5,5 6,5 7,5 8,5 9,5
```

- A função `seq` permite especificar um incremento diferente de 1 e -1:

```
seq(1, 10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
seq(10, 1)
```

```
##  [1] 10  9  8  7  6  5  4  3  2  1
```

```
seq(1, 10, 0.5)
```

```
##  [1]  1,0  1,5  2,0  2,5  3,0  3,5  4,0  4,5  5,0  5,5  6,0  6,5  7,0
## [14]  7,5  8,0  8,5  9,0  9,5 10,0
```

- A função `rep` cria vetores com elementos repetidos:

```
rep(1, 10)
```

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

```
rep(c(1, 2), 10)
```

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

```
rep(c(1, 2), each = 10)
```

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

```
rep(c(1, 2), c(3, 4))
```

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

• O número que aparece na saída, entre colchetes, é o índice do primeiro elemento daquela linha:

```
rep(1, 1000)
```

```
##    [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##   [33] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##   [65] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
##   [97] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [129] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [161] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [193] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [225] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [257] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [289] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [321] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [353] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [385] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [417] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [449] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [481] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [513] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [545] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [577] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [609] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [641] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [673] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [705] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [737] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [769] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [801] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [833] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [865] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [897] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [929] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [961] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [993] 1 1 1 1 1 1 1 1
```

- Vetores começam no índice 1!!!!
- Tudo é vetor:

```r
10
```

```
## [1] 10
```

```r
is.vector(10)
```

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

```r
length(10)
```

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

- O valor `NA` significa ausência de informação:

```r
notas <- c(10, 8, NA, 7, 10)
notas
```

```
## [1] 10  8 NA  7 10
```

2.6 Operações com vetores

- Adição e outras operações aritméticas::

```r
v3 <- c(10, 20)
```

```r
v1
```

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

```r
v2
```

```
## [1] 3 4 5
```

```r
v3
```

```
## [1] 10 20
```

  - R recicla o vetor mais curto, mas avisa quando o comprimento do maior vetor não é múltiplo inteiro do comprimento do menor vetor.

    ```r
    v1 + v3
    ```

    ```
    ## [1] 11 22
    ```

    ```r
    v1 + v2
    ```

    ```
    ## Warning in v1 + v2: longer object length is not a multiple of shorter
    ## object length
    ```

    ```
    ## [1] 4 6 6
    ```

    ```r
    v1 <- c(1, 2)
    v4 <- c(3, 4, 5, 6, 7, 8)
    v1 + v4
    ```

    ```
    ## [1]  4  6  6  8  8 10
    ```

    ```r
    v1 - v4
    ```

    ```
    ## [1] -2 -2 -4 -4 -6 -6
    ```

```
v1 * v4
```

```
## [1]  3  8  5 12  7 16
```

```
v1 / v4
```

```
## [1] 0,3333333 0,5000000 0,2000000 0,3333333 0,1428571 0,2500000
```

- Funções úteis para vetores:
  - Somar todos os elementos:

```
sum(v4)
```

```
## [1] 33
```

  - O valor `NA` se propaga em operações aritméticas:

```
notas
```

```
## [1] 10  8 NA  7 10
```

```
sum(notas)
```

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

  - Para ignorar os valores `NA`, use o argumento `na.rm`:

```
sum(notas, na.rm = TRUE)
```

```
## [1] 35
```

  - Média de todos os elementos:

```
mean(notas, na.rm = TRUE)
```

```
## [1] 8,75
```

```
sum(notas, na.rm = TRUE) / 4
```

```
## [1] 8,75
```

2.7 Indexação

- Com um valor inteiro:

```
v5 <- 1:50 * 4
v5
```

```
##  [1]   4   8  12  16  20  24  28  32  36  40  44  48  52  56  60  64
## [17]  68  72  76  80  84  88  92  96 100 104 108 112 116 120 124 128
## [33] 132 136 140 144 148 152 156 160 164 168 172 176 180 184 188 192
## [49] 196 200
```

```
v5[10]
```

```
## [1] 40
```

- Com um vetor de inteiros:

```
v5[1:10]
```

```
##  [1]  4  8 12 16 20 24 28 32 36 40
```

```
v5[c(2, 10, 13, 30)]
```

```
## [1]   8  40  52 120
```

```
v5[seq(2, 50, 2)]
```

```
##  [1]   8  16  24  32  40  48  56  64  72  80  88  96 104 112 120 128
## [17] 136 144 152 160 168 176 184 192 200
```

- Com um vetor de booleanos, os elementos indexados por `TRUE` são selecionados:

```
v4
```

```
## [1] 3 4 5 6 7 8
```

```
v4[c(TRUE, TRUE, FALSE, FALSE, TRUE, FALSE)]
```

```
## [1] 3 4 7
```

- Uma condição produz um vetor de booleanos:

```
v4 > 5
```

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

- Logo, podemos indexar com uma condição:

```
v4[ v4 > 5 ]
```

```
## [1] 6 7 8
```

```
v4[ v4 %% 2 != 0 ]
```

```
## [1] 3 5 7
```

```
notas[!is.na(notas)]
```

```
## [1] 10  8  7 10
```

- Para especificar os elementos a não selecionar, use índices negativos.

```
v5
```

```
##  [1]   4   8  12  16  20  24  28  32  36  40  44  48  52  56  60  64
## [17]  68  72  76  80  84  88  92  96 100 104 108 112 116 120 124 128
## [33] 132 136 140 144 148 152 156 160 164 168 172 176 180 184 188 192
## [49] 196 200
```

```
v5[-1]
```

```
##  [1]   8  12  16  20  24  28  32  36  40  44  48  52  56  60  64  68
## [17]  72  76  80  84  88  92  96 100 104 108 112 116 120 124 128 132
## [33] 136 140 144 148 152 156 160 164 168 172 176 180 184 188 192 196
## [49] 200
```

```
v5[-c(1, 4, 20)]
```

```
##  [1]   8  12  20  24  28  32  36  40  44  48  52  56  60  64  68  72
## [17]  76  84  88  92  96 100 104 108 112 116 120 124 128 132 136 140
## [33] 144 148 152 156 160 164 168 172 176 180 184 188 192 196 200
```

2.8

Vídeo 2

https://youtu.be/3GEc1oiKDrU

2.9

Simular lançamentos de dados

- Vamos criar um dado de 6 lados. Basta um vetor:

```
(dado <- 1:6)
```

```
## [1] 1 2 3 4 5 6
```

- Para lançar este dado uma vez, usamos `sample`:

```
sample(dado, 1)
```

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

- Para lançar o dado várias vezes:

```
n <- 6
sample(dado, n, replace = TRUE)
```

```
## [1] 4 3 3 6 4 2
```

- Observe que, para permitir que o mesmo valor apareça mais de uma vez, precisamos usar `replace = TRUE` — a amostragem será feita com reposição.
- Uma função para retornar a soma de 2 dados:

```
lancar2 <- function() {

  dado <- 1:6
  lancamentos <- sample(dado, size = 2, replace = TRUE)
  sum(lancamentos)

}
```

```
lancar2()
```

```
## [1] 12
```

- Vamos generalizar a função:
  - O número de lados do dado é passado como argumento.

- A quantidade de dados é passada como argumento.

```
lancar <- function(n = 2, k = 6) {

  dado <- 1:k
  lancamentos <- sample(dado, size = n, replace = TRUE)
  sum(lancamentos)

}
```

```
lancar()
```

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

```
lancar(n = 4, k = 10)
```

```
## [1] 19
```

- Vamos lançar os 2 dados 10 mil vezes, usando a função `replicate`:

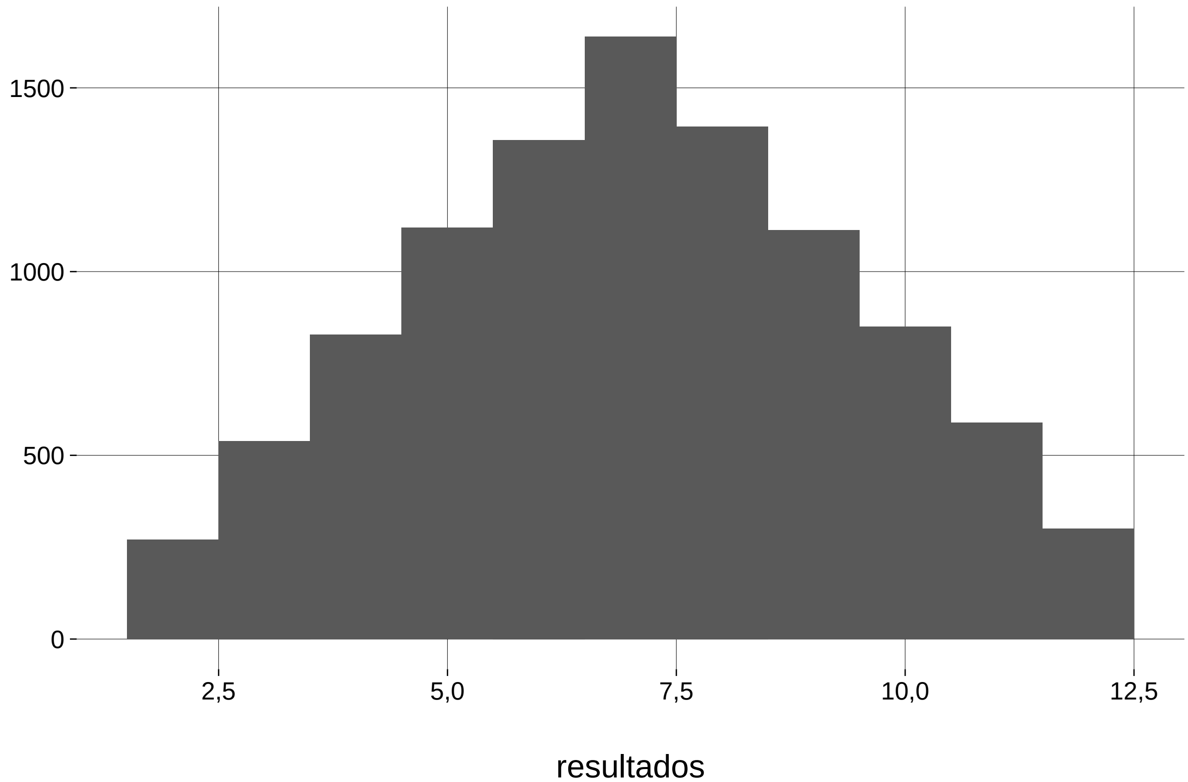
```
resultados <- replicate(1e4, lancar())
```

2.10 Visualização

- A função `qplot`, do pacote `tidyverse`, produz um gráfico adequado aos argumentos recebidos — aqui, um histograma:

```
qplot(resultados, bins = 11)
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```



- Em um capítulo sobre visualização, você vai aprender a configurar melhor a aparência de gráficos como este.

2.11 Dados viciados

- Vamos modificar a função `lancar` para receber um vetor com as probabilidades dos lados:

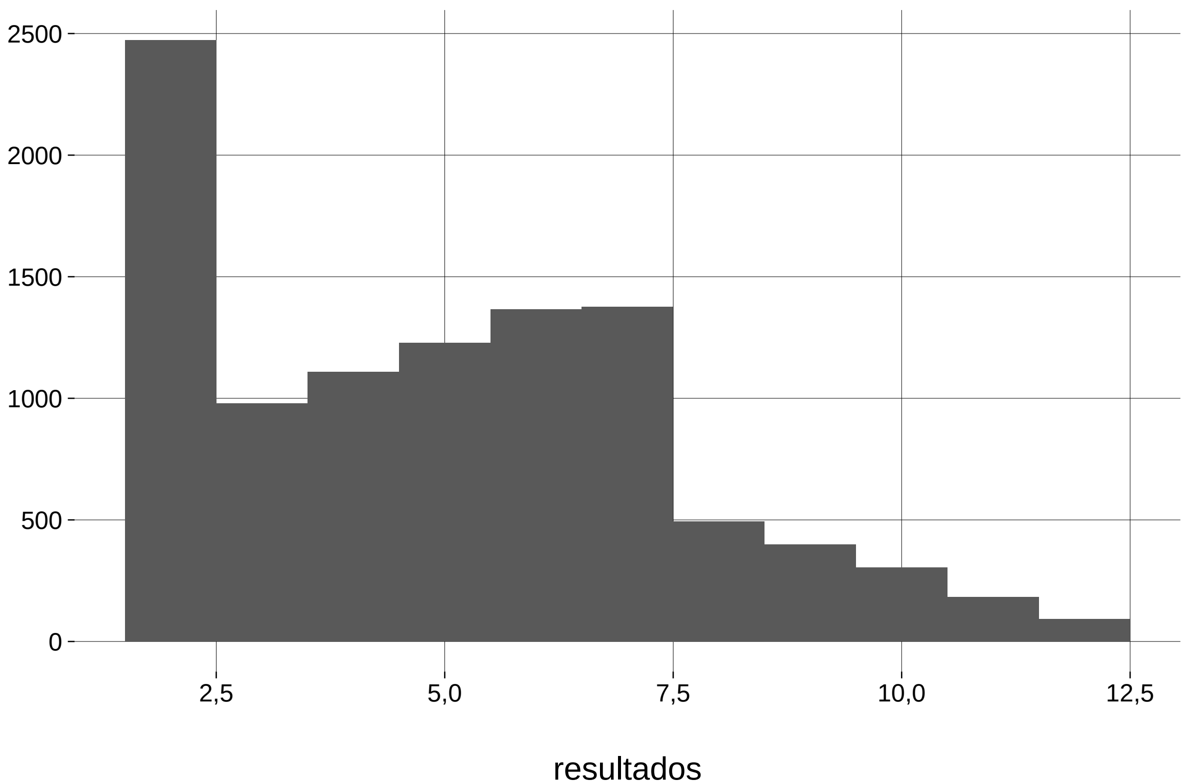
```
lancar <- function(
  qtde = 2,
  lados = 6,
  probs = rep(1/lados, lados)
) {

  dado <- 1:lados
  resultados <- sample(
    dado,
    size = qtde,
    replace = TRUE,
    prob = probs
  )
  sum(resultados)
```

```
}
```

- Lançando dados viciados:

```
resultados <- replicate(1e4, lancar(probs = c(1/2, rep(1/10, 5))))
```

```
qplot(resultados, bins = 11)
```



2.12 Exercícios

1. Para criar sua conta no RStudio Cloud, acesse https://rstudio.cloud/.
2. Se você preferir instalar o R no seu computador, acesse
   - https://cran.r-project.org/ para baixar e instalar o R, e
   - https://rstudio.com/products/rstudio/download/ para baixar e instalar o RStudio, um IDE específico para R.
3. Abra o RStudio Cloud ou o seu RStudio instalado localmente.
4. Crie um novo projeto. Sempre trabalhe em projetos para ter seus arquivos organizados.

5. Para instalar o `swirl` (pacote do R para exercícios interativos), execute o seguinte comando no console do RStudio:

```
install.packages("swirl")
```

6. Para instalar os exercícios de introdução a R, execute os seguintes comandos no console do RStudio:

```
library(swirl)
install_course_github('fnaufel', 'introR')
```

7. Mude o idioma para português e execute o `swirl`.

```
select_language('portuguese', append_rprofile = TRUE)
swirl()
```

8. Na primeira execução, você vai precisar se identificar (qualquer nome serve). Com essa identificação, o `swirl` vai registrar o seu progresso nas lições.

9. No `swirl`, as perguntas são mostradas no console. Você também deve responder no console.

10. Às vezes, um *script* será aberto no editor de textos para que você complete um programa. Quando seu programa estiver pronto, salve o arquivo e digite `submit()` no console para o `swirl` processar o *script*.

11. O `swirl` dá instruções claras no console. Na dúvida, digite `info()` no *prompt* do R (>).

12. Se, em vez do *prompt* do R, o console mostrar reticências (`...`), tecle *Enter*.

13. Se nada funcionar, tecle *ESC*.

14. Para sair do `swirl()`, digite `bye()` no *prompt* do R.

15. Para voltar para os exercícios, digite

```
library(swirl)
swirl()
```

16. Se, quando você tentar instalar os exercícios, acontecer um erro, desinstale todos os cursos com os comandos

```
library(swirl)
uninstall_all_courses()
```

e tente instalar os exercícios novamente.

CAPÍTULO 3

Introdução ao `tidyverse`

Busque mais informações sobre os pacotes que compõem o `tidyverse` nas referências recomendadas.

3.1 Criando uma *tibble*

- Uma *tibble* é uma tabela retangular.
- Cada coluna é um vetor:

```
cores <- tibble(
  pessoa = c('João', 'Maria', 'Pedro', 'Ana'),
  'cor favorita' = c('azul', 'rosa', 'preto', 'branco')
)
```

```
cores
```

```
## # A tibble: 4 x 2
##   pessoa `cor favorita`
##   <chr>  <chr>
## 1 João   azul
## 2 Maria  rosa
```

```
## 3 Pedro  preto
## 4 Ana    branco
```

- Isto é um pouco diferente da maneira como estamos acostumados a ver tabelas (como uma coleção de linhas, em vez de uma coleção de colunas).
- A função `tribble` permite a entrada de forma mais natural, linha a linha. Lembre-se de usar ~ antes dos nomes das colunas.

```
cores <- tribble(
  ~pessoa, ~'cor favorita',
   "João",           "azul",
  "Maria",           "rosa",
  "Pedro",          "preto",
    "Ana",         "branco"
)
```

```
cores
```

```
## # A tibble: 4 x 2
##   pessoa `cor favorita`
##   <chr>  <chr>
## 1 João   azul
## 2 Maria  rosa
## 3 Pedro  preto
## 4 Ana    branco
```

Mesmo que você crie uma *tibble* linha a linha, o R vai continuar tratando sua *tibble* como uma coleção de colunas.

É importante lembrar disto para entender a forma como R manipula estas tabelas.

- Se uma coluna não puder ser armazenada em um vetor, a coluna será uma lista (com vetores como elementos):

```
cores <- tibble(
  pessoa = c('João', 'Maria', 'Pedro', 'Ana'),
  'cor favorita' = list(
    c('azul', 'roxo'),
    c('rosa', 'magenta'),
    NA,
    'branco'
  )
)
```

```
cores
```

```
## # A tibble: 4 x 2
##   pessoa `cor favorita`
##   <chr>  <list>
## 1 João   <chr [2]>
## 2 Maria  <chr [2]>
## 3 Pedro  <lgl [1]>
## 4 Ana    <chr [1]>
```

- Use `View()` para examinar interativamente o conteúdo de uma coluna-lista:

```
cores %>% View()
```

3.2 Operador de *pipe* (%>%)

- O `tidyverse` inclui o pacote `magrittr`, que contém o operador `%>%`, chamado *pipe*.[1]
- A idéia é facilitar a leitura de composições de funções. O código

```
y <- h(g(f(x)))
```

pode ser escrito como

```
y <- x %>% f() %>% g() %>% h()
```

- Esta segunda versão é mais fiel à ordem em que as operações acontecem.
- Na verdade, R tem um operador de atribuição para a direita, mas poucas pessoas recomendam usá-lo:

```
x %>% f() %>% g() %>% h() -> y
```

- Se `f`, `g` e `h` forem funções de um argumento só, os parênteses podem ser omitidos:

```
y <- x %>% f %>% g %>% h
```

- Se a função `f` tiver outros argumentos, escreva-os normalmente na chamada a `f`:

```
y <- x %>% mean(na.rm = TRUE)
```

- O *pipe* `EXP %>% f(...)` sempre insere o resultado da expressão `EXP` como o primeiro argumento da função `f`.
- Se você precisar que o resultado da expressão `EXP` seja inserido em outra posição na lista de argumentos de `f`, use um ponto "`.`" para isso:

[1] Por que o nome do pacote e o nome do operador formam um trocadilho?

```
x %>% consultar(df, .)
```

equivale a

```
consultar(df, x)
```

3.3 Formato *tidy*

- Nossa última versão da *tibble* `cores` é um pouco mais complexa do que deveria ser:

```
cores
```

```
## # A tibble: 4 x 2
##   pessoa `cor favorita`
##   <chr>  <list>
## 1 João   <chr [2]>
## 2 Maria  <chr [2]>
## 3 Pedro  <lgl [1]>
## 4 Ana    <chr [1]>
```

- O formato *tidy* exige que
  1. Cada linha da *tibble* corresponda a uma observação sobre um indivíduo,
  2. Cada coluna corresponda a uma variável observada, e
  3. Cada célula contenha um valor da variável.
- Na *tibble* `cores`, a primeira e a segunda exigências são satisfeitas, mas a terceira não, pois algumas células contém valores múltiplos.
- A *tibble* não está no formato *tidy*.
- Podemos "extrair" estes vetores "aninhados" usando o comando `unnest`, do pacote `tidyr`:

```
cores <- cores %>%
  unnest(`cor favorita`)
```

```
cores
```

```
## # A tibble: 6 x 2
##   pessoa `cor favorita`
##   <chr>  <chr>
## 1 João   azul
## 2 João   roxo
## 3 Maria  rosa
```

```
## 4 Maria  magenta
## 5 Pedro  <NA>
## 6 Ana    branco
```

- A maioria das funções do `tidyverse` exige que as *tibbles* estejam neste formato *tidy*.
- Um exemplo mais complexo é o *dataset* `billboard`, com as seguintes colunas (para cada música que estava no *top 100* da Billboard no ano de 2000):
    - Nome do artista ou banda;
    - Nome da música;
    - Data em que a música entrou no *top 100* da Billboard;
    - Para cada uma das 76 semanas seguintes, a posição da música no *top 100*.

```
billboard %>% glimpse()
```

```
## Rows: 317
## Columns: 79
## $ artist       <chr> "2 Pac", "2Ge+her", "3 Doors Down", "3 Doors Dow~
## $ track        <chr> "Baby Don't Cry (Keep...", "The Hardest Part Of ~
## $ date.entered <date> 2000-02-26, 2000-09-02, 2000-04-08, 2000-10-21,~
## $ wk1          <dbl> 87, 91, 81, 76, 57, 51, 97, 84, 59, 76, 84, 57, ~
## $ wk2          <dbl> 82, 87, 70, 76, 34, 39, 97, 62, 53, 76, 84, 47, ~
## $ wk3          <dbl> 72, 92, 68, 72, 25, 34, 96, 51, 38, 74, 75, 45, ~
## $ wk4          <dbl> 77, NA, 67, 69, 17, 26, 95, 41, 28, 69, 73, 29, ~
## $ wk5          <dbl> 87, NA, 66, 67, 17, 26, 100, 38, 21, 68, 73, 23,~
## $ wk6          <dbl> 94, NA, 57, 65, 31, 19, NA, 35, 18, 67, 69, 18, ~
## $ wk7          <dbl> 99, NA, 54, 55, 36, 2, NA, 35, 16, 61, 68, 11, 2~
## $ wk8          <dbl> NA, NA, 53, 59, 49, 2, NA, 38, 14, 58, 65, 9, 17~
## $ wk9          <dbl> NA, NA, 51, 62, 53, 3, NA, 38, 12, 57, 73, 9, 17~
## $ wk10         <dbl> NA, NA, 51, 61, 57, 6, NA, 36, 10, 59, 83, 11, 1~
## $ wk11         <dbl> NA, NA, 51, 61, 64, 7, NA, 37, 9, 66, 92, 1, 17,~
## $ wk12         <dbl> NA, NA, 51, 59, 70, 22, NA, 37, 8, 68, NA, 1, 3,~
## $ wk13         <dbl> NA, NA, 47, 61, 75, 29, NA, 38, 6, 61, NA, 1, 3,~
## $ wk14         <dbl> NA, NA, 44, 66, 76, 36, NA, 49, 1, 67, NA, 1, 7,~
## $ wk15         <dbl> NA, NA, 38, 72, 78, 47, NA, 61, 2, 59, NA, 4, 10~
## $ wk16         <dbl> NA, NA, 28, 76, 85, 67, NA, 63, 2, 63, NA, 8, 17~
## $ wk17         <dbl> NA, NA, 22, 75, 92, 66, NA, 62, 2, 67, NA, 12, 2~
## $ wk18         <dbl> NA, NA, 18, 67, 96, 84, NA, 67, 2, 71, NA, 22, 2~
## $ wk19         <dbl> NA, NA, 18, 73, NA, 93, NA, 83, 3, 79, NA, 23, 2~
## $ wk20         <dbl> NA, NA, 14, 70, NA, 94, NA, 86, 4, 89, NA, 43, 4~
## $ wk21         <dbl> NA, NA, 12, NA, NA, NA, NA, NA, 5, NA, NA, 44, 4~
## $ wk22         <dbl> NA, NA, 7, NA, NA, NA, NA, NA, 5, NA, NA, NA, 50~
## $ wk23         <dbl> NA, NA, 6, NA, NA, NA, NA, NA, 6, NA, NA, NA, NA~
## $ wk24         <dbl> NA, NA, 6, NA, NA, NA, NA, NA, 9, NA, NA, NA, NA~
## $ wk25         <dbl> NA, NA, 6, NA, NA, NA, NA, NA, 13, NA, NA, NA, N~
## $ wk26         <dbl> NA, NA, 5, NA, NA, NA, NA, NA, 14, NA, NA, NA, N~
```

```
## $ wk27        <dbl> NA, NA, 5, NA, NA, NA, NA, NA, 16, NA, NA, NA, N~
## $ wk28        <dbl> NA, NA, 4, NA, NA, NA, NA, NA, 23, NA, NA, NA, N~
## $ wk29        <dbl> NA, NA, 4, NA, NA, NA, NA, NA, 22, NA, NA, NA, N~
## $ wk30        <dbl> NA, NA, 4, NA, NA, NA, NA, NA, 33, NA, NA, NA, N~
## $ wk31        <dbl> NA, NA, 4, NA, NA, NA, NA, NA, 36, NA, NA, NA, N~
## $ wk32        <dbl> NA, NA, 3, NA, NA, NA, NA, NA, 43, NA, NA, NA, N~
## $ wk33        <dbl> NA, NA, 3, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ wk34        <dbl> NA, NA, 3, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ wk35        <dbl> NA, NA, 4, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ wk36        <dbl> NA, NA, 5, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ wk37        <dbl> NA, NA, 5, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ wk38        <dbl> NA, NA, 9, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ wk39        <dbl> NA, NA, 9, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ wk40        <dbl> NA, NA, 15, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk41        <dbl> NA, NA, 14, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk42        <dbl> NA, NA, 13, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk43        <dbl> NA, NA, 14, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk44        <dbl> NA, NA, 16, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk45        <dbl> NA, NA, 17, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk46        <dbl> NA, NA, 21, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk47        <dbl> NA, NA, 22, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk48        <dbl> NA, NA, 24, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk49        <dbl> NA, NA, 28, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk50        <dbl> NA, NA, 33, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk51        <dbl> NA, NA, 42, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk52        <dbl> NA, NA, 42, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk53        <dbl> NA, NA, 49, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk54        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk55        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk56        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk57        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk58        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk59        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk60        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk61        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk62        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk63        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk64        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk65        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk66        <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk67        <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk68        <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk69        <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk70        <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk71        <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk72        <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk73        <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk74        <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
```

```
## $ wk75         <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ wk76         <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
```

- Vamos renomear as colunas:

```
bb <- billboard %>%
  rename(
    artista = artist,
    musica = track,
    entrou = date.entered
  )
```

```
bb %>% head()
```

```
## # A tibble: 6 x 79
##   artista    musica entrou       wk1   wk2   wk3   wk4   wk5   wk6   wk7
##   <chr>      <chr>  <date>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 2 Pac      Baby ~ 2000-02-26    87    82    72    77    87    94    99
## 2 2Ge+her    The H~ 2000-09-02    91    87    92    NA    NA    NA    NA
## 3 3 Doors ~ Krypt~ 2000-04-08    81    70    68    67    66    57    54
## 4 3 Doors ~ Loser  2000-10-21    76    76    72    69    67    65    55
## 5 504 Boyz  Wobbl~ 2000-04-15    57    34    25    17    17    31    36
## 6 98^0      Give ~ 2000-08-19    51    39    34    26    26    19     2
## # i 69 more variables: wk8 <dbl>, wk9 <dbl>, wk10 <dbl>, wk11 <dbl>,
## #   wk12 <dbl>, wk13 <dbl>, wk14 <dbl>, wk15 <dbl>, wk16 <dbl>,
## #   wk17 <dbl>, wk18 <dbl>, wk19 <dbl>, wk20 <dbl>, wk21 <dbl>,
## #   wk22 <dbl>, wk23 <dbl>, wk24 <dbl>, wk25 <dbl>, wk26 <dbl>,
## #   wk27 <dbl>, wk28 <dbl>, wk29 <dbl>, wk30 <dbl>, wk31 <dbl>,
## #   wk32 <dbl>, wk33 <dbl>, wk34 <dbl>, wk35 <dbl>, wk36 <dbl>,
## #   wk37 <dbl>, wk38 <dbl>, wk39 <dbl>, wk40 <dbl>, wk41 <dbl>, ...
```

- O que é uma observação neste conjunto de dados?

  A posição, em uma semana, de uma música que esteve no *top* 100 da *Billboard* durante o ano *2000*.

- Quais são as variáveis que qualificam cada observação?

  - O artista,
  - O título da música,
  - A posiçao da música no *top* 100 da *Billboard* em cada uma das 76 semanas depois que ela entrou na lista.

- Este último item é complexo, e o criador da *tibble* decidiu criar uma coluna por semana.

- Uma decisão ruim, pois existe informação embutida nos nomes das colunas. A coluna `wk68` corresponde à posição da música na semana 68 após ela entrar na lista, mas o número da semana só aparece no nome da coluna!

- Isto nunca deve acontecer. A informação deve sempre estar nas células.
- Vamos simplificar as coisas criando duas colunas:
  - `semana`, com o número da semana; perceba que esta informação vem dos nomes das colunas,
  - `pos`, com a posição da música naquela semana; esta informação vem das células.
- A *tibble*, que antes era larga, vai ser mais estreita e mais longa.
- A função `pivot_longer`, do pacote `tidyr`, vai fazer o trabalho — inclusive extraindo os números das semanas dos nomes das colunas:

```
bb_tidy <- bb %>%
  pivot_longer(
    wk1:wk76,
    names_to = 'semana',
    names_prefix = 'wk',
    names_transform = list(
      semana = as.integer
    ),
    values_to = 'pos'
  )
```

```
bb_tidy
```

```
## # A tibble: 24.092 x 5
##   artista musica                  entrou     semana   pos
##   <chr>   <chr>                   <date>      <int> <dbl>
## 1 2 Pac   Baby Don't Cry (Keep... 2000-02-26      1    87
## 2 2 Pac   Baby Don't Cry (Keep... 2000-02-26      2    82
## 3 2 Pac   Baby Don't Cry (Keep... 2000-02-26      3    72
## 4 2 Pac   Baby Don't Cry (Keep... 2000-02-26      4    77
## 5 2 Pac   Baby Don't Cry (Keep... 2000-02-26      5    87
## 6 2 Pac   Baby Don't Cry (Keep... 2000-02-26      6    94
## # i 24.086 more rows
```

- O R só mostra, por *default*, as 1000 primeiras linhas de uma *tibble*.
- Na verdade, o número de linhas da tabela original era

```
bb %>% nrow()
```

```
## [1] 317
```

- O número de linhas, depois de `pivot_longer`, ficou:

```
bb_tidy %>% nrow()
```

```
## [1] 24092
```

- Existem linhas onde `pos` tem o valor `NA`. São resultado da organização original dos dados, onde o `NA` indicava que a música não estava no *top* 100 naquela semana.
- No novo formato, a ausência da linha com aquele número de semana já basta para indicar isto. Então, vamos eliminar as linhas onde `pos` é `NA`.
- A função `filter` mantém as linhas que satisfazem a condição dada; por isso, a condição é "`pos` não é `NA`":

```
bb_tidy <- bb_tidy %>%
  filter(!is.na(pos))
```

```
bb_tidy
```

```
## # A tibble: 5.307 x 5
##   artista musica                  entrou     semana   pos
##   <chr>   <chr>                   <date>      <int> <dbl>
## 1 2 Pac   Baby Don't Cry (Keep... 2000-02-26      1    87
## 2 2 Pac   Baby Don't Cry (Keep... 2000-02-26      2    82
## 3 2 Pac   Baby Don't Cry (Keep... 2000-02-26      3    72
## 4 2 Pac   Baby Don't Cry (Keep... 2000-02-26      4    77
## 5 2 Pac   Baby Don't Cry (Keep... 2000-02-26      5    87
## 6 2 Pac   Baby Don't Cry (Keep... 2000-02-26      6    94
## # i 5.301 more rows
```

- O número de linhas ficou

```
bb_tidy %>% nrow()
```

```
## [1] 5307
```

3.3.1 Exercícios

- Todas as semanas deste conjunto de dados são do ano 2000?
- Qual é o tipo do primeiro argumento da função `filter()`?

3.4 Manipulando os dados

3.4.1 Criando novas colunas: `mutate`, `transmute`

- O *data frame*[2] `cars` tem dados (de 1920!) sobre as distâncias de frenagem (em pés) de um carro viajando a diversas velocidades (em milhas por hora):

```
cars
```

```
## # A tibble: 50 x 2
##   speed  dist
##   <dbl> <dbl>
## 1     4     2
## 2     4    10
## 3     7     4
## 4     7    22
## 5     8    16
## 6     9    10
## # i 44 more rows
```

- Vamos criar colunas novas com os valores convertidos para km/h e metros; além disso, uma coluna com a taxa de frenagem:

```
cars %>%
  mutate(
    velocidade = speed * 1.6,
    distancia = dist * .33,
    taxa = velocidade / distancia
  )
```

```
## # A tibble: 50 x 5
##   speed  dist velocidade distancia  taxa
##   <dbl> <dbl>      <dbl>     <dbl> <dbl>
## 1     4     2        6.4      0.66  9.70
## 2     4    10        6.4      3.3   1.94
## 3     7     4       11.2      1.32  8.48
## 4     7    22       11.2      7.26  1.54
## 5     8    16       12.8      5.28  2.42
## 6     9    10       14.4      3.3   4.36
## # i 44 more rows
```

- Perceba que as colunas antigas continuam lá. Se quiser manter apenas as colunas novas, use `transmute`:

[2] Considere *data frame* como sinônimo de *tibble*. Na verdade, *tibbles* formam um superconjunto de *data frames*: todo *data frame* é uma *tibble*, mas nem toda *tibble* é um *data frame*.

```
cars %>%
  transmute(
    velocidade = speed * 1.6,
    distancia = dist * .33,
    taxa = velocidade / distancia
  )
```

```
## # A tibble: 50 x 3
##   velocidade distancia  taxa
##        <dbl>     <dbl> <dbl>
## 1        6.4      0.66  9.70
## 2        6.4      3.3   1.94
## 3       11.2      1.32  8.48
## 4       11.2      7.26  1.54
## 5       12.8      5.28  2.42
## 6       14.4      3.3   4.36
## # i 44 more rows
```

- Ou use o argumento `.keep` de `mutate` para escolher com mais precisão. Veja a ajuda de `mutate`.

3.4.2 Selecionando colunas: `select`, `distinct`, `pull`

- Vamos voltar à nossa *tibble* dos *top* 100 da *Billboard*.
- Para ver só a coluna de artistas:

```
bb_tidy %>%
  select(artista)
```

```
## # A tibble: 5.307 x 1
##   artista
##   <chr>
## 1 2 Pac
## 2 2 Pac
## 3 2 Pac
## 4 2 Pac
## 5 2 Pac
## 6 2 Pac
## # i 5.301 more rows
```

- Para eliminar as repetições:

```
bb_tidy %>%
  select(artista) %>%
  distinct()
```

```
## # A tibble: 228 x 1
##   artista
##   <chr>
## 1 2 Pac
## 2 2Ge+her
## 3 3 Doors Down
## 4 504 Boyz
## 5 98^0
## 6 A*Teens
## # i 222 more rows
```

- Para ver artistas e músicas:

```
bb_tidy %>%
  select(artista, musica) %>%
  distinct()
```

```
## # A tibble: 317 x 2
##   artista      musica
##   <chr>        <chr>
## 1 2 Pac        Baby Don't Cry (Keep...
## 2 2Ge+her      The Hardest Part Of ...
## 3 3 Doors Down Kryptonite
## 4 3 Doors Down Loser
## 5 504 Boyz     Wobble Wobble
## 6 98^0         Give Me Just One Nig...
## # i 311 more rows
```

- Para especificar colunas a não mostrar, use o sinal de menos "-":

```
bb_tidy %>%
  select(-c(entrou, semana, pos))
```

```
## # A tibble: 5.307 x 2
##   artista musica
##   <chr>   <chr>
## 1 2 Pac   Baby Don't Cry (Keep...
## 2 2 Pac   Baby Don't Cry (Keep...
## 3 2 Pac   Baby Don't Cry (Keep...
## 4 2 Pac   Baby Don't Cry (Keep...
## 5 2 Pac   Baby Don't Cry (Keep...
## 6 2 Pac   Baby Don't Cry (Keep...
## # i 5.301 more rows
```

- Para extrair uma coluna na forma de vetor (`unique` é uma função do R base, aplicável a vetores):

```r
bb_tidy %>%
  pull(artista) %>%
  unique()
```

```
##   [1] "2 Pac"                          "2Ge+her"
##   [3] "3 Doors Down"                   "504 Boyz"
##   [5] "98^0"                           "A*Teens"
##   [7] "Aaliyah"                        "Adams, Yolanda"
##   [9] "Adkins, Trace"                  "Aguilera, Christina"
##  [11] "Alice Deejay"                   "Allan, Gary"
##  [13] "Amber"                          "Anastacia"
##  [15] "Anthony, Marc"                  "Avant"
##  [17] "BBMak"                          "Backstreet Boys, The"
##  [19] "Badu, Erkyah"                   "Baha Men"
##  [21] "Barenaked Ladies"               "Beenie Man"
##  [23] "Before Dark"                    "Bega, Lou"
##  [25] "Big Punisher"                   "Black Rob"
##  [27] "Black, Clint"                   "Blaque"
##  [29] "Blige, Mary J."                 "Blink-182"
##  [31] "Bloodhound Gang"                "Bon Jovi"
##  [33] "Braxton, Toni"                  "Brock, Chad"
##  [35] "Brooks & Dunn"                  "Brooks, Garth"
##  [37] "Byrd, Tracy"                    "Cagle, Chris"
##  [39] "Cam'ron"                        "Carey, Mariah"
##  [41] "Carter, Aaron"                  "Carter, Torrey"
##  [43] "Changing Faces"                 "Chesney, Kenny"
##  [45] "Clark Family Experience"        "Clark, Terri"
##  [47] "Common"                         "Counting Crows"
##  [49] "Creed"                          "Cyrus, Billy Ray"
##  [51] "D'Angelo"                       "DMX"
##  [53] "Da Brat"                        "Davidson, Clay"
##  [55] "De La Soul"                     "Destiny's Child"
##  [57] "Diffie, Joe"                    "Dion, Celine"
##  [59] "Dixie Chicks, The"              "Dr. Dre"
##  [61] "Drama"                          "Dream"
##  [63] "Eastsidaz, The"                 "Eiffel 65"
##  [65] "Elliott, Missy \"Misdemeanor\"" "Eminem"
##  [67] "En Vogue"                       "Estefan, Gloria"
##  [69] "Evans, Sara"                    "Eve"
##  [71] "Everclear"                      "Fabian, Lara"
##  [73] "Fatboy Slim"                    "Filter"
##  [75] "Foo Fighters"                   "Fragma"
##  [77] "Funkmaster Flex"                "Ghostface Killah"
##  [79] "Gill, Vince"                    "Gilman, Billy"
##  [81] "Ginuwine"                       "Goo Goo Dolls"
##  [83] "Gray, Macy"                     "Griggs, Andy"
##  [85] "Guy"                            "Hanson"
```

```
##  [87] "Hart, Beth"                  "Heatherly, Eric"
##  [89] "Henley, Don"                 "Herndon, Ty"
##  [91] "Hill, Faith"                 "Hoku"
##  [93] "Hollister, Dave"             "Hot Boys"
##  [95] "Houston, Whitney"            "IMx"
##  [97] "Ice Cube"                    "Ideal"
##  [99] "Iglesias, Enrique"           "J-Shin"
## [101] "Ja Rule"                     "Jackson, Alan"
## [103] "Jagged Edge"                 "Janet"
## [105] "Jay-Z"                       "Jean, Wyclef"
## [107] "Joe"                         "John, Elton"
## [109] "Jones, Donell"               "Jordan, Montell"
## [111] "Juvenile"                    "Kandi"
## [113] "Keith, Toby"                 "Kelis"
## [115] "Kenny G"                     "Kid Rock"
## [117] "Kravitz, Lenny"              "Kumbia Kings"
## [119] "LFO"                         "LL Cool J"
## [121] "Larrieux, Amel"              "Lawrence, Tracy"
## [123] "Levert, Gerald"              "Lil Bow Wow"
## [125] "Lil Wayne"                   "Lil' Kim"
## [127] "Lil' Mo"                     "Lil' Zane"
## [129] "Limp Bizkit"                 "Lonestar"
## [131] "Lopez, Jennifer"             "Loveless, Patty"
## [133] "Lox"                         "Lucy Pearl"
## [135] "Ludacris"                    "M2M"
## [137] "Madison Avenue"              "Madonna"
## [139] "Martin, Ricky"               "Mary Mary"
## [141] "Master P"                    "McBride, Martina"
## [143] "McEntire, Reba"              "McGraw, Tim"
## [145] "McKnight, Brian"             "Messina, Jo Dee"
## [147] "Metallica"                   "Montgomery Gentry"
## [149] "Montgomery, John Michael"    "Moore, Chante"
## [151] "Moore, Mandy"                "Mumba, Samantha"
## [153] "Musiq"                       "Mya"
## [155] "Mystikal"                    "N'Sync"
## [157] "Nas"                         "Nelly"
## [159] "Next"                        "Nine Days"
## [161] "No Doubt"                    "Nu Flavor"
## [163] "Offspring, The"              "Paisley, Brad"
## [165] "Papa Roach"                  "Pearl Jam"
## [167] "Pink"                        "Price, Kelly"
## [169] "Profyle"                     "Puff Daddy"
## [171] "Q-Tip"                       "R.E.M."
## [173] "Rascal Flatts"               "Raye, Collin"
## [175] "Red Hot Chili Peppers"       "Rimes, LeAnn"
## [177] "Rogers, Kenny"               "Ruff Endz"
## [179] "Sammie"                      "Santana"
## [181] "Savage Garden"               "SheDaisy"
```

```
## [183] "Sheist, Shade"                 "Shyne"
## [185] "Simpson, Jessica"              "Sisqo"
## [187] "Sister Hazel"                  "Smash Mouth"
## [189] "Smith, Will"                   "Son By Four"
## [191] "Sonique"                       "SoulDecision"
## [193] "Spears, Britney"               "Spencer, Tracie"
## [195] "Splender"                      "Sting"
## [197] "Stone Temple Pilots"           "Stone, Angie"
## [199] "Strait, George"                "Sugar Ray"
## [201] "TLC"                           "Tamar"
## [203] "Tamia"                         "Third Eye Blind"
## [205] "Thomas, Carl"                  "Tippin, Aaron"
## [207] "Train"                         "Trick Daddy"
## [209] "Trina"                         "Tritt, Travis"
## [211] "Tuesday"                       "Urban, Keith"
## [213] "Usher"                         "Vassar, Phil"
## [215] "Vertical Horizon"              "Vitamin C"
## [217] "Walker, Clay"                  "Wallflowers, The"
## [219] "Westlife"                      "Williams, Robbie"
## [221] "Wills, Mark"                   "Worley, Darryl"
## [223] "Wright, Chely"                 "Yankee Grey"
## [225] "Yearwood, Trisha"              "Ying Yang Twins"
## [227] "Zombie Nation"                 "matchbox twenty"
```

3.4.3

Filtrando linhas: `filter`, `slice`

- Apenas as músicas da Britney Spears:

```
bb_tidy %>%
  filter(artista == 'Spears, Britney')
```

```
## # A tibble: 51 x 5
##   artista         musica                  entrou     semana   pos
##   <chr>           <chr>                   <date>      <int> <dbl>
## 1 Spears, Britney From The Bottom Of M... 2000-01-29      1    76
## 2 Spears, Britney From The Bottom Of M... 2000-01-29      2    59
## 3 Spears, Britney From The Bottom Of M... 2000-01-29      3    52
## 4 Spears, Britney From The Bottom Of M... 2000-01-29      4    52
## 5 Spears, Britney From The Bottom Of M... 2000-01-29      5    14
## 6 Spears, Britney From The Bottom Of M... 2000-01-29      6    14
## # i 45 more rows
```

- Apenas músicas que chegaram à posição 1, sem mostrar a coluna `pos`:

```
bb_tidy %>%
```

```
  filter(pos == 1) %>%
  select(-pos)
```

```
## # A tibble: 55 x 4
##   artista             musica                  entrou     semana
##   <chr>               <chr>                   <date>      <int>
## 1 Aaliyah             Try Again               2000-03-18     14
## 2 Aguilera, Christina Come On Over Baby (A... 2000-08-05     11
## 3 Aguilera, Christina Come On Over Baby (A... 2000-08-05     12
## 4 Aguilera, Christina Come On Over Baby (A... 2000-08-05     13
## 5 Aguilera, Christina Come On Over Baby (A... 2000-08-05     14
## 6 Aguilera, Christina What A Girl Wants       1999-11-27      8
## # i 49 more rows
```

- Apenas músicas que chegaram à posição 1 em menos de 10 semanas, mostrando apenas artista e música:

```
bb_tidy %>%
  filter(pos == 1, semana < 10) %>%
  distinct(artista, musica)
```

```
## # A tibble: 5 x 2
##   artista             musica
##   <chr>               <chr>
## 1 Aguilera, Christina What A Girl Wants
## 2 Destiny's Child     Independent Women Pa...
## 3 Madonna             Music
## 4 Santana             Maria, Maria
## 5 Sisqo               Incomplete
```

- As funções da família `slice` filtram linhas de diversas maneiras.
- De acordo com seus índices (números de linha):

```
bb_tidy %>%
  slice(c(1, 1000, 5000))
```

```
## # A tibble: 3 x 5
##   artista                  musica                entrou     semana   pos
##   <chr>                    <chr>                 <date>      <int> <dbl>
## 1 2 Pac                    Baby Don't Cry (Keep~ 2000-02-26      1    87
## 2 Clark Family Experience  Meanwhile Back At Th~ 2000-11-18      3    81
## 3 Vassar, Phil             Carlene               2000-03-04      3    64
```

```
bb_tidy %>%
  slice_head(n = 4)
```

```
## # A tibble: 4 x 5
##   artista musica                          entrou     semana   pos
```

```
##   <chr>   <chr>                  <date>      <int> <dbl>
## 1 2 Pac   Baby Don't Cry (Keep... 2000-02-26      1    87
## 2 2 Pac   Baby Don't Cry (Keep... 2000-02-26      2    82
## 3 2 Pac   Baby Don't Cry (Keep... 2000-02-26      3    72
## 4 2 Pac   Baby Don't Cry (Keep... 2000-02-26      4    77
```

```r
bb_tidy %>%
  slice_tail(n = 4)
```

```
## # A tibble: 4 x 5
##   artista          musica entrou     semana   pos
##   <chr>            <chr>  <date>      <int> <dbl>
## 1 matchbox twenty Bent   2000-04-29     36    37
## 2 matchbox twenty Bent   2000-04-29     37    38
## 3 matchbox twenty Bent   2000-04-29     38    38
## 4 matchbox twenty Bent   2000-04-29     39    48
```

- De acordo com a ordenação de uma coluna ou de uma função das colunas:

```r
bb_tidy %>%
  slice_min(pos)
```

```
## # A tibble: 55 x 5
##   artista            musica                  entrou     semana   pos
##   <chr>              <chr>                   <date>      <int> <dbl>
## 1 Aaliyah            Try Again               2000-03-18     14     1
## 2 Aguilera, Christina Come On Over Baby (A... 2000-08-05     11     1
## 3 Aguilera, Christina Come On Over Baby (A... 2000-08-05     12     1
## 4 Aguilera, Christina Come On Over Baby (A... 2000-08-05     13     1
## 5 Aguilera, Christina Come On Over Baby (A... 2000-08-05     14     1
## 6 Aguilera, Christina What A Girl Wants       1999-11-27      8     1
## # i 49 more rows
```

```r
bb_tidy %>%
  slice_max(semana)
```

```
## # A tibble: 1 x 5
##   artista musica entrou     semana   pos
##   <chr>   <chr>  <date>      <int> <dbl>
## 1 Creed   Higher 1999-09-11     65    49
```

- Aleatoriamente, criando uma amostra:

```r
bb_tidy %>%
  slice_sample(n = 5)
```

```
## # A tibble: 5 x 5
##   artista          musica                  entrou     semana   pos
```

```
##   <chr>           <chr>                  <date>      <int> <dbl>
## 1 Rimes, LeAnn    Can't Fight The Moon... 2000-09-09      8    78
## 2 Mya             Case Of The Ex (What... 2000-08-19     20     4
## 3 Houston, Whitney I Learned From The B... 2000-02-19     11    82
## 4 Ja Rule         Between Me And You     2000-09-16      1    85
## 5 Urban, Keith    Your Everything        2000-07-15     13    84
```

- Veja a ajuda de `slice` para saber mais sobre estas funções. Por exemplo:
  - `slice_min` e `slice_max` podem considerar ou não empates.
  - Você pode especificar uma proporção de linhas (usando `prop`) em vez da quantidade de linhas (`n`).
  - Você pode fazer amostragem com reposição, ou com probabilidades diferentes para cada linha.

3.4.4 Ordenando linhas: `arrange`

- Por título, sem repetições:

```
bb_tidy %>%
  select(musica) %>%
  distinct() %>%
  arrange(musica)
```

```
## # A tibble: 316 x 1
##   musica
##   <chr>
## 1 (Hot S**t) Country G...
## 2 3 Little Words
## 3 911
## 4 A Country Boy Can Su...
## 5 A Little Gasoline
## 6 A Puro Dolor (Purest...
## # i 310 more rows
```

- Por título, sem repetições, em ordem inversa:

```
bb_tidy %>%
  select(musica) %>%
  distinct() %>%
  arrange(desc(musica))
```

```
## # A tibble: 316 x 1
##   musica
##   <chr>
## 1 www.memory
```

```
## 2 Your Everything
## 3 You're A God
## 4 You'll Always Be Lov...
## 5 You Won't Be Lonely ...
## 6 You Should've Told M...
## # i 310 more rows
```

3.4.5 Contando linhas: `count`

- Quantas semanas cada artista ficou nos *top* 100? Duas músicas na mesma semana contam como duas semanas.

```
bb_tidy %>%
  count(artista, sort = TRUE)
```

```
## # A tibble: 228 x 2
##   artista            n
##   <chr>          <int>
## 1 Creed            104
## 2 Lonestar          95
## 3 Destiny's Child   92
## 4 N'Sync            74
## 5 Sisqo             74
## 6 3 Doors Down      73
## # i 222 more rows
```

- Quantas semanas cada música ficou nos *top* 100?

```
bb_tidy %>%
  count(musica, sort = TRUE)
```

```
## # A tibble: 316 x 2
##   musica                 n
##   <chr>              <int>
## 1 Higher                57
## 2 Amazed                55
## 3 Breathe               53
## 4 Kryptonite            53
## 5 With Arms Wide Open   47
## 6 I Wanna Know          44
## # i 310 more rows
```

- Se houve músicas com o mesmo nome, mas de artistas diferentes, o código acima está errado. O certo é

```
bb_tidy %>%
  count(musica, artista, sort = TRUE)
```

```
## # A tibble: 317 x 3
##   musica              artista          n
##   <chr>               <chr>        <int>
## 1 Higher              Creed           57
## 2 Amazed              Lonestar        55
## 3 Breathe             Hill, Faith     53
## 4 Kryptonite          3 Doors Down    53
## 5 With Arms Wide Open Creed           47
## 6 I Wanna Know        Joe             44
## # i 311 more rows
```

De fato, há uma diferença de uma linha.

3.4.5.1

Exercício

- Ache o título da música que tem dois artistas diferentes.

  Sugestão: conte por música e artista primeiro, depois só por música.

3.4.6

Agrupando linhas: `group_by` e `summarize`

- Qual foi a melhor posição que cada artista alcançou?

```
bb_tidy %>%
  group_by(artista) %>%
  summarize(melhor = min(pos)) %>%
  arrange(melhor)
```

```
## # A tibble: 228 x 2
##   artista             melhor
##   <chr>                <dbl>
## 1 Aaliyah                  1
## 2 Aguilera, Christina      1
## 3 Carey, Mariah            1
## 4 Creed                    1
## 5 Destiny's Child          1
## 6 Iglesias, Enrique        1
## # i 222 more rows
```

- Qual foi a melhor posição que cada música alcançou?

```
bb_tidy %>%
  group_by(artista, musica) %>%
  summarize(melhor = min(pos)) %>%
  arrange(melhor)
```

```
## `summarise()` has grouped output by 'artista'. You can override using
## the `.groups` argument.
```

```
## # A tibble: 317 x 3
##   artista             musica                  melhor
##   <chr>               <chr>                    <dbl>
## 1 Aaliyah             Try Again                    1
## 2 Aguilera, Christina Come On Over Baby (A...      1
## 3 Aguilera, Christina What A Girl Wants            1
## 4 Carey, Mariah       Thank God I Found Yo...      1
## 5 Creed               With Arms Wide Open          1
## 6 Destiny's Child     Independent Women Pa...      1
## # i 311 more rows
```

- Quando usamos `summarize`, só o agrupamento mais interno é desfeito. Isto significa que o resultado acima ainda está agrupado por `artista`.
- Quantas semanas cada artista ficou na posição 1?

  A função `n()` é uma maneira conveniente de obter o número de linhas de um grupo (ou, se não houver agrupamento, de toda a *tibble*); mas `n()` só pode ser chamada em certos contextos, como `summarise()` ou `mutate()`.

```
bb_tidy %>%
  filter(pos == 1) %>%
  group_by(artista) %>%
  summarize(semanas = n()) %>%
  arrange(desc(semanas))
```

```
## # A tibble: 15 x 2
##   artista             semanas
##   <chr>                 <int>
## 1 Destiny's Child          14
## 2 Santana                  10
## 3 Aguilera, Christina       6
## 4 Madonna                   4
## 5 Savage Garden             4
## 6 Iglesias, Enrique         3
## # i 9 more rows
```

- Perceba que `count`, que vimos mais acima, faz agrupamentos do mesmo modo:

```
bb_tidy %>%
```

```
  filter(pos == 1) %>%
  count(artista, sort = TRUE)
```

```
## # A tibble: 15 x 2
##   artista                n
##   <chr>              <int>
## 1 Destiny's Child       14
## 2 Santana               10
## 3 Aguilera, Christina    6
## 4 Madonna                4
## 5 Savage Garden          4
## 6 Iglesias, Enrique      3
## # i 9 more rows
```

- Uma pergunta diferente: quais são os artistas cujas músicas apareceram no *top* 100 mais tempo depois do lançamento da música?

```
bb_tidy %>%
  group_by(artista) %>%
  summarize(semanas = max(semana)) %>%
  arrange(desc(semanas))
```

```
## # A tibble: 228 x 2
##   artista          semanas
##   <chr>              <int>
## 1 Creed                 65
## 2 Lonestar              64
## 3 3 Doors Down          53
## 4 Hill, Faith           53
## 5 Joe                   44
## 6 Vertical Horizon      41
## # i 222 more rows
```

- Qual a posição média de cada música? Lembre-se de que eliminamos as linhas com `NA`; logo, a média vai ser sobre a quantidade de semanas em que a música esteve na lista.

```
media1 <- bb_tidy %>%
  group_by(artista, musica) %>%
  summarize(media = mean(pos), .groups = 'drop') %>%
  arrange(media)
```

```
media1
```

```
## # A tibble: 317 x 3
##   artista                          musica                  media
##   <chr>                            <chr>                   <dbl>
## 1 "Santana"                        Maria, Maria             10.5
```

```
## 2 "Madonna"                        Music                    13.5
## 3 "N'Sync"                         Bye Bye Bye              14.3
## 4 "Elliott, Missy \"Misdemeanor\"" Hot Boyz                 14.3
## 5 "Destiny's Child"                Independent Women Pa...  14.8
## 6 "Iglesias, Enrique"              Be With You              15.8
## # i 311 more rows
```

- E se quisermos a média sobre o número de semanas desde a entrada da música até a última semana em que a música apareceu na lista?

```
media2 <- bb_tidy %>%
  group_by(artista, musica) %>%
  summarize(media = sum(pos)/max(semana), .groups = 'drop') %>%
  arrange(media)
```

```
media2
```

```
## # A tibble: 317 x 3
##   artista                          musica                   media
##   <chr>                            <chr>                    <dbl>
## 1 "Santana"                        Maria, Maria              10.5
## 2 "Madonna"                        Music                     13.5
## 3 "N'Sync"                         Bye Bye Bye               14.3
## 4 "Elliott, Missy \"Misdemeanor\"" Hot Boyz                  14.3
## 5 "Destiny's Child"                Independent Women Pa...   14.8
## 6 "Iglesias, Enrique"              Be With You               15.8
## # i 311 more rows
```

As primeiras linhas são iguais, mas os resultados são diferentes:

```
identical(media1, media2)
```

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

3.5 Exercícios

1. Vamos trabalhar com um conjunto de dados sobre super-heróis.

   Carregue o `tidyverse` com o comando

   ```
   library(tidyverse)
   ```

   Execute o seguinte comando para ler os dados para uma *tibble*:

```r
arquivo <- paste0(
  'https://github.com/fnaufel/',
  'probestr/raw/master/data/',
  'heroes_information.csv'
)

herois_info <- read_csv(
  arquivo,
  na = c('', '-', 'NA')
) %>%
  # Eliminar a primeira coluna (números de série)
  select(-1) %>%
  # Renomear colunas restantes
  rename(
    nome = name,
    sexo = Gender,
    olhos = 'Eye color',
    raça = Race,
    cabelos = 'Hair color',
    altura = Height,
    editora = Publisher,
    pele = 'Skin color',
    lado = Alignment,
    peso = Weight
  )
```

2. Quantas linhas tem a *tibble*?

3. Existem heróis que aparecem em mais de uma linha?

4. Quantas editoras diferentes existem na *tibble*? Liste-as em ordem decrescente de quantidade de heróis.

5. Vamos colocar todas as editores menores em uma classe só.

   Na coluna `editora`, substitua

   - 'Marvel Comics' por 'Marvel',
   - 'DC Comics' por 'DC', e
   - todas as outras editoras pelo termo 'Outras'.

   Dica: use a função `case_when()`, do `tidyverse`.

6. Confira, novamente, a quantidade de valores diferentes na coluna `editora`.

7. Existem heróis sem informação de editora. Quantos? Quais são?

8. Altere novamente a coluna `editora`, colocando o valor 'Outras' para os heróis sem informação de editora. Use a função `if_else()` (com *underscore*, não a função `ifelse`).

9. Confira, mais uma vez, a quantidade de valores diferentes na coluna `editora`.

10. Existem heróis sem informação de sexo? Quantos? Para estes heróis, coloque o valor 'Desconhecido' na coluna `sexo`.
11. Qual a altura mínima? Qual a altura máxima? Substitua as alturas negativas por `NA`.
12. Qual o peso mínimo? Qual o peso máximo? Substitua os pesos negativos por `NA`.
13. Qual é o peso médio de todos os heróis? Ignore os valores `NA`.
14. Qual é a altura média de todos os heróis? Ignore os valores `NA`.
15. Qual é a altura média dos heróis, por editora? Ignore os valores `NA`.
16. Quais são os 3 heróis mais altos de cada sexo?
17. Quais são as 3 cores de olhos mais comuns para cada sexo?
18. Liste, por editora, as quantidades de heróis do bem, do mal, e neutros.
19. Quantas raças diferentes existem?
20. Qual a quantidade de raças diferentes de cada editora?
21. **DESAFIO:** Liste as raças que só pertencem a uma única editora.

    Existem várias maneiras de fazer isto. Experimente várias, até achar uma que seja mais elegante.

3.6 Examinando *tibbles* intermediárias

- O pacote `ViewPipeSteps` serve para exibir (no console ou em *tabs* no RStudio) as *tibbles* que são resultados de cada passo em uma sequência de comandos montada com o *pipe* `%>%`.
- Instale o pacote com o comando

```
install.packages("ViewPipeSteps")
```

- Carregue o pacote com

```
library(ViewPipeSteps)
```

- Para exibir, no console, as *tibbles* intermediárias, acrescente `print_pipe_steps(all = TRUE)` após o último passo do *pipe*:

```
resultado <- bb_tidy %>%
  group_by(artista, musica) %>%
  summarize(media = sum(pos)/max(semana), .groups = 'drop') %>%
  arrange(media) %>%
  print_pipe_steps(all = TRUE)
```

```
## 1. bb_tidy

## # A tibble: 5.307 x 5
##   artista musica                  entrou     semana   pos
##   <chr>   <chr>                   <date>      <int> <dbl>
## 1 2 Pac   Baby Don't Cry (Keep... 2000-02-26      1    87
## 2 2 Pac   Baby Don't Cry (Keep... 2000-02-26      2    82
## 3 2 Pac   Baby Don't Cry (Keep... 2000-02-26      3    72
## 4 2 Pac   Baby Don't Cry (Keep... 2000-02-26      4    77
## 5 2 Pac   Baby Don't Cry (Keep... 2000-02-26      5    87
## 6 2 Pac   Baby Don't Cry (Keep... 2000-02-26      6    94
## # i 5.301 more rows

## 2. group_by(artista, musica)

## # A tibble: 5.307 x 5
##   artista musica                  entrou     semana   pos
##   <chr>   <chr>                   <date>      <int> <dbl>
## 1 2 Pac   Baby Don't Cry (Keep... 2000-02-26      1    87
## 2 2 Pac   Baby Don't Cry (Keep... 2000-02-26      2    82
## 3 2 Pac   Baby Don't Cry (Keep... 2000-02-26      3    72
## 4 2 Pac   Baby Don't Cry (Keep... 2000-02-26      4    77
## 5 2 Pac   Baby Don't Cry (Keep... 2000-02-26      5    87
## 6 2 Pac   Baby Don't Cry (Keep... 2000-02-26      6    94
## # i 5.301 more rows

## 3. summarize(media = sum(pos)/max(semana), .groups = "drop")

## # A tibble: 317 x 3
##   artista      musica                  media
##   <chr>        <chr>                   <dbl>
## 1 2 Pac        Baby Don't Cry (Keep...  85.4
## 2 2Ge+her      The Hardest Part Of ...  90
## 3 3 Doors Down Kryptonite               26.5
## 4 3 Doors Down Loser                    67.1
## 5 504 Boyz     Wobble Wobble            56.2
## 6 98^0         Give Me Just One Nig...  37.6
## # i 311 more rows

## 4. arrange(media)

## # A tibble: 317 x 3
##   artista                          musica                  media
##   <chr>                            <chr>                   <dbl>
## 1 "Santana"                        Maria, Maria             10.5
## 2 "Madonna"                        Music                    13.5
## 3 "N'Sync"                         Bye Bye Bye              14.3
## 4 "Elliott, Missy \"Misdemeanor\"" Hot Boyz                 14.3
## 5 "Destiny's Child"                Independent Women Pa...  14.8
## 6 "Iglesias, Enrique"              Be With You              15.8
## # i 311 more rows
```

- Para exibir as *tibbles* intermediárias em *tabs* do RStudio (como com a função `View()`), você pode usar o *addin* `viewPipeChain`, que também faz parte deste pacote. Veja o exemplo no site do pacote.

CAPÍTULO 4

Visualização com ggplot2

Busque mais informações sobre os pacotes `tidyverse` e `ggplot2` nas referências recomendadas.

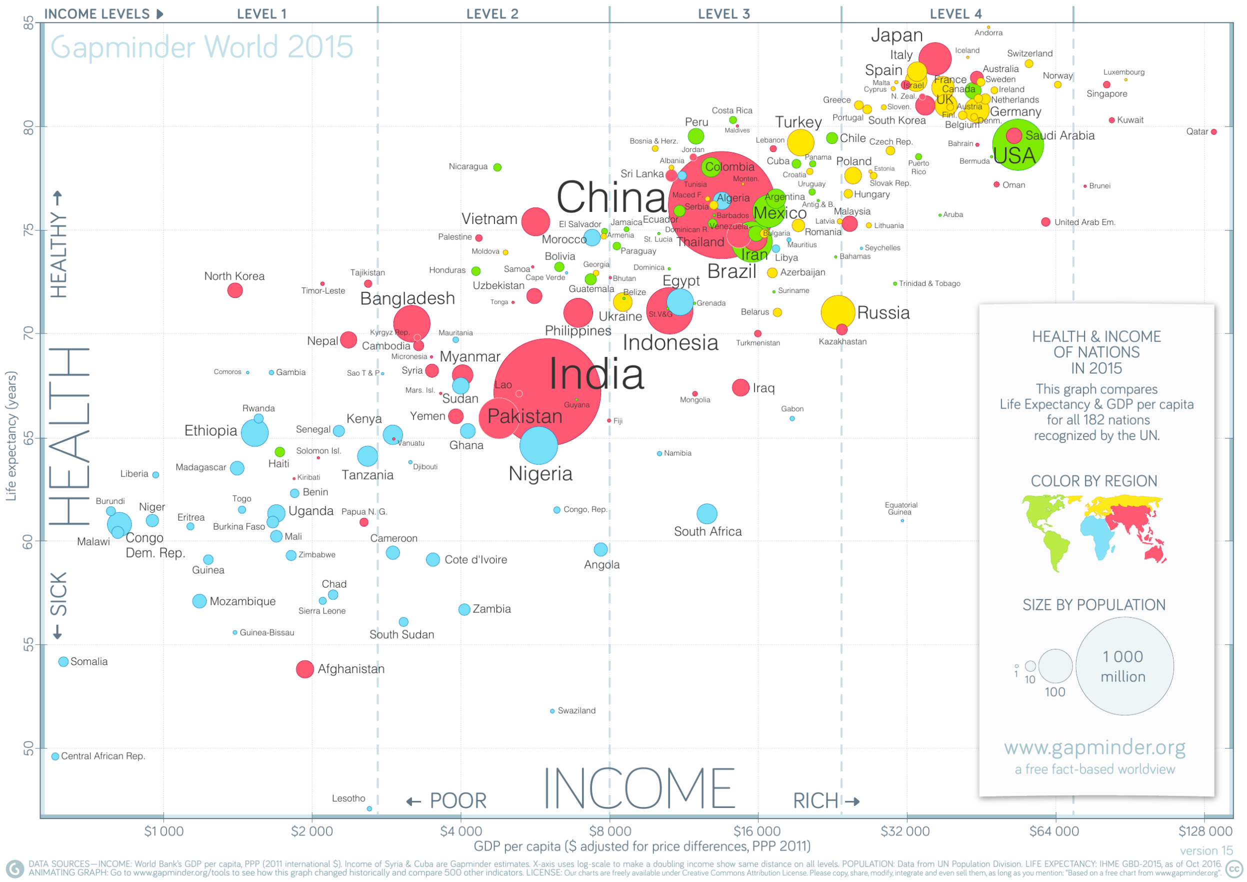
4.1 Vídeo 1

https://youtu.be/OBpNjqIIyhI

4.2 Componentes de um gráfico ggplot2

4.2.1 Geometrias e mapeamentos estéticos (*mappings*)

- Observe o gráfico abaixo, obtido de https://www.gapminder.org/downloads/updated-gapminder-world-poster-2015/.



- O gráfico mostra como, em cada país, a saúde (mais precisamente, a expectativa de vida) se relaciona com a riqueza (mais precisamente, o PIB *per capita*).
- Além da expectativa de vida e o do PIB *per capita*, o gráfico traz mais informações sobre cada país.
- Cada país é representado por um ponto (a geometria).
- Informações sobre cada país são representadas por características do ponto correspondente (as estéticas):

Variável	Geometria	Estética
PIB *per capita*	ponto	posição x
Expectativa de vida	ponto	posição y
População	ponto	tamanho
Continente	ponto	cor

- Você pode usar outras estéticas para representer informações:
  - Cor de preenchimento.
  - Cor do traço.
  - Tipo do traço (sólido, pontilhado, tracejado etc.).
  - Forma (círculo, quadrado, triângulo etc.).

- Opacidade.
- etc.

- Você pode usar outras geometrias:
  - Linhas.
  - Barras ou colunas.
  - Caixas.
  - etc.

4.2.2 Escalas (*scales*)

- As escalas controlam os detalhes da aparência da geometria e do mapeamento (eixos, cores etc.).
- Os eixos do gráfico acima são escalas contínuas, com valores reais.
- Observe o eixo horizontal. Os valores não aumentam linearmente, mas sim exponencialmente: cada passo à direita equivale a *dobrar* o valor do PIB. O eixo horizontal segue uma escala logarítmica.
- Os tamanhos dos pontos formam uma escala discreta, com 4 valores possíveis (veja a legenda no canto inferior direito do gráfico).
- As cores também formam uma escala discreta.

4.2.3 Rótulos (*labels*)

- O gráfico também representa informação na forma de texto.
- Além de rótulos (por exemplo, o texto que identifica cada eixo), o texto também pode, ele mesmo, ser uma geometria, com suas próprias estéticas: observe como o nome de cada país é escrito em um tamanho proporcional à sua população.

4.2.4 Outros componentes

- Coordenadas:
  - Este gráfico usa coordenadas cartesianas, com eixos x e y.
  - Existem gráficos que usam um sistema de coordenadas polares.
- Temas:
  - Incluem todos os elementos “decorativos”: cor de fundo, linhas de grade, etc. Ajudam a facilitar a leitura e a interpretação.

  - No gráfico acima, um detalhe interessante do tema é a divisão de cada eixo em segmentos claros e segmentos escuros.
- Legendas (*guides*).
- Facetas:
  - Às vezes, um gráfico é composto por múltiplos subgráficos.
  - Cada subgráfico é uma faceta.
  - Facetas evitam que informações demais sejam apresentadas no mesmo lugar.

4.3 Conjunto de dados

- Nossos exemplos de gráficos vão usar dados sobre o sono de diversos mamíferos.
- O conjunto de dados se chama `msleep` e está incluído no pacote `ggplot2`.
- Para ver a documentação, digite

```
library(ggplot2)
?msleep
```

- Vamos atribuir o conjunto de dados à variável `df`:

```
df <- msleep
df
```

```
## # A tibble: 83 x 11
##   name               genus vore  order conservation sleep_total sleep_rem
##   <chr>              <chr> <chr> <chr> <chr>              <dbl>     <dbl>
## 1 Cheetah            Acin~ carni Carn~ lc                  12.1      NA
## 2 Owl monkey         Aotus omni  Prim~ <NA>                17         1.8
## 3 Mountain beaver    Aplo~ herbi Rode~ nt                  14.4       2.4
## 4 Greater short-t~ Blar~ omni  Sori~ lc                  14.9       2.3
## 5 Cow                Bos   herbi Arti~ domesticated         4         0.7
## 6 Three-toed sloth Brad~ herbi Pilo~ <NA>                14.4       2.2
## # i 77 more rows
## # i 4 more variables: sleep_cycle <dbl>, awake <dbl>, brainwt <dbl>,
## #   bodywt <dbl>
```

- Vamos examinar a estrutura — usando R base:

```
str(df)
```

```
## tibble [83 x 11] (S3: tbl_df/tbl/data.frame)
##  $ name        : chr [1:83] "Cheetah" "Owl monkey" "Mountain beaver" ...
##  $ genus       : chr [1:83] "Acinonyx" "Aotus" "Aplodontia" ...
```

```
##  $ vore        : chr [1:83] "carni" "omni" "herbi" ...
##  $ order       : chr [1:83] "Carnivora" "Primates" "Rodentia" ...
##  $ conservation: chr [1:83] "lc" NA "nt" ...
##  $ sleep_total : num [1:83] 12,1 17 14,4 14,9 4 14,4 8,7 7 ...
##  $ sleep_rem   : num [1:83] NA 1,8 2,4 2,3 0,7 2,2 1,4 NA ...
##  $ sleep_cycle : num [1:83] NA NA NA 0,133 ...
##  $ awake       : num [1:83] 11,9 7 9,6 9,1 20 9,6 15,3 17 ...
##  $ brainwt     : num [1:83] NA 0,0155 NA 0,00029 0,423 NA NA NA ...
##  $ bodywt      : num [1:83] 50 0,48 1,35 0,019 ...
```

- Podemos usar `glimpse`, uma função do `tidyverse`:

```
glimpse(df)
```

```
## Rows: 83
## Columns: 11
## $ name         <chr> "Cheetah", "Owl monkey", "Mountain beaver", "Gre~
## $ genus        <chr> "Acinonyx", "Aotus", "Aplodontia", "Blarina", "B~
## $ vore         <chr> "carni", "omni", "herbi", "omni", "herbi", "herb~
## $ order        <chr> "Carnivora", "Primates", "Rodentia", "Soricomorp~
## $ conservation <chr> "lc", NA, "nt", "lc", "domesticated", NA, "vu", ~
## $ sleep_total  <dbl> 12,1, 17,0, 14,4, 14,9, 4,0, 14,4, 8,7, 7,0, 10,~
## $ sleep_rem    <dbl> NA, 1,8, 2,4, 2,3, 0,7, 2,2, 1,4, NA, 2,9, NA, 0~
## $ sleep_cycle  <dbl> NA, NA, NA, 0,1333333, 0,6666667, 0,7666667, 0,3~
## $ awake        <dbl> 11,9, 7,0, 9,6, 9,1, 20,0, 9,6, 15,3, 17,0, 13,9~
## $ brainwt      <dbl> NA, 0,01550, NA, 0,00029, 0,42300, NA, NA, NA, 0~
## $ bodywt       <dbl> 50,000, 0,480, 1,350, 0,019, 600,000, 3,850, 20,~
```

- Para examinar só as primeiras linhas do *data frame*:

```
head(df)
```

```
## # A tibble: 6 x 11
##   name              genus vore  order conservation sleep_total sleep_rem
##   <chr>             <chr> <chr> <chr> <chr>              <dbl>     <dbl>
## 1 Cheetah           Acin~ carni Carn~ lc                  12.1      NA  
## 2 Owl monkey        Aotus omni  Prim~ <NA>                17         1.8
## 3 Mountain beaver   Aplo~ herbi Rode~ nt                  14.4       2.4
## 4 Greater short-t~ Blar~ omni  Sori~ lc                  14.9       2.3
## 5 Cow               Bos   herbi Arti~ domesticated         4         0.7
## 6 Three-toed sloth Brad~ herbi Pilo~ <NA>                14.4       2.2
## # i 4 more variables: sleep_cycle <dbl>, awake <dbl>, brainwt <dbl>,
## #   bodywt <dbl>
```

- Para examinar o *data frame* interativamente:

```
view(df)
```

- Podemos produzir um sumário dos dados usando o pacote *summarytools* (que já foi carregado neste documento):

```
df %>% dfSummary() %>% print()
```

Variável	Estatísticas / Valores	Freqs (% de Válidos)	Faltante
name [character]	1. African elephant	1 (1,2%)	0 (0,0%)
	2. African giant pouched rat	1 (1,2%)	
	3. African striped mouse	1 (1,2%)	
	4. Arctic fox	1 (1,2%)	
	5. Arctic ground squirrel	1 (1,2%)	
	6. Asian elephant	1 (1,2%)	
	7. Baboon	1 (1,2%)	
	8. Big brown bat	1 (1,2%)	
	9. Bottle-nosed dolphin	1 (1,2%)	
	10. Brazilian tapir	1 (1,2%)	
	[73 outros]	73 (88,0%)	
genus [character]	1. Panthera	3 (3,6%)	0 (0,0%)
	2. Spermophilus	3 (3,6%)	
	3. Equus	2 (2,4%)	
	4. Vulpes	2 (2,4%)	
	5. Acinonyx	1 (1,2%)	
	6. Aotus	1 (1,2%)	
	7. Aplodontia	1 (1,2%)	
	8. Blarina	1 (1,2%)	
	9. Bos	1 (1,2%)	
	10. Bradypus	1 (1,2%)	
	[67 outros]	67 (80,7%)	
vore [character]	1. carni	19 (25,0%)	7 (8,4%)
	2. herbi	32 (42,1%)	
	3. insecti	5 (6,6%)	
	4. omni	20 (26,3%)	
order [character]	1. Rodentia	22 (26,5%)	0 (0,0%)
	2. Carnivora	12 (14,5%)	
	3. Primates	12 (14,5%)	
	4. Artiodactyla	6 (7,2%)	
	5. Soricomorpha	5 (6,0%)	
	6. Cetacea	3 (3,6%)	
	7. Hyracoidea	3 (3,6%)	
	8. Perissodactyla	3 (3,6%)	
	9. Chiroptera	2 (2,4%)	
	10. Cingulata	2 (2,4%)	
	[9 outros]	13 (15,7%)	

Variável	Estatísticas / Valores	Freqs (% de Válidos)	Faltante
conservation [character]	1. cd 2. domesticated 3. en 4. lc 5. nt 6. vu	2 (3,7%) 10 (18,5%) 4 (7,4%) 27 (50,0%) 4 (7,4%) 7 (13,0%)	29 (34,9%)
sleep_total [numeric]	Média (dp) : 10,4 (4,5) mín < mediana < máx: 1,9 < 10,1 < 19,9 IQE (CV) : 5,9 (0,4)	65 valores distintos	0 (0,0%)
sleep_rem [numeric]	Média (dp) : 1,9 (1,3) mín < mediana < máx: 0,1 < 1,5 < 6,6 IQE (CV) : 1,5 (0,7)	32 valores distintos	22 (26,5%)
sleep_cycle [numeric]	Média (dp) : 0,4 (0,4) mín < mediana < máx: 0,1 < 0,3 < 1,5 IQE (CV) : 0,4 (0,8)	22 valores distintos	51 (61,4%)
awake [numeric]	Média (dp) : 13,6 (4,5) mín < mediana < máx: 4,1 < 13,9 < 22,1 IQE (CV) : 5,9 (0,3)	65 valores distintos	0 (0,0%)
brainwt [numeric]	Média (dp) : 0,3 (1) mín < mediana < máx: 0 < 0 < 5,7 IQE (CV) : 0,1 (3,5)	53 valores distintos	27 (32,5%)
bodywt [numeric]	Média (dp) : 166,1 (786,8) mín < mediana < máx: 0 < 1,7 < 6654 IQE (CV) : 41,6 (4,7)	82 valores distintos	0 (0,0%)

- Vemos que há muitos `NA` em diversas variáveis. Para nossos exemplos simples de visualização, vamos usar as colunas
    - `name`
    - `genus`
    - `order`
    - `sleep_total`
    - `awake`
    - `bodywt`
    - `brainwt`
- Mas... a coluna que mostra a dieta (`vore`) tem só 7 `NA`. Quais são?

```
df %>%
```

```
  filter(is.na(vore)) %>%
  select(name)
```

```
## # A tibble: 7 x 1
##   name
##   <chr>
## 1 Vesper mouse
## 2 Desert hedgehog
## 3 Deer mouse
## 4 Phalanger
## 5 Rock hyrax
## 6 Mole rat
## # i 1 more row
```

- OK. Vamos manter a coluna `vore` também, apesar dos `NA`. Quando formos usar esta variável, tomaremos cuidado.
- Também… a coluna `bodywt` tem 0 como valor mínimo. Como assim?

```
df %>%
  filter(bodywt < 1) %>%
  select(name, bodywt) %>%
  arrange(bodywt)
```

```
## # A tibble: 35 x 2
##   name                          bodywt
##   <chr>                          <dbl>
## 1 Lesser short-tailed shrew      0.005
## 2 Little brown bat               0.01
## 3 Greater short-tailed shrew     0.019
## 4 Deer mouse                     0.021
## 5 House mouse                    0.022
## 6 Big brown bat                  0.023
## # i 29 more rows
```

- Ah, sem problema. A função `dfSummary` arredondou estes pesos para 0. Os valores de verdade ainda estão na *tibble*.
- Vamos criar uma *tibble* nova, só com as colunas que nos interessam:

```
sono <- df %>%
  select(
    name, order, genus, vore, bodywt,
    brainwt, awake, sleep_total
  )
```

- Vamos ver o sumário:

```
sono %>% dfSummary() %>% print()
```

Variável	Estatísticas / Valores	Freqs (% de Válidos)	Faltante
name	1. African elephant	1 (1,2%)	0
[character]	2. African giant pouched rat	1 (1,2%)	(0,0%)
	3. African striped mouse	1 (1,2%)	
	4. Arctic fox	1 (1,2%)	
	5. Arctic ground squirrel	1 (1,2%)	
	6. Asian elephant	1 (1,2%)	
	7. Baboon	1 (1,2%)	
	8. Big brown bat	1 (1,2%)	
	9. Bottle-nosed dolphin	1 (1,2%)	
	10. Brazilian tapir	1 (1,2%)	
	[73 outros]	73 (88,0%)	
order	1. Rodentia	22 (26,5%)	0
[character]	2. Carnivora	12 (14,5%)	(0,0%)
	3. Primates	12 (14,5%)	
	4. Artiodactyla	6 (7,2%)	
	5. Soricomorpha	5 (6,0%)	
	6. Cetacea	3 (3,6%)	
	7. Hyracoidea	3 (3,6%)	
	8. Perissodactyla	3 (3,6%)	
	9. Chiroptera	2 (2,4%)	
	10. Cingulata	2 (2,4%)	
	[9 outros]	13 (15,7%)	
genus	1. Panthera	3 (3,6%)	0
[character]	2. Spermophilus	3 (3,6%)	(0,0%)
	3. Equus	2 (2,4%)	
	4. Vulpes	2 (2,4%)	
	5. Acinonyx	1 (1,2%)	
	6. Aotus	1 (1,2%)	
	7. Aplodontia	1 (1,2%)	
	8. Blarina	1 (1,2%)	
	9. Bos	1 (1,2%)	
	10. Bradypus	1 (1,2%)	
	[67 outros]	67 (80,7%)	
vore	1. carni	19 (25,0%)	7
[character]	2. herbi	32 (42,1%)	(8,4%)
	3. insecti	5 (6,6%)	
	4. omni	20 (26,3%)	
bodywt	Média (dp) : 166,1 (786,8)	82 valores distintos	0
[numeric]	mín < mediana < máx:		(0,0%)
	0 < 1,7 < 6654		
	IQE (CV) : 41,6 (4,7)		
brainwt	Média (dp) : 0,3 (1)	53 valores distintos	27
[numeric]	mín < mediana < máx:		(32,5%)
	0 < 0 < 5,7		
	IQE (CV) : 0,1 (3,5)		

Variável	Estatísticas / Valores	Freqs (% de Válidos)	Faltante
awake [numeric]	Média (dp) : 13,6 (4,5) mín < mediana < máx: 4,1 < 13,9 < 22,1 IQE (CV) : 5,9 (0,3)	65 valores distintos	0 (0,0%)
sleep_total [numeric]	Média (dp) : 10,4 (4,5) mín < mediana < máx: 1,9 < 10,1 < 19,9 IQE (CV) : 5,9 (0,4)	65 valores distintos	0 (0,0%)

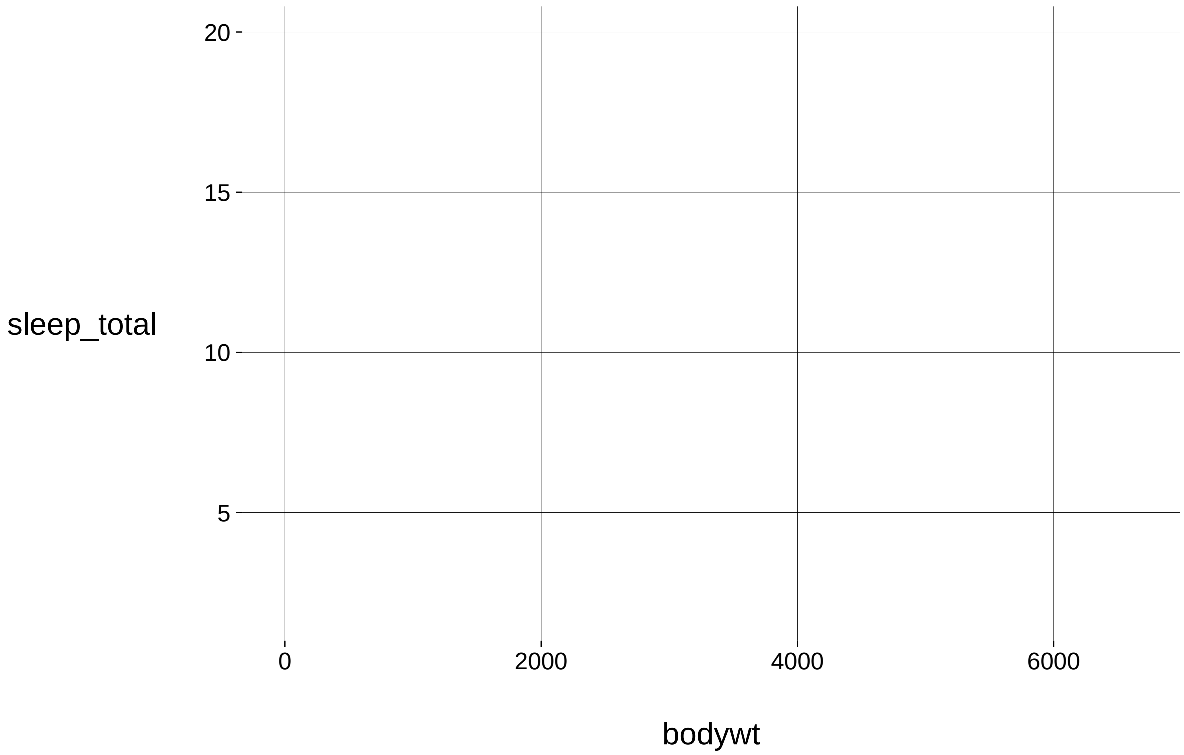
4.4 Gráficos de dispersão (*scatter plots*)

- Servem para visualizar a *relação* entre duas variáveis quantitativas.
- Essa relação *não* é necessariamente de causa e efeito.
- Isto é, a variável do eixo horizontal não determina, necessariamente, os valores da variável do eixo vertical.
- Pense em associação, correlação, não em causalidade.
- Troque as variáveis de eixo, se ajudar a deixar isto claro.

4.4.1 Horas de sono e peso corporal

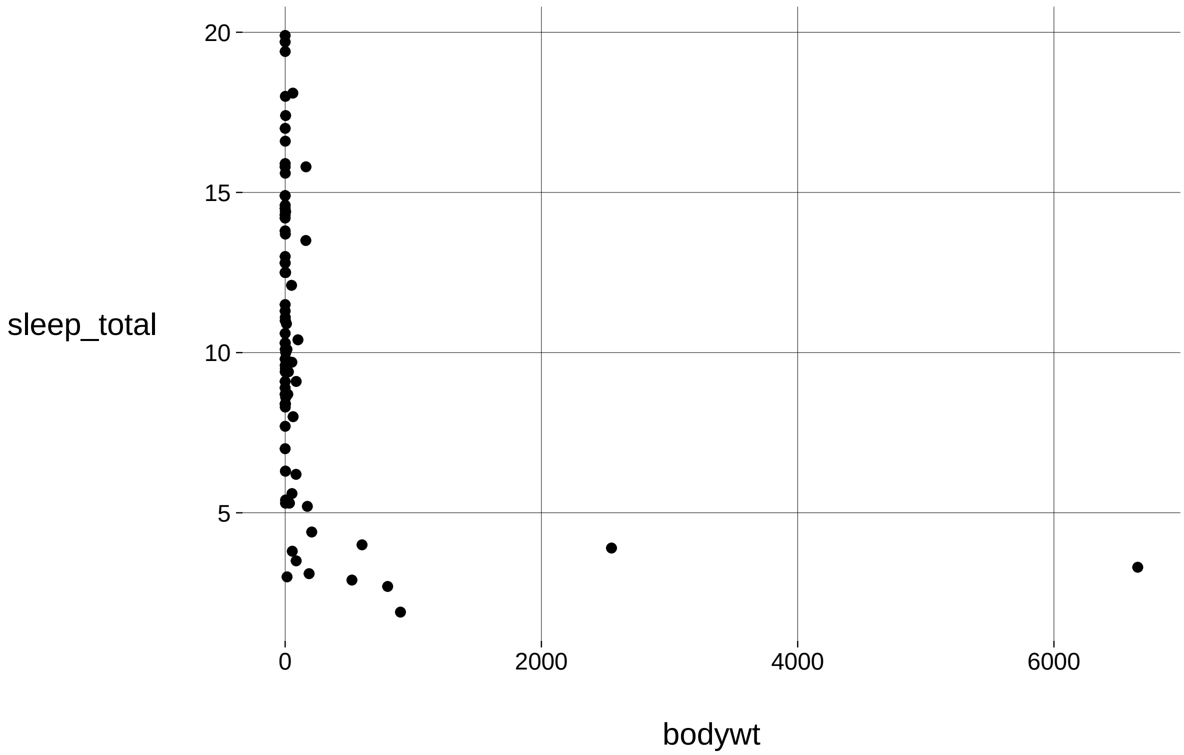
- Como as variáveis `sleep_total` e `bodywt` estão relacionadas?

```
sono %>%
  ggplot(aes(x = bodywt, y = sleep_total))
```



- O que houve? Cadê os pontos?
- O problema foi que só especificamos o mapeamento estético (com `aes`, que são as iniciais de *aesthetics*). Faltou a geometria.

```
sono %>%
  ggplot(aes(x = bodywt, y = sleep_total)) +
  geom_point()
```



- Que horror.
- A única coisa que percebemos aqui é que os mamíferos muito pesados dormem menos de 5 horas por noite.
- Estes animais muito pesados estão estragando a escala do eixo x.
- Que animais são estes?

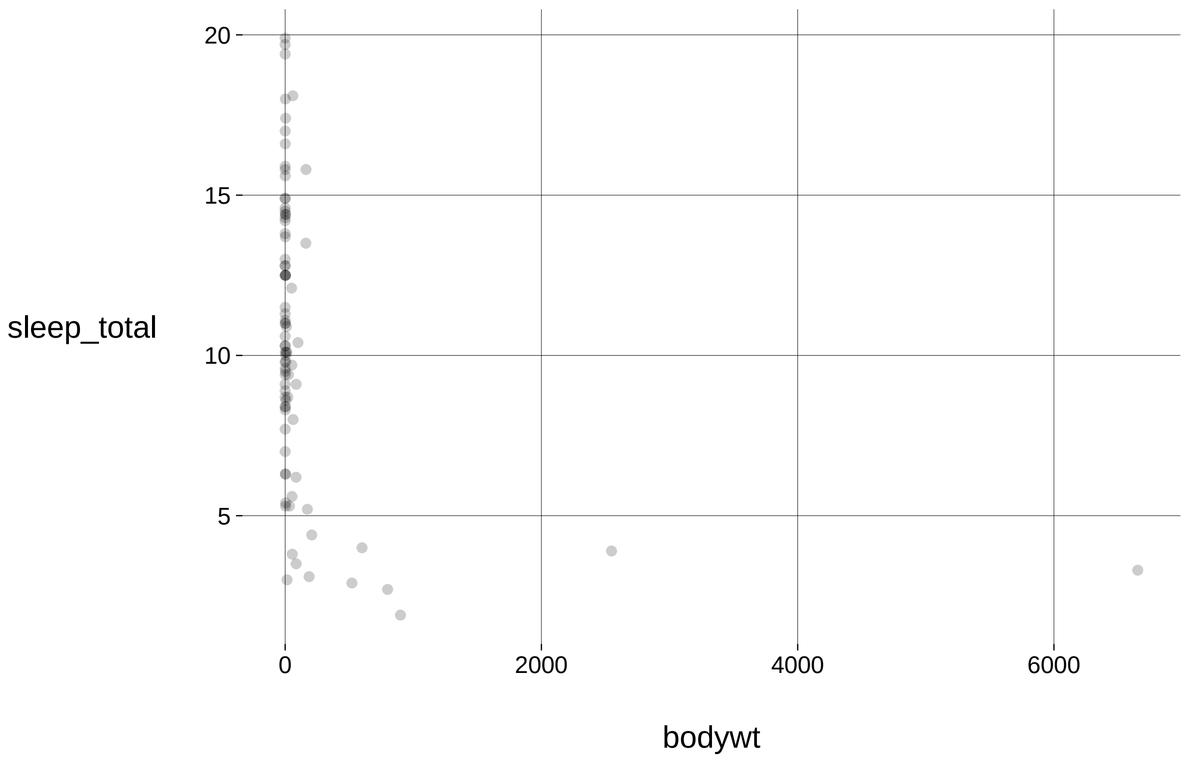
```
sono %>%
  filter(bodywt > 250) %>%
  select(name, bodywt) %>%
  arrange(bodywt)
```

```
## # A tibble: 6 x 2
##   name              bodywt
##   <chr>              <dbl>
## 1 Horse               521
## 2 Cow                 600
## 3 Pilot whale         800
## 4 Giraffe             900.
## 5 Asian elephant     2547
## 6 African elephant   6654
```

- Além disso, há muitos pontos sobrepostos. Em bom português, temos um problema de *overplotting*.
- Existem diversas maneiras de lidar com isso.

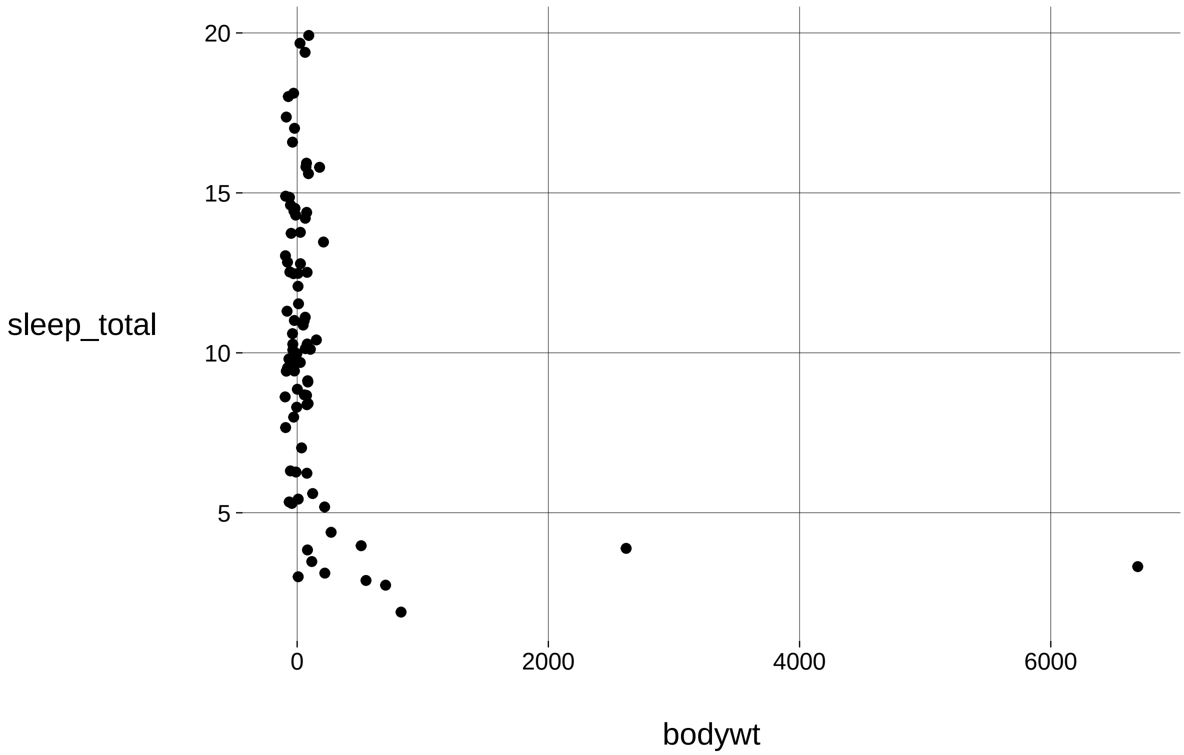
- A primeira delas é alterando a opacidade dos pontos. Isto é um ajuste na geometria apenas, pois a opacidade, aqui, não representa informação nenhuma.

```
sono %>%
  ggplot(aes(x = bodywt, y = sleep_total)) +
    geom_point(alpha = 0.2)
```



- Outra maneira é usar `geom_jitter` em vez de `geom_point`. "*Jitter*" significa "tremer". As posições dos pontos são ligeiramente perturbadas, para evitar colisões. Perdemos precisão, mas a visualização fica melhor.

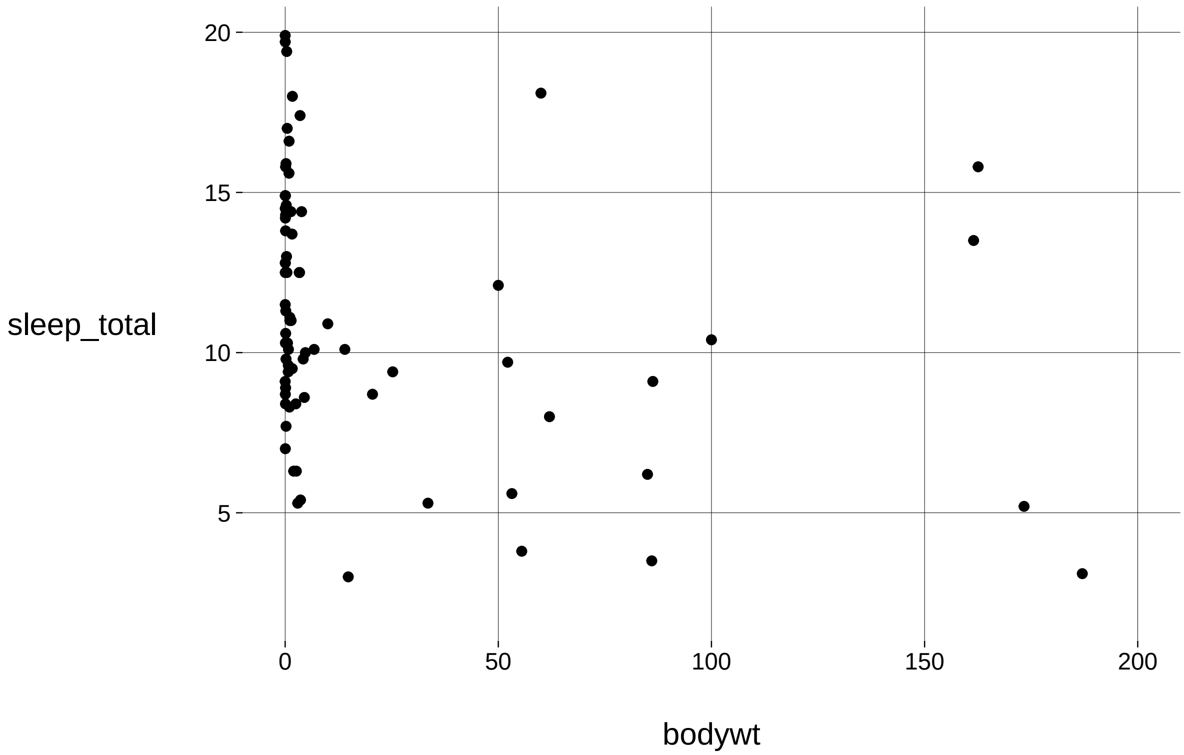
```
sono %>%
  ggplot(aes(x = bodywt, y = sleep_total)) +
    geom_jitter(width = 100)
```



- Vamos mudar os limites do gráfico para nos concentrarmos nos animais menos pesados. Observe que isto é um ajuste na escala.

```
sono %>%
  ggplot(aes(x = bodywt, y = sleep_total)) +
    geom_point() +
    scale_x_continuous(limits = c(0, 200))
```

```
## Warning: Removed 7 rows containing missing values (`geom_point()`).
```



- Nestes limites, a relação entre horas de sono e peso não é mais tão pronunciada.

4.4.2

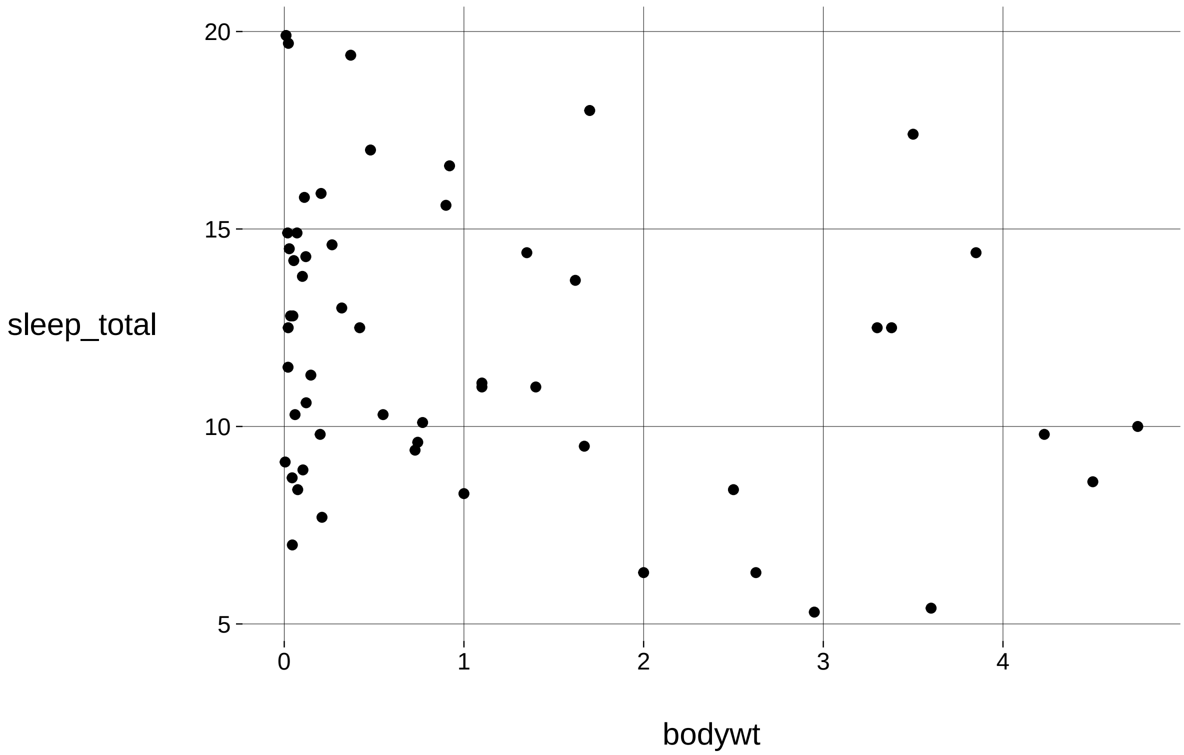
Horas de sono e peso corporal para animais pequenos

- Vamos restringir o gráfico a animais com no máximo 5kg.

```
limite <- 5
```

- Em vez de mudar a escala do gráfico, vamos filtrar as linhas do *data frame*:

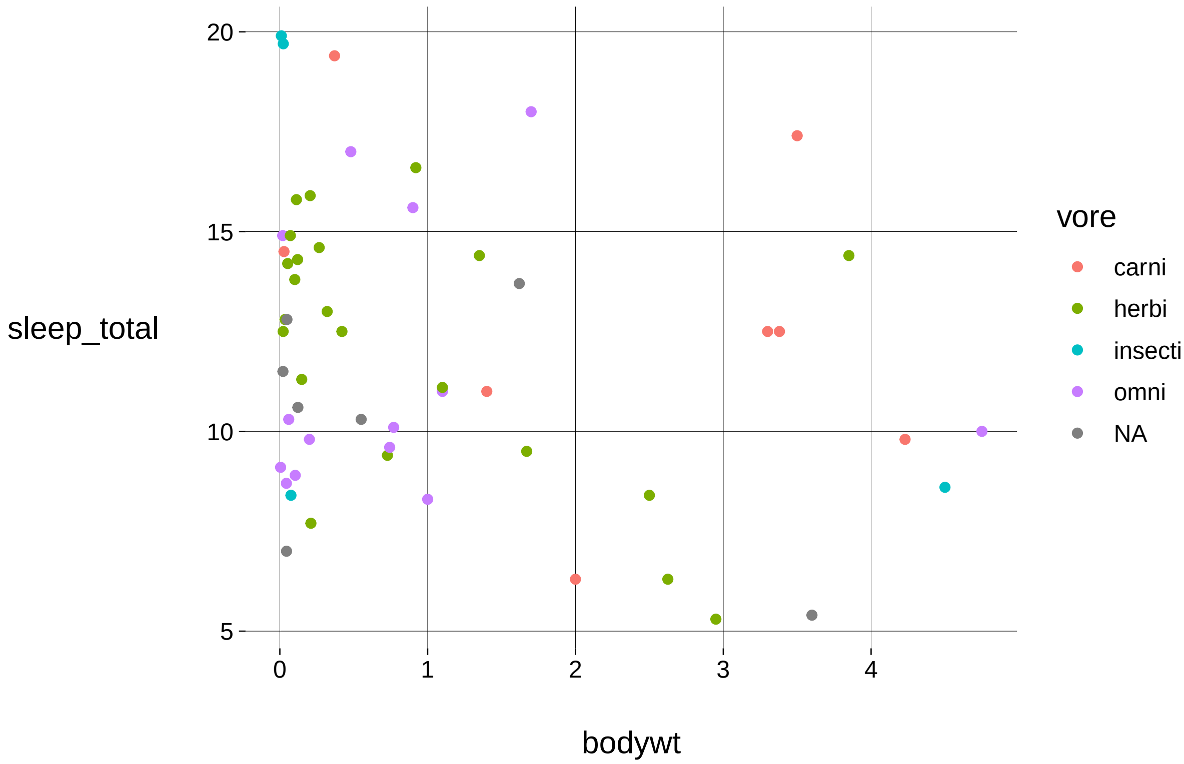
```
sono %>%
  filter(bodywt < limite) %>%
  ggplot(aes(x = bodywt, y = sleep_total)) +
    geom_point()
```



4.4.3 Incluindo a dieta

- Com a estética `color`. Observe como a legenda aparece automaticamente.

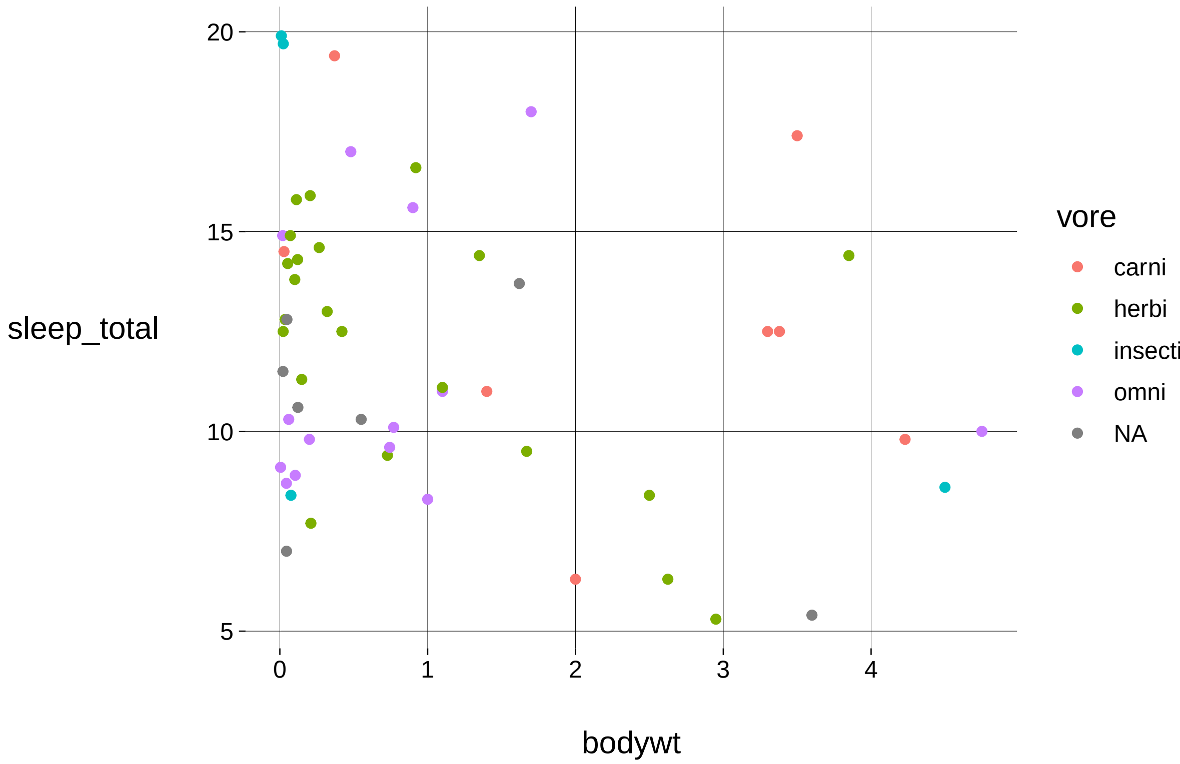
```
sono %>%
  filter(bodywt < limite) %>%
  ggplot(aes(x = bodywt, y = sleep_total, color = vore)) +
    geom_point()
```



4.4.4 A estética pode ser especificada na `geom`

- Compare com o código anterior.

```
sono %>%
  filter(bodywt < limite) %>%
  ggplot() +
    geom_point(aes(x = bodywt, y = sleep_total, color = vore))
```



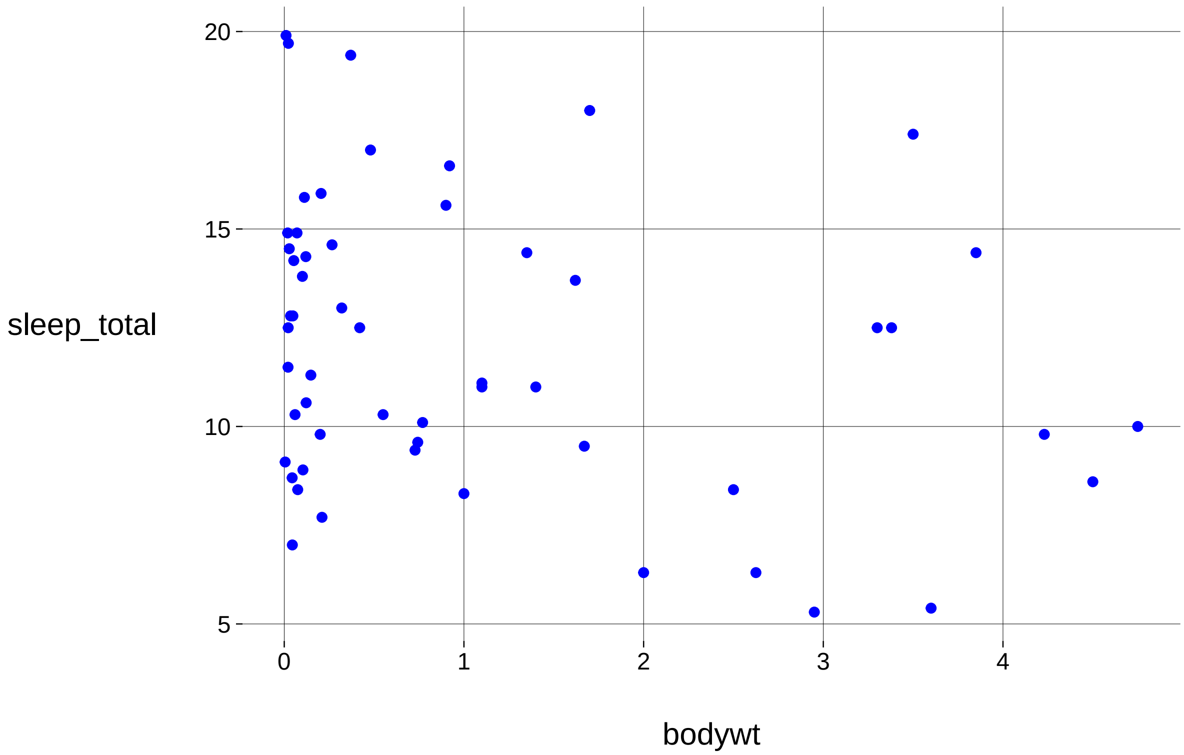
- Fazendo deste modo, a estética só vale para uma geometria. Se você acrescentar outras geometrias (linhas, por exemplo), a estética não valerá para elas.

4.4.5

Aparência fixa ou dependendo de variável?

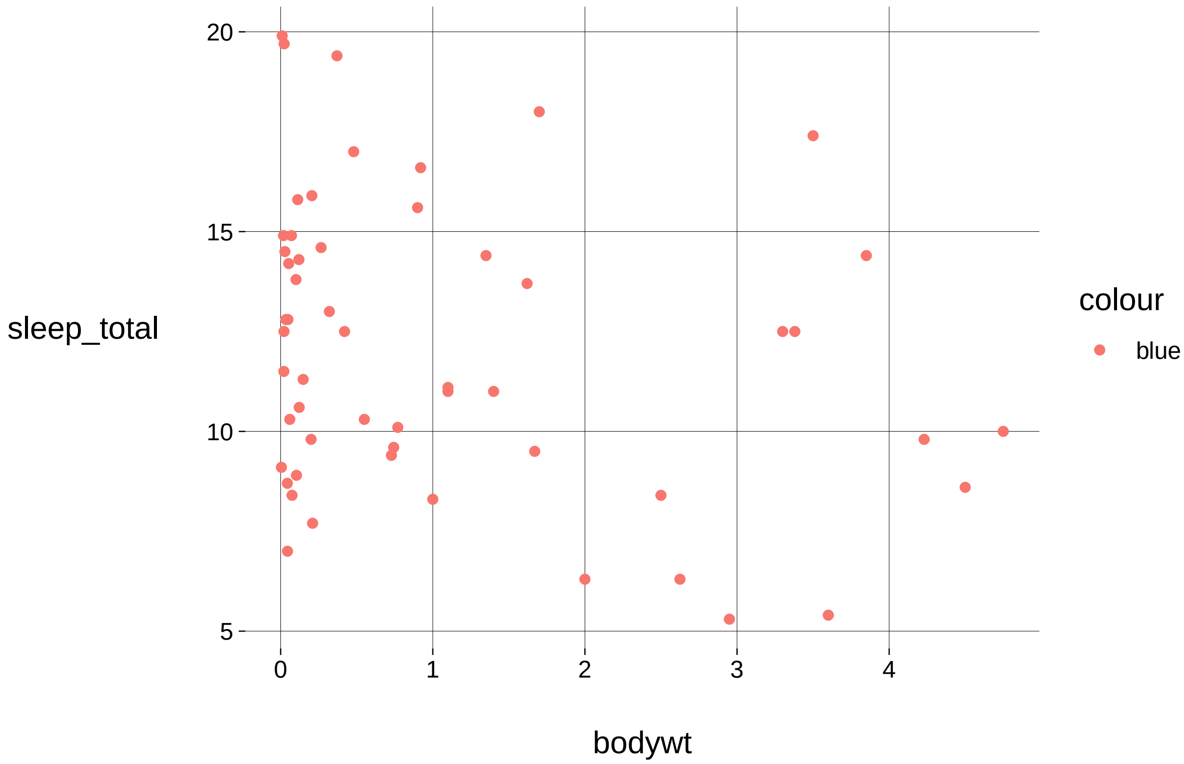
- Se for fixa, não é estética. Não representa informação.
- Se depender de variável, é estética. Representa informação.
- Compare o último *chunk* acima com:

```
sono %>%
  filter(bodywt < limite) %>%
  ggplot() +
    geom_point(aes(x = bodywt, y = sleep_total), color = 'blue')
```



- Se for uma estética, precisa estar associada a uma variável, não a um valor fixo. Um erro comum seria fazer:

```
sono %>%
  filter(bodywt < limite) %>%
  ggplot() +
    geom_point(aes(x = bodywt, y = sleep_total, color = 'blue'))
```

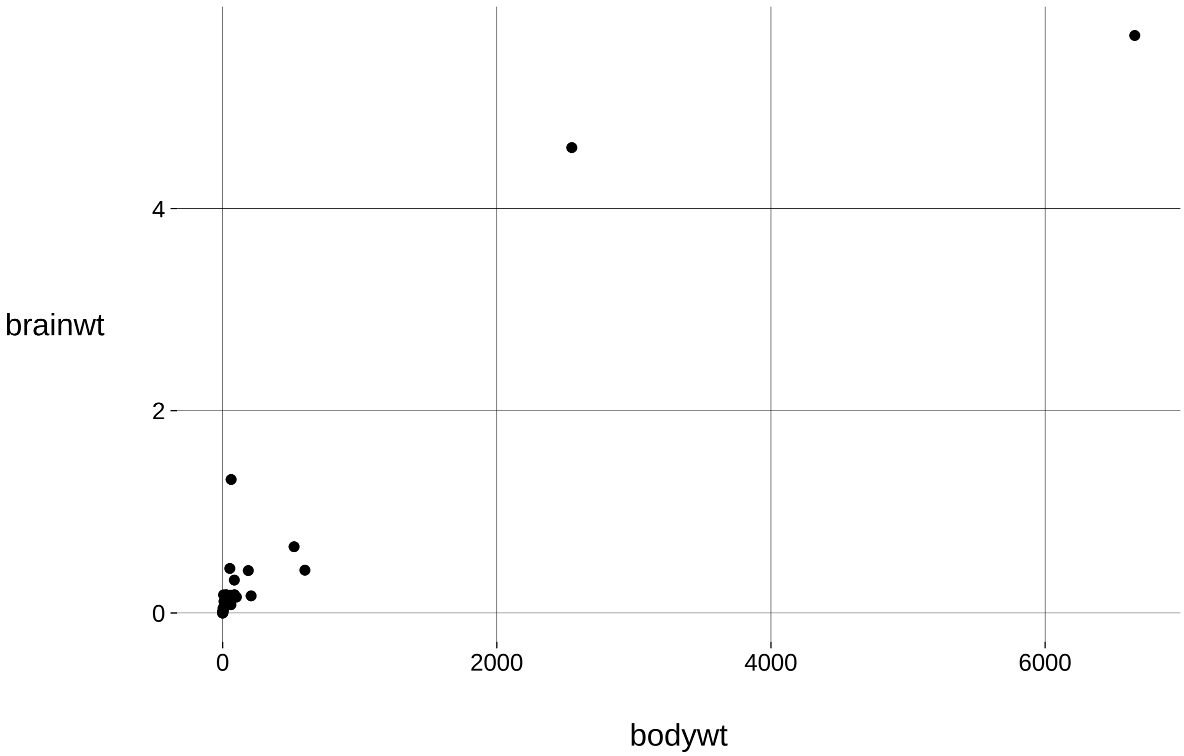


4.4.6 Uma correlação mais clara

- Peso cerebral versus peso corporal:

```
sono %>%
  ggplot(aes(x = bodywt, y = brainwt)) +
    geom_point()
```

```
## Warning: Removed 27 rows containing missing values (`geom_point()`).
```



- A mensagem de aviso (*warning*) diz que há 27 valores faltantes (`NA`) em `bodywt` ou `brainwt`. De fato:

```
sono %>%
  filter(is.na(bodywt)) %>%
  count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1     0
```

```
sono %>%
  filter(is.na(brainwt)) %>%
  count()
```

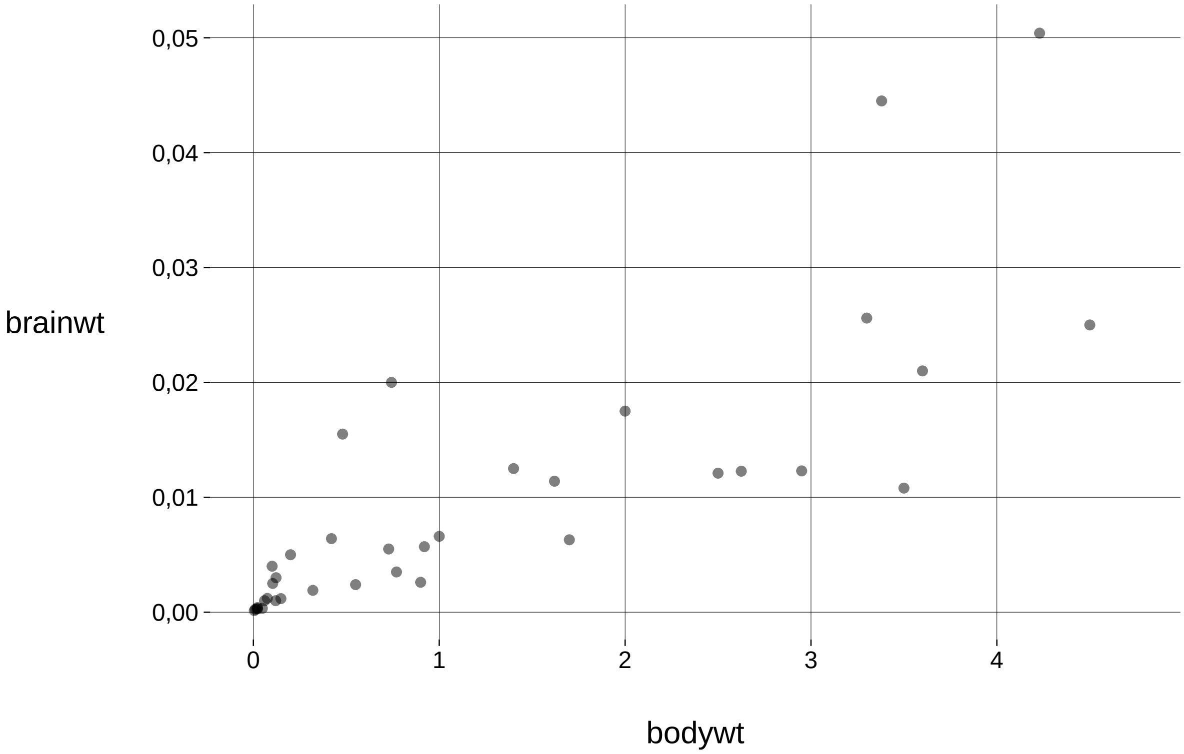
```
## # A tibble: 1 x 1
##       n
##   <int>
## 1    27
```

- Vamos restringir aos animais mais leves e mudar a opacidade:

```
sono %>%
  filter(bodywt < limite) %>%
  ggplot(aes(x = bodywt, y = brainwt)) +
```

```
    geom_point(alpha = .5)
```

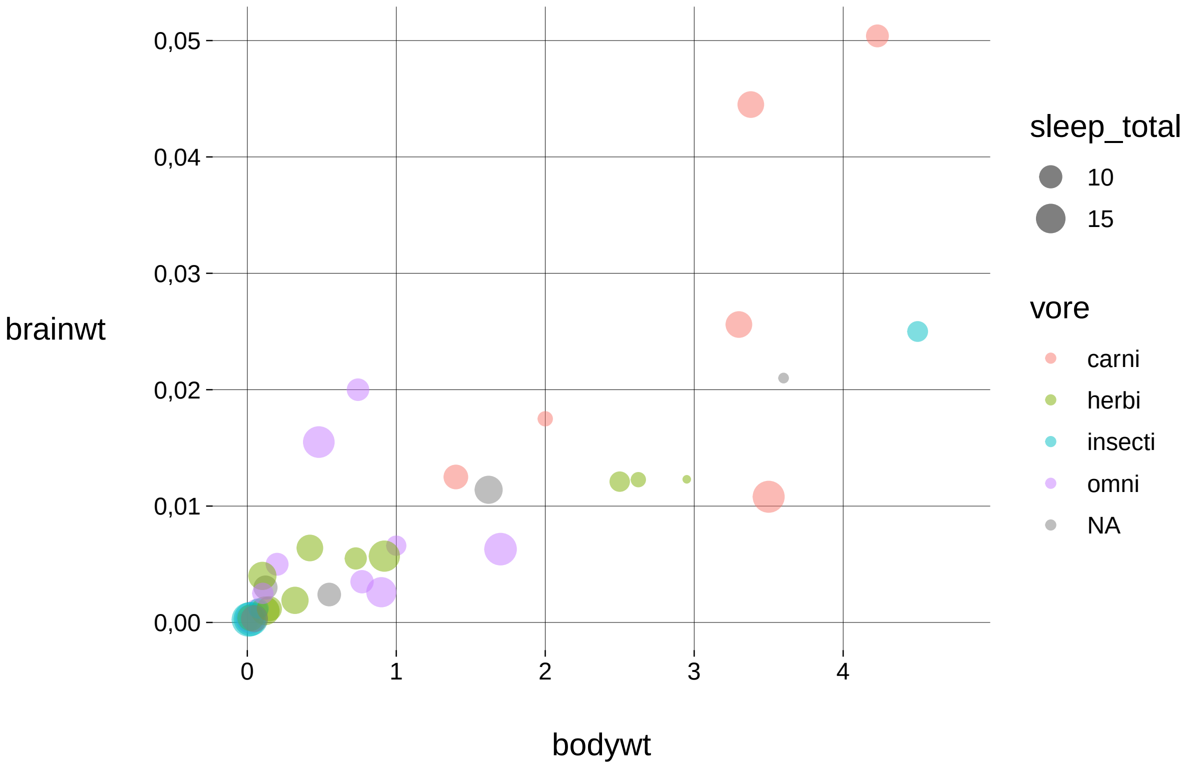
```
## Warning: Removed 18 rows containing missing values (`geom_point()`).
```



- Vamos incluir horas de sono e dieta. Observe as estéticas usadas.

```
sono %>%
  filter(bodywt < limite) %>%
  ggplot(
    aes(
      x = bodywt,
      y = brainwt,
      size = sleep_total,
      color = vore
    )
  ) +
    geom_point(alpha = .5)
```

```
## Warning: Removed 18 rows containing missing values (`geom_point()`).
```

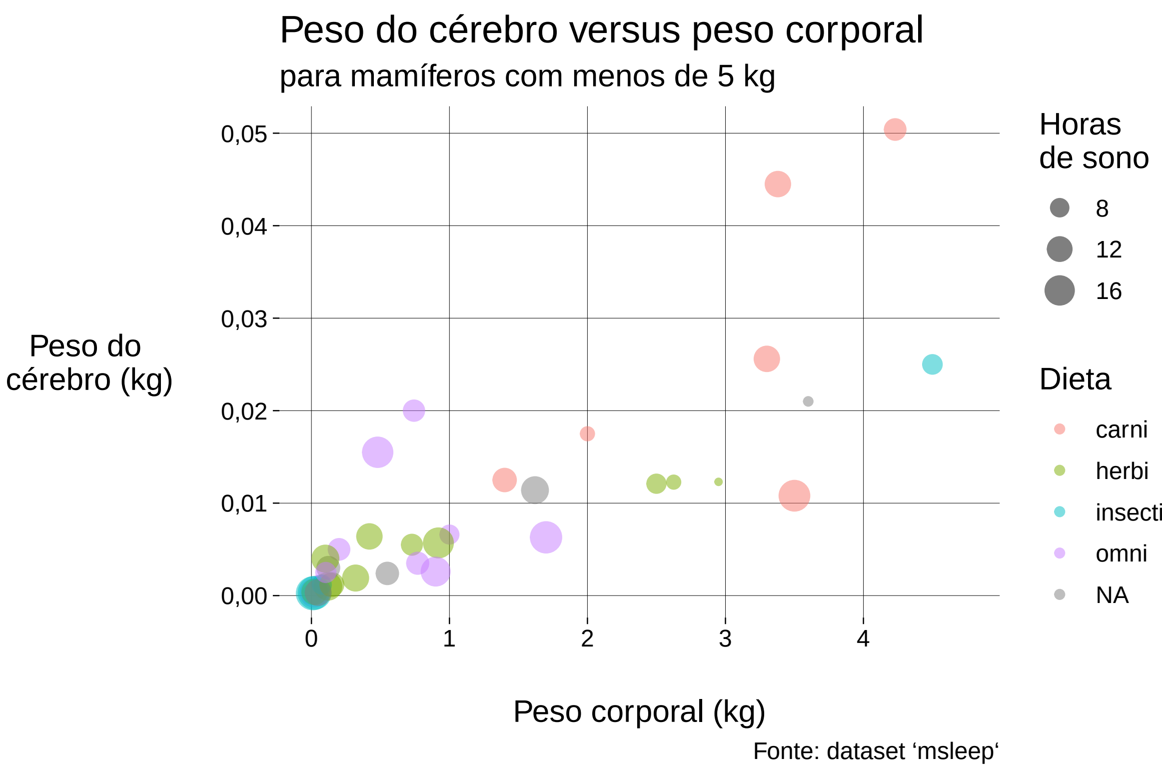


- Vamos mudar a escala dos tamanhos e incluir rótulos:

```
grafico <- sono %>%
  filter(bodywt < limite) %>%
  ggplot(
    aes(
      x = bodywt,
      y = brainwt,
      size = sleep_total,
      color = vore
    )
  ) +
    geom_point(alpha = .5) +
    scale_size(
      breaks = seq(0, 24, 4)
    ) +
    labs(
      title = 'Peso do cérebro versus peso corporal',
      subtitle = paste0(
        'para mamíferos com menos de ',
        limite,
        ' kg'
      ),
      caption = 'Fonte: dataset `msleep`',
      x = 'Peso corporal (kg)',
```

```
      y = 'Peso do\n cérebro (kg)',
      color = 'Dieta',
      size = 'Horas\nde sono'
    )

grafico
```

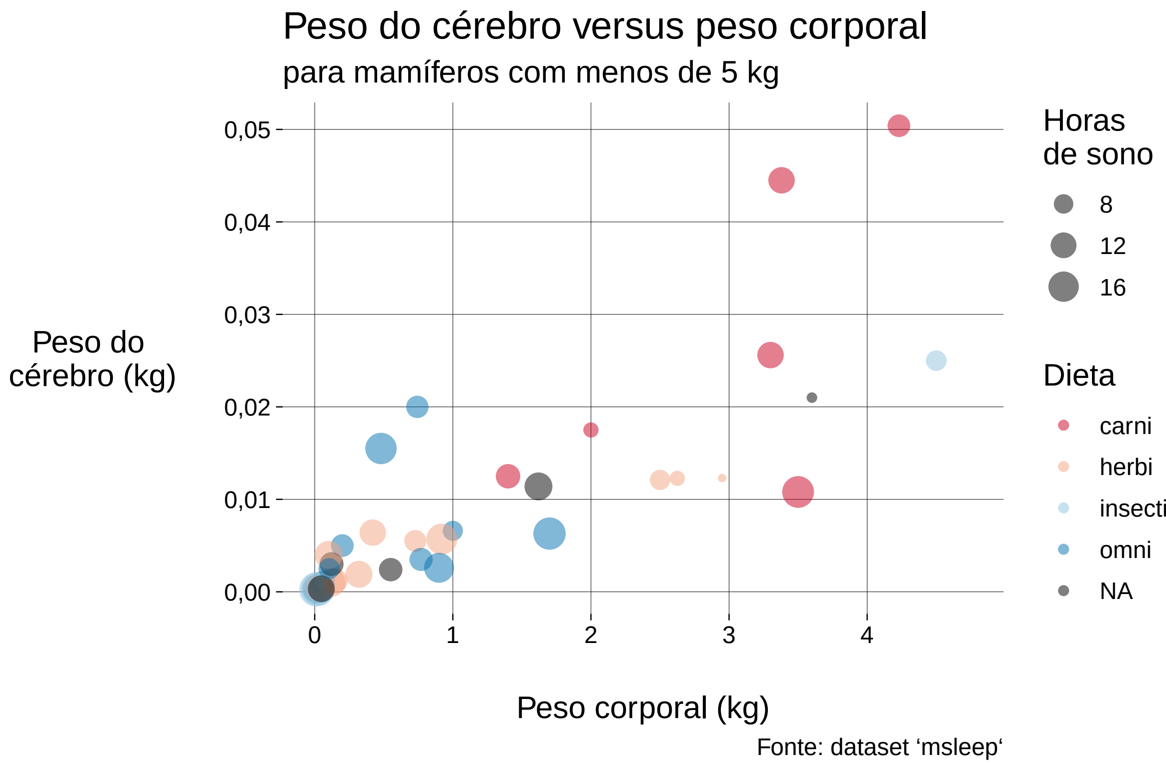
```
## Warning: Removed 18 rows containing missing values (`geom_point()`).
```



- Vamos mudar as cores usadas para a dieta, usando uma escala diferente.

```
grafico2 <- grafico +
  scale_color_discrete(
    palette = 'RdBu',
    na.value = 'black',
    type = scale_color_brewer
  )

grafico2
```

```
## Warning: Removed 18 rows containing missing values (`geom_point()`).
```



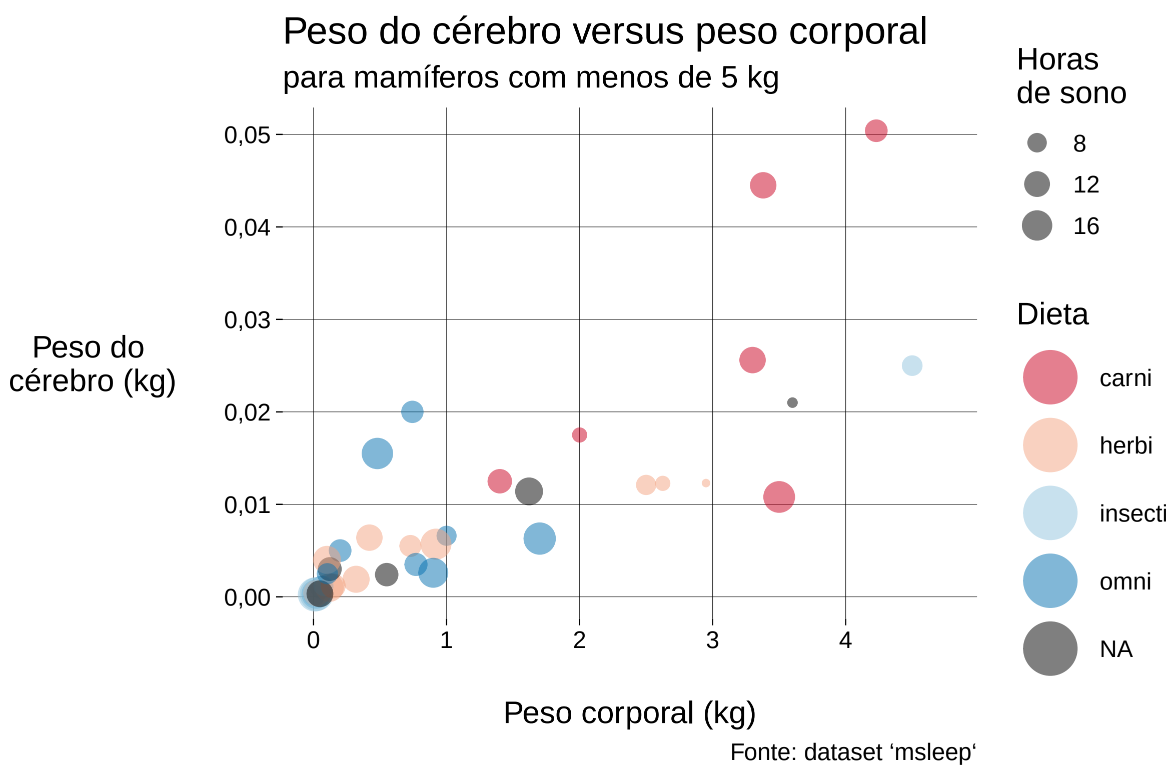
- Observe como usamos o gráfico já salvo na variável `grafico` e simplesmente acrescentamos a nova escala. Este tipo de "montagem" de gráficos `ggplot2` é bem conveniente, para evitar repetição de código.

- Um último ajuste na aparência: os pontos na legenda "Dieta" estão pequenos demais. Quase não identificamos as cores deles.

  Vamos usar a função `guides` para modificar (*override*) a estética `color` — apenas na legenda, não nos pontos mostrados no gráfico, cujos tamanhos representam o número de horas de sono — tornando o tamanho maior. Leia mais sobre `override.aes` neste *link* (em inglês).

```
grafico3 <- grafico2 +
  guides(color = guide_legend(override.aes = list(size = 10)))

grafico3
```

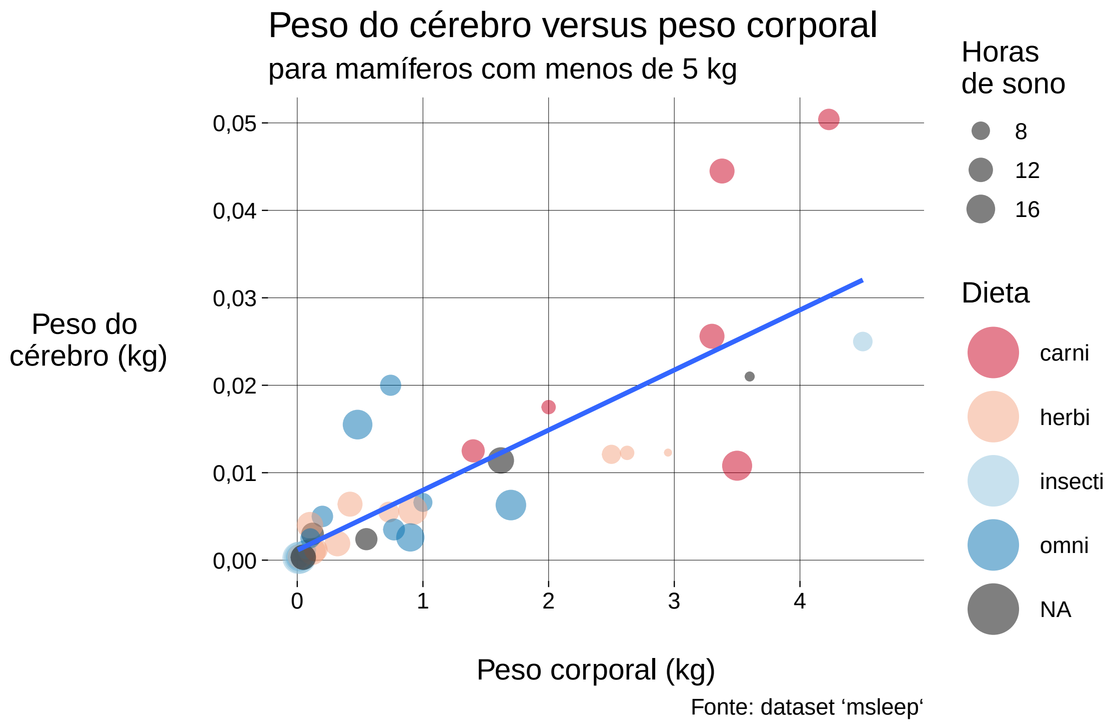
```
## Warning: Removed 18 rows containing missing values (`geom_point()`).
```



- Agora podemos finalmente comentar sobre a informação que o gráfico mostra sobre os dados:
  - De fato, existe uma correlação entre peso cerebral e peso corporal: quanto maior o peso corporal, maior o peso cerebral. Nada surprenndente.
  - Podemos fazer o `ggplot2` traçar uma reta de regressão com a geometria `geom_smooth`. Vamos falar mais sobre correlação em um capítulo futuro.

```
grafico4 <- grafico3 +
  geom_smooth(
    aes(group = 1),
    show.legend = FALSE,
    method = 'lm',
    se = FALSE,
    linewidth = 1
  )

grafico4
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



- Todos os carnívoros têm peso corporal maior que 1kg e peso cerebral maior ou igual a 10g.
- Só um carnívoro dorme 8 horas ou menos. Qual?
- Todos os insetívoros — com exceção de um (qual?) — são muito leves e dormem muito.
- Todos os onívoros têm menos de 2kg de peso corporal e 20g ou menos de peso cerebral.

4.5 Vídeo 2

https://youtu.be/c-LoZ9e8xWc

4.6 Histogramas e cia.

- A idéia agora é agrupar indivíduos em classes, dependendo do valor de uma variável quantitativa.

4.6.1 Distribuições de frequência

- Vamos nos concentrar nas horas de sono.

```
sono$sleep_total
```

```
##  [1] 12,1 17,0 14,4 14,9  4,0 14,4  8,7  7,0 10,1  3,0  5,3  9,4 10,0
## [14] 12,5 10,3  8,3  9,1 17,4  5,3 18,0  3,9 19,7  2,9  3,1 10,1 10,9
## [27] 14,9 12,5  9,8  1,9  2,7  6,2  6,3  8,0  9,5  3,3 19,4 10,1 14,2
## [40] 14,3 12,8 12,5 19,9 14,6 11,0  7,7 14,5  8,4  3,8  9,7 15,8 10,4
## [53] 13,5  9,4 10,3 11,0 11,5 13,7  3,5  5,6 11,1 18,1  5,4 13,0  8,7
## [66]  9,6  8,4 11,3 10,6 16,6 13,8 15,9 12,8  9,1  8,6 15,8  4,4 15,6
## [79]  8,9  5,2  6,3 12,5  9,8
```

- Antes de montar o histograma, vamos construir uma distribuição de frequência.
- A amplitude é a diferença entre o valor máximo e o valor mínimo. A função `range` não retorna a amplitude, mas sim os valores mínimo e máximo:

```
sono$sleep_total %>% range()
```

```
## [1]  1,9 19,9
```

- Vamos decidir que cada classe vai ter 2 horas. A função `cut` substitui os valores do vetor pelos nomes das classes:

```
sono$sleep_total %>%
  cut(breaks = seq(0, 20, 2), right = FALSE)
```

```
##  [1] [12,14) [16,18) [14,16) [14,16) [4,6)   [14,16) [8,10)  [6,8)
##  [9] [10,12) [2,4)   [4,6)   [8,10)  [10,12) [12,14) [10,12) [8,10)
## [17] [8,10)  [16,18) [4,6)   [18,20) [2,4)   [18,20) [2,4)   [2,4)
## [25] [10,12) [10,12) [14,16) [12,14) [8,10)  [0,2)   [2,4)   [6,8)
## [33] [6,8)   [8,10)  [8,10)  [2,4)   [18,20) [10,12) [14,16) [14,16)
## [41] [12,14) [12,14) [18,20) [14,16) [10,12) [6,8)   [14,16) [8,10)
## [49] [2,4)   [8,10)  [14,16) [10,12) [12,14) [8,10)  [10,12) [10,12)
## [57] [10,12) [12,14) [2,4)   [4,6)   [10,12) [18,20) [4,6)   [12,14)
## [65] [8,10)  [8,10)  [8,10)  [10,12) [10,12) [16,18) [12,14) [14,16)
## [73] [12,14) [8,10)  [8,10)  [14,16) [4,6)   [14,16) [8,10)  [4,6)
## [81] [6,8)   [12,14) [8,10)
## 10 Levels: [0,2) [2,4) [4,6) [6,8) [8,10) [10,12) [12,14) ... [18,20)
```

- A função `table` faz a contagem dos elementos de cada classe:

```
sono$sleep_total %>%
  cut(breaks = seq(0, 20, 2), right = FALSE) %>%
  table(dnn = 'Horas de sono') %>%
```

```
  as.data.frame()
```

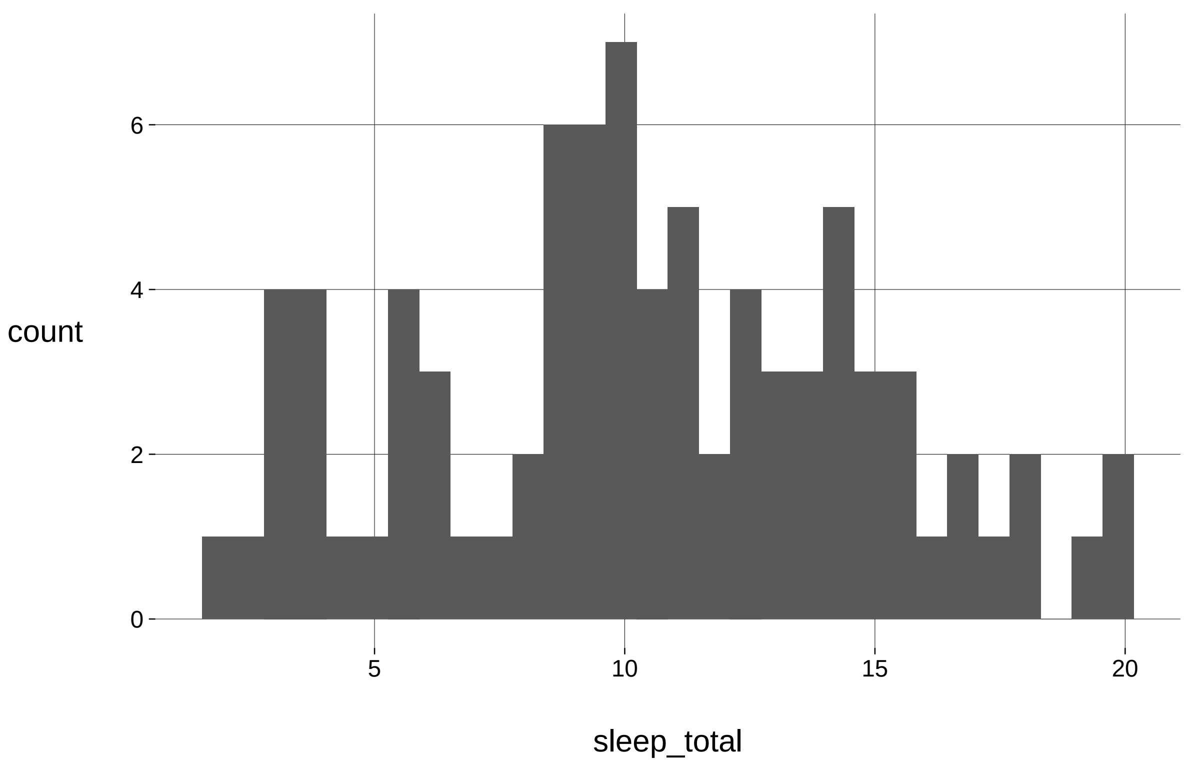
```
## # A tibble: 10 x 2
##   Horas.de.sono  Freq
##   <fct>         <int>
## 1 [0,2)             1
## 2 [2,4)             8
## 3 [4,6)             7
## 4 [6,8)             5
## 5 [8,10)           17
## 6 [10,12)          14
## # i 4 more rows
```

4.6.2 Histograma

- Na verdade, o `ggplot2` já faz esses cálculos para nós.
- O *default* é criar 30 classes (*bins*):

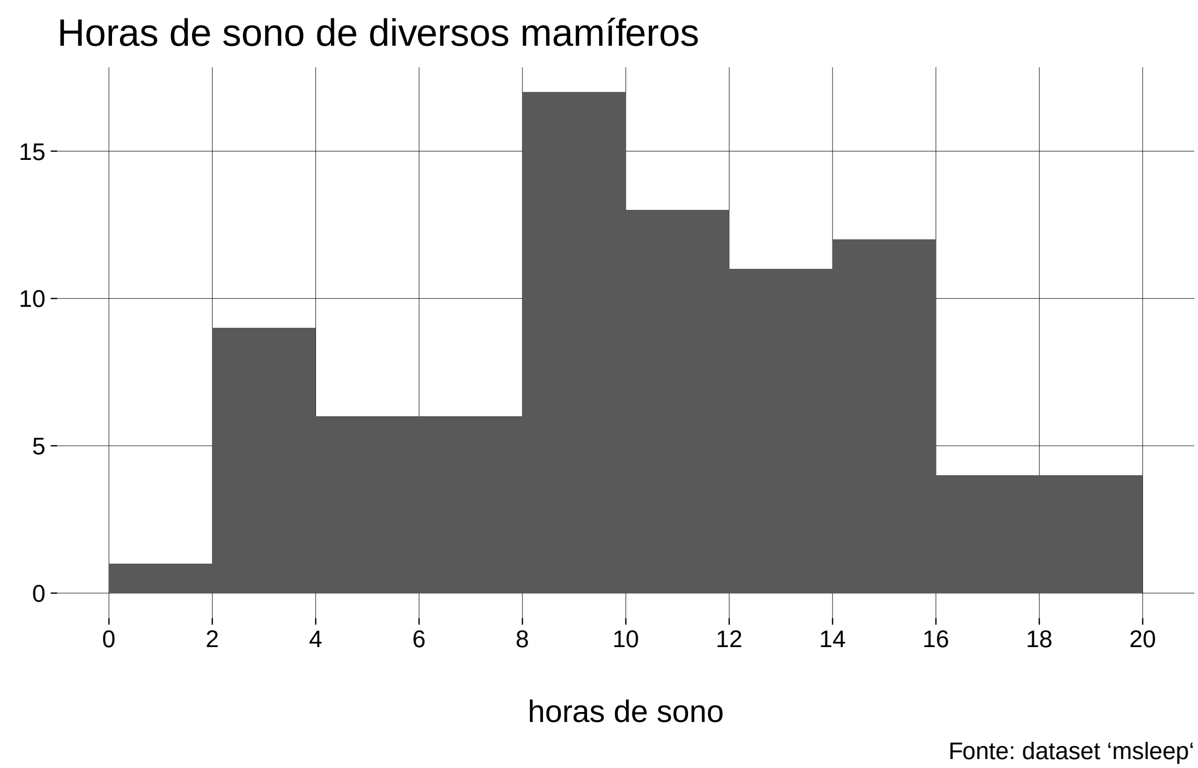
```
sono %>%
  ggplot(aes(x = sleep_total)) +
    geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



- Vamos mudar isto passando um vetor de limites das classes (*breaks*). Vamos acrescentar rótulos também:

```
sono %>%
  ggplot(aes(x = sleep_total)) +
    geom_histogram(breaks = seq(0, 20, 2)) +
    scale_x_continuous(breaks = seq(0, 20, 2)) +
    labs(
      title = 'Horas de sono de diversos mamíferos',
      x = 'horas de sono',
      y = NULL,
      caption = 'Fonte: dataset `msleep`'
    )
```



- Nossas impressões:
  - A classe que mais tem elementos é a de 8 a 10 horas.
  - A distribuição é mais ou menos simétrica.
  - A distribuição tem forma aproximada de sino: há poucos mamíferos com valores extremos de horas de sono; a maioria está próxima do valor médio:

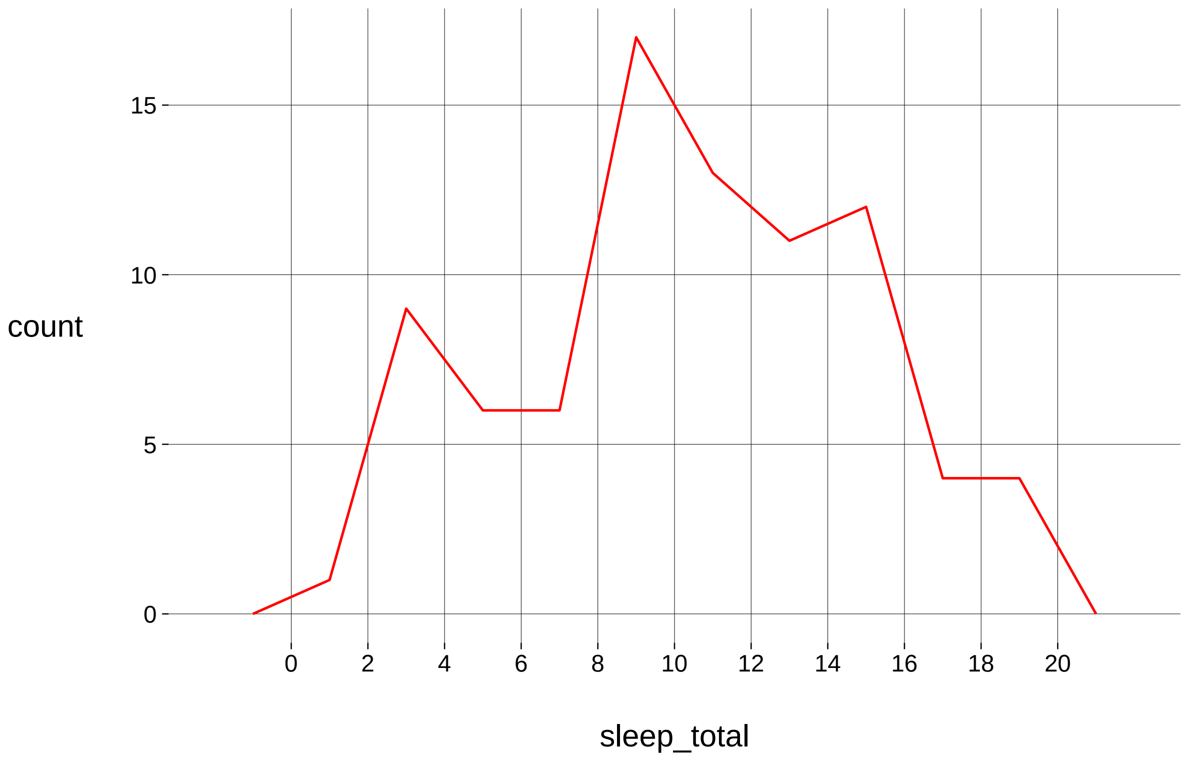
    ```
    mean(sono$sleep_total)
    ```

    ```
    ## [1] 10,43373
    ```

4.6.3 Polígono de frequência

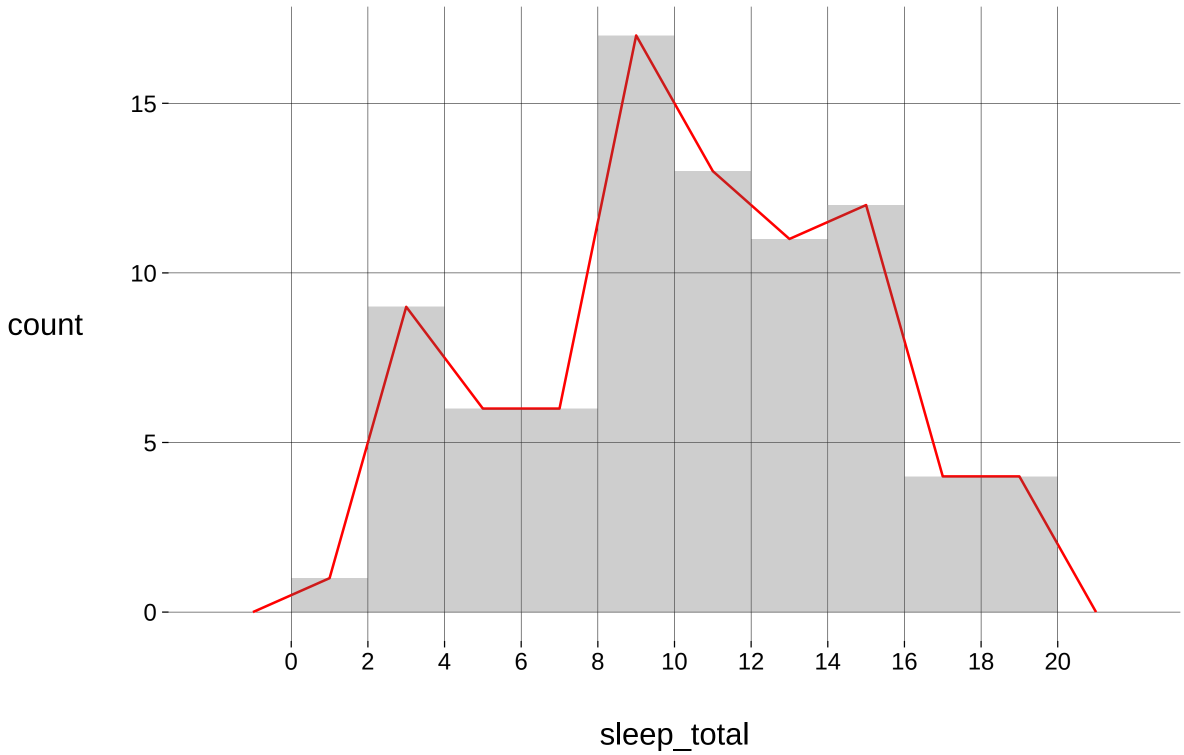
- Em vez das barras do histograma, podemos desenhar uma linha ligando seus topos.
- O resultado é um polígono de frequência.

```
pf <- sono %>%
  ggplot(aes(x = sleep_total)) +
    geom_freqpoly(breaks = seq(0, 20, 2), color = 'red') +
    scale_x_continuous(breaks = seq(0, 20, 2))

pf
```



- Vamos sobrepor o polígono de frequência ao histograma, para deixar claro o que está acontecendo:

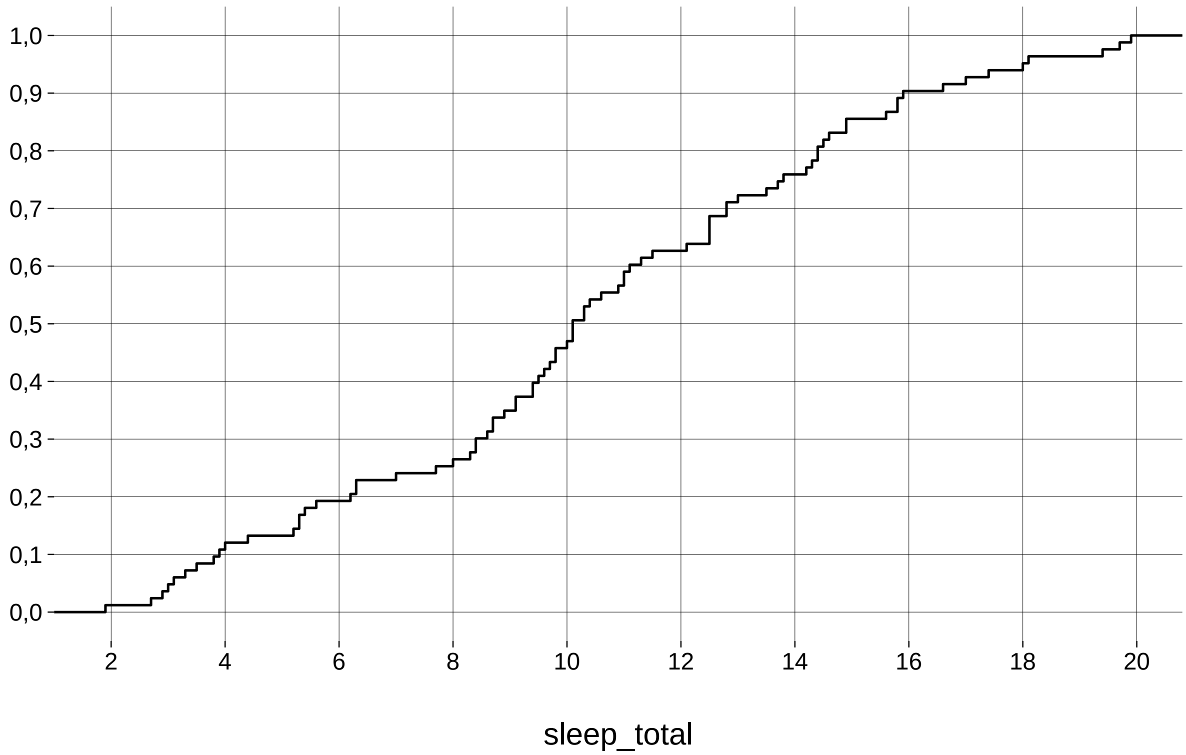
```
pf + geom_histogram(breaks = seq(0, 20, 2), alpha = .3)
```



4.7 Ogiva

- A ogiva é um gráfico que mostra a frequência acumulada: para cada valor v da variável no eixo x, a proporção de indivíduos com valor menor ou igual a v.
- A geometria `geom_step` gera o gráfico de uma função degrau.
- Cada geometria está ligada a uma `stat`, um algoritmo para computar o que vai ser desenhado. Aqui, passamos para a geometria a função `ecdf` (*empirical cumulative distribution function*), do pacote `stats`, que calcula as frequências acumuladas.

```
sono %>%
  ggplot(aes(x = sleep_total)) +
    geom_step(stat = 'ecdf') +
    scale_x_continuous(breaks = seq(0, 20, 2)) +
    scale_y_continuous(breaks = seq(0, 1, .1)) +
    labs(y = NULL)
```



- Com a ogiva, podemos obter informações difíceis de visualizar no histograma. Por exemplo:
  - Cerca de 20% dos mamíferos têm menos de 6 horas de sono.
  - Cerca de metade dos mamíferos têm menos de 10 horas de sono.
  - Cerca de 10% dos mamíferos têm mais de 16 horas de sono.

4.8 Ramos e folhas

- No início dos anos 1900, quando estatísticas eram feitas à mão, Arthur Bowley criou os diagramas de ramos e folhas.
- Um diagrama de ramos e folhas é, basicamente, uma listagem de todos os valores de uma variável, agrupados de maneira que todos os valores de uma classe (i.e., de uma linha) têm os algarismos iniciais dentro de um intervalo.
- Para as horas de sono dos mamíferos:

```
sono$sleep_total %>%
  stem()
```

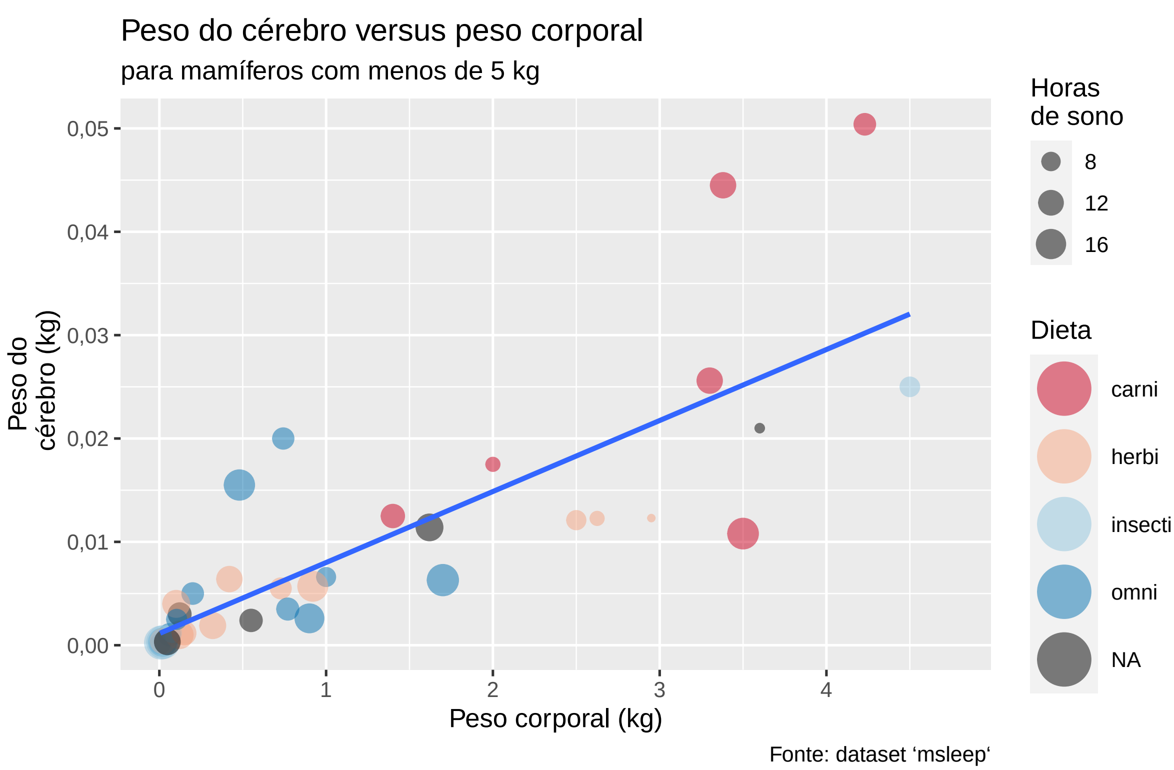
```
##
##   The decimal point is at the |
##
```

```
##    0 | 9
##    2 | 79013589
##    4 | 0423346
##    6 | 23307
##    8 | 03446779114456788
##   10 | 01113346900135
##   12 | 15555880578
##   14 | 234456996889
##   16 | 604
##   18 | 01479
```

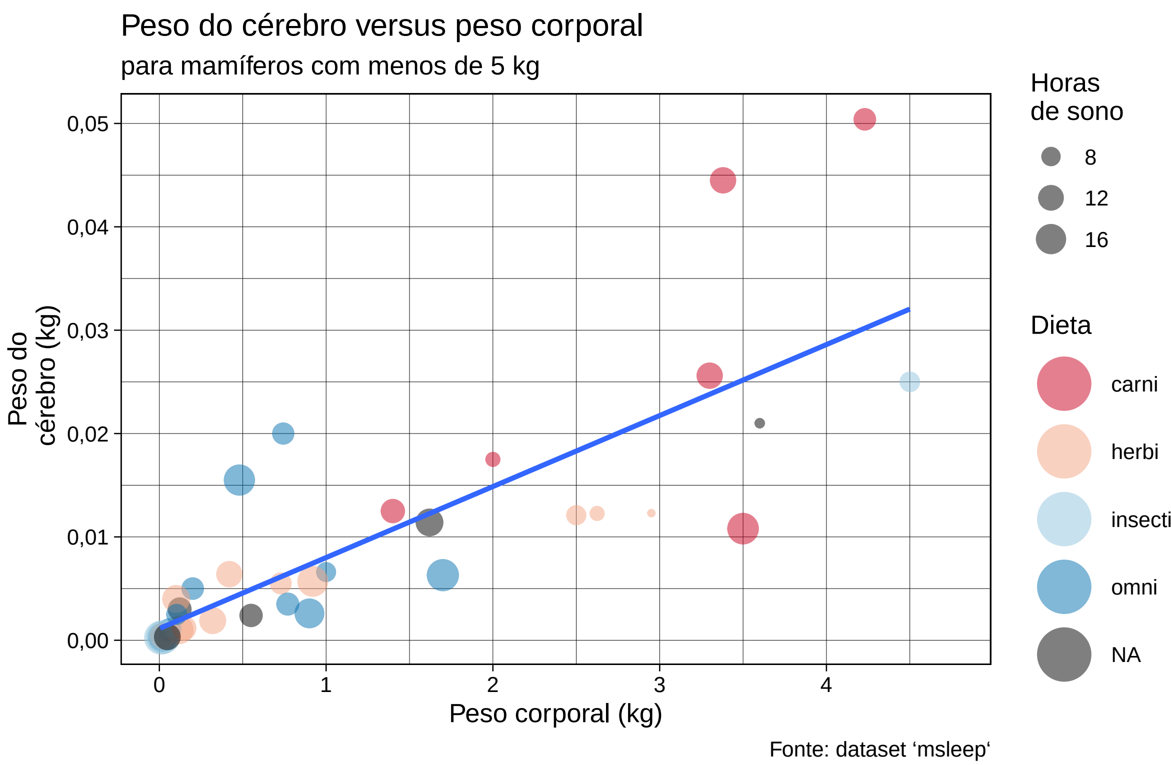
- A primeira linha representa um indivíduo com $0{,}9$ horas de sono.
- A penúltima linha representa 3 valores:
  - $16{,}6$
  - $17{,}0$
  - $17{,}4$

4.9 Personalização do tema

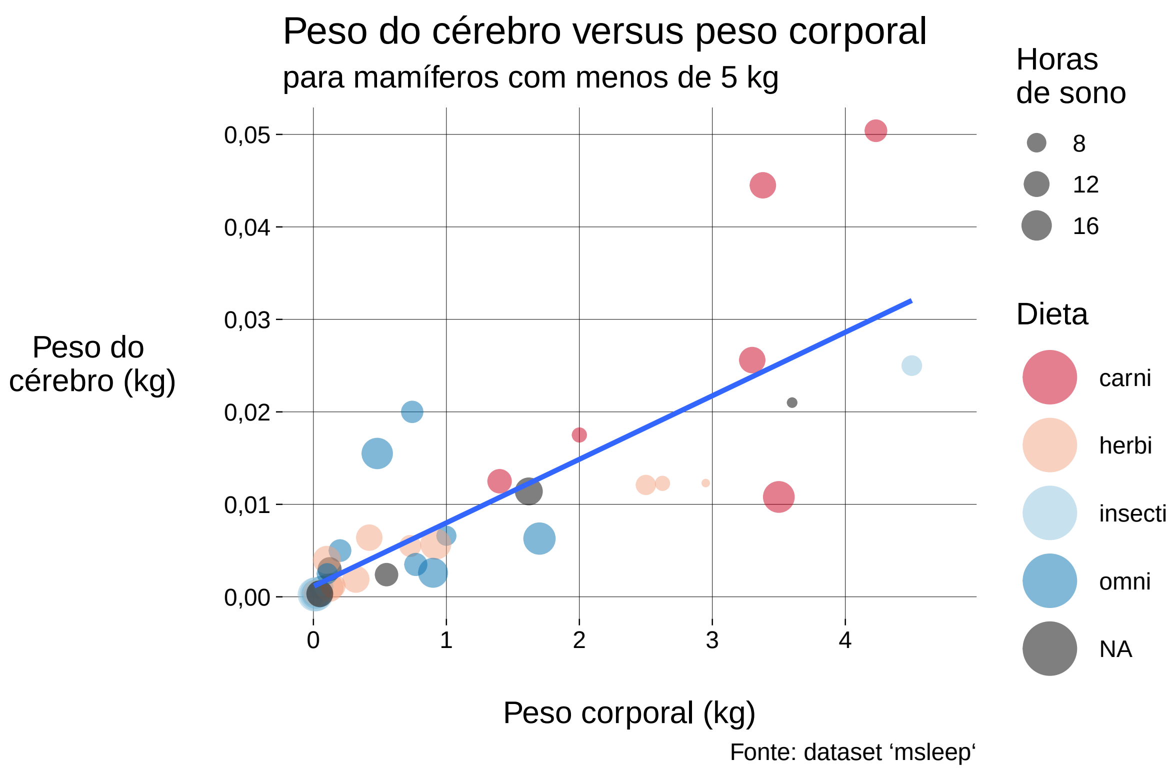
- O `ggplot2` tem um tema *default*, chamado `theme_gray`, que gera o *scatterplot* de um exemplo anterior deste capítulo do seguinte modo:



- Para este material, escolhi o tema `theme_linedraw`, que usa linhas pretas sobre fundo branco:



- Para deixar os gráficos mais leves e facilitar a leitura, fiz as seguintes alterações no tema:
  - Mudei o tamanho do texto dos rótulos.
  - Fiz o rótulo do eixo y aparecer na horizontal; embora isto ocupe um pouco mais de espaço, evita que o leitor tenha que girar a cabeça para ler o rótulo.
  - Eliminei as linhas dos eixos, para o gráfico ficar mais leve.
  - Eliminei a moldura da área de dados, para o gráfico ficar mais leve.
  - Eliminei a grade secundária, para o gráfico ficar mais leve.
- O resultado é



- Os meus comandos para alterar o tema são

```
# Tamanho do texto depende do formato de saída (html ou pdf):
plot_text_size = ifelse(is_html_output(), 12, 13)

# Tema mais leve:
theme_set(
  theme_linedraw() +
    theme(
      # Tamanho do texto
      text = element_text(size = plot_text_size),
      # Eixo y
      axis.title.y.left = element_text(
        # Nunca girar o rótulo do eixo y
        angle = 0,
        # Separar o rótulo do eixo um pouco
        margin = margin(r = 20),
        # Posicionar verticalmente no meio
        vjust = .5
      ),
      # Eixo y secundário (à direita), quando presente
      axis.title.y.right = element_text(
        # Nunca girar o rótulo do eixo y
        angle = 0,
        # Separar o rótulo do eixo um pouco
```

```
        margin = margin(l = 20),
        # Posicionar verticalmente no meio
        vjust = .5
      ),
      # Não colocar marcas no eixo y secundário
      axis.ticks.y.right = element_blank(),
      # Separar o eixo x do rótulo um pouco mais
      axis.title.x.bottom = element_text(
        margin = margin(t = 20)
      ),
      # Eliminar linhas dos eixos
      axis.line = element_blank(),
      # Eliminar a moldura da área de dados
      panel.border = element_blank(),
      # Eliminar a grade secundária
      panel.grid.minor = element_blank()
    )
)
```

4.10 Exercícios

Não se esqueça de incluir títulos nos gráficos e rótulos nos eixos.

4.10.1 Peso cerebral e peso corporal

1. Observe os comandos que geraram o gráfico `grafico4`.
2. O que acontece se você retirar `aes(group = 1)` da chamada a `geom_smooth`? Explique.
3. O que acontece se você mudar `show.legend = FALSE` para `show.legend = TRUE` na chamada a `geom_smooth`? Explique.
4. O que acontece se você mudar `se = FALSE` para `se = TRUE` na chamada a `geom_smooth`? Explique.
5. Acrescente ao gráfico a camada `facet_wrap(~vore)`. O que acontece?
6. Examine o *data frame* `sono` e identifique o único insetívoro com mais de 4kg.
7. Instale o pacote `gg_repel` e acrescente ao gráfico `grafico4` (não facetado) a geometria `geom_label_repel` (consulte a ajuda) para rotular o mamífero insetívoro identificado no item anterior com o seu nome, sem cobrir outros pontos do gráfico. Cuidado para não alterar a legenda que já existe.

4.10.2

Peso cerebral e horas de sono

Use o *data frame* `sono` definido como

```
library(ggplot2)

sono <- msleep %>%
  select(
    name, order, genus, vore, bodywt,
    brainwt, awake, sleep_total
  )
```

1. Construa um histograma da variável `brainwt`. Escolha o número de classes que você achar melhor. O que acontece com os valores `NA`?
2. Descubra que função da forma `scale_x_...` usar para fazer com que o eixo x tenha uma escala logarítmica. Gere um novo histograma.
3. Qual dos dois histogramas é melhor para responder a pergunta "*Qual a faixa de peso cerebral que tem mais animais?*" de forma satisfatória?
4. Construa um *scatter plot* de horas de sono versus peso do cérebro. Você percebe alguma correlação entre estas variáveis? Se precisar, concentre-se em um subconjunto dos dados.
5. Usando `geom_smooth` (leia a respeito), sobreponha uma reta de regressão ao gráfico de dispersão, usando o método `lm` e sem o erro padrão (i.e., com `se = FALSE`). O que você observa? Discuta.

4.10.3

Igualdade de gênero entre furacões?

Este artigo tenta achar uma relação entre o gênero do nome de um furacão e a quantidade de vítimas fatais provocadas por ele.

Os dados estão no pacote `DAAG`, que deve ser instalado:

```
if (!require(DAAG))
  install.packages("DAAG")
```

Vamos usar apenas algumas das variáveis, com nomes em português.

```r
df <- hurricNamed %>%
  as_tibble() %>%
  transmute(
    id = paste(Year, Name, sep = '-'),
    nome = Name,
    ano = Year,
    velocidade = LF.WindsMPH * 1.8,      # convertido para km/h
    pressao = LF.PressureMB,             # mbar
    prejuizo = BaseDam2014 %>% round(), # milhões de dólares de 2014
    mortes = deaths,
    genero = mf
  )
```

1. Crie histogramas para as seguintes variáveis, escolhendo a quantidade de barras que você achar melhor.

   - `velocidade`
   - `prejuizo`
   - `mortes`

   Não se esqueça de incluir títulos nos gráficos e rótulos nos eixos.

   Comente os histogramas.

2. Os histogramas de prejuízos e mortes não ficaram bons. Vamos gerar histogramas transformados.

   No *data frame*, crie duas novas colunas:

   - `logprejuizo`: *logaritmo* do prejuízo (na base 10)
   - `logmortes`: *logaritmo* do número de mortes (na base 10)

   Agora, gere histogramas destas duas novas variáveis.

3. O que significa o valor do logaritmo do prejuízo na base 10?

4. O que significa o valor do logaritmo do número de mortes na base 10?

5. Por que o histograma do logaritmo do número de mortes vem com uma mensagem de aviso?

6. Por que isto não acontece com o logaritmo do prejuízo?

7. Faça um gráfico de dispersão com `pressao` no eixo y e `velocidade` no eixo x.

8. Usando `geom_smooth` (leia a respeito), sobreponha uma reta de regressão ao gráfico, usando o método `lm` e sem o erro padrão (i.e., com `se = FALSE`). O que você observa? Discuta.

9. Faça um gráfico de dispersão com `logmortes` no eixo y e `pressao` no eixo x.

10. Usando `geom_smooth` (leia a respeito), sobreponha uma reta de regressão ao gráfico, usando o método `lm` e sem o erro padrão (i.e., com `se = FALSE`). O que você observa? Discuta.
11. Faça um gráfico de dispersão com `logmortes` no eixo y e `pressao` no eixo x, com pontos coloridos de acordo com o gênero do nome do furacão.
12. Usando `geom_smooth` (leia a respeito), sobreponha retas de regressão ao gráfico, uma para cada gênero, usando o método `lm` e sem o erro padrão (i.e., com `se = FALSE`). O que você observa? Discuta.

> Visualizações como esta ajudam a explorar os dados, mas não servem para testar rigorosamente a hipótese de que furacões mulheres matam mais do que furacões homens.
>
> Mais adiante no curso, vamos aprender a fazer testes mais rigorosos sobre hipóteses como esta.

CAPÍTULO 5

Visualização com ggplot2 (continuação)

Busque mais informações sobre os pacotes `tidyverse` e `ggplot2` nas referências recomendadas.

5.1 Vídeo 1

https://youtu.be/TjgLDeIQHIc

5.2 *Boxplots*

5.2.1 Conjunto de dados

- Vamos continuar a trabalhar com os dados sobre as horas de sono de alguns mamíferos:

```
sono <- msleep %>%
  select(name, vore, order, sleep_total)
```

```
sono
```

```
## # A tibble: 83 x 4
##   name                       vore  order        sleep_total
##   <chr>                      <chr> <chr>              <dbl>
## 1 Cheetah                    carni Carnivora           12.1
## 2 Owl monkey                 omni  Primates            17
## 3 Mountain beaver            herbi Rodentia            14.4
## 4 Greater short-tailed shrew omni  Soricomorpha        14.9
## 5 Cow                        herbi Artiodactyla         4
## 6 Three-toed sloth           herbi Pilosa              14.4
## # i 77 more rows
```

5.2.2 Mediana e quartis

- Para entender *boxplots*, precisamos, antes, entender algumas medidas.
- Se tomarmos as quantidades de horas de sono de todos os animais do conjunto de dados e classificarmos estas quantidades em ordem crescente, vamos ter:

```
horas <- sono %>%
  pull(sleep_total) %>%
  sort()

horas
```

```
##  [1]  1,9  2,7  2,9  3,0  3,1  3,3  3,5  3,8  3,9  4,0  4,4  5,2  5,3
## [14]  5,3  5,4  5,6  6,2  6,3  6,3  7,0  7,7  8,0  8,3  8,4  8,4  8,6
## [27]  8,7  8,7  8,9  9,1  9,1  9,4  9,4  9,5  9,6  9,7  9,8  9,8 10,0
## [40] 10,1 10,1 10,1 10,3 10,3 10,4 10,6 10,9 11,0 11,0 11,1 11,3 11,5
## [53] 12,1 12,5 12,5 12,5 12,5 12,8 12,8 13,0 13,5 13,7 13,8 14,2 14,3
## [66] 14,4 14,4 14,5 14,6 14,9 14,9 15,6 15,8 15,8 15,9 16,6 17,0 17,4
## [79] 18,0 18,1 19,4 19,7 19,9
```

- Quantos valores são?

```
length(horas)
```

```
## [1] 83
```

- O valor que está bem no meio desta fila — i.e., na posição 42 — é a mediana:

```
horas[ceiling(length(horas) / 2)]
```

```
## [1] 10,1
```

- Em R:

```
median(horas)
```

```
## [1] 10,1
```

Mediana e média são coisas muito diferentes.

Por acaso, neste exemplo, a média das horas é próxima da mediana:

```
mean(horas)
```

```
## [1] 10,43373
```

Isto costuma acontecer quando a distribuição dos dados é aproximadamente simétrica.

- Os quartis são os valores que estão nas posições $\frac{1}{4}$, $\frac{1}{2}$ e $\frac{3}{4}$ da fila. São o primeiro, segundo e terceiro quartis, respectivamente.

```
horas[
  c(
    ceiling(length(horas) / 4),
    ceiling(length(horas) / 2),
    ceiling(3 * length(horas) / 4)
  )
]
```

```
## [1]  7,7 10,1 13,8
```

- Sim, a mediana é o segundo quartil.
- Em R, a função `quantile` generaliza esta idéia: dado um número q entre 0 e 1, o quantil (com "N") q é o elemento que está na posição que corresponde à fração q da fila ordenada.

```
horas %>% quantile(c(.25, .5, .75))
```

```
##   25%   50%   75%
##  7,85 10,10 13,75
```

- Na verdade, R tem 9 algoritmos diferentes para calcular os quantis de uma amostra! Leia a ajuda da função `quantile` para conhecê-los.
- As diferenças entre nossos cálculos "à mão" e os resultados retornados por `quantile` são porque, em algumas situações, `quantile` calcula uma média ponderada entre elementos vizinhos. Por isso, `quantile` pode retornar valores que nem estão no vetor.
- Em R, a função `summary` mostra o mínimo, os quartis (com "R"), a média, e o máximo de um vetor:

```r
summary(horas)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    1,90    7,85   10,10   10,43   13,75   19,90
```

5.2.3 Média × mediana

- Vamos ver um exemplo simples para entender a diferença entre a média e a mediana.
- Imagine o seguinte vetor com as receitas mensais de algumas pessoas (em milhares de reais:)

```r
receitas <- c(1, 2, 2, 3.5, 1, 4, 1)
```

- Eis a mediana e a média deste vetor:

```r
summary(receitas)[c('Median', 'Mean')]
```

```
##   Median     Mean 
## 2,000000 2,071429
```

- A mediana e a média são bem próximas.
- Imagine, agora, que adicionamos ao vetor um sujeito com receita mensal de 100 mil reais:

```r
receitas <- c(1, 2, 2, 3.5, 1, 4, 1, 100)
```

- Eis a nova mediana e a nova média:

```r
summary(receitas)[c('Median', 'Mean')]
```

```
##  Median    Mean 
##  2,0000 14,3125
```

- O sujeito com a receita de 2 mil reais continua no meio da fila, mas a média (que é a soma de todas as receitas, dividida pelo número de indivíduos) ficou muito diferente.
- A receita do novo sujeito é um valor discrepante, ou, em inglês, um *outlier*.

Conclusão:

A mediana é robusta, pouco afetada por *outliers*.

A média é pouco robusta, muito sensível a *outliers*.

5.2.4 Intervalo interquartil (IQR) e *outliers*

- Qual fração dos elementos está entre o primeiro e o terceiro quartis?

```
length(
  horas[between(horas, quantile(horas, .25), quantile(horas, .75))]
) /
length(
  horas
)
```

```
## [1] 0,4939759
```

- Metade do total de elementos está entre o primeiro e o terceiro quartis.
- Este é o chamado intervalo interquartil (*interquartile range*, em inglês).
- No nosso vetor `horas`, os limites do IQR são

```
quantile(horas, c(.25, .75))
```

```
##   25%   75%
##  7,85 13,75
```

- O comprimento deste intervalo é calculado pela função `IQR`:

```
IQR(horas)
```

```
## [1] 5,9
```

- Valores muito abaixo do primeiro quartil podem ser considerados discrepantes (*outliers*), mas quão abaixo?
- A resposta (puramente convencional) é $1{,}5 \times$ IQR abaixo do primeiro quartil.
- No nosso vetor `horas`, isto significa valores abaixo de

```
limite_inferior <- quantile(horas, .25) - 1.5 * IQR(horas)

unname(limite_inferior)
```

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

- Neste caso, não há *outliers*:

```
horas[horas < limite_inferior]
```

```
## numeric(0)
```

- Da mesma forma, valores muito acima do terceiro quartil podem ser considerados discrepantes (*outliers*), mas quão acima?
- De novo, a resposta (puramente convencional) é $1{,}5 \times$ IQR acima do terceiro quartil.
- No nosso vetor `horas`, isto significa valores acima de

```
limite_superior <- quantile(horas, .75) + 1.5 * IQR(horas)

unname(limite_superior)
```

```
## [1] 22,6
```

- Neste caso, também não há *outliers*:

```
horas[horas > limite_superior]
```

```
## numeric(0)
```

- Outro exemplo: vamos tomar apenas os mamíferos onívoros:

```
onivoros <- sono %>%
  filter(vore == 'omni')

onivoros
```

```
## # A tibble: 20 x 4
##   name                       vore  order        sleep_total
##   <chr>                      <chr> <chr>              <dbl>
## 1 Owl monkey                 omni  Primates            17
## 2 Greater short-tailed shrew omni  Soricomorpha        14.9
## 3 Grivet                     omni  Primates            10
## 4 Star-nosed mole            omni  Soricomorpha        10.3
## 5 African giant pouched rat  omni  Rodentia             8.3
## 6 Lesser short-tailed shrew  omni  Soricomorpha         9.1
## # i 14 more rows
```

- Vamos extrair o vetor de horas de sono:

```
horas <- onivoros %>%
  pull(sleep_total)

horas
```

```
##  [1] 17,0 14,9 10,0 10,3  8,3  9,1 18,0 10,1 10,9  9,8  8,0 10,1  9,7
## [14]  9,4 11,0  8,7  9,6  9,1 15,6  8,9
```

- Vamos calcular o primeiro e terceiro quartis:

```
quartis <- horas %>%
  quantile(c(.25, .75))

quartis
```

```
##    25%    75%
##  9,100 10,925
```

- Vamos achar o IQR:

```
IQR(horas)
```

```
## [1] 1,825
```

- E os limites a partir dos quais os valores são *outliers*:

```
limites <- quartis + c(-1, 1) * 1.5 * IQR(horas)

unname(limites)
```

```
## [1]  6,3625 13,6625
```

- Existem *outliers* inferiores?

```
onivoros %>%
  filter(sleep_total < limites[1])
```

```
## # A tibble: 0 x 4
## # i 4 variables: name <chr>, vore <chr>, order <chr>,
## #   sleep_total <dbl>
```

Não.

- Existem *outliers* superiores?

```
onivoros %>%
  filter(sleep_total > limites[2])
```

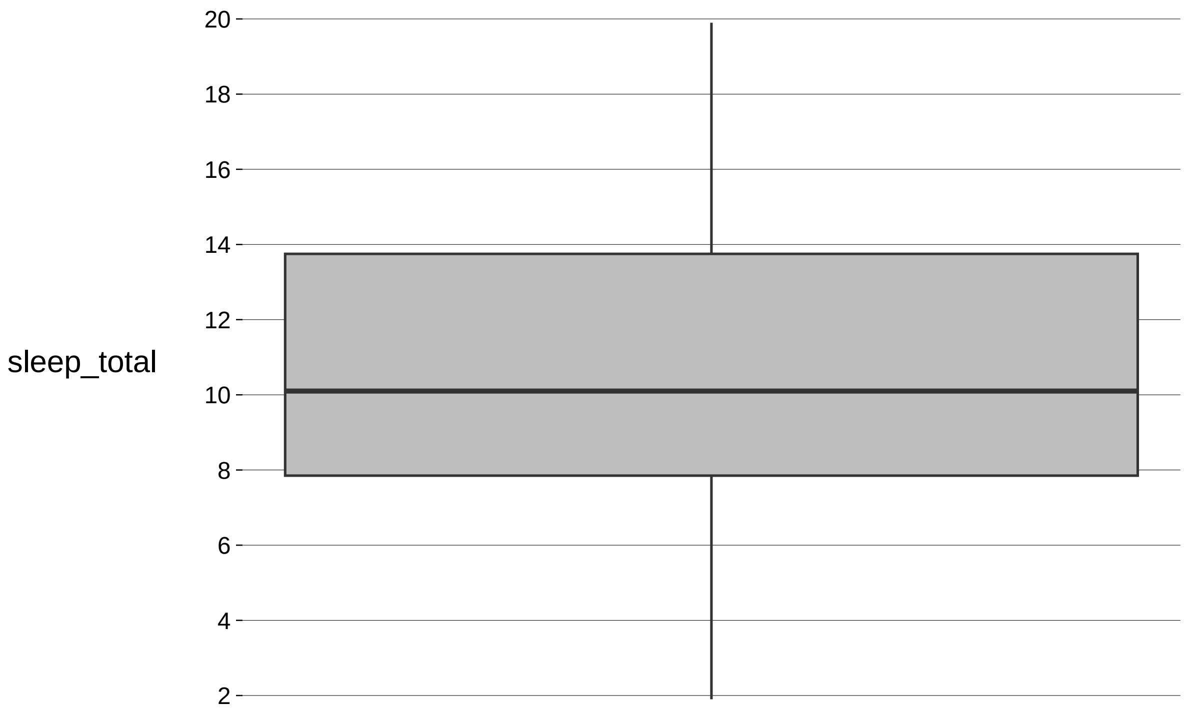
```
## # A tibble: 4 x 4
##   name                           vore  order           sleep_total
##   <chr>                          <chr> <chr>                 <dbl>
## 1 Owl monkey                     omni  Primates               17
## 2 Greater short-tailed shrew omni  Soricomorpha           14.9
## 3 North American Opossum         omni  Didelphimorphia        18
## 4 Tenrec                         omni  Afrosoricida           15.6
```

Sim! Estes animais dormem demais em comparação com os outros onívoros.

5.2.5 Gerando boxplots

- Um *boxplot* é uma representação visual dos valores que calculamos acima.
- No `ggplot2`, a geometria `geom_boxplot` constrói *boxplots*:

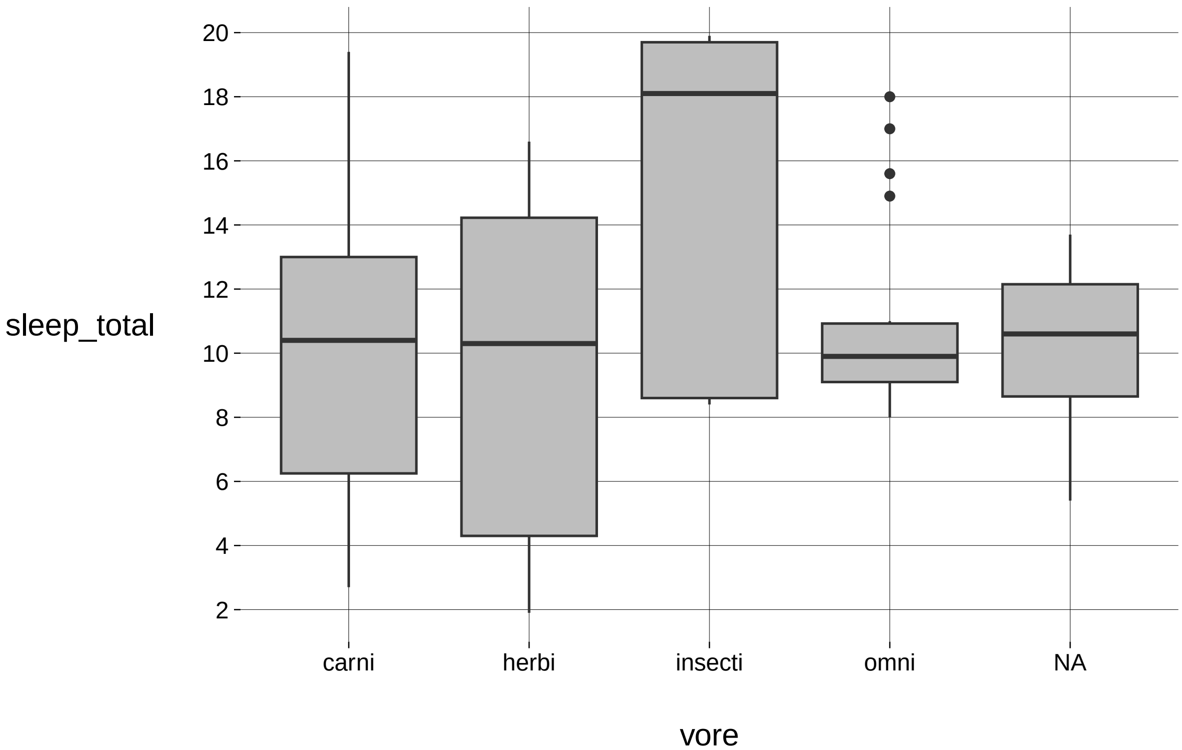
```
sono %>%
  ggplot(aes(y = sleep_total)) +
    geom_boxplot(fill = 'gray') +
    scale_x_continuous(breaks = NULL) +
    scale_y_continuous(breaks = seq(0, 20, 2))
```



- A caixa vai do valor do primeiro quartil (embaixo) até o terceiro quartil (em cima).
- A linha horizontal dentro da caixa representa o valor da mediana.
- As linhas verticais acima e abaixo da caixa (pitorescamente chamadas de "bigodes") vão até o limite inferior (primeiro quartil $-1{,}5 \times$ IQR) e até o limite superior (terceiro quartil $+1{,}5 \times$ IQR).
- Neste *boxplot*, não há *outliers*.
- Podemos usar a posição x para desenhar vários *boxplots*, um para cada dieta:

```
sono %>%
  ggplot(aes(x = vore, y = sleep_total)) +
```

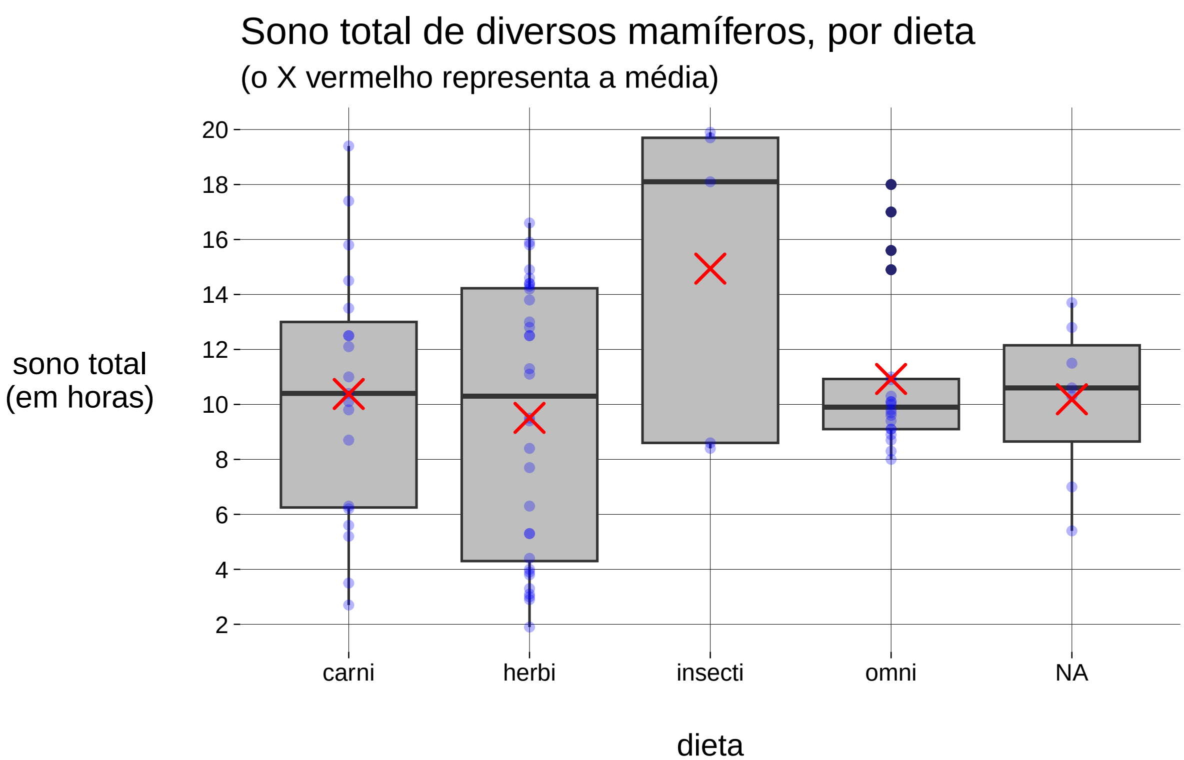
```
    geom_boxplot(fill = 'gray') +
    scale_y_continuous(breaks = seq(0, 20, 2))
```



- No *boxplot* de onívoros, os *outliers* aparecem como pontos isolados, acima da caixa, além dos alcances do bigode superior (aliás, onde está bigode superior?).
- *Boxplots* lado a lado são úteis para compararmos grupos diferentes de dados.
- Veja como, com exceção dos insetívoros, as medianas dos grupos são parecidas.
- Veja como carnívoros, insetívoros e herbívoros apresentam maior variação, enquanto onívoros e animais sem dieta registrada apresentam menor variação.
- Vamos combinar, em um só gráfico
  - Os pontos representando os animais,
  - Os *boxplots*,
  - As médias (que podem estar próximas ou distantes das medianas).

```
sono %>%
  ggplot(aes(x = vore, y = sleep_total)) +
    geom_boxplot(fill = 'gray') +
    scale_y_continuous(breaks = seq(0, 20, 2)) +
    geom_point(
      color = 'blue',
      alpha = .3
    ) +
```

```
    stat_summary(
      fun = mean,
      geom = 'point',
      color = 'red',
      shape = 'cross',
      size = 5,
      stroke = 1
    ) +
    labs(
      title = 'Sono total de diversos mamíferos, por dieta',
      subtitle = '(o X vermelho representa a média)',
      x = 'dieta',
      y = 'sono total\n(em horas)'
    )
```



- Quando a caixa é longa, o IQR é grande, e os valores estão muito espalhados; é o caso dos herbívoros e insetívoros.
- Quando a caixa é curta, o IQR é pequeno, e os valores estão pouco espalhados; é o caso dos onívoros. Como o IQR é pequeno, os 4 mamíferos com mais de 14 horas de sono são *outliers*.
- Observe, ainda, como os *outliers* "puxam" a média dos onívoros para cima.

5.3 Vídeo 2

https://youtu.be/QqnOvgBXJ-s

5.4 Gráficos de barras e de colunas

5.4.1 Conjunto de dados

- O R tem um *array* de 3 dimensões com dados sobre as cores dos cabelos e dos olhos de 592 alunos e alunas de uma universidade americana em 1974.
- Se pedirmos para o R exibir os dados, veremos duas matrizes, uma para cada sexo:

```
HairEyeColor
```

```
## , , Sex = Male
##
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    32   11    10     3
##   Brown    53   50    25    15
##   Red      10   10     7     7
##   Blond     3   30     5     8
##
## , , Sex = Female
##
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    36    9     5     2
##   Brown    66   34    29    14
##   Red      16    7     7     7
##   Blond     4   64     5     8
```

- Vamos transformar este *array* em um *data frame*.
- O *array* contém apenas os totais de cada classe. Vamos usar a função `uncount` para gerar uma linha para cada aluno:

```
df_orig <- as.data.frame(HairEyeColor) %>%
  uncount(Freq) %>%
  as_tibble()
```

```
df_orig
```

```
## # A tibble: 592 x 3
##   Hair  Eye   Sex
##   <fct> <fct> <fct>
## 1 Black Brown Male
## 2 Black Brown Male
## 3 Black Brown Male
## 4 Black Brown Male
## 5 Black Brown Male
## 6 Black Brown Male
## # i 586 more rows
```

- O `ggplot2` e os outros pacotes do `tidyverse` foram projetados para trabalhar com *data frames* neste formato, com uma observação (um indivíduo, um elemento) por linha. É o chamado formato *tidy*.
- Usando vetores com elementos nomeados, podemos traduzir o conteúdo do *data frame* para português:

```
cabelo <- c(
  'Brown' = 'castanhos',
  'Blond' = 'louros',
  'Black' = 'pretos',
  'Red' = 'ruivos'
)

olhos <- c(
  'Brown' = 'castanhos',
  'Blue' = 'azuis',
  'Hazel' = 'avelã',
  'Green' = 'verdes'
)

sexo <- c(
  'Male' = 'homem',
  'Female' = 'mulher'
)

df <- df_orig %>%
  transmute(
    cabelos = cabelo[Hair],
    olhos = olhos[Eye],
    sexo = sexo[Sex]
  )
```

- Um sumário:

```
df %>% dfSummary() %>% print()
```

Variável	Estatísticas / Valores	Freqs (% de Válidos)	Faltante
cabelos [character]	1. castanhos 2. louros 3. pretos 4. ruivos	108 (18,2%) 286 (48,3%) 71 (12,0%) 127 (21,5%)	0 (0,0%)
olhos [character]	1. avelã 2. azuis 3. castanhos 4. verdes	93 (15,7%) 215 (36,3%) 220 (37,2%) 64 (10,8%)	0 (0,0%)
sexo [character]	1. homem 2. mulher	279 (47,1%) 313 (52,9%)	0 (0,0%)

5.4.2 Gerando gráficos de barras

- Um gráfico de barras contém uma barra para cada valor de uma variável categórica.
- Usamos `geom_bar` para gerar um gráfico de barras de cores de cabelo:

```
df %>%
  ggplot(aes(x = cabelos)) +
    geom_bar() +
    labs(y = NULL)
```

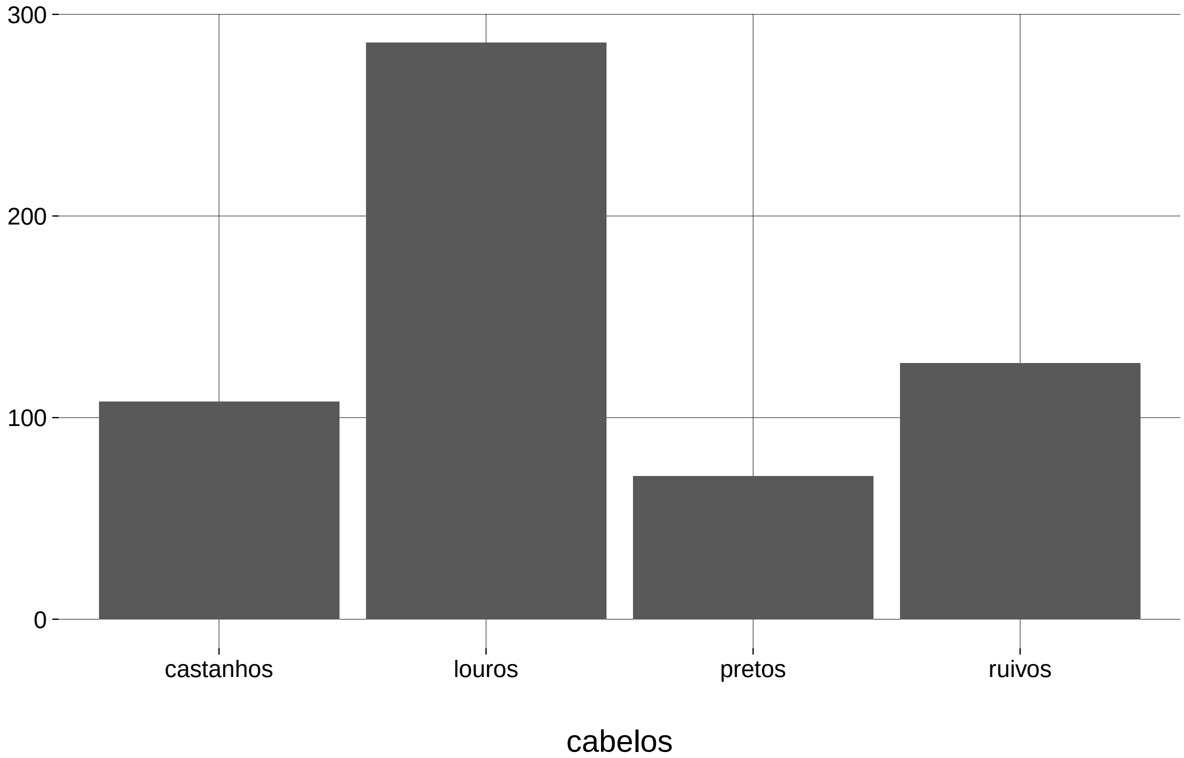


Gráfico de barras × histograma:

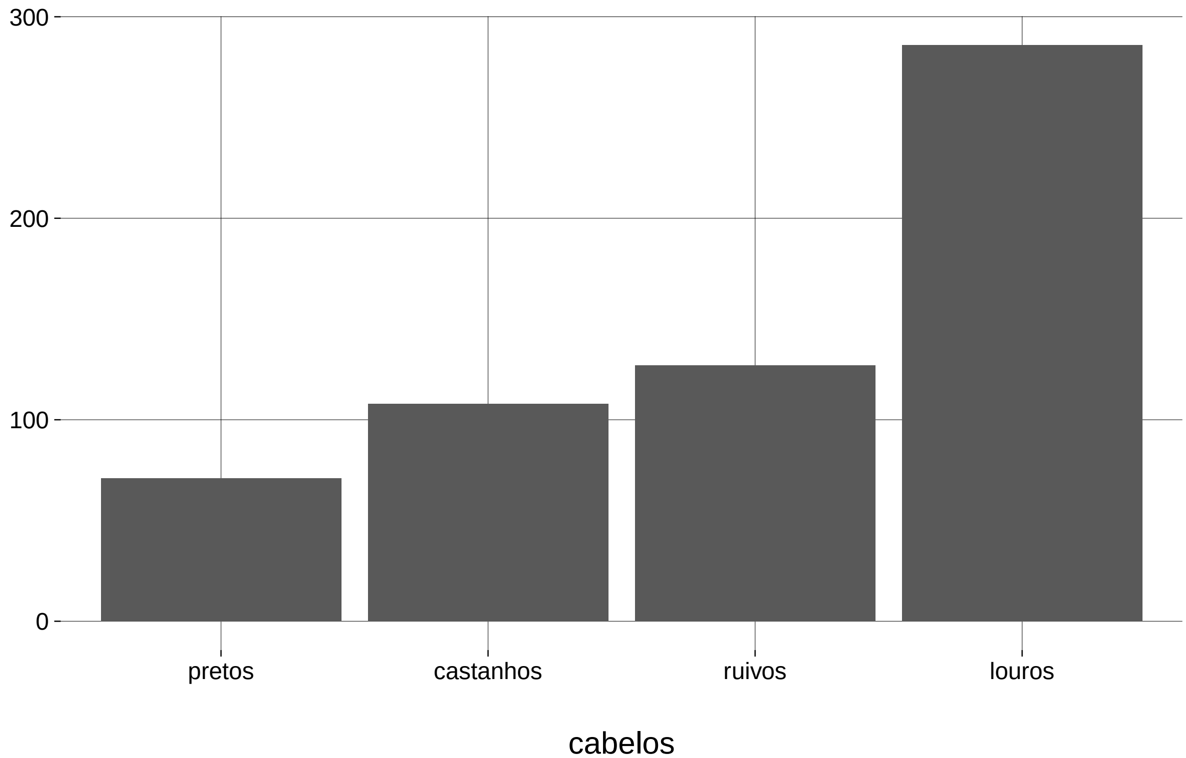
- Os dois tipos de gráficos mostram a frequência (quantidade de elementos) no eixo vertical.
- No gráfico de barras:
  * A variável é categórica (nominal).
  * Cada barra corresponde a um valor da variável.
  * As barras não se tocam, enfatizando o fato de que a variável é categórica.
- No histograma (veja o exemplo):
  * A variável é quantitativa (intervalar ou racional).
  * Cada barra corresponde a uma classe de valores da variável.
  * As barras se tocam, para enfatizar que as classes são contíguas.

- Um gráfico de barras é mais legível quando as barras são mostradas em ordem crescente ou decrescente.
- Embora os valores da variável `cabelos` sejam *strings*, podemos aplicar a eles funções que manipulam fatores.
- A função `fct_infreq`, do pacote `forcats`, ordena os valores em ordem decrescente

de frequência.

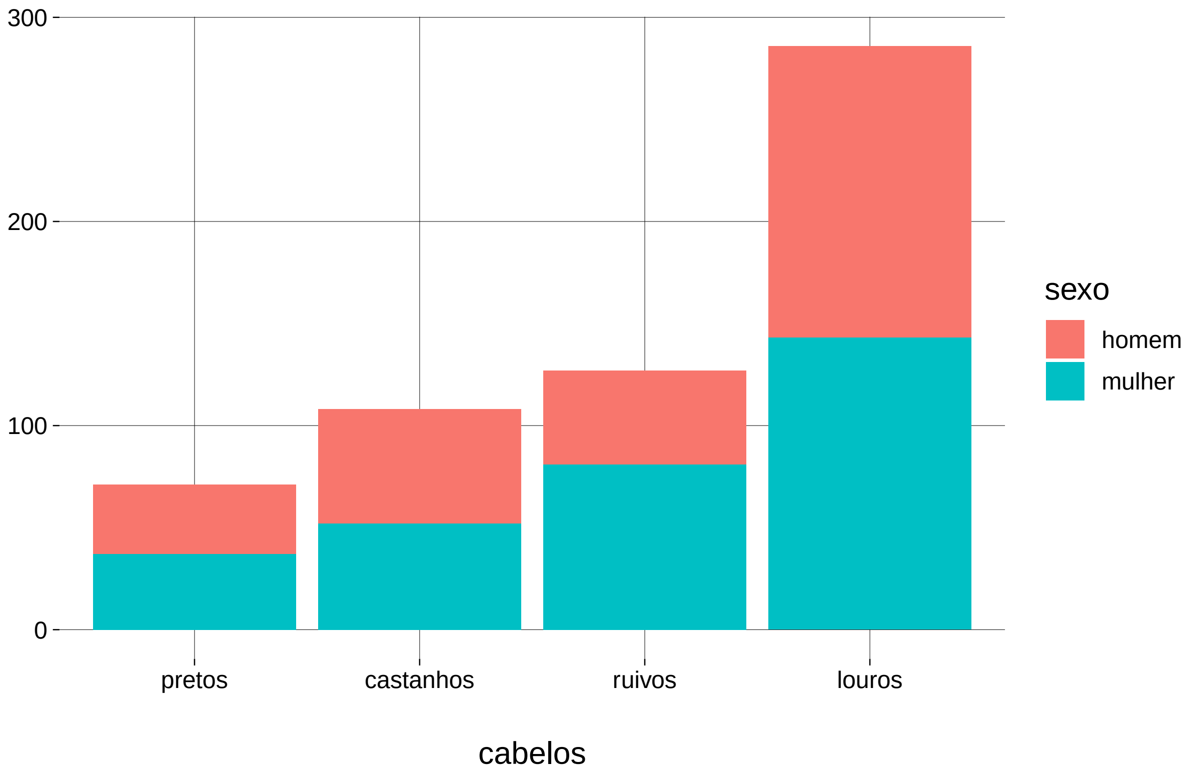
- A função `fct_rev`, também do pacote `forcats`, inverte a ordenação.

```
df %>%
  ggplot(aes(x = fct_rev(fct_infreq(cabelos)))) +
    geom_bar() +
    labs(
      x = 'cabelos',
      y = NULL
    )
```



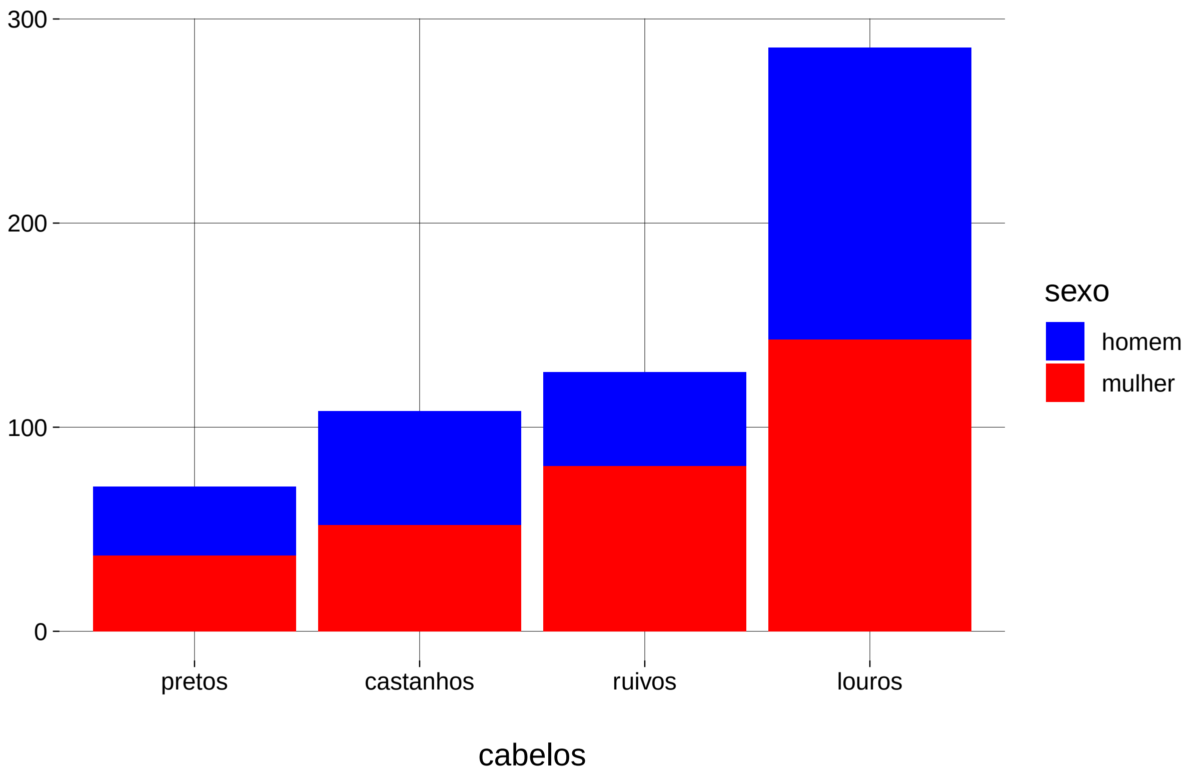
- A posição x e a altura de cada barra são estéticas: a posição x representa a cor dos cabelos, e a altura representa a frequência daquela cor.
- Vamos acrescentar mais uma estética: a cor de preenchimento vai representar o sexo.

```
df %>%
  ggplot(aes(x = fct_rev(fct_infreq(cabelos)), fill = sexo)) +
    geom_bar() +
    labs(
      x = 'cabelos',
      y = NULL
    )
```



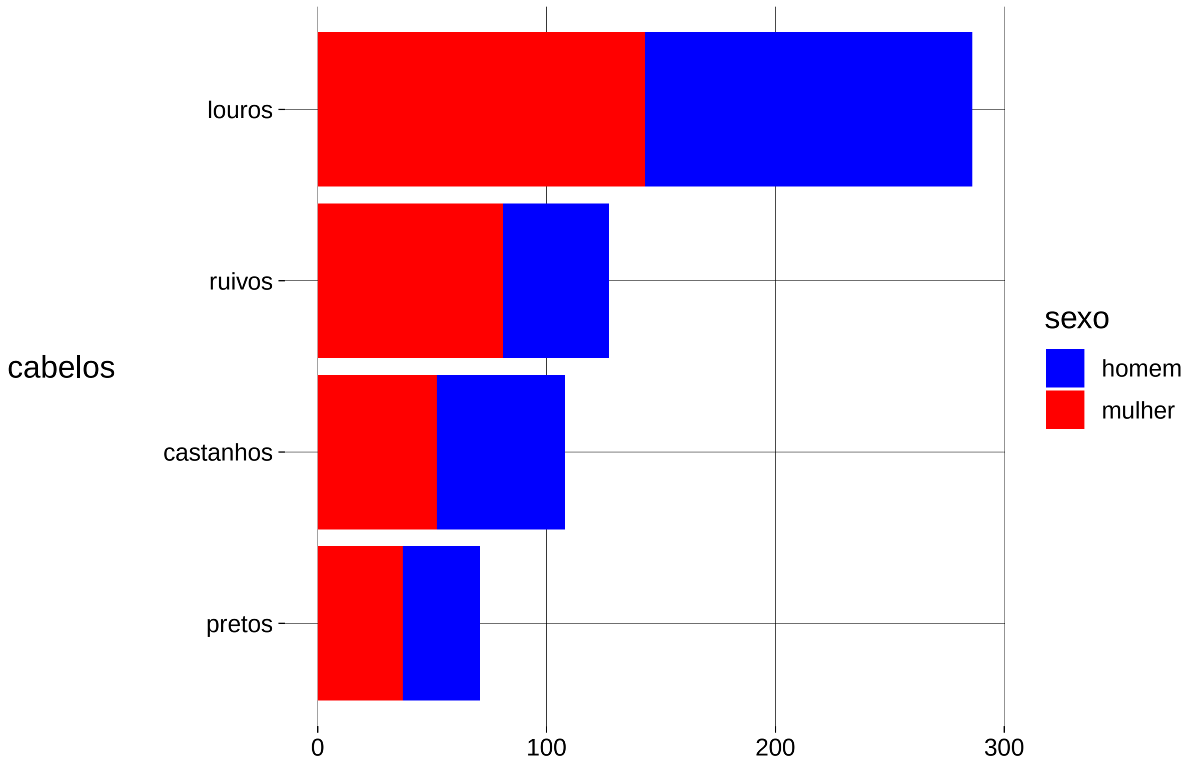
- Se a cor dos homens incomoda você, altere a escala que especifica o preenchimento (`scale_fill_discrete`):

```
df %>%
  ggplot(aes(x = fct_rev(fct_infreq(cabelos)), fill = sexo)) +
    geom_bar() +
    scale_fill_discrete(type = c('blue', 'red')) +
    labs(
      x = 'cabelos',
      y = NULL
    )
```



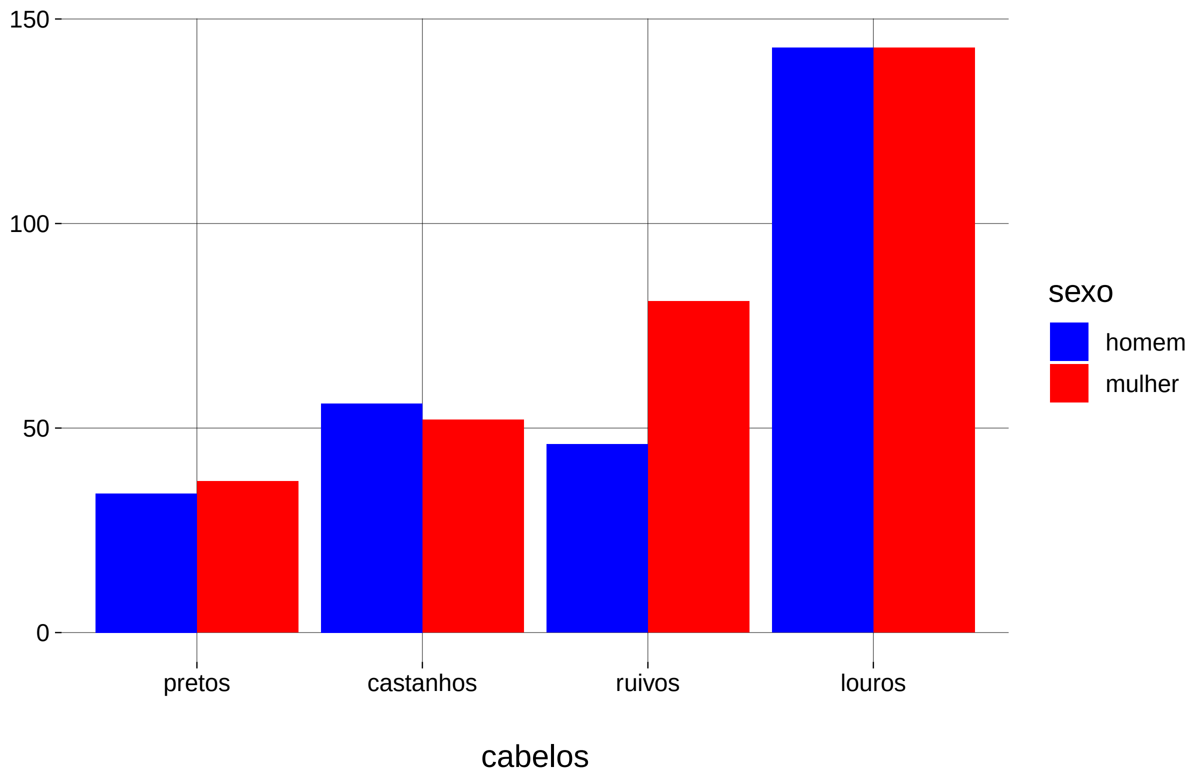
- Podemos fazer um gráfico de barras horizontais com `coord_flip`. Isto geralmente é útil quando os rótulos das barras são longos:

```
df %>%
  ggplot(aes(x = fct_rev(fct_infreq(cabelos)), fill = sexo)) +
    geom_bar() +
    scale_fill_discrete(type = c('blue', 'red')) +
    labs(
      x = 'cabelos',
      y = NULL
    ) +
    coord_flip()
```



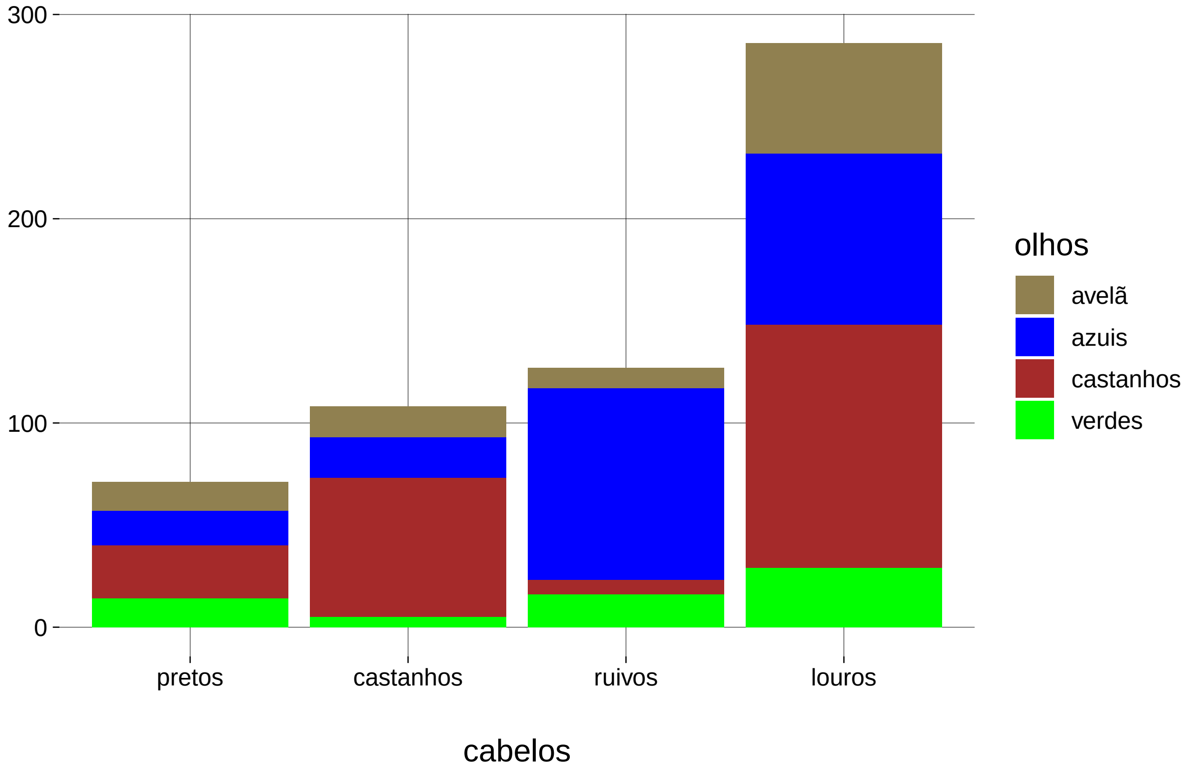
- Você consegue dizer se há mais homens ou mulheres com cabelos pretos? E castanhos? E ruivos?
- Se posicionarmos as barras lado a lado, fica mais fácil responder.
- Usamos o argumento `position = 'dodge'` de `geom_bar`. "*Dodge*" significa "esquivar-se", em inglês.

```
df %>%
  ggplot(aes(x = fct_rev(fct_infreq(cabelos)), fill = sexo)) +
    geom_bar(position = 'dodge') +
    labs(
      x = 'cabelos',
      y = NULL
    ) +
    scale_fill_discrete(type = c('blue', 'red'))
```



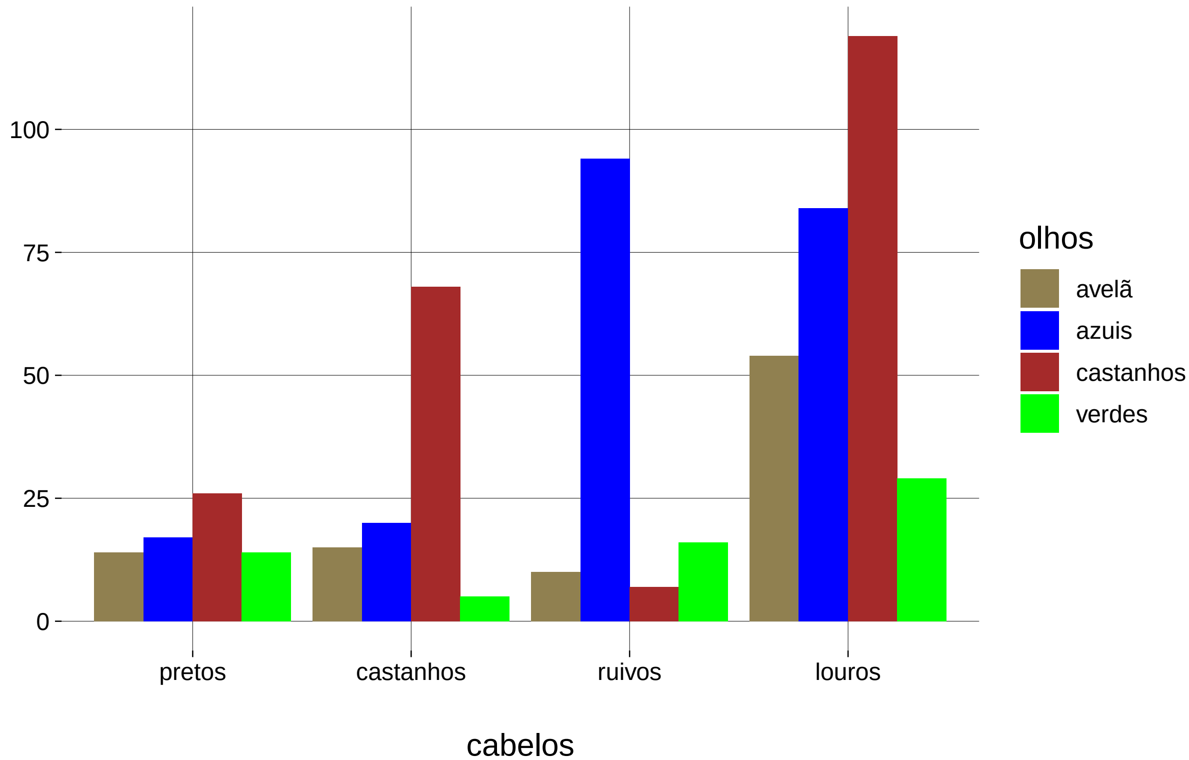
- Agora vamos examinar a relação entre as cores dos olhos e as cores dos cabelos:

```
df %>%
  ggplot(aes(x = fct_rev(fct_infreq(cabelos)), fill = olhos)) +
    geom_bar() +
    scale_fill_discrete(
      type = c('#908050', 'blue', 'brown', 'green')
    ) +
    labs(
      x = 'cabelos',
      y = NULL
    )
```



- Ou, com barras lado a lado:

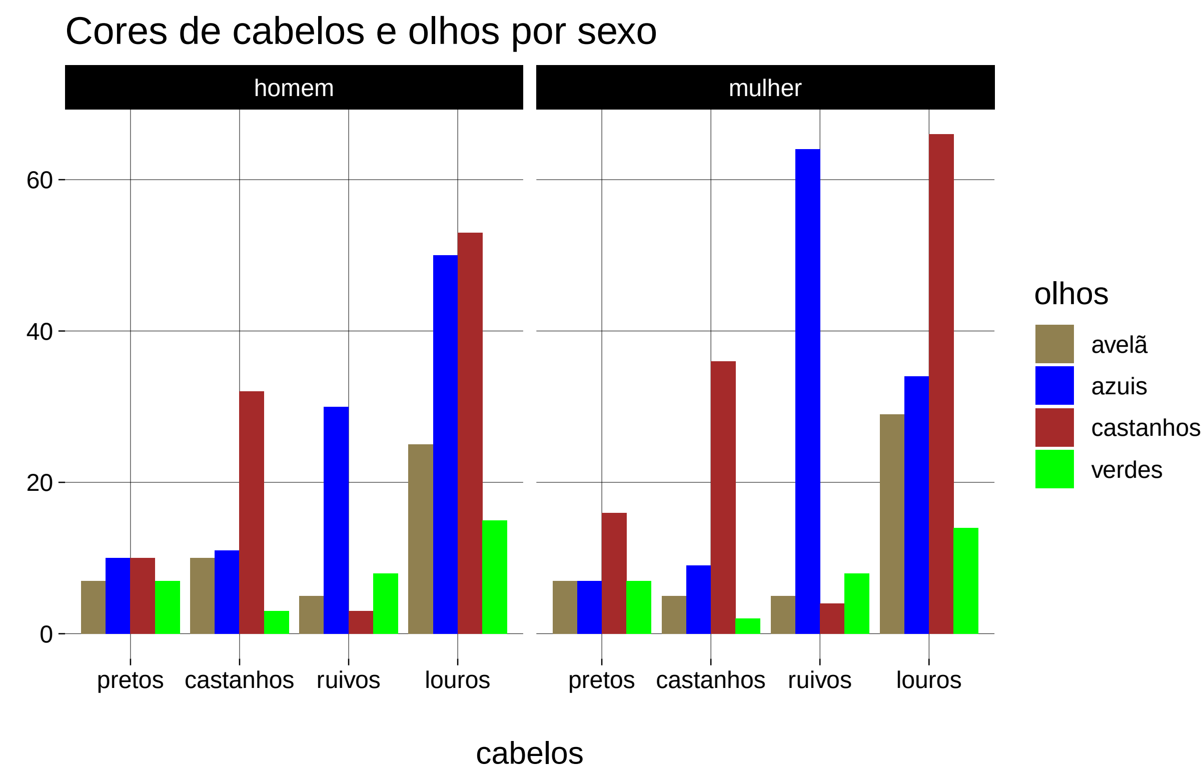
```
df %>%
  ggplot(aes(x = fct_rev(fct_infreq(cabelos)), fill = olhos)) +
    geom_bar(position = 'dodge') +
    scale_fill_discrete(
      type = c('#908050', 'blue', 'brown', 'green')
    ) +
    labs(
      x = 'cabelos',
      y = NULL
    )
```



- Observações e perguntas:
  1. Há mais pessoas louras de olhos castanhos do que louras de olhos azuis. O esperado não seria mais pessoas louras de olhos azuis? Pessoas louras de olhos castanhos pintaram os cabelos?
  2. Há muito mais ruivos de olhos azuis do que ruivos de olhos verdes. Não deveria ser o contrário? Também são pessoas que pintaram os cabelos de ruivo? Ou houve erro no registro das cores dos olhos?
- Para incluir o sexo, podemos facetar o gráfico. Usando `facet_wrap`[1], geramos dois subgráficos lado a lado:

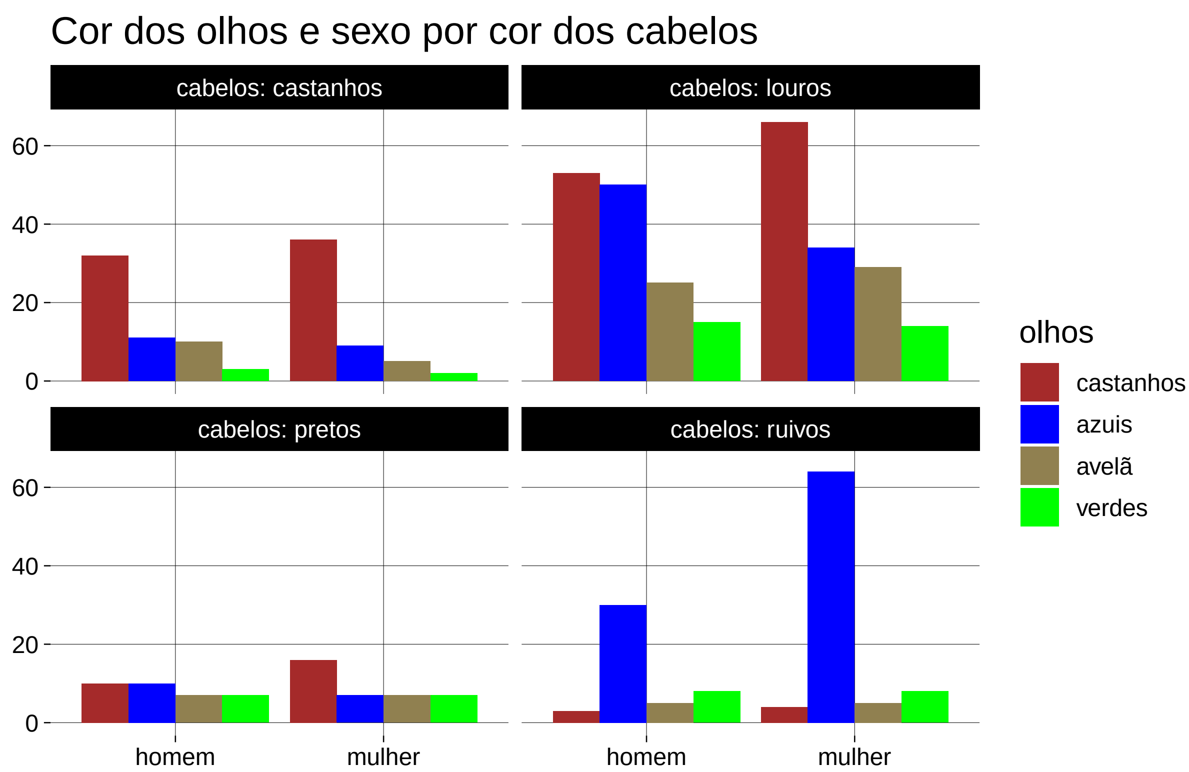
```
df %>%
  ggplot(aes(x = fct_rev(fct_infreq(cabelos)), fill = olhos)) +
    geom_bar(position = 'dodge') +
    scale_fill_discrete(type = c('#908050', 'blue', 'brown', 'green')) +
    facet_wrap(~sexo) +
    labs(
      title = 'Cores de cabelos e olhos por sexo',
      y = NULL,
      x = 'cabelos'
    )
```

[1] O nome da variável segundo a qual facetar deve aparecer depois de um ~.



- Se a quantidade grande de pessoas louras de olhos castanhos (em comparação com pessoas louras de olhos azuis) for por causa da pintura de cabelos, então o gráfico acima mostra que as mulheres pintam os cabelos de louro com mais frequência do que os homens.
- Quando facetamos por cor de cabelos, também podemos observar as mesmas diferenças entre homens e mulheres:

```
df %>%
  ggplot(aes(x = sexo, fill = fct_infreq(olhos))) +
    geom_bar(position = 'dodge') +
    facet_wrap(~cabelos, labeller = label_both) +
    scale_fill_discrete(type = c('brown', 'blue', '#908050', 'green')) +
    labs(
      x = NULL,
      y = NULL,
      fill = 'olhos',
      title = 'Cor dos olhos e sexo por cor dos cabelos'
    )
```



5.4.3 *Data frame* já contendo os totais

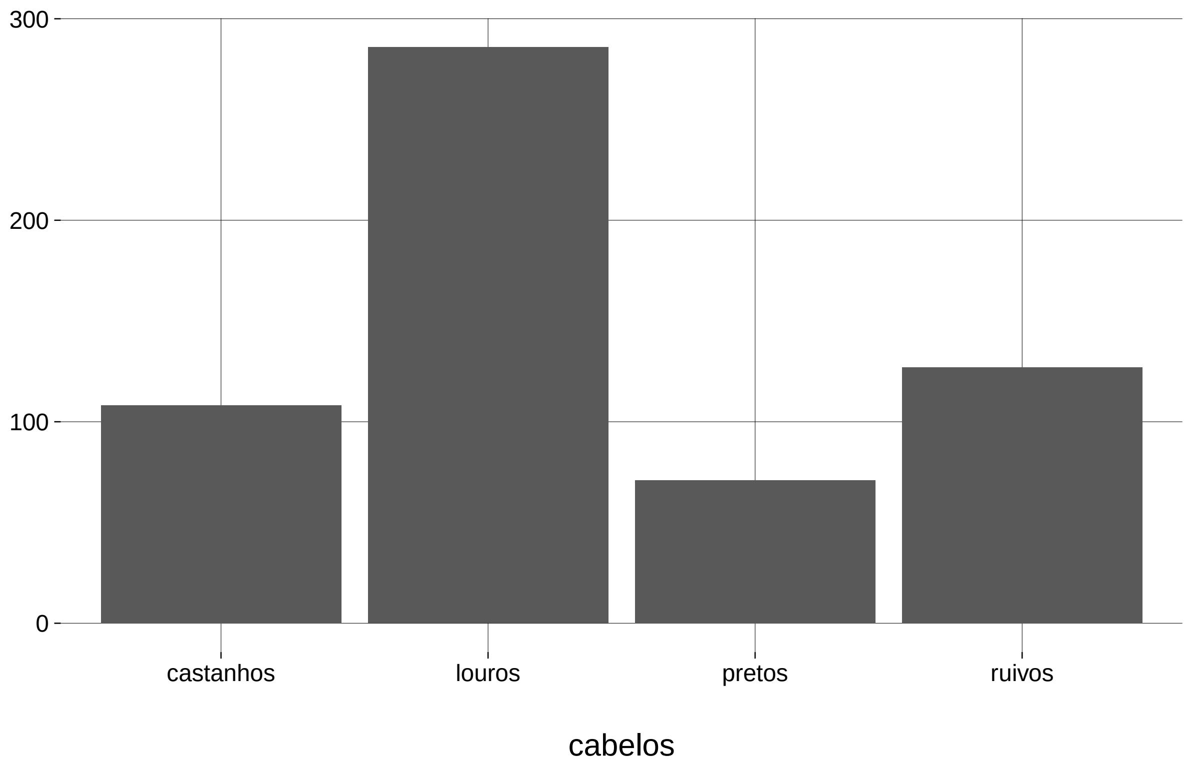
- Você percebeu que `geom_bar` analisa o *data frame* e calcula as frequências necessárias para construir o gráfico.
- Em algumas situações, o *data frame* já contém as frequências (em vez de conter uma linha por indivíduo).
- Vamos usar `count` para criar um *data frame* assim:

```
df_tot <- df %>%
  count(sexo, cabelos, olhos)

df_tot
```

```
## # A tibble: 32 x 4
##   sexo  cabelos   olhos         n
##   <chr> <chr>     <chr>     <int>
## 1 homem castanhos avelã        10
## 2 homem castanhos azuis        11
## 3 homem castanhos castanhos    32
## 4 homem castanhos verdes        3
## 5 homem louros    avelã        25
## 6 homem louros    azuis        50
## # i 26 more rows
```

- Para 4 cores de cabelo, 4 cores de olhos, e 2 sexos, são 32 combinações possíveis.
- Com este *data frame*, podemos gerar todos os gráficos anteriores usando `geom_col` no lugar de `geom_bar`. Por exemplo:

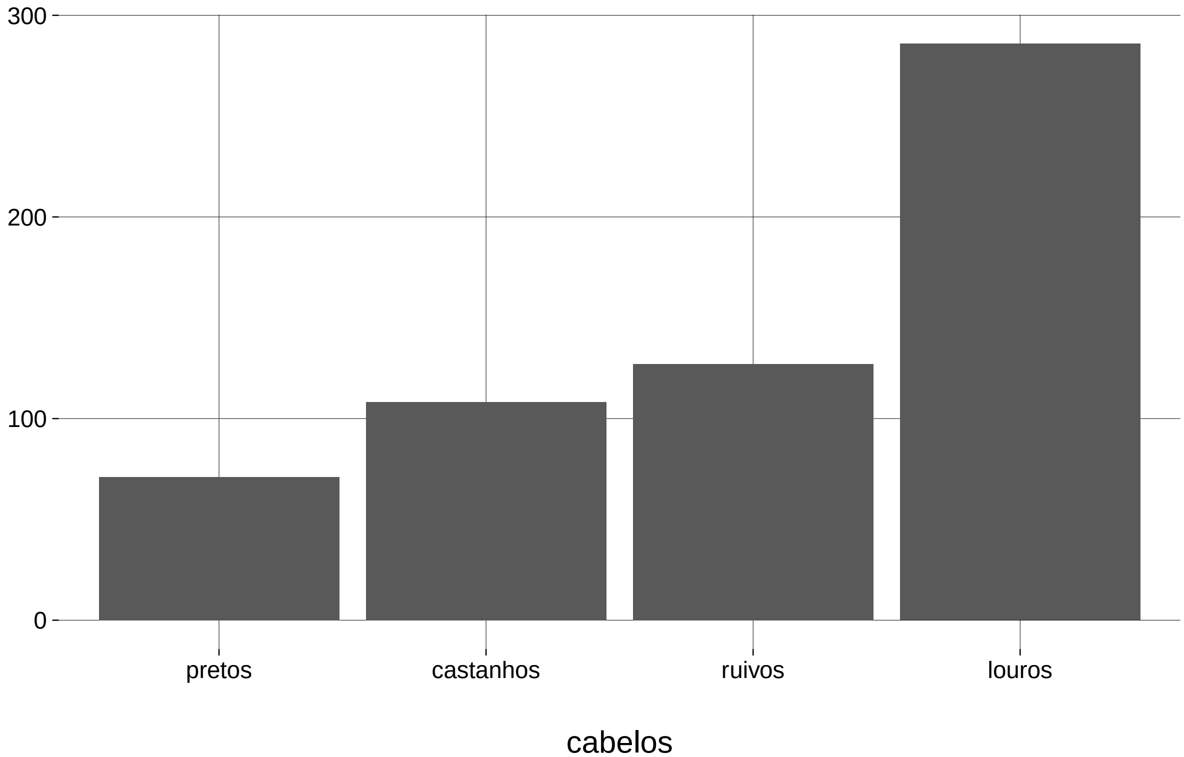
```
df_tot %>%
  ggplot(aes(x = cabelos, y = n)) +
    geom_col() +
    labs(
      y = NULL
    )
```



- Com `geom_col`, precisamos passar a estética y (no nosso exemplo, a variável `n`, que contém as frequências).
- Para ordenar as barras, usamos a função `fct_reorder`, que ordena os níveis de um fator (`cabelos`) de acordo com o resultado de uma função (`sum`) aplicada sobre os valores de outra variável (`n`):

```
df_tot %>%
  ggplot(aes(x = fct_reorder(cabelos, n, sum), y = n)) +
    geom_col() +
    labs(
      x = 'cabelos',
      y = NULL
```

```
    )
```



5.5 Gráficos de linha e séries temporais

5.5.1 Conjunto de dados

- O R tem uma matriz com as quantidades de telefones em várias regiões do mundo ao longo de vários anos:

```
WorldPhones
```

```
##      N.Amer Europe Asia S.Amer Oceania Africa Mid.Amer
## 1951  45939  21574 2876   1815    1646     89      555
## 1956  60423  29990 4708   2568    2366   1411      733
## 1957  64721  32510 5230   2695    2526   1546      773
## 1958  68484  35218 6662   2845    2691   1663      836
## 1959  71799  37598 6856   3000    2868   1769      911
## 1960  76036  40341 8220   3145    3054   1905     1008
## 1961  79831  43173 9053   3338    3224   2005     1076
```

- Os números representam milhares.
- Os números dos anos são os nomes das linhas da matriz.

- Vamos transformar esta matriz em uma *tibble*:

```
fones <- WorldPhones %>%
  as_tibble(rownames = 'Ano') %>%
  mutate(Ano = as.numeric(Ano))

fones
```

```
## # A tibble: 7 x 8
##     Ano N.Amer Europe  Asia S.Amer Oceania Africa Mid.Amer
##   <dbl>  <dbl>  <dbl> <dbl>  <dbl>   <dbl>  <dbl>    <dbl>
## 1  1951  45939  21574  2876   1815    1646     89      555
## 2  1956  60423  29990  4708   2568    2366   1411      733
## 3  1957  64721  32510  5230   2695    2526   1546      773
## 4  1958  68484  35218  6662   2845    2691   1663      836
## 5  1959  71799  37598  6856   3000    2868   1769      911
## 6  1960  76036  40341  8220   3145    3054   1905     1008
## # i 1 more row
```

- Esta *tibble* não está no formato *tidy*. Queremos que cada linha corresponda a uma observação, contendo
  - Ano,
  - Região,
  - Quantidade de telefones.
- Usamos a função `pivot_longer` para mudar o formato da *tibble*:

```
fones_long <- fones %>%
  pivot_longer(
    cols = -Ano,
    names_to = 'Região',
    values_to = 'n'
  )

fones_long
```

```
## # A tibble: 49 x 3
##     Ano Região      n
##   <dbl> <chr>   <dbl>
## 1  1951 N.Amer  45939
## 2  1951 Europe  21574
## 3  1951 Asia     2876
## 4  1951 S.Amer   1815
## 5  1951 Oceania  1646
## 6  1951 Africa     89
## # i 43 more rows
```

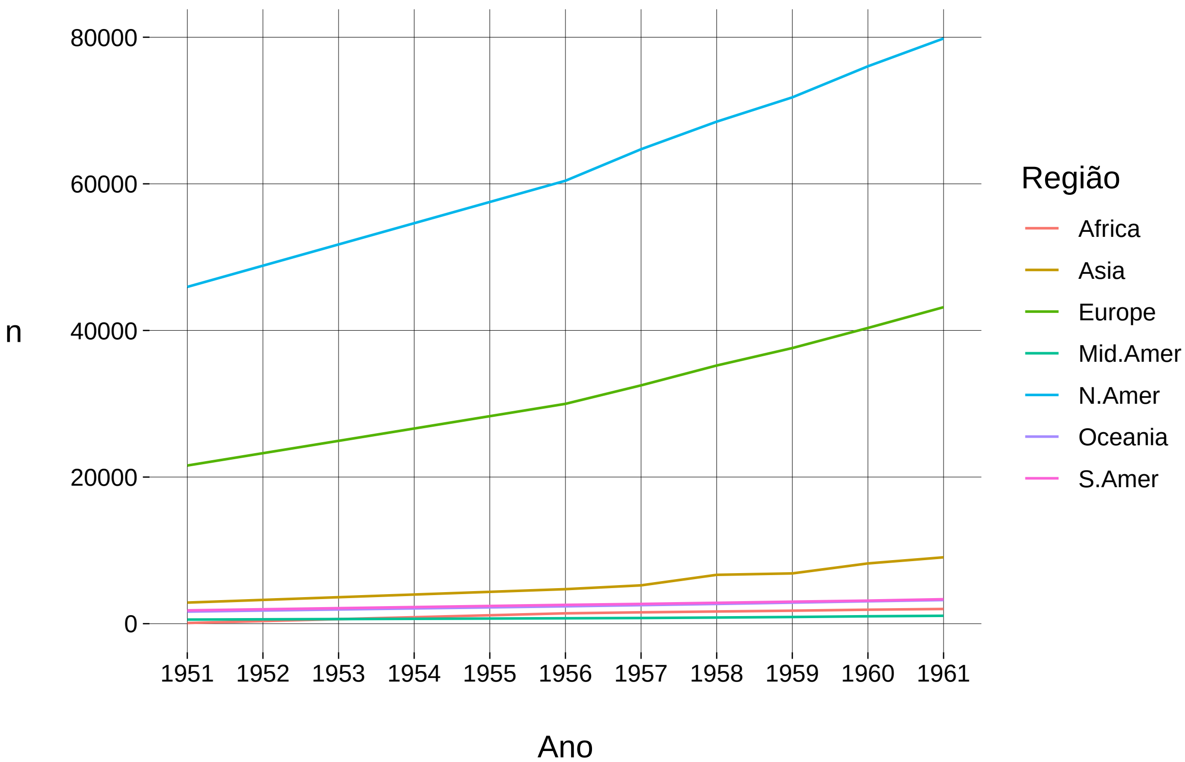
- Confira: antes, tínhamos 7 anos, com 7 quantidades por ano, uma quantidade por região. Eram 49 quantidades. Agora temos uma *tibble* de 49 linhas.

5.5.2

Gerando gráficos de linha

- A geometria `geom_line` gera gráficos de linha. Perceba como geramos uma linha por região:

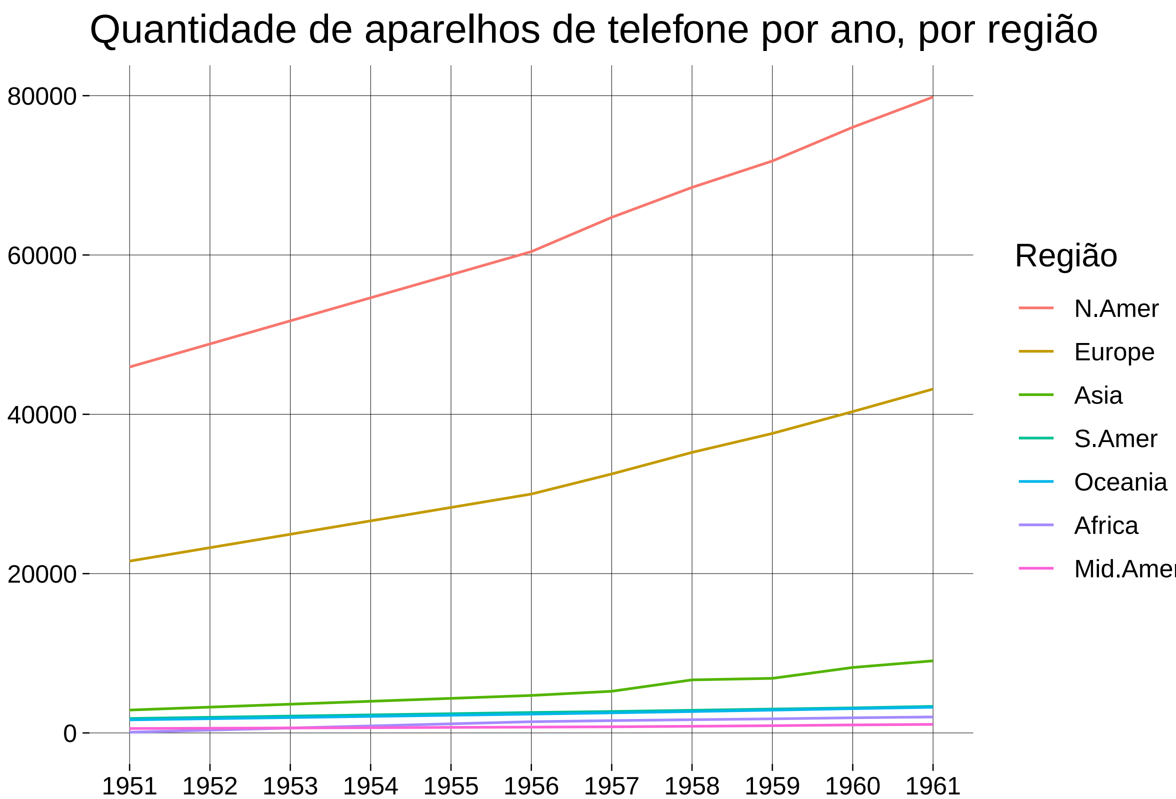
```
fones_long %>%
  ggplot(aes(x = Ano, y = n, color = Região)) +
    geom_line() +
    scale_x_continuous(breaks = 1951:1961)
```



- Embora a legenda associe uma cor a cada região, a leitura seria mais fácil se a ordem das regiões na legenda coincidisse com a posição das linhas na borda direita da grade:

```
fones_long %>%
  ggplot(
      aes(
        x = Ano,
        y = n,
```

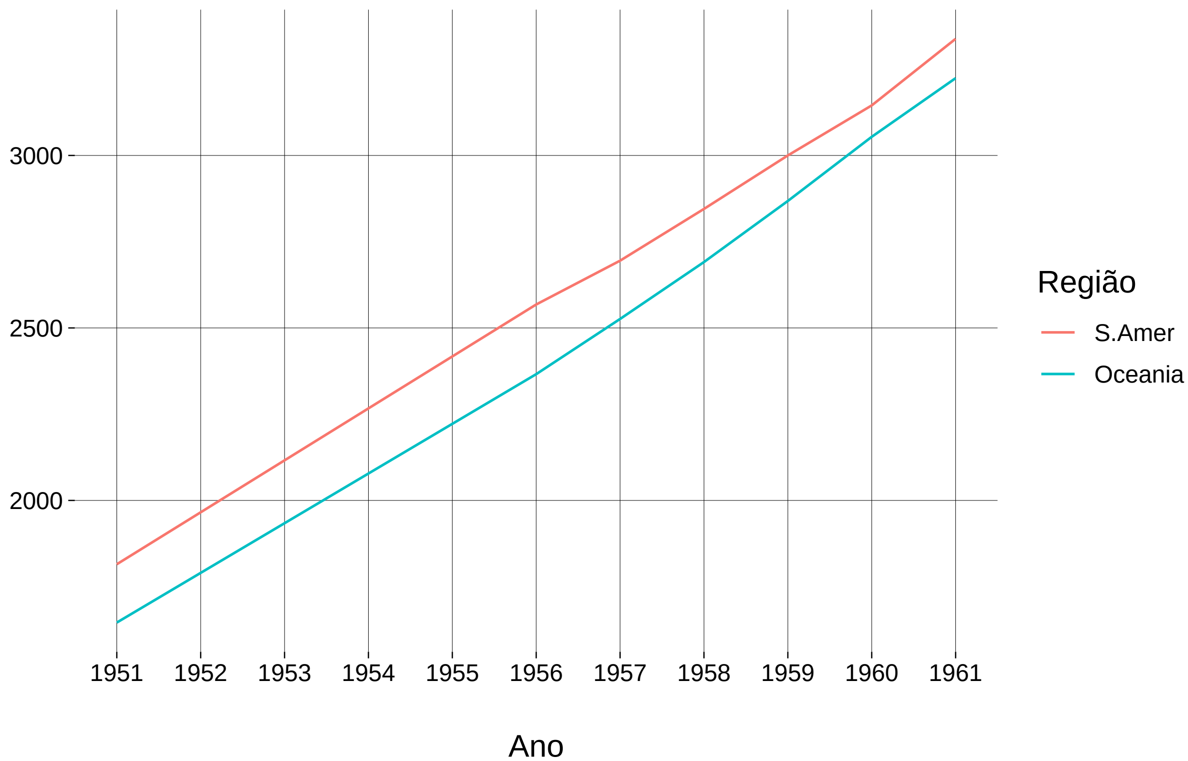
```
      color = fct_rev(fct_reorder(Região, n, max))
    )
  ) +
    geom_line() +
    scale_x_continuous(breaks = 1951:1961) +
    labs(
      color = 'Região',
      y = '',
      x = NULL,
      title = 'Quantidade de aparelhos de telefone por ano, por região'
    )
```



- Parece que está faltando uma linha, mas o que acontece é que as quantidades da América do Sul e da Oceania são bem parecidas:

```
fones_long %>%
  filter(Região %in% c('S.Amer', 'Oceania')) %>%
  ggplot(
    aes(
      x = Ano,
      y = n,
      color = fct_rev(fct_reorder(Região, n, max))
    )
  ) +
```

```
    geom_line() +
    scale_x_continuous(breaks = 1951:1961) +
    labs(y = NULL, color = 'Região')
```



- Estamos tratando estes dados como simples números, mas, na verdade, este conjunto de dados é uma série temporal (*time series*).
- R tem todo um conjunto de funções para tratar séries temporais, calcular tendências, achar padrões cíclicos, fazer estimativas, e gerar gráficos específicos, entre outras coisas.
- Mas não vamos falar mais sobre séries temporais aqui.
- O pacote `tsibble` oferece maneiras de trabalhar com séries temporais de maneira *tidy*. Você pode ler a documentação do pacote entrando

```
library(tsibble)
?`tsibble-package`
```

5.6 Exercícios

5.6.1 O bigode dos onívoros

- Examine o *data frame* `sono` para descobrir o que houve com o bigode superior do *boxplot* dos onívoros neste gráfico.

5.6.2 Usando `geom_col`

- Use `geom_col` para reproduzir, a partir do *data frame* `df_tot`, todos os gráficos que foram gerados com `geom_bar` na seção Gerando gráficos de barras.

5.7 Referências sobre visualização e R

> Busque mais informações sobre os pacotes `tidyverse` e `ggplot2` nas referências recomendadas.

CAPÍTULO 6

Medidas

6.1 Vídeo

https://youtu.be/C96MOP4YlaY

6.2 Medidas de centralidade

6.2.1 Média

- A média de uma população é escrita como μ, e é definida como

$$\mu = \frac{\sum_{i=1}^{N} x_i}{N}$$

  - $\sum_{i=1}^{N} x_i$ é a soma de todos os dados da população.
  - N é a quantidade de elementos na população.

- A média de uma amostra é escrita como $\bar{x}$, e é definida como:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

  - $\sum_{i=1}^{n} x_i$ é a soma de todos os dados da amostra.
  - n é a quantidade de elementos na amostra.
- O cálculo é essencialmente o mesmo. Só mudam os símbolos: N versus n, e μ versus $\bar{x}$.

6.2.1.1

Exemplo

- Idades dos alunos de uma turma:

```
idades <- c(
  20, 20, 20, 20, 20, 20, 21, 21, 21, 21,
  22, 22, 22, 23, 23, 23, 23, 24, 24,
  65
)
```

- Média com o velhinho de 65 anos:

```
mean(idades)
```

```
## [1] 23,75
```

- Média sem o velhinho:

```
mean(idades[-length(idades)])
```

```
## [1] 21,57895
```

6.2.2

Mediana

- Já aprendemos sobre a mediana na seção sobre *boxplots*.
- A idéia é que, depois de ordenar os dados, 50% dos dados estarão à esquerda da mediana, e 50% à direita.
- A mediana não é tão sensível a *outliers* quanto à média.

6.2.2.1 Exemplo

- Mediana com o velhinho:

```
median(idades)
```

```
## [1] 21,5
```

- Mediana sem o velhinho:

```
median(idades[-length(idades)])
```

```
## [1] 21
```

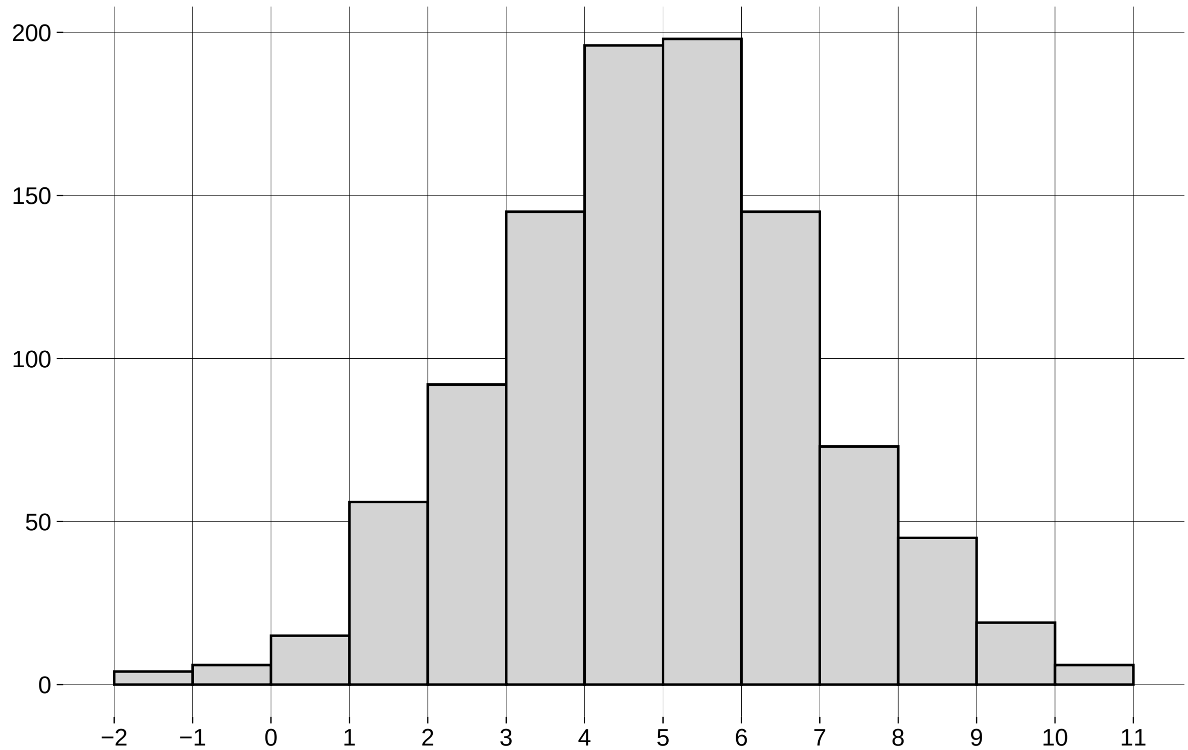
6.2.3 Moda

- A moda é o valor mais frequente do conjunto de dados.
- Pode haver mais de uma moda.
- Não existe uma função para a moda em R base. Por quê?
- Por incrível que pareça, é complicado definir a moda de forma a conseguir resultados interessantes.
- Por exemplo, vamos definir um conjunto de 1000 valores numéricos distribuídos normalmente[1], com média igual a 5 e desvio-padrão[2] igual a 2:

```
normal <- rnorm(1000, mean = 5, sd = 2)
```

- O histograma dos nossos dados é

[1]Mais sobre a distribuição normal no capítulo ???.
[2]Mais sobre o desvio-padrão daqui a pouco.



- Vamos calcular a moda com a função `mfv` (*most frequent value*), do pacote `modeest`:

```
# Pacote com funções para calcular modas
library(modeest)
```

```
## Registered S3 method overwritten by 'rmutil':
##   method         from
##   print.response httr
```

```
# Por causa de um bug na função mfv,
# precisamos de números com ponto decimal
# (em vez de vírgula):
options(OutDec = '.')
mfv(normal)
```

```
##     [1] -1.9265930 -1.4986917 -1.4718853 -1.2578215 -0.4664627
##     [6] -0.3880051 -0.3538267 -0.3484132 -0.2694697 -0.1108932
##    [11]  0.2139091  0.3230286  0.3829877  0.4891109  0.5069779
##    [16]  0.5273603  0.5361215  0.6625981  0.7083582  0.7394972
##    [21]  0.7421879  0.8112960  0.8817900  0.9560357  0.9766202
##    [26]  1.0346385  1.0620985  1.1632375  1.1700127  1.2083555
##    [31]  1.2207841  1.2459278  1.2754860  1.2830844  1.3275824
##    [36]  1.3366422  1.3837080  1.4089045  1.4175873  1.4348363
##    [41]  1.5425942  1.5652945  1.5698081  1.6126346  1.6195220
##    [46]  1.6549103  1.6597038  1.6720581  1.6849994  1.6913447
##    [51]  1.6958754  1.7010331  1.7070768  1.7218485  1.7344382
```

```
##  [56] 1.7345651 1.7352704 1.7392026 1.7420443 1.7463935
##  [61] 1.7561119 1.7971487 1.8266784 1.8480056 1.8842732
##  [66] 1.8943385 1.8998501 1.9125756 1.9197125 1.9238422
##  [71] 1.9262631 1.9277299 1.9286435 1.9414269 1.9570007
##  [76] 1.9661235 1.9666985 1.9741026 1.9860700 1.9888090
##  [81] 1.9976685 2.0043179 2.0167819 2.0177090 2.0728997
##  [86] 2.0940168 2.0947897 2.0984780 2.0995363 2.1099430
##  [91] 2.1257317 2.1285370 2.1371903 2.1393343 2.1456756
##  [96] 2.1574570 2.1580540 2.1684531 2.1789002 2.2615040
## [101] 2.2935560 2.3027327 2.3349573 2.3475047 2.3554650
## [106] 2.3695001 2.3718656 2.3904645 2.4063588 2.4145472
## [111] 2.4147992 2.4161936 2.4413403 2.4487785 2.4863993
## [116] 2.4914152 2.4940613 2.5110697 2.5117487 2.5192173
## [121] 2.5196009 2.5228421 2.5347647 2.5374632 2.5489902
## [126] 2.5591866 2.5648240 2.5719513 2.5749926 2.5887586
## [131] 2.6134987 2.6220035 2.6400358 2.6476885 2.6650499
## [136] 2.6859623 2.6883345 2.6940601 2.7031469 2.7127904
## [141] 2.7301993 2.7538212 2.7631841 2.7706671 2.7887026
## [146] 2.8038107 2.8045817 2.8093283 2.8095689 2.8166559
## [151] 2.8279150 2.8372313 2.8480472 2.8501581 2.8538009
## [156] 2.8679292 2.8782234 2.9062005 2.9207295 2.9211006
## [161] 2.9256746 2.9283912 2.9466351 2.9497451 2.9503653
## [166] 2.9559740 2.9641444 2.9690222 2.9749103 2.9767511
## [171] 2.9790977 2.9791313 2.9949325 3.0105486 3.0252443
## [176] 3.0283474 3.0297790 3.0414937 3.0470658 3.0585377
## [181] 3.0754463 3.0773041 3.0788403 3.0834757 3.0849566
## [186] 3.0926895 3.1081362 3.1153681 3.1246473 3.1247128
## [191] 3.1419399 3.1501968 3.1520497 3.1602103 3.1614756
## [196] 3.1669997 3.1776238 3.1926477 3.1930213 3.1937149
## [201] 3.2122734 3.2153028 3.2185763 3.2242406 3.2310536
## [206] 3.2316218 3.2414286 3.2460327 3.2468608 3.2526630
## [211] 3.2684902 3.2753820 3.2845035 3.2854222 3.2870938
## [216] 3.2968768 3.3034996 3.3042059 3.3059649 3.3396726
## [221] 3.3410369 3.3428925 3.3438778 3.3453556 3.3561916
## [226] 3.3730437 3.3804187 3.3837334 3.3877500 3.3881049
## [231] 3.4046696 3.4081254 3.4100968 3.4181618 3.4189010
## [236] 3.4215078 3.4247067 3.4280766 3.4306235 3.4512519
## [241] 3.4567618 3.4663810 3.4709358 3.4843819 3.4869525
## [246] 3.4905317 3.4942088 3.5066791 3.5101215 3.5131987
## [251] 3.5164528 3.5230973 3.5268697 3.5273756 3.5437712
## [256] 3.5476410 3.5570168 3.5576278 3.5657346 3.5917266
## [261] 3.6061191 3.6114772 3.6164928 3.6180392 3.6307112
## [266] 3.6372067 3.6442835 3.6522458 3.6524747 3.6560711
## [271] 3.6618792 3.6633754 3.6746809 3.6749344 3.6890945
## [276] 3.7064633 3.7166391 3.7219931 3.7269461 3.7364986
## [281] 3.7442996 3.7443569 3.7521223 3.7545717 3.7652701
## [286] 3.7689417 3.7730069 3.7746560 3.8022058 3.8041437
## [291] 3.8107664 3.8375739 3.8399887 3.8408500 3.8489678
```

```
## [296] 3.8749125 3.8829826 3.8832816 3.8858105 3.8896494
## [301] 3.9019089 3.9021555 3.9087718 3.9090158 3.9127081
## [306] 3.9182066 3.9189401 3.9360107 3.9410914 3.9450631
## [311] 3.9471778 3.9531335 3.9557663 3.9652081 3.9787048
## [316] 3.9844370 3.9862676 3.9909766 4.0018877 4.0035086
## [321] 4.0044664 4.0075907 4.0078887 4.0098348 4.0141509
## [326] 4.0149491 4.0240762 4.0260814 4.0351443 4.0388776
## [331] 4.0426032 4.0461559 4.0473051 4.0473173 4.0544149
## [336] 4.0620825 4.0657638 4.0692244 4.0720358 4.0738388
## [341] 4.0788830 4.0844871 4.0883855 4.0899251 4.0977841
## [346] 4.1232008 4.1256116 4.1256315 4.1299565 4.1528931
## [351] 4.1607001 4.1666645 4.1713082 4.1764403 4.1848025
## [356] 4.1861808 4.1919282 4.1955165 4.1980811 4.2070211
## [361] 4.2082107 4.2088074 4.2088728 4.2111280 4.2126819
## [366] 4.2170879 4.2208184 4.2215355 4.2268435 4.2303319
## [371] 4.2417842 4.2429866 4.2545837 4.2549810 4.2594125
## [376] 4.2637104 4.2664347 4.2712625 4.2742576 4.2820905
## [381] 4.2840494 4.2861482 4.3080479 4.3104642 4.3214429
## [386] 4.3245759 4.3396521 4.3474537 4.3676549 4.3740301
## [391] 4.3740889 4.3767156 4.3814502 4.3851900 4.3977723
## [396] 4.3993704 4.4000863 4.4015097 4.4027532 4.4098371
## [401] 4.4180578 4.4288419 4.4304450 4.4305695 4.4317678
## [406] 4.4375998 4.4510267 4.4523478 4.4555947 4.4580083
## [411] 4.4580427 4.4646581 4.4660212 4.4693851 4.4731168
## [416] 4.4760053 4.4848328 4.4853460 4.4866322 4.4884123
## [421] 4.4897766 4.4916896 4.4936158 4.4954957 4.4999996
## [426] 4.5043454 4.5145498 4.5153738 4.5282213 4.5299207
## [431] 4.5391150 4.5396222 4.5433858 4.5470008 4.5548739
## [436] 4.5630148 4.5674734 4.5791828 4.5836611 4.5855594
## [441] 4.5996896 4.6102192 4.6141977 4.6327837 4.6525936
## [446] 4.6526935 4.6532583 4.6566037 4.6590430 4.6610506
## [451] 4.6657560 4.6691845 4.6713815 4.6766340 4.6800150
## [456] 4.6862329 4.6888021 4.6929400 4.6930322 4.6931933
## [461] 4.6995499 4.6997033 4.7006378 4.7008177 4.7056330
## [466] 4.7065599 4.7154402 4.7194023 4.7255305 4.7263951
## [471] 4.7265421 4.7286734 4.7349940 4.7359897 4.7360926
## [476] 4.7466064 4.7533750 4.7579690 4.7605242 4.7606527
## [481] 4.7737893 4.7755134 4.7824133 4.7890550 4.7938058
## [486] 4.7946265 4.7971370 4.8052672 4.8075184 4.8371398
## [491] 4.8407560 4.8410625 4.8468023 4.8616721 4.8632723
## [496] 4.8659861 4.8733587 4.8739604 4.8851560 4.8945912
## [501] 4.9013215 4.9016677 4.9018370 4.9209084 4.9314027
## [506] 4.9347310 4.9390263 4.9456024 4.9499630 4.9513016
## [511] 4.9559175 4.9735537 4.9784670 4.9816651 5.0068493
## [516] 5.0069194 5.0142430 5.0162785 5.0193634 5.0216696
## [521] 5.0302427 5.0307550 5.0309904 5.0317653 5.0336039
## [526] 5.0375595 5.0380487 5.0431719 5.0480693 5.0522959
## [531] 5.0572029 5.0716609 5.0758565 5.0863098 5.0885567
```

```
## [536] 5.0891452 5.0892725 5.0893393 5.0925766 5.0968330
## [541] 5.0972405 5.0972729 5.0983657 5.1015254 5.1101170
## [546] 5.1287656 5.1303977 5.1316570 5.1317926 5.1328576
## [551] 5.1337699 5.1370859 5.1603358 5.1691351 5.1715161
## [556] 5.1749059 5.1797073 5.1812087 5.1850365 5.1854445
## [561] 5.1885995 5.1916252 5.1928916 5.2081966 5.2281327
## [566] 5.2359827 5.2421074 5.2610794 5.2691197 5.2780620
## [571] 5.2783710 5.2824253 5.2851120 5.2902889 5.2967086
## [576] 5.3047874 5.3056040 5.3072264 5.3094533 5.3177487
## [581] 5.3184327 5.3206378 5.3242656 5.3305280 5.3436406
## [586] 5.3503903 5.3507646 5.3531137 5.3599812 5.3679888
## [591] 5.3816345 5.3864377 5.3867899 5.3883341 5.3903003
## [596] 5.3920099 5.4071636 5.4110083 5.4119125 5.4162126
## [601] 5.4217663 5.4251480 5.4271890 5.4343311 5.4462551
## [606] 5.4464264 5.4553183 5.4591303 5.4627854 5.4645645
## [611] 5.4673215 5.4701788 5.4730203 5.4765146 5.4770486
## [616] 5.4836009 5.4927150 5.4936635 5.4962081 5.5016947
## [621] 5.5047442 5.5085423 5.5111333 5.5140670 5.5393868
## [626] 5.5412066 5.5431290 5.5472432 5.5529919 5.5635431
## [631] 5.5649545 5.5671526 5.5707744 5.5728497 5.5767701
## [636] 5.5810664 5.5838820 5.5910691 5.5917111 5.6022851
## [641] 5.6057307 5.6242484 5.6302153 5.6391946 5.6409600
## [646] 5.6443198 5.6455875 5.6480638 5.6605448 5.6759820
## [651] 5.6778118 5.6852070 5.6935190 5.6956771 5.6982469
## [656] 5.6992676 5.7061214 5.7073372 5.7103084 5.7165140
## [661] 5.7295071 5.7312803 5.7327709 5.7339094 5.7426836
## [666] 5.7485545 5.7742760 5.7747946 5.7783842 5.7794210
## [671] 5.7824942 5.7912902 5.7923376 5.7953145 5.8083087
## [676] 5.8127451 5.8142727 5.8234481 5.8286719 5.8311662
## [681] 5.8347901 5.8349883 5.8365941 5.8403936 5.8416048
## [686] 5.8444112 5.8473007 5.8485798 5.8602569 5.8617535
## [691] 5.8676956 5.8744076 5.8774614 5.8787231 5.8793596
## [696] 5.8845652 5.8882921 5.8971924 5.9060964 5.9102828
## [701] 5.9108441 5.9109451 5.9298057 5.9307429 5.9621275
## [706] 5.9683778 5.9715571 5.9764709 5.9859938 5.9942076
## [711] 5.9983217 5.9983909 6.0006873 6.0058286 6.0244501
## [716] 6.0277736 6.0329336 6.0374886 6.0391522 6.0432055
## [721] 6.0513905 6.0539102 6.0543776 6.0667948 6.0766743
## [726] 6.0816867 6.0852201 6.0886070 6.1117938 6.1124816
## [731] 6.1136405 6.1182878 6.1186518 6.1518351 6.1664418
## [736] 6.1782913 6.1807417 6.1829255 6.1898064 6.1976959
## [741] 6.1978285 6.1997033 6.2097471 6.2216143 6.2265449
## [746] 6.2267807 6.2312646 6.2430541 6.2459138 6.2490455
## [751] 6.2537551 6.2548635 6.2595198 6.2640936 6.2752197
## [756] 6.2796986 6.2834658 6.2859076 6.2862360 6.2895780
## [761] 6.3091303 6.3220513 6.3235245 6.3439751 6.3441254
## [766] 6.3455075 6.3465804 6.3497147 6.3517395 6.3523513
## [771] 6.3598857 6.3662389 6.3974153 6.3985160 6.4124987
```

```
## [776]  6.4138751  6.4148056  6.4272295  6.4312958  6.4392074
## [781]  6.4414924  6.4594249  6.4624366  6.4636097  6.4664497
## [786]  6.4788953  6.4869526  6.4896149  6.4900861  6.4961244
## [791]  6.5109867  6.5148088  6.5172403  6.5183205  6.5265642
## [796]  6.5435498  6.5499153  6.5619505  6.5622732  6.5763126
## [801]  6.5826290  6.5830432  6.5854835  6.5883071  6.5960104
## [806]  6.6018977  6.6044534  6.6385337  6.6476484  6.6526329
## [811]  6.6777462  6.6790037  6.6908569  6.6995312  6.7006949
## [816]  6.7036041  6.7061678  6.7070122  6.7138332  6.7202536
## [821]  6.7265394  6.7269660  6.7370431  6.7457201  6.7462602
## [826]  6.7489740  6.7491353  6.7919408  6.7940077  6.7981815
## [831]  6.7997994  6.8133359  6.8230604  6.8356989  6.8414490
## [836]  6.8449503  6.8465211  6.8534231  6.8551702  6.8677069
## [841]  6.8719497  6.8782379  6.8815858  6.8885408  6.8920992
## [846]  6.8984974  6.9165059  6.9278619  6.9448174  6.9507788
## [851]  6.9540295  6.9562802  6.9633871  6.9647698  6.9713744
## [856]  6.9853693  6.9942686  7.0072740  7.0316665  7.0325018
## [861]  7.0475999  7.0660247  7.0683453  7.0750238  7.0754651
## [866]  7.1050852  7.1333110  7.1353915  7.1464480  7.1489599
## [871]  7.1541706  7.2052405  7.2068367  7.2133803  7.2322567
## [876]  7.2422767  7.2440874  7.2442380  7.2463337  7.2811319
## [881]  7.2858325  7.2945341  7.3135492  7.3186987  7.3217963
## [886]  7.3285718  7.3300160  7.3319673  7.3366124  7.3392583
## [891]  7.3754304  7.3855355  7.3862097  7.4017574  7.4072420
## [896]  7.4155765  7.4242618  7.4276725  7.4295569  7.4388847
## [901]  7.4494105  7.4671699  7.4718814  7.5456898  7.5713748
## [906]  7.5996357  7.6206445  7.6333525  7.6397456  7.6717104
## [911]  7.6936502  7.6993946  7.7074600  7.7223811  7.7292946
## [916]  7.7388118  7.7960709  7.8060369  7.8152508  7.8465848
## [921]  7.8609892  7.8724821  7.8777069  7.8900096  7.8932715
## [926]  7.8996549  7.9126044  7.9201991  7.9393163  7.9839662
## [931]  8.0055157  8.0384331  8.0541625  8.0733042  8.0867943
## [936]  8.1073980  8.1160752  8.1507564  8.1594979  8.1671327
## [941]  8.1968307  8.1972939  8.1978373  8.2525628  8.2764934
## [946]  8.3247006  8.3305590  8.3401037  8.3531913  8.3633059
## [951]  8.3676811  8.4213082  8.4425913  8.4667611  8.4769230
## [956]  8.4848659  8.5103649  8.5263922  8.5374870  8.6081435
## [961]  8.6388579  8.6473770  8.6519291  8.6608282  8.6949107
## [966]  8.7576059  8.7804296  8.8057690  8.8196210  8.8232612
## [971]  8.8388018  8.8810651  8.9247695  8.9354609  8.9571836
## [976]  9.0181139  9.0499487  9.0671158  9.0726142  9.0732075
## [981]  9.0947751  9.1026147  9.1081024  9.1827373  9.2211812
## [986]  9.2383904  9.2875543  9.2937174  9.3400519  9.4023306
## [991]  9.4077138  9.4447565  9.6992618  9.8166295 10.1137783
## [996] 10.1961144 10.2523109 10.2770788 10.4529059 10.7863280
```

```
# Voltamos para a vírgula como separador decimal:
options(OutDec = ',')
```

- O que houve?!
- O problema é que não há valores repetidos no conjunto de dados! Por isso, todos os 1000 valores são modais.
- Uma maneira de evitar isto é definir a moda como o centro do intervalo mais curto que contém metade dos dados. Usamos a função `mlv` (*most likely value*):

```
moda <- mlv(normal, method = 'venter')
moda
```

```
## [1] 5,200738
```

- Esta moda estimada pode nem estar no conjunto de dados:

```
moda %in% normal
```

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

- Mas o resultado de `mlv()` é útil, pois nos diz que, embora não haja valores repetidos, valores próximos de 5 são mais frequentes, como mostra o histograma.

6.2.3.1 *Exercícios*

- Arrendonde os valores no vetor `normal` para 2 casas decimais e ache a(s) moda(s)
  1. com a função `mfv`, e
  2. com a função `mlv`, usando o método `venter`.

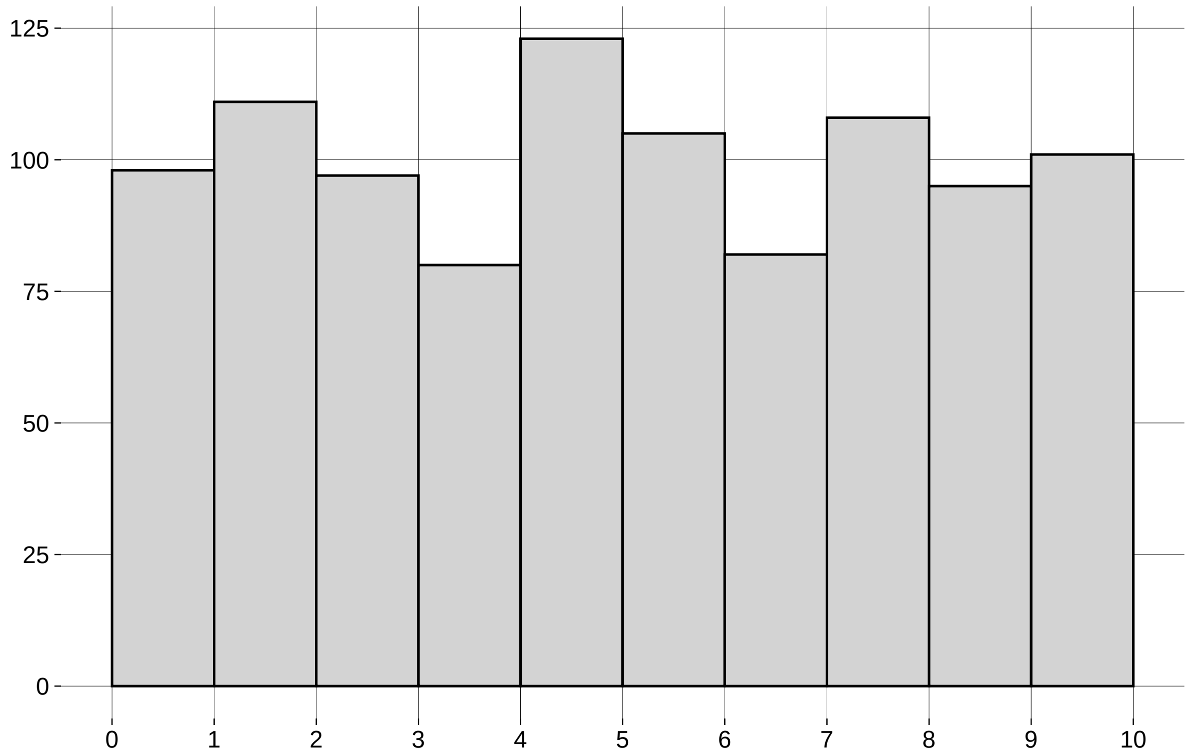
  Considerando o histograma, qual das respostas você prefere? Por quê?

6.3 Formas de uma distribuição

- A forma do histograma mostra aspectos importantes da distribuição dos dados.

6.3.1 Distribuição Uniforme

- Se o histograma tem todas as barras aproximadamente da mesma altura, dizemos que a distribuição é uniforme:



- A distribuição uniforme não tem moda, já que todos os valores têm aproximadamente a mesma frequência.

6.3.2 Simetria

- Se o histograma for simétrico (i.e., os lados esquerdo e direito são "espelhados"), dizemos que a distribuição é simétrica.
- A distribuição normal do exemplo acima é simétrica.
- A distribuição uniforme também é simétrica.
- Para distribuições simétricas, a média, a mediana e a moda (quando existe e é única) são bem próximas.
  - Para a distribuição normal do exemplo:

```
mean(normal)
```

```
## [1] 4,90222
```

```
median(normal)
```

```
## [1] 4,897956
```

```
mlv(normal, method = 'venter')
```

```
## [1] 5,200738
```

- Para a distribuição uniforme do exemplo:

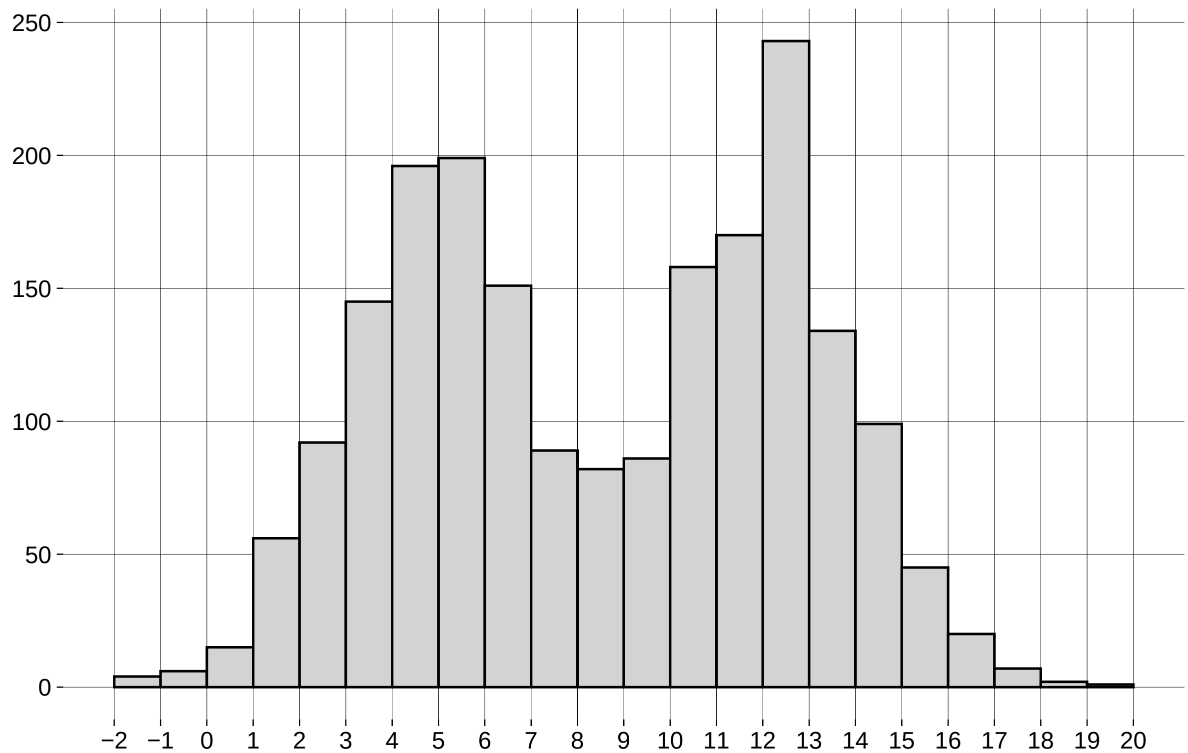
```
mean(uniforme)
```

```
## [1] 4,996517
```

```
median(uniforme)
```

```
## [1] 4,929388
```

- Uma distribuição pode ser simétrica, mas ter duas (ou mais) modas diferentes:



- Algumas distribuições não são simétricas, mas têm uma cauda longa à esquerda ou à direita.
- Dependendo da cauda, as distribuições são chamadas de assimétricas à esquerda ou assimétricas à direita.
- Um exemplo: receitas anuais (em milhões de dólares) de CEOs de grandes empresas:

```
df <- read_csv(
  './data/CEO_Salary_2012.csv',
  show_col_types = FALSE
)
glimpse(df)
```

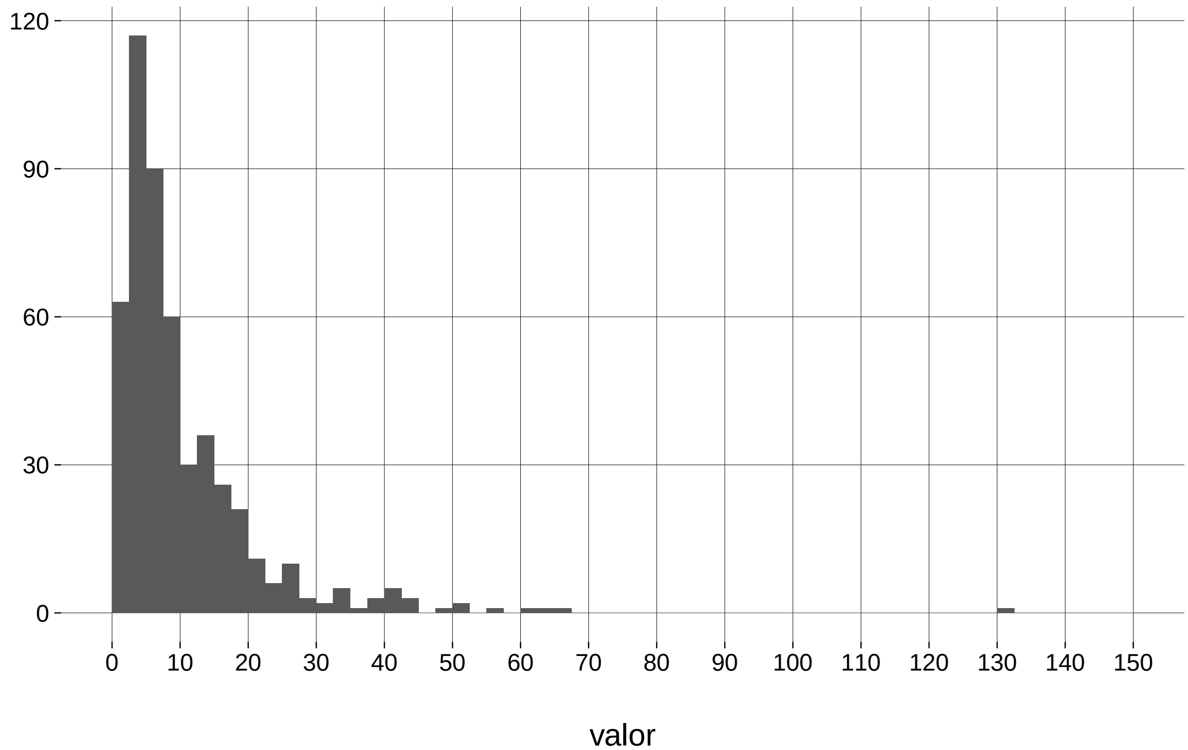
```
## Rows: 500
## Columns: 9
## $ Rank                  <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, ~
## $ Name                  <chr> "John H Hammergren", "Ralph Lauren", "M~
## $ Company               <chr> "McKesson", "Ralph Lauren", "Vornado Re~
## $ `1-Year Pay ($mil)`   <dbl> 131,190, 66,650, 64,405, 60,940, 55,790~
## $ `5 Year Pay ($mil)`   <dbl> 285,020, 204,060, NA, 60,940, 96,110, 1~
## $ `Shares Owned ($mil)` <dbl> 51,9, 5010,4, 171,7, 8582,3, 21,5, 47,3~
## $ Age                   <dbl> 53, 72, 55, 67, 59, 57, 55, 59, 61, 60,~
## $ Efficiency            <dbl> 121, 84, NA, NA, 138, 36, 12, NA, 91, 1~
## $ `Log Pay`             <dbl> 8,117901, 7,823800, 7,808920, 7,784902,~
```

- Vamos usar apenas os nomes e os valores anuais:

```
salarios <- df %>%
  select(Name, valor = `1-Year Pay ($mil)`)
```

- Um histograma:

```
salarios %>%
  ggplot(aes(x = valor)) +
    geom_histogram(breaks = seq(0, 150, 2.5)) +
    scale_x_continuous(breaks = seq(0, 150, 10)) +
    labs(y = NULL)
```



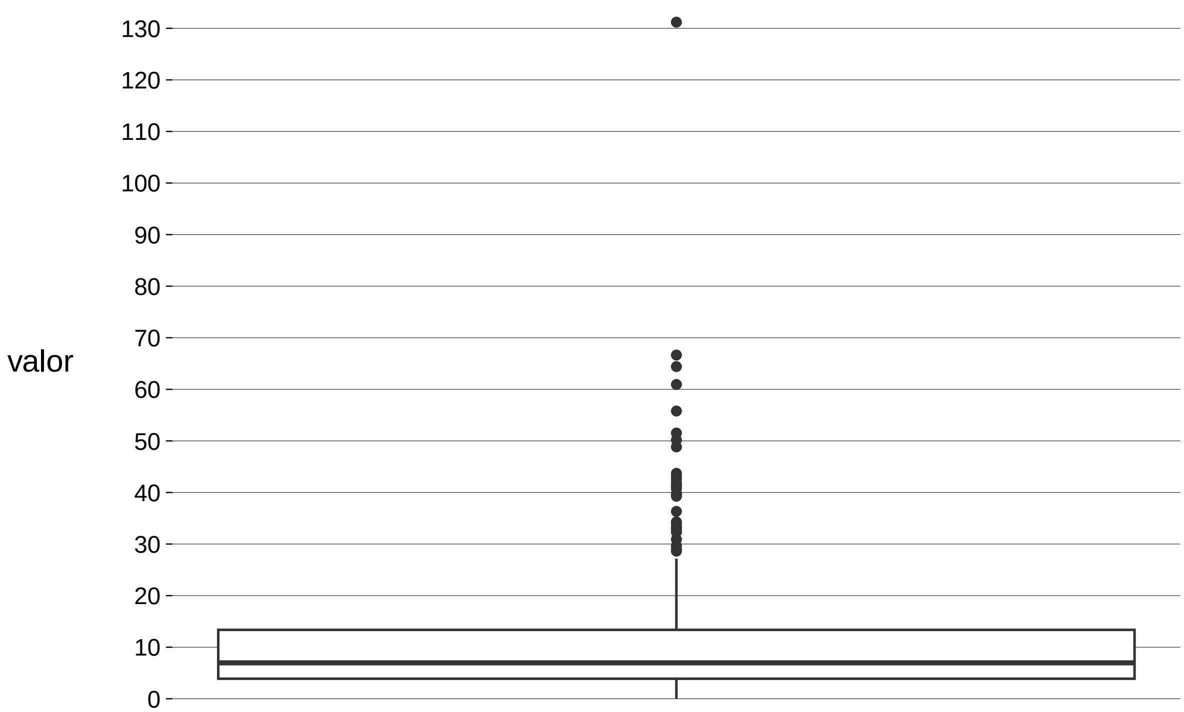
- É uma distribuição assimétrica à direita: a maior parte dos CEOs têm receitas anuais "baixas", de menos de 10 milhões. À medida que examinamos valores maiores, a quantidade de CEOs vai diminuindo lentamente.
- Observe que a longa cauda à direita "puxa" a média para um valor mais alto do que a mediana.
- A moda, que corresponde à barra mais alta do histograma, é menor que a mediana (e que a média):

```
sumario <- salarios %>%
  summarise(
    moda = mlv(valor, method = 'venter'),
    mediana = median(valor),
    media = mean(valor)
  )

sumario
```

```
## # A tibble: 1 x 3
##    moda mediana media
##   <dbl>   <dbl> <dbl>
## 1  4.60    6.97  10.5
```

- Em um *boxplot*, também é possível detectar a assimetria pela grande quantidade de *outliers* em um extremo:

```
salarios %>%
  ggplot(aes(y = valor)) +
    geom_boxplot() +
    scale_x_continuous(breaks = NULL) +
    scale_y_continuous(breaks = seq(0, 150, 10))
```



- Com distribuições assimétricas à esquerda, a situação se inverte: a média é menor que a mediana, que é menor que a moda.

6.3.3 Exercícios

- Ache um conjunto de dados com uma distribuição assimétrica à esquerda.
- Faça um histograma.
- Calcule a média, a mediana, e a moda dos dados.

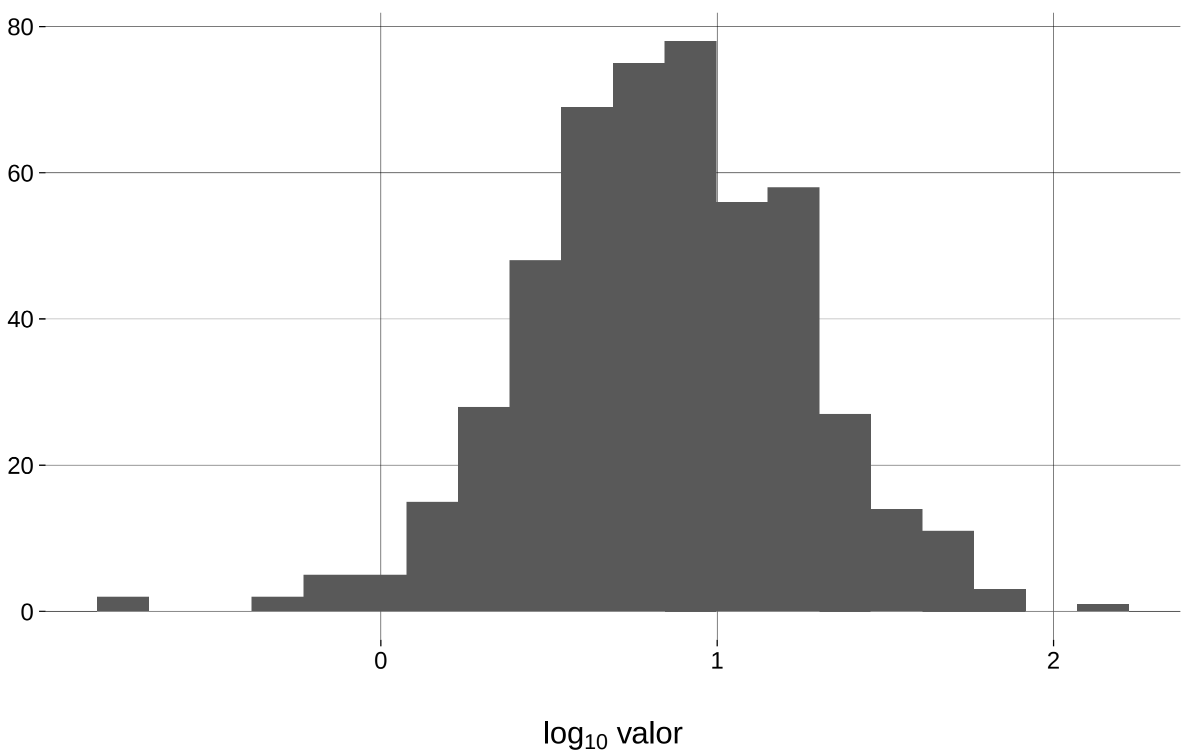
6.4 Re-expressão

- Muitas vezes, é recomendável transformar a escala dos dados para que uma distribuição assimétrica se torne simétrica.
- No exemplo das receitas dos CEOs, podemos tomar os logaritmos dos valores, em vez dos valores:

```
salarios_log <- salarios %>%
  mutate(log_valor = log10(valor))
```

```
salarios_log %>%
  ggplot(aes(x = log_valor)) +
    geom_histogram(bins = 20) +
    labs(
      x = TeX('$\\log_{10}$ valor'),
      y = NULL
    )
```

```
## Warning: Removed 3 rows containing non-finite values (`stat_bin()`).
```



- O logaritmo de um número na base 10 é, essencialmente, a quantidade de dígitos do número, vista como uma grandeza contínua.
- Logaritmos negativos vêm de valores entre 0 e 1.

- Logaritmo zero vem do valor 1.
- Valores iguais ou menores que zero não têm logaritmo definido.
- Por isso a mensagem de aviso sobre 3 valores removidos. São valores iguais a zero:

```
salarios_log %>%
  filter(valor == 0)
```

```
## # A tibble: 3 x 3
##   Name                 valor log_valor
##   <chr>                <dbl>     <dbl>
## 1 Malon Wilkus             0      -Inf
## 2 Matthew J Lambiase       0      -Inf
## 3 Larry Page               0      -Inf
```

- Uma vantagem desta escala logarítmica é que podemos entender melhor o histograma. Os dados não estão amontoados de um lado só.

6.4.1 Exercícios

- Quais são os registros com $\log_{10}$ valor < 0?
- Faça um *boxplot* dos logaritmos das receitas.

6.5 Medidas de posição

6.5.1 Quantis

- Na seção sobre *boxplots*, falamos sobre quantis, que são medidas de posição.
- Em R, a função `quantile` calcula quantis de um vetor:

```
salarios %>%
  pull(valor) %>%
  quantile()
```

```
##        0%       25%       50%       75%      100%
##   0,00000   3,88500   6,96750  13,36125 131,19000
```

- Você pode passar frações entre 0 e 1 para `quantile`. Por exemplo, para calcular o primeiro, o quinto, e o décimo percentis[3] das receitas dos CEOs:

[3]Um percentil é um quantil da forma $k/100$, para k natural, $0 \leq k \leq 100$.

```
salarios %>%
  pull(valor) %>%
  quantile(c(.01, .05, .1))
```

```
##      1%      5%     10%
## 0,48695 1,48405 2,19400
```

6.6 Medidas de dispersão

- Tão importantes quanto as medidas de centralidade são as medidas de dispersão (ou espalhamento).
- Elas informam o quanto os dados variam.

6.6.1 Amplitude

- Uma medida simples é a diferença entre o valor máximo e o valor mínimo.
- Lembrando do nosso exemplo das idades dos alunos:

```
idades
```

```
##  [1] 20 20 20 20 20 20 21 21 21 21 22 22 22 23 23 23 23 24 24 65
```

- A função `range` retorna o mínimo e o máximo:

```
range(idades)
```

```
## [1] 20 65
```

- A amplitude destes dados é, então

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

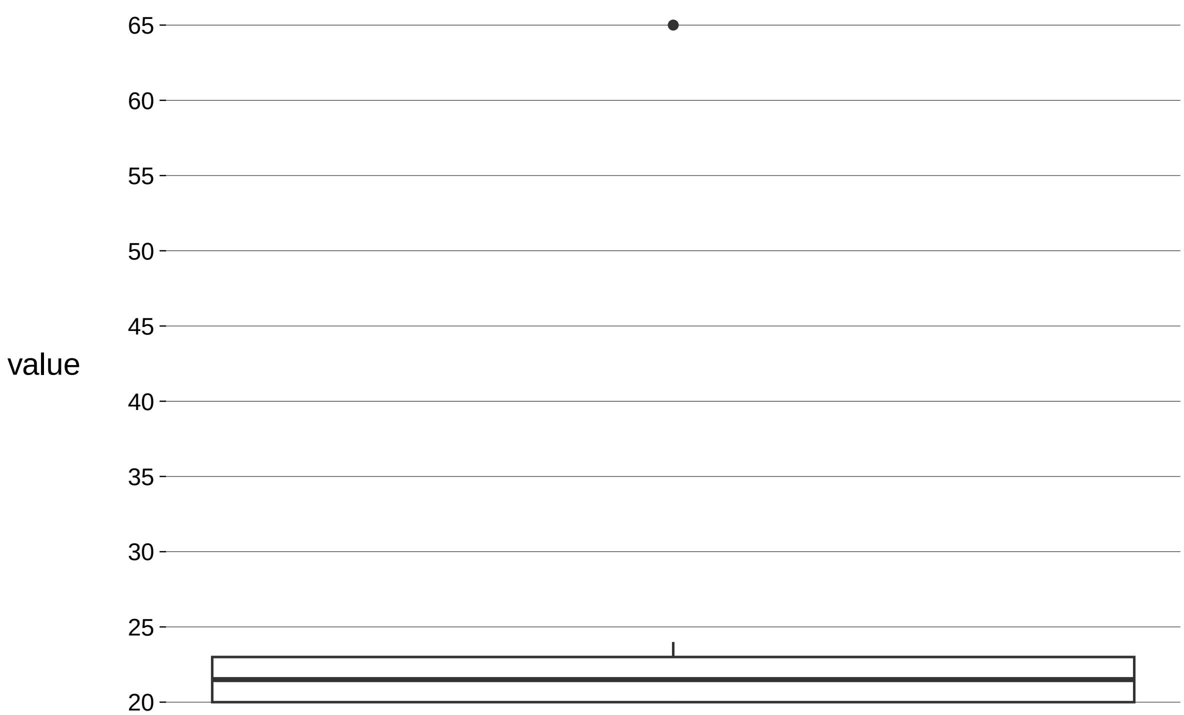
```
## [1] 45
```

- A diferença de idade entre o aluno mais novo e o mais velho é de 45 anos, um valor alto, por causa do velhinho.

6.6.2 IQR

- Na seção sobre *boxplots*, também falamos sobre o intervalo interquartil (IQR).
- No *boxplot*, é a altura da caixa. Para as idades dos alunos:

```
idades %>%
  as_tibble() %>%
  ggplot(aes(y = value)) +
    geom_boxplot() +
    scale_x_continuous(breaks = NULL) +
    scale_y_continuous(breaks = seq(20, 70, 5))
```



- O IQR é a diferença entre o primeiro e o terceiro quartis:

```
summary(idades)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   20,00   20,00   21,50   23,75   23,00   65,00
```

```
unname(summary(idades)[5] - summary(idades)[2])
```

```
## [1] 3
```

```
IQR(idades)
```

```
## [1] 3
```

- Ou seja, os 50% centrais dos alunos têm idade entre 20 e 23 anos, um IQR de 3.
- É uma variação pequena, porém mais fiel à realidade do que a amplitude, que é alta por causa do velhinho.
- Quanto maior o IQR, mais espalhados estão os dados.

6.6.3 Variância

- Agora, vamos trabalhar com os pesos (kg) e alturas (m) de um time de basquete:

```
medidas <- tibble(
  altura = .025 *
    c(72, 74, 68, 76, 74, 69, 72, 79, 70, 69, 77, 73),
  peso = 0.45 *
    c(180, 168, 225, 201, 189, 192, 197, 162, 174, 171, 185, 210)
)

medidas
```

```
## # A tibble: 12 x 2
##   altura  peso
##    <dbl> <dbl>
## 1   1.8   81
## 2   1.85  75.6
## 3   1.7  101.
## 4   1.9   90.4
## 5   1.85  85.0
## 6   1.72  86.4
## # i 6 more rows
```

```
summary(medidas$altura)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1,700   1,744   1,812   1,819   1,863   1,975
```

```
summary(medidas$peso)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   72,90   77,96   84,15   84,53   89,10  101,25
```

- A variância é a maneira mais usada de medir o espalhamento em torno da média.

- Para calcular a variância das alturas e a variância dos pesos, precisamos calcular valores intermediários.
- O desvio de um valor é a diferença entre o valor e a média. O desvio pode ser positivo ou negativo.

```
d_medidas <- medidas %>%
  mutate(
    d_altura = altura - mean(altura),
    d_peso = peso - mean(peso)
  )

d_medidas
```

```
## # A tibble: 12 x 4
##   altura  peso d_altura d_peso
##    <dbl> <dbl>    <dbl>  <dbl>
## 1   1.8   81     -0.0188 -3.53
## 2   1.85  75.6    0.0312 -8.92
## 3   1.7  101.    -0.119  16.7
## 4   1.9   90.4    0.0813  5.92
## 5   1.85  85.0    0.0312  0.525
## 6   1.72  86.4   -0.0938  1.88
## # i 6 more rows
```

- Vamos calcular o desvio médio das alturas e o desvio médio dos pesos:

```
d_medidas %>%
  summarize(
    d_medio_altura = mean(d_altura),
    d_medio_peso = mean(d_peso)
  )
```

```
## # A tibble: 1 x 2
##   d_medio_altura d_medio_peso
##            <dbl>        <dbl>
## 1              0    -3.55e-15
```

- Não foi uma boa idéia. O desvio médio sempre é igual a zero.[4] (O R pode mostrar algum valor diferente de zero por causa da precisão limitada dos números de ponto flutuante.)
- Como resolver isto? Elevando os desvios ao quadrado:

```
dq_medidas <- d_medidas %>%
  mutate(
    dq_altura = d_altura^2,
```

[4] Você vai provar isto em um exercício.

```
    dq_peso = d_peso^2
  )

dq_medidas
```

```
## # A tibble: 12 x 6
##   altura  peso d_altura d_peso dq_altura dq_peso
##    <dbl> <dbl>    <dbl>  <dbl>     <dbl>   <dbl>
## 1   1.8   81    -0.0188 -3.53   0.000352  12.4
## 2   1.85  75.6   0.0312 -8.92   0.000977  79.7
## 3   1.7  101.   -0.119  16.7    0.0141   280.
## 4   1.9   90.4   0.0813  5.92   0.00660   35.1
## 5   1.85  85.0   0.0312  0.525  0.000977   0.276
## 6   1.72  86.4  -0.0938  1.88   0.00879    3.52
## # i 6 more rows
```

- Agora temos os desvios quadrados, que são todos positivos.
- O desvio quadrado médio vai ser a variância:

```
dq_medidas %>%
  summarize(
    var_altura = mean(dq_altura),
    var_peso = mean(dq_peso)
  )
```

```
## # A tibble: 1 x 2
##   var_altura var_peso
##        <dbl>    <dbl>
## 1    0.00678     63.3
```

- Uma vantagem da variância é que *outliers* (que têm desvios quadrados maiores) contribuem mais do que elementos próximos à média (que têm desvios quadrados menores).
- Uma desvantagem da variância é que a sua unidade é o quadrado da unidade dos valores.
- Neste exemplo, as unidades são m^2 e kg^2!

6.6.4 Desvio-padrão

- É melhor trabalhar com a raiz quadrada da variância, que chamamos de desvio-padrão.
- As unidades são as mesmas que as unidades dos dados.

```
dq_medidas %>%
  summarize(
    dp_altura = sqrt(mean(dq_altura)),
    dp_peso = sqrt((mean(dq_peso)))
  )
```

```
## # A tibble: 1 x 2
##   dp_altura dp_peso
##       <dbl>   <dbl>
## 1    0.0824    7.96
```

- Claro que o R tem funções para calcular isso: `var` e `sd` (*standard deviation*):

```
medidas %>%
  summarize(
    altura_var = var(altura),
    altura_dp = sd(altura),
    peso_var = var(peso),
    peso_dp = sd(peso)
  )
```

```
## # A tibble: 1 x 4
##   altura_var altura_dp peso_var peso_dp
##        <dbl>     <dbl>    <dbl>   <dbl>
## 1    0.00740    0.0860     69.1    8.31
```

- Mas os valores são diferentes dos que calculamos. Por quê?

6.6.5 Definições

- Para uma população com N elementos e média μ, a variância é

$$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}$$

e o desvio-padrão é

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}}$$

- Para uma amostra com n elementos e média $\bar{x}$, a variância é

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

e o desvio-padrão é

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

- Nós calculamos a versão populacional destas medidas.
- R calcula a versão amostral destas medidas.
- Reveja os cálculos e entenda a diferença.
- Note, também, que as medidas populacionais são representadas por letras gregas — μ, σ^2, σ —, enquanto as medidas amostrais são representadas por letras latinas — $\bar{x}$, s^2, s.

> Mais adiante no curso, você vai entender por que o denominador da variância amostral é $n-1$, em vez de n.
>
> Nada é por acaso, nem mesmo em Estatística.

6.6.6 Exercícios

- Quando a variância e o desvio-padrão de um conjunto de dados são iguais a zero?
- Mostre que o desvio médio de qualquer conjunto de valores é igual a zero.

  Ou seja, considere o conjunto

  $$\{x_1, x_2, \dots, x_n\}$$

  e prove que

  $$\sum_{i=1}^{n}(x_i - \bar{x}) = 0$$

  Manipule apenas as variáveis x_i. Não use exemplos, pois eles não provam o enunciado geral.

  Dica: lembre que $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$.

6.7 Coeficiente de variação

- Em um conjunto de dados, o desvio-padrão é uma medida importante da variação dos dados.
- Mas a unidade do desvio-padrão muda de um conjunto de dados para outro: alturas em metros, pesos em quilos etc.
- Podemos eliminar as unidades expressando o desvio-padrão em termos da média.
- O resultado é a fração $\frac{\sigma}{\mu}$ (na população) ou $\frac{s}{\bar{x}}$ na amostra.
- Esta fração é o coeficiente de variação (CV).
- O CV não tem unidades.
- Para as alturas do exemplo dos jogadores de basquete:

  A média das alturas é $1{,}82$ metros.

  O desvio-padrão das alturas é $0{,}09$ metros.

  O CV é aproximadamente $0{,}0473$.

  ```
  statip::cv(medidas$altura)
  ```

  ```
  ## [1] 0,04729982
  ```

  Em outras palavras, para as alturas, um desvio-padrão corresponde a $4{,}73\%$ da média.

- Para os pesos:

  A média dos pesos é $84{,}53$ quilos.

  O desvio-padrão dos pesos é $8{,}31$ quilos.

  O CV é aproximadamente $0{,}0983$.

  ```
  statip::cv(medidas$peso)
  ```

  ```
  ## [1] 0,09834649
  ```

  Em outras palavras, para os pesos, um desvio-padrão corresponde a $9{,}83\%$ da média.

- Segundo estes valores, a variação dos pesos é cerca de 2 vezes maior do que a variação das alturas.

O coeficiente de variação sempre faz sentido para dados do nível racional (veja a definição) — i.e., dados onde o zero é absoluto.

Para dados apenas intervalares, o uso do CV pode levar a conclusões absurdas, como você terá chance de ver no exercício.

6.7.1 Exercícios

- Considere o seguinte conjunto de temperaturas (em graus Celsius):

```
celsius <- c(0, 10, 20, 30, 40)
```

- E as mesmas temperaturas (em graus Fahrenheit):

```
fahrenheit <- 9 * celsius / 5 + 32
```

- Calcule para cada um dos dois vetores acima:
  1. A média,
  2. O desvio-padrão,
  3. O coeficiente de variação.
- As temperaturas são as mesmas (apenas em unidades diferentes), mas os CVs são diferentes. Por quê?
- Agora, convertemos as mesmas temperaturas para a escala Kelvin:

```
kelvin <- celsius + 273.15
```

- E para a escala Rankine:

```
rankine <- fahrenheit + 459.67
```

- Calcule para cada um dos dois vetores acima:
  1. A média,
  2. O desvio-padrão,
  3. O coeficiente de variação.
- Compare:
  1. As médias de `celsius` e `kelvin`,
  2. As médias de `fahrenheit` e `rankine`,
  3. Os desvios-padrão de `celsius` e `kelvin`,

4. Os desvios-padrão de `fahrenheit` e `rankine`,
5. Os coeficientes de variação de `kelvin` e `rankine`.

- Explique o que houve.

6.8 Escores-padrão

- Para qualquer conjunto de dados, a unidade usada é uma escolha arbitrária.
- Para alturas, por exemplo, podemos usar metros, centímetros, pés, polegadas etc.
- A escolha de unidades é tão arbitrária que podemos escolher uma unidade (que dificilmente vai ter nome) que faça com que a média do conjunto de dados seja zero e que o desvio-padrão seja igual a 1.
- Isto equivale a tomar, como unidade, o *desvio-padrão acima da média*.
- Os valores, nesta nova unidade, são chamados de escores-padrão.
- Dizemos que os valores foram padronizados.
- Vamos usar as alturas dos jogadores de basquete.
- Para fazer a altura média virar zero, basta subtrair, de cada altura, a altura média:

```
alturas <- medidas$altura
mean(alturas)
```

```
## [1] 1,81875
```

```
alturas_deslocadas <- alturas - mean(alturas)
mean(alturas_deslocadas)
```

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

- Para fazer o desvio-padrão ser igual a 1, basta dividir estes valores pelo desvio-padrão dos dados originais:

```
sd(alturas)
```

```
## [1] 0,08602656
```

```
alturas_padronizadas <- alturas_deslocadas / sd(alturas)
sd(alturas_padronizadas)
```

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

- Tome, por exemplo, o seguinte jogador:

```
altura <- alturas[1]
altura
```

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

```
altura_padronizada <- (alturas[1] - mean(alturas)) / sd(alturas)
altura_padronizada
```

```
## [1] -0,217956
```

Faça as contas: o valor da altura padronizada deste jogador significa que a altura dele está $0{,}217956$ desvios-padrão *abaixo* da altura média.

- No geral:
  - Se a média for $\bar{x}$, e
  - Se o desvio-padrão for s,
  - Os escores-padrão z_i vão ser

$$z_i = \frac{x_i - \bar{x}}{s}$$

- Em R, a função `scale` faz isso:

```
medidas <- medidas %>%
  mutate(altura_padronizada = scale(altura)[,1])

medidas %>%
  select(altura, altura_padronizada)
```

```
## # A tibble: 12 x 2
##   altura altura_padronizada
##    <dbl>              <dbl>
## 1   1.8              -0.218
## 2   1.85              0.363
## 3   1.7              -1.38
## 4   1.9               0.944
## 5   1.85              0.363
## 6   1.72             -1.09
## # i 6 more rows
```

```
mean(medidas$altura_padronizada)
```

```
## [1] -0,00000000000000004610683
```

```
sd(medidas$altura_padronizada)
```

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

- A função `scale` foi feita para receber e retornar matrizes. Como estamos trabalhando com vetores, usamos `scale(altura)[,1]` para tomar apenas a primeira (e única) coluna do resultado.

6.8.1 Exercícios

- Por que, quando calculamos as alturas deslocadas divididas pelo desvio-padrão das alturas, temos certeza de que a média dos valores resultantes não mudou?
- Padronize os pesos dos jogadores de basquete.
- Confira a média e o desvio-padrão dos pesos padronizados.
- Crie um *scatterplot* de peso por altura.
- Crie um *scatterplot* de peso padronizado por altura padronizada.
- Compare os dois *scatterplots*. O que muda de um para outro?

6.9 Teorema de Tchebychev

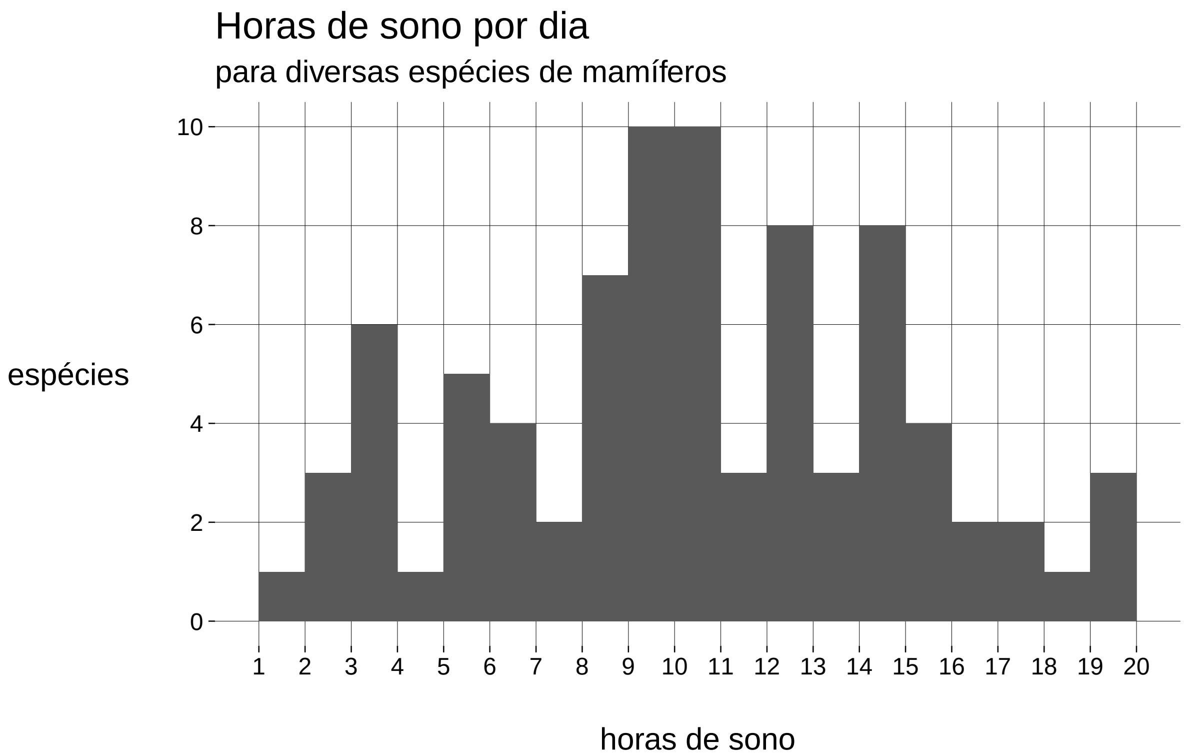
- Grosso modo, quanto mais alto o desvio-padrão, maior é a distância da média até os valores.
- Ou seja, quanto menor o desvio-padrão, maior é a proporção de valores que estão próximos à média.
- O teorema de Tchebychev quantifica esta idéia:

  Em *qualquer* distribuição, a proporção de valores dentro de $\pm k$ desvios-padrão $(k > 1)$ da média é de, *no mínimo*

$$1 - \frac{1}{k^2}$$

6.9.1 Exemplo

- Lembre-se do conjunto de dados sobre os totais de horas de sono de diversos mamíferos:



- Média e desvio-padrão:

```
media <- mean(df$value)
media
```

```
## [1] 10,43373
```

```
dp <- sd(df$value)
dp
```

```
## [1] 4,450357
```

- Qual a proporção de espécies que estão a $1{,}3$ ou menos desvios-padrão de distância da média?

```
k <- 1.3
inicio <- media - k * dp
inicio
```

```
## [1] 4,648271
```

```
fim <- media + k * dp
fim
```

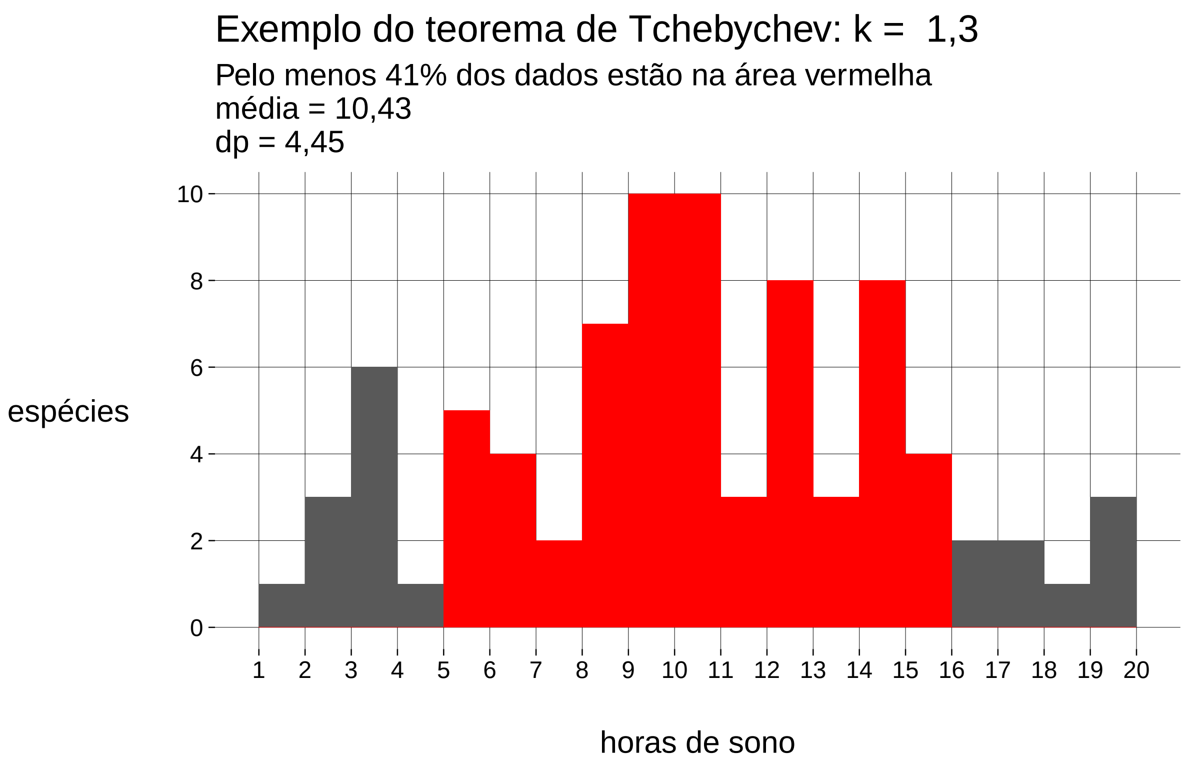
```
## [1] 16,2192
```

- O teorema diz que no mínimo a seguinte proporção das espécies está dentro deste intervalo:

```
proporcao_teorema <- 1 - 1 / k^2
proporcao_teorema
```

```
## [1] 0,408284
```

- **Graficamente:**



- **Vamos conferir:**

```
total_especies <- df %>% nrow()
total_especies
```

```
## [1] 83
```

```
especies_intervalo <- df %>%
  filter(value >= inicio & value <= fim) %>%
  nrow()
especies_intervalo
```

```
## [1] 64
```

```
proporcao_real <- especies_intervalo / total_especies
proporcao_real
```

```
## [1] 0,7710843
```

- Como o teorema usa apenas a média e o desvio-padrão, e mais nenhuma informação sobre a distribuição dos valores — forma, simetria etc. — o que ele garante é, muitas vezes, mais fraco do que a realidade.
- Neste exemplo, o teorema garantia no mínimo $40{,}83\%$ das espécies a $1{,}30$ ou menos desvios-padrão de distância da média.
- A proporção verdadeira é $77{,}11\%$ das espécies.
- O teorema está certo (claro), mas, sem mais informações sobre a distribuição dos dados, o teorema não pode ser mais preciso.

6.9.2 Exercício

- Uma loja recebe uma média de 20 clientes por dia, com desvio-padrão de 2 clientes.
- Em um ano (365 dias), em quantos dias, no mínimo, o número de clientes ficou entre 16 e 24, inclusive?

CAPÍTULO 7

Probabilidades

7.1 Vídeo 1

https://youtu.be/hJfyzRzEs44

7.2 Espaço amostral

- Para falar em probabilidades, precisamos falar de experimentos, resultados, espaços amostrais, e eventos.
- Um experimento probabilístico é um experimento cujo resultado exato é desconhecido *a priori*; mais ainda: executar o experimento diversas vezes, nas mesmas condições, pode produzir resultados diferentes.
- O espaço amostral é o conjunto de todos os resultados possíveis de um experimento probabilístico, representados de alguma forma.
- Exemplos:
    1. Experimento: lançar uma moeda;
       Espaço amostral: $\{K, C\}$ (onde K é cara, C é coroa).

2. Experimento: lançar 2 moedas;
   Espaço amostral: $\{(K, K), (K, C), (C, K), (C, C)\}$.
3. Experimento: lançar um dado;
   Espaço amostral: $\{1, 2, 3, 4, 5, 6\}$.
4. Experimento: lançar 2 dados:
   Espaço amostral: $\{(1, 1), (1, 2), \dots, (6, 5), (6, 6)\}$.

7.3 Evento

- Um evento é um subconjunto do espaço amostral; ou seja, um evento é um conjunto de resultados.
- Exemplos:
  1. Lançar uma moeda e obter cara:
     $\{K\}$.
  2. Lançar 2 moedas e obter resultados iguais:
     $\{(K, K), (C, C)\}$.
  3. Lançar um dado e obter um número maior que 4:
     $\{5, 6\}$.
  4. Lançar 2 dados e obter 2 números iguais:
     $\{(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6)\}$.
- Dizemos que o evento A ocorreu se o experimento foi realizado e o resultado obtido está no conjunto que corresponde ao evento A.

7.4 Análise Combinatória

- Para calcularmos probabilidades, vamos precisar contar a quantidade de certos objetos complexos (formados por partes menores).
- Existem técnicas de contagem, que são assunto de Análise Combinatória.
- Exemplos:
  1. Quantas senhas de 6 caracteres (dentre letras e dígitos apenas) existem, sem distinguir entre minúsculas e maiúsculas?
  2. E se não puder haver repetição de caracteres?
- Se você nunca tiver estudado técnicas de contagem, ou se quiser revisar ou aprender mais, consulte o excelente livro Morgado et al. (2004).

7.5 Probabilidade clássica

- Nesta abordagem simples — que pode não ser a correta para o problema que estamos tentando resolver —, cada resultado do espaço amostral tem a mesma chance de ocorrer.
- Ou seja, para um evento A, a probabilidade $P(A)$ é

$$P(A) = \frac{\text{Qtde de resultados em } A}{\text{Qtde de resultados no espaço amostral}}$$

- Exemplo: de um baralho normal, de 52 cartas, qual a probabilidade de escolher uma carta ao acaso e obter
  1. Uma carta de ouros?
  2. Uma carta vermelha?
  3. Uma carta de figura (J, Q ou K)?
  4. Uma carta de ouros, copas, paus ou espadas?
  5. Um carta de um naipe verde?

> Só podemos usar este raciocínio se todos os resultados do experimento tiverem a mesma probabilidade de ocorrer.
>
> Como a carta é escolhida ao acaso, esta condição é satisfeita neste exemplo.

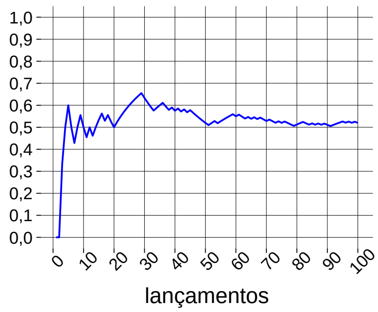
7.6 Probabilidade empírica

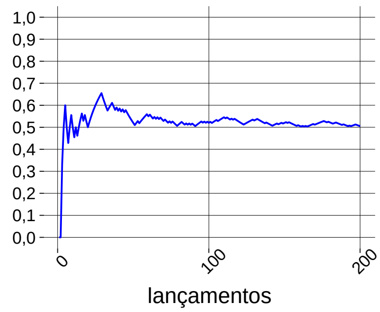
- Baseada em repetições de um experimento probabilístico.
- Nesta abordagem, a probabilidade de um evento é sua frequência relativa:

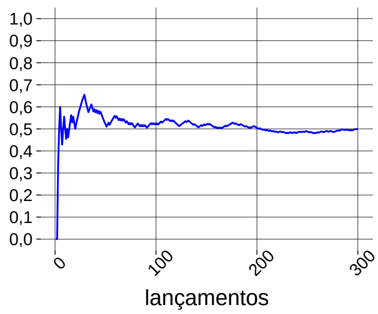
$$P(A) = \frac{\text{Qtde de ocorrências de } A}{\text{Qtde total de repetições do experimento}}$$

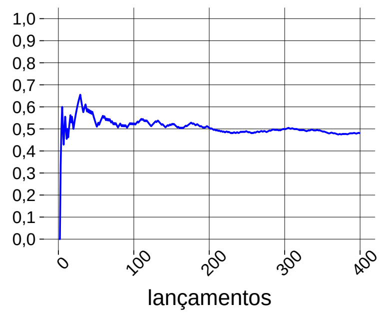
- Esta abordagem é fácil de usar quando é possível repetir o experimento muitas vezes, nas mesmas condições (lançar uma moeda, escolher uma carta de um baralho).
- Em outros casos (calcular a probabilidade de um candidato vencer uma eleição), não é possível repetir o experimento nas mesmas condições.

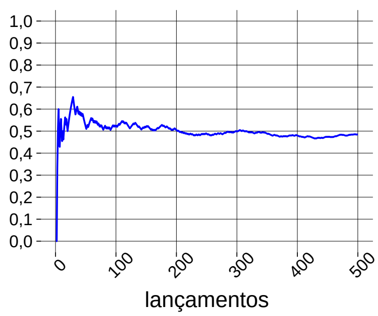
- Exemplo: se lançarmos uma moeda não-viciada muitas vezes, a proporção de caras vai ser aproximadamente $0{,}5$. Os gráficos abaixo mostram como, à medida que o número de lançamentos aumenta (no eixo horizontal), a proporção de caras (no eixo vertical) vai se aproximando de $0{,}5$:

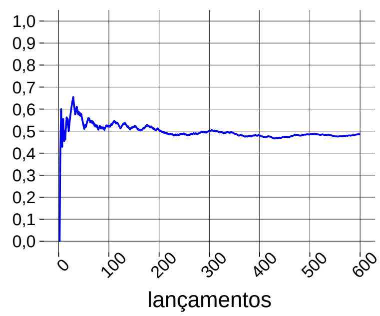

1,0
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0,0
0
10
20
30
40
50
60
70
80
90
100
lançamentos

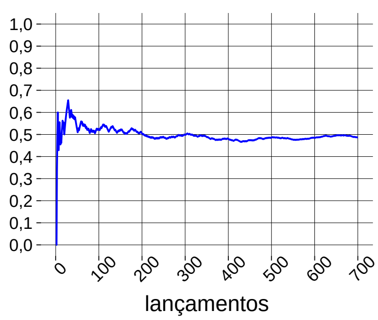

1,0
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0,0
0
100
200
lançamentos

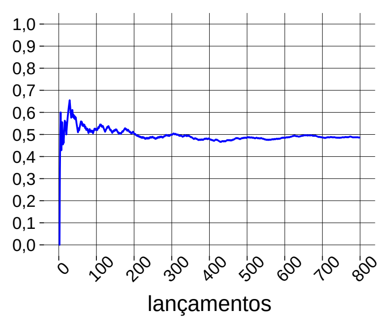

1,0
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0,0
0
100
200
300
lançamentos

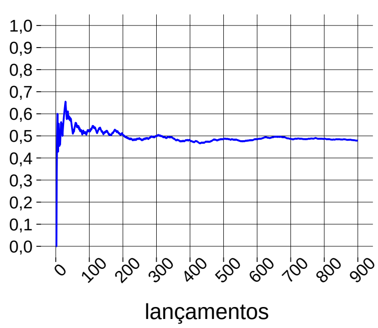

1,0
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0,0
0
100
200
300
400
lançamentos

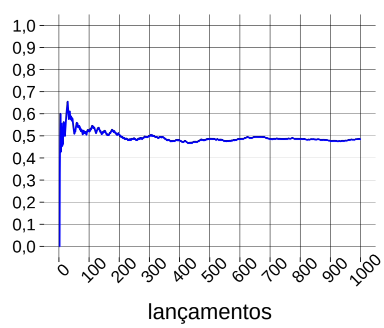

1,0
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0,0
0
100
200
300
400
500
lançamentos


1,0
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0,0
0
100
200
300
400
500
600
lançamentos


1,0
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0,0
0
100
200
300
400
500
600
700
lançamentos


1,0
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0,0
0
100
200
300
400
500
600
700
800
lançamentos


1,0
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0,0
0
100
200
300
400
500
600
700
800
900
lançamentos


1,0
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0,0
0
100
200
300
400
500
600
700
800
900
1000
lançamentos

- A lei dos grandes números é um resultado matemático que diz, essencialmente, que, quando o número n de repetições de um experimento tende a infinito, a frequência relativa de um evento tende à sua probabilidade real.

> Um erro comum é achar que, se houve poucas caras nos lançamentos mais recentes, então a probabilidade de o resultado ser cara no próximo lançamento é maior, para que a proporção de caras fique mais perto de $0{,}5$.
>
> A lei dos grandes números fala sobre os resultados do experimento quando n tende ao infinito, não no futuro próximo.
>
> Em lançamentos independentes de uma moeda não-viciada, a probabilidade de cara sempre é $0{,}5$.

7.7 Probabilidade subjetiva

- Outra interpretação de probabilidades se baseia na crença — a estimativa de um agente sobre a ocorrência de um evento.
- Uma maneira de quantificar a crença é através de apostas justas.
- Por exemplo, você aposta com um amigo que
  - Se o seu time de basquete[1] vencer o próximo jogo contra o dele, ele pagará $\$3$ para você.
  - Se o time dele vencer o próximo jogo contra o seu, você pagará $\$1$ para ele.
- Se você considera justa esta aposta, então você crê que a probabilidade de o time dele vencer é 3 vezes maior do que a probabilidade de o seu time vencer.
- Como a soma das probabilidades de um evento e do evento complementar deve ser 1, isto equivale a dizer que

$$P(\text{seu time vencer}) = 1/4 \qquad \text{e} \qquad P(\text{time dele vencer}) = 3/4$$

- Em mais detalhes:
  - Você pode receber menos com uma probabilidade maior,
  - Seu amigo pode receber mais com uma probabilidade menor,
  - A razão entre as quantias (3) é contrabalançada exatamente pela razão entre as probabilidades ($1/3$).

[1] Considere que, no basquete, o empate é impossível.

7.8

Formalização de probabilidades

- Para trabalhar matematicamente com probabilidades, é preciso definir as "regras do jogo".
- Tudo que se pode concluir sobre probabilidades é consequência dos seguintes axiomas, formulados por Kolmogorov em 1933:
    1. $0 \leq P(A) \leq 1$, para qualquer evento A.
    2. $P(\Omega) = 1$, onde Ω é o espaço amostral (o conjunto de todos os resultados possíveis do experimento em questão);
    3. $P(\bar{A}) = 1 - P(A)$, onde $\bar{A}$ é o evento complementar de A (i.e., o evento que corresponde a A não ocorrer)
    4. $P(A_1 \cup \cdots \cup A_n) = P(A_1) + \cdots + P(A_n)$, onde $A_1, \ldots, A_n$ são eventos disjuntos dois a dois (i.e., A_i e A_j não podem ocorrer ao mesmo tempo, para todo par (i, j) com $i \neq j$).
- Mostre, a partir dos axiomas acima, que

$$P(\emptyset) = 0$$

Dica

$$\emptyset = \bar{\Omega}$$

7.9

Eventos independentes (explicação informal)

- Se a ocorrência de A não influencia a ocorrência de B, nem vice-versa, dizemos que os eventos A e B são independentes.
- Exemplo:
    - O experimento é lançar um dado duas vezes.
    - A é o evento o primeiro lançamento deu um número par.
    - B é o evento o segundo lançamento deu 6.
    - Saber se A aconteceu não nos ajuda em nada a estimar se B aconteceu.
    - Aqui, A e B são independentes.

- Outro exemplo:
  - O experimento é lançar um dado duas vezes.
  - A é o evento o primeiro lançamento deu um número menor que 3.
  - B é o evento a soma dos dois lançamentos é maior que 8.
  - Agora, saber se A aconteceu ajuda a estimar se B aconteceu.
  - Na verdade, se A aconteceu, B é impossível (a probabilidade de B, dado A, é 0).
  - Se A não aconteceu, a probabilidade de B é $5/12$.
  - Aqui, A e B não são independentes.
- A probabilidade de os eventos A e B acontecerem ao mesmo tempo é escrita como

$$P(A, B) \qquad \text{ou como} \qquad P(A \cap B)$$

- Quando A e B são independentes,

$$P(A, B) = P(A) \cdot P(B)$$

- Ou seja, quando A e B são independentes, a probabilidade de A e B acontecerem ao mesmo tempo é igual ao produto das probabilidades de A e de B.
- Mais adiante, vamos ver uma definição formal de independência, e vamos provar esta última igualdade.

7.10

$P(A \cup B)$ com A e B não-disjuntos

- Um dos axiomas de probabilidade fala sobre a probabilidade da união de vários eventos disjuntos (sem elementos em comum):

$$P(A_1 \cup \cdots \cup A_n) = P(A_1) + \cdots + P(A_n)$$

- E se os eventos não forem disjuntos?
- Veja a figura abaixo:

  ???
- Imagine que a probabilidade de um evento é proporcional à sua área nesta figura.
- Se você somar a área de A com a área de B, você vai estar contando duas vezes a área comum aos dois (a área que corresponde a $A \cap B$).
- Por isso, o certo é "descontar" esta área.

- O resultado é

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

- Exemplo: suponha que 25% das pessoas têm cachorro, 29% das pessoas têm gato, e 12% das pessoas têm cachorro e gato.
- Qual a probabilidade de que uma pessoa tenha gato ou cachorro ou ambos?

$$\begin{aligned} P(\text{cachorro } \cup \text{ gato}) &= P(\text{cachorro}) + P(\text{gato}) - P(\text{cachorro } \cap \text{ gato}) \\ &= 0{,}25 + 0{,}29 - 0{,}12 \\ &= 0{,}42 \end{aligned}$$

- No geral, para n eventos $A_1, \dots, A_n$:

$$\begin{aligned} P(A_1 \cup \dots \cup A_n) = {} & P(A_1) + \dots + P(A_n) \\ & - P(A_1 \cap A_2) - \dots - P(A_{n-1} \cap A_n) \\ & + P(A_1 \cap A_2 \cap A_3) + \dots + P(A_{n-2} \cap A_{n-1} \cap A_n) \\ & \dots \\ & \pm P(A_1 \cap \dots \cap A_n) \end{aligned}$$

- Na última linha
  - o sinal vai ser $+$ se n for ímpar;
  - o sinal vai ser $-$ se n for par;
  - poderíamos escrever, então, $(-1)^{n+1} \cdot P(A_1 \cap \dots \cap A_n)$.
- Escreva, seguindo o padrão acima, a expressão para

$$P(A \cup B \cup C)$$

7.11

Problema do aniversário

7.11.1

Solução teórica

- Em uma sala estão 25 pessoas escolhidas ao acaso.
- Qual a probabilidade de que pelo menos 2 delas façam aniversário no mesmo dia do ano?
- Premissas:

- Os dias dos aniversários das pessoas são independentes.
- Cada dia do ano tem a mesma probabilidade de ser o aniversário de alguém.
- Vamos ignorar anos bissextos. Cada ano tem 365 dias.

- Queremos achar $P(I)$, onde I é o evento de que pelo menos duas pessoas têm aniversários iguais.
- Vamos calcular a probabilidade $P(N)$ de que não haja aniversários iguais.
- Este evento N é o complementar do evento I, i.e., $N = \bar{I}$.
- Então, $P(I) = 1 - P(N)$.
- $P(N)$ é a probabilidade de que todos os aniversários caiam em dias diferentes:
  - A pessoa 1 pode ter nascido em qualquer dia do ano.
  - A pessoa 2 precisa ter nascido em algum dos outros 364 dias. A probabilidade é $\frac{364}{365}$.
  - A pessoa 3 precisa ter nascido em algum dos outros 363 dias. A probabilidade é $\frac{363}{365}$.
  - ...
  - A pessoa 25 precisa ter nascido em algum dos outros 341 dias. A probabilidade é $\frac{341}{365}$.
- Como os nascimentos são independentes, a probabilidade de todos os eventos acontecerem juntos é o produto das probabilidades:

$$P(N) = \frac{364}{365} \cdot \frac{363}{365} \cdot \dots \cdot \frac{341}{365} = \frac{364 \cdot 363 \cdot \dots \cdot 341}{365^{24}}$$

- O que dá

```
pn <- prod((364:341)/365)
pn
```

```
## [1] 0,4313003
```

- Então, $P(I)$ é

```
1 - pn
```

```
## [1] 0,5686997
```

- Surpreso? Com 25 pessoas na sala, é mais provável haver do que não haver coincidência de aniversários!

7.11.2 Simulação

- Vamos simular milhares de salas com 25 pessoas satisfazendo as premissas e ver em quantas delas há coincidência de aniversários. Examine o código abaixo:

```
nsalas <- 1e4
npessoas <- 25

coincidencia <- function(sala) {

  # Se a quantidade de valores únicos for diferente
  # da quantidade total de valores, então há repetição
  !(length(unique(sala)) == length(sala))

}

gerar_e_testar <- function(npessoas) {

  # Escolhemos, ao acaso, npessoas números entre 1 e 365,
  # com reposição
  sala <- sample(1:365, npessoas, replace = TRUE)

  # Testamos se há alguma coincidência de aniversários
  coincidencia(sala)

}

simular <- function(npessoas, nsalas) {

  resultados <- replicate(nsalas, gerar_e_testar(npessoas))

  # Como resultados é um vetor booleano, tirar a média
  # vai dar a proporção de resultados verdadeiros,
  # que é a probabilidade.
  mean(resultados)

}

simular(npessoas, nsalas)
```

```
## [1] 0,5666
```

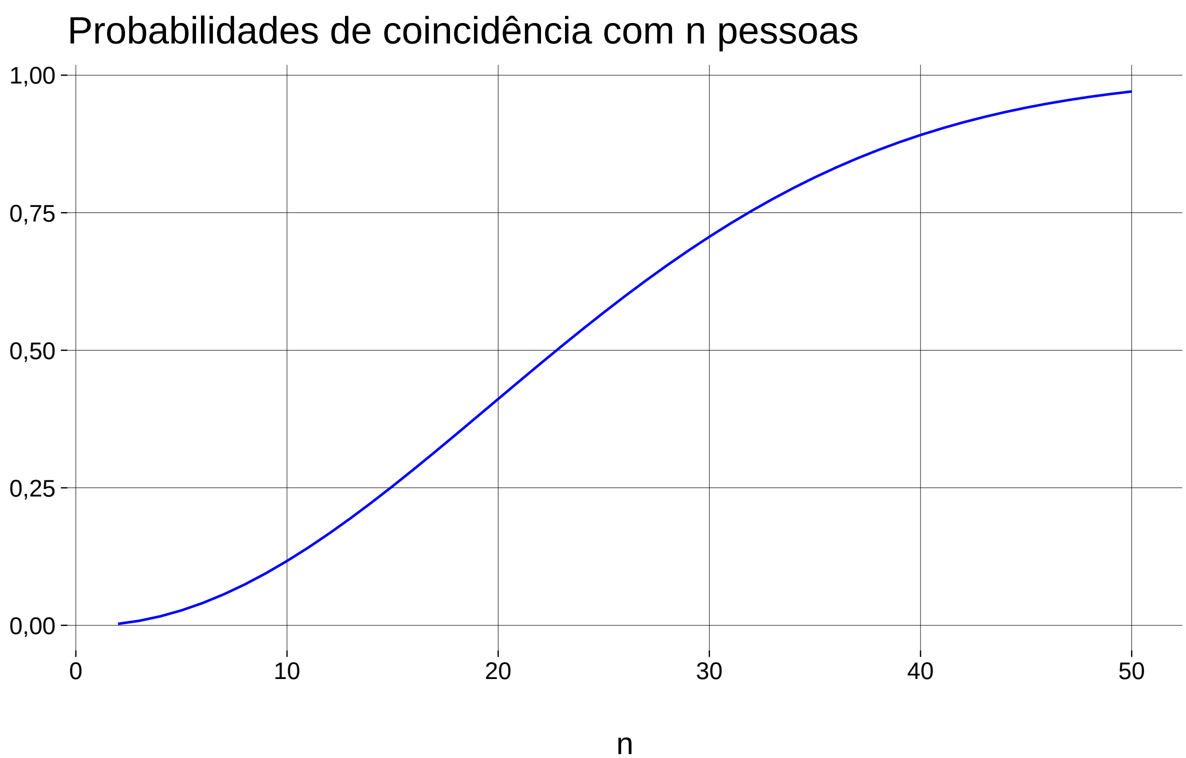
7.11.3

Para diferentes valores de $n \in \{2, 3, \dots, 50\}$

Soluções teóricas

- Vamos calcular as probabilidades de coincidência para diferentes quantidades n de pessoas na sala e fazer um gráfico:

```
npessoas <- 2:50

p <- function(n) {

  # Fórmula geral, para n pessoas
  1 - prod((364:(366 - n))/365)

}

probs <- sapply(npessoas, p)

grafico <- probs %>%
  as_tibble() %>%
  ggplot(aes(x = npessoas, y = value)) +
    geom_line(color = 'blue') +
    labs(
      title = 'Probabilidades de coincidência com n pessoas',
      y = NULL,
      x = 'n'
    )

grafico
```

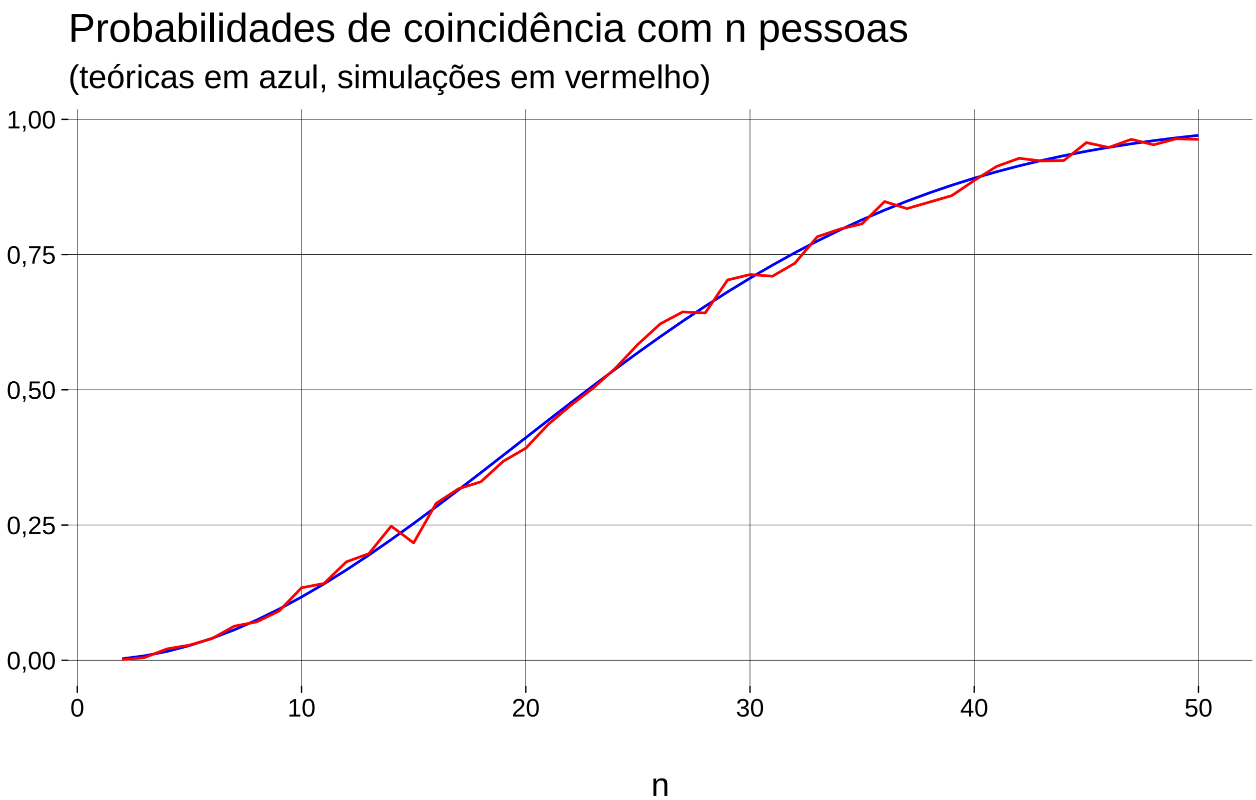


- Este problema é tão usado em cursos de probabilidade que o R oferece as funções `pbirthday` e `qbirthday`.
- Leia a ajuda de `pbirthday` e recrie o gráfico acima usando esta função.
- Leia a ajuda de `qbirthday` e responda:
  - Quantas pessoas são necessárias para que a probabilidade de uma ou mais coincidências seja de pelo menos 50%?
  - Quantas pessoas são necessárias para que a probabilidade de uma ou mais coincidências seja de pelo menos 90%?
  - Quantas pessoas são necessárias para que haja uma probabilidade de pelo menos 50% de que 5 ou mais pessoas façam aniversário no mesmo dia?

Simulação

```
nsalas <- 1e3
npessoas <- 2:50

probs_sim <- sapply(npessoas, simular, nsalas)

grafico +
  geom_line(
    data = as_tibble(probs_sim),
```

```
    mapping = aes(y = value),
    color = 'red'
  ) +
  labs(
    subtitle = '(teóricas em azul, simulações em vermelho)'
  )
```



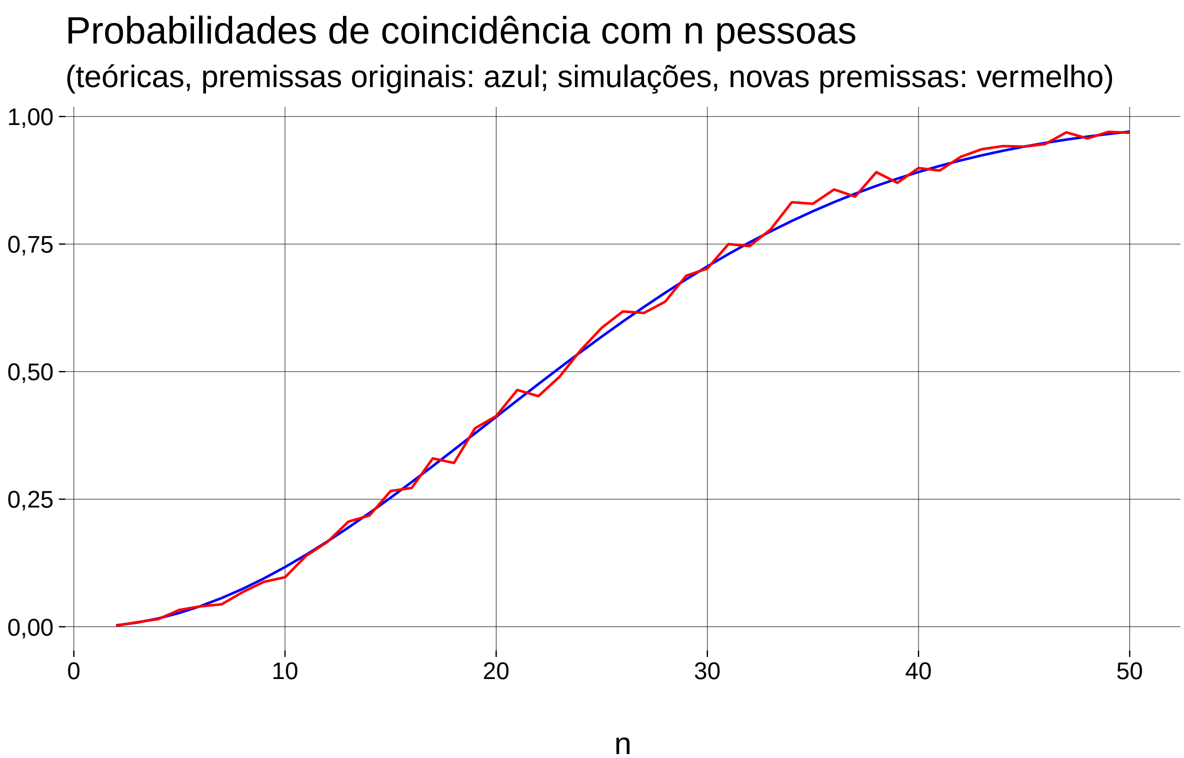
7.11.4 Premissas mais realistas

- Vamos considerar anos bissextos. O total de dias muda para 366, mas um dos dias (29 de fevereiro) tem $1/4$ da probabilidade de um dia normal de ser o aniversário de alguém.
- Além disso, vamos supor que haja 165 dias em que a probabilidade de alguém nascer é 25% maior do que nos 200 dias normais.
- A solução teórica é bem mais complexa do que no caso uniforme!
- Vamos fazer apenas a simulação.
- Preste atenção no vetor `pesos`, que representam as probabilidades de dias diferentes:
  - 200 dias normais têm peso 4;
  - 165 dias mais prováveis têm peso 5;

  - 1 dia (29 de fevereiro) tem peso 1.

- Estes pesos não são probabilidades, porque a soma deles não é 1.
- A função `sample` normaliza automaticamente estes pesos.
- Normalizar significa dividir todos os valores pela mesma constante, de forma que a soma seja 1.

```
nsalas <- 1e3
npessoas <- 2:50

pesos <- c(
  rep(4, 200),      # dias normais
  rep(5, 165),      # dias mais prováveis
  1                 # 29 de fevereiro
)

gerar_e_testar <- function(npessoas, pesos) {

  sala <- sample(1:366, npessoas, replace = TRUE, prob = pesos)
  coincidencia(sala)

}

simular <- function(npessoas, nsalas, pesos) {

  resultados <- replicate(nsalas, gerar_e_testar(npessoas, pesos))
  mean(resultados)

}

novas_probs <- sapply(npessoas, simular, nsalas, pesos)

grafico +
  geom_line(
    data = as_tibble(novas_probs),
    mapping = aes(y = value),
    color = 'red'
  ) +
  labs(
    subtitle = paste(
      '(teóricas, premissas originais: azul;',
      'simulações, novas premissas: vermelho)'
    )
  )
```



- As novas premissas não mudaram muita coisa.
- Escreva a versão normalizada do vetor `pesos`.

7.12 Exercícios

7.12.1 Semanas com mais nascimentos

- Imagine que 50% dos nascimentos de um ano aconteçam em um período de 15 semanas, e o restante dos nascimentos seja distribuído de maneira uniforme no restante do ano. Ignore anos bissextos.
- Faça simulações como na seção anterior ($2 \leq n \leq 50$) e construa o gráfico comparando com as probabilidades teóricas (com as premissas originais).
- Interprete o resultado.

7.12.2

Pôquer

- Uma mão de pôquer consiste de 5 cartas retiradas ao acaso de um baralho de 32 cartas (4 naipes, cada um com cartas 7, 8, 9, 10, J, Q, K, A).
- Calcule as seguintes probabilidades teoricamente e através de simulações.
  1. Qual a probabilidade de obter uma mão sem ases?
  2. Qual a probabilidade de obter 4 ases?
  3. Qual a probabilidade de obter uma sequência (7 a J, 8 a Q, 9 a K, ou 10 a A) de naipes quaisquer?
  4. Qual a probabilidade de obter uma sequência (7 a J, 8 a Q, 9 a K, ou 10 a A) do mesmo naipe?

7.12.3

Dados

- Calcule as seguintes probabilidades teoricamente e através de simulações.
  1. Você lança um dado não-viciado 6 vezes. Qual a probabilidade de que saiam os 6 números?
  2. Idem, se você lançar o dado 10 vezes.

7.13

Vídeo 2

https://youtu.be/NVP-MwtGp0Q

7.14

Probabilidade condicional

- Em um mesmo experimento, saber que um evento B aconteceu pode dar informação sobre um outro evento A.
- Por exemplo, ao lançar um dado, a probabilidade de A — conseguir um 6 — é de $1/6$.
- Se formos informados que o evento B — o lançamento deu um número maior que 3 — ocorreu, então a probabilidade de ter conseguido um 6 passa para $1/3$.
- Escrevemos

$$P(A) = 1/6$$

e

$$P(A \mid B) = 1/3$$

- $P(A \mid B)$ é a probabilidade de A ocorrer, dado que B ocorreu.
- É uma probabilidade condicional. Estamos condicionando sobre B.

7.14.1

Exemplo: Titanic

- A seguinte tabela mostra as quantidades de pessoas no Titanic, categorizadas como sobreviventes ou não, e divididas pela classe:

```
##           Classe
## Sobreviveu    1    2    3 Tripulação Total
##        Não  122  167  528        673  1490
##        Sim  203  118  178        212   711
##        Total 325  285  706        885  2201
```

Probabilidade de ser tripulante

- Escolha uma das 2.201 pessoas ao acaso.
- Qual é a probabilidade de a pessoa escolhida ser um tripulante?
- Esta é uma probabilidade não-condicional: basta dividir o total de tripulantes pelo total de pessoas:

$$P(\text{tripulante}) = \frac{885}{2.201}$$

- A tabela está na variável `tit_tab`. Em R, podemos indexar uma tabela pelos nomes. O primeiro índice corresponde à linha, o segundo à coluna:

```
ptrip <-
  tit_tab['Total', 'Tripulação'] / tit_tab['Total', 'Total']

ptrip
```

```
## [1] 0,40209
```

Probabilidade de não ser tripulante

- Escolha uma das 2.201 pessoas ao acaso.
- Qual é a probabilidade de a pessoa escolhida não ser um tripulante?
- Esta é uma probabilidade não-condicional: basta dividir o total de não-tripulantes pelo total de pessoas:

$$P(\text{não-tripulante}) = \frac{325 + 285 + 706}{2.201}$$

- Em R, podemos selecionar várias células de uma tabela; basta usar um vetor como índice:

```
ntrip <- sum(tit_tab['Total', c('1', '2', '3')])

ntrip / tit_tab['Total', 'Total']
```

```
## [1] 0,59791
```

- Mas nem era preciso fazer este cálculo. Basta perceber que "ser tripulante" e "ser não-tripulante" são eventos complementares. Daí,

$$P(\text{não-tripulante}) = 1 - P(\text{tripulante})$$

```
1 - ptrip
```

```
## [1] 0,59791
```

Probabilidade de sobreviver

- Escolha uma das 2.201 pessoas ao acaso.
- Qual é a probabilidade de a pessoa escolhida ter sobrevivido?
- Esta é uma probabilidade não-condicional: basta dividir o total de sobreviventes pelo total de pessoas:

$$P(\text{sobrevivente}) = \frac{711}{2.201}$$

```
tit_tab['Sim', 'Total'] / tit_tab['Total', 'Total']
```

```
## [1] 0,323035
```

Probabilidade de ser de primeira classe

- Escolha uma das 2.201 pessoas ao acaso.
- Qual é a probabilidade de a pessoa escolhida ser da primeira classe?
- Esta é uma probabilidade não-condicional: basta dividir o total de passageiros da primeira classe pelo total de pessoas:

$$P(\text{1ª classe}) = \frac{325}{2.201}$$

```
tit_tab['Total', '1'] / tit_tab['Total', 'Total']
```

```
## [1] 0,1476602
```

Probabilidade de sobreviver E *ser de primeira classe*

- Escolha uma das 2.201 pessoas ao acaso.
- Qual é a probabilidade de a pessoa escolhida ter sobrevivido e ser da primeira classe?
- Isto é uma probabilidade conjunta — a probabilidade de dois eventos terem ocorrido ao mesmo tempo. Ainda não é uma probabilidade condicional.
- Queremos saber a proporção de pessoas, do total de pessoas a bordo, que eram de primeira classe e sobreviveram.

$$P(\text{sobrevivente da 1ª classe}) = \frac{203}{2.201}$$

```
tit_tab['Sim', '1'] / tit_tab['Total', 'Total']
```

```
## [1] 0,0922308
```

Probabilidade de uma pessoa da primeira classe sobreviver

- Escolha uma das 2.201 pessoas ao acaso.
- Qual é a probabilidade de a pessoa escolhida ter sobrevivido, dado que a pessoa estava na primeira classe?
- Isto é uma probabilidade condicional, escrita como

$$P(\text{sobrevivente} \ \mid \ \text{1ª classe})$$

- Cuidado, agora.
- Já sabemos que a pessoa é da primeira classe. Logo, restringimos o universo a estas 325 pessoas. O denominador vai ser o total de passageiros da primeira classe:

$$P(\text{sobrevivente} \ \mid \ \text{1ª classe}) = \frac{203}{325}$$

```
tit_tab['Sim', '1'] / tit_tab['Total', '1']
```

```
## [1] 0,6246154
```

- Perceba que

$$P(\text{sobreviveu} \mid \text{1ª classe})$$

é o mesmo que

$$\frac{P(\text{sobreviveu} \ \cap \ \text{1ª classe})}{P(\text{1ª classe})}$$

Probabilidade de um sobrevivente ser da primeira classe

- Escolha uma das 2.201 pessoas ao acaso.
- Qual é a probabilidade de a pessoa escolhida ser da primeira classe, dado que ela sobreviveu?
- Isto é outra probabilidade condicional, escrita como

$$P(\text{1ª classe} \ \mid \ \text{sobreviveu})$$

- Não é a mesma probabilidade que $P(\text{sobreviveu} \mid \text{1ª classe})$.

- Em português:
  - $P($sobreviveu $\mid$ 1ª classe$)$ é a probabilidade de
    - A pessoa sobreviver, dado que era da primeira classe;
    - Equivalentemente: alguém da primeira classe sobreviver.
  - $P($1ª classe $\mid$ sobreviveu$)$ é a probabilidade de
    - A pessoa ter sido da primeira classe, dado que sobreviveu;
    - Equivalentemente: alguém que sobreviveu ter sido da primeira classe.
  - Releia até entender.
- Agora, restringimos o universo às pessoas que sobreviveram. Dentre estas, quantas são da primeira classe?

$$P(\text{1ª classe} \mid \text{sobreviveu}) = \frac{203}{711}$$

```
tit_tab['Sim', '1'] / tit_tab['Sim', 'Total']
```

```
## [1] 0,2855134
```

Este é um exemplo de que $P(A \mid B)$ pode ser diferente de $P(B \mid A)$.

7.14.2 Definição de probabilidade condicional

- Como vimos nos exemplos, para calcular $P(A \mid B)$, restringimos o universo aos elementos onde B acontece, e, deste universo, verificamos quantos elementos também correspondem a A acontecer — isto é, elementos onde $A \cap B$ acontece.
- Em termos de frequência relativa:

$$\frac{\text{ocorrências de } A \cap B}{\text{ocorrências de } B}$$

- Em termos de probabilidade, a definição é

$$P(A \mid B) \quad = \quad \frac{P(A \cap B)}{P(B)}$$

7.14.3 Exercícios

No Titanic,

1. Qual a probabilidade de um tripulante sobreviver?
2. Qual a probabilidade de um sobrevivente ser tripulante?
3. Qual a probabilidade de um não-tripulante sobreviver?
4. Qual a probabilidade de um sobrevivente não ser tripulante?
5. Compare as probabilidades condicionais de uma pessoa sobreviver dado que
   a. Ela estava na 1ª classe. (Já calculada no exemplo acima: $0{,}62$.)
   b. Ela estava na 2ª classe.
   c. Ela estava na 3ª classe.
   d. Ela era da tripulação.

   Que conclusões você tira?

7.15 Probabilidade conjunta

- Imagine que queremos calcular a probabilidade de dois eventos A e B acontecerem ao mesmo tempo.
- Ou seja, queremos descobrir a probabilidade conjunta $P(A \cap B)$.
- Muitas vezes, é difícil calcular esta probabilidade diretamente.
- A fórmula para calcular $P(A \mid B)$ nos dá uma maneira de calcular $P(A \cap B)$:

$$P(A \mid B) \;=\; \frac{P(A \cap B)}{P(B)} \quad \Longleftrightarrow \quad P(A \cap B) \;=\; P(A \mid B) \cdot P(B)$$

- Ou, invertendo A e B,

$$P(B \mid A) \;=\; \frac{P(A \cap B)}{P(A)} \quad \Longleftrightarrow \quad P(A \cap B) \;=\; P(B \mid A) \cdot P(A)$$

- Em palavras:
  - A probabilidade de A e B acontecerem juntos é a probabilidade de A dado que B aconteceu, multiplicada pela probabilidade de B.
  - Ou, invertendo A e B, a probabilidade de A e B acontecerem juntos é a probabilidade de B dado que A aconteceu, multiplicada pela probabilidade de A.

7.16 Independência

- Mais acima, vimos que, para dois eventos independentes A e B, a probabilidade conjunta $P(A \cap B)$ é igual a $P(A) \cdot P(B)$.
- Olhando para as fórmulas acima para a probabilidade conjunta, se A e B forem independentes, então

$$
\begin{aligned}
P(A \cap B) &= P(A \mid B) \cdot P(B) \\
&= P(A) \cdot P(B)
\end{aligned}
$$

  o que nos diz que

$$
P(A \mid B) = P(A)
$$

- Da mesma forma,

$$
P(B \mid A) = P(B)
$$

- Em palavras: saber que um dos eventos ocorreu não altera a probabilidade do outro evento.
- Qualquer uma das 3 igualdades pode ser tomada como a definição formal de eventos independentes.

7.16.1 Exemplos

Estar na primeira classe e sobreviver são independentes?

- A probabilidade de sobreviver, dado que a pessoa estava na 1ª classe, é

$$
P(\text{sobreviver} \mid \text{1ª classe}) = \frac{203}{325}
$$

```
tit_tab['Sim', '1'] / tit_tab['Total', '1']
```

```
## [1] 0,6246154
```

- Mas a probabilidade (incondicional) de sobreviver é

$$P(\text{sobreviver}) = \frac{711}{2.201}$$

```
tit_tab['Sim', 'Total'] / tit_tab['Total', 'Total']
```

```
## [1] 0,323035
```

- Como as probabilidades são diferentes, os eventos não são independentes.
- Verifique se $P(\text{sobreviver} \cap \text{1ª classe}) = P(\text{sobreviver}) \cdot P(\text{1ª classe})$.

Faltas e turno de trabalho

- Numa empresa:
  - 75 funcionários trabalham no turno diurno, com um número de faltas de 3 por semana.
  - 25 funcionários trabalham no turno noturno, com um número de faltas de 1 por semana.
  - Faltar é independente do turno de trabalho?
- Vamos construir uma tabela:

```
faltas <- array(
  c(3, 1, 72, 24),
  dim = c(2, 2)
) %>%
  addmargins()

dimnames(faltas) = list(
    'Turno' = c('Diurno', 'Noturno', 'Total'),
    'Presença' = c('Faltou', 'Presente', 'Total')
  )

faltas
```

```
##          Presença
## Turno      Faltou Presente Total
##   Diurno        3       72    75
##   Noturno       1       24    25
##   Total         4       96   100
```

- A probabilidade (incondicional) de faltar é

$$P(\text{Faltou}) = \frac{4}{100}$$

```
faltas['Total', 'Faltou'] / faltas['Total', 'Total']
```

```
## [1] 0,04
```

- A probabilidade de faltar no turno diurno é

$$P(\text{Faltou} \mid \text{Diurno}) = \frac{3}{75}$$

```
faltas['Diurno', 'Faltou'] / faltas['Diurno', 'Total']
```

```
## [1] 0,04
```

- Como as probabilidades são iguais, os eventos são independentes.
- Verifique que $P(\text{Faltou}) = P(\text{Faltou} \mid \text{Noturno})$.

7.17 Probabilidade total

7.17.1 Exemplo

- Dentre 80 homens, 30 têm olhos azuis.
- Dentre 50 mulheres, 20 têm olhos azuis.
- Neste grupo de pessoas, qual a probabilidade de ter olhos azuis?
- Homens e mulheres formam uma partição deste grupo — i.e., cada pessoa só pode ser homem ou mulher (não ambos) e cada pessoa precisa ser homem ou mulher.
- O evento "ter olhos azuis" se subdivide em dois casos mutuamente exclusivos:
  1. Ter olhos azuis e ser homem;
  2. Ter olhos azuis e ser mulher.
- Vamos chamar os eventos de
  - A = ter olhos azuis
  - H = ser homem
  - M = ser mulher
- Calculamos a probabilidade $P(A)$ somando as probabilidades dos dois casos:

$$P(A) = P(A \cap H) + P(A \cap M)$$

- Então,

$$P(A) = \frac{30}{130} + \frac{20}{130} = \frac{50}{130} \approx 0{,}39$$

- Ou, antes de somar, podemos transformar as probabilidades conjuntas em produtos de uma probabilidade condicional por uma probabilidade não-condicional:

$$\begin{aligned} P(A) &= P(A \cap H) + P(A \cap M) \\ &= P(A \mid H)P(H) + P(A \mid M)P(M) \end{aligned}$$

- O que nos dá o mesmo resultado:

$$\begin{aligned} P(A) &= \frac{30}{80} \cdot \frac{80}{130} + \frac{20}{50} \cdot \frac{50}{130} \\ &= \frac{50}{130} \\ &\approx 0{,}39 \end{aligned}$$

7.18

Teorema de Bayes

7.18.1

Exemplo

- De todos os *emails*, 60% são *spam*:

$$P(\text{spam}) = 0{,}6$$

- De todos os *emails* que são *spam*, 90% contêm a palavra “compre”:

$$P(\text{compre} \mid \text{spam}) = 0{,}9$$

- De todos os *emails* (*spam* ou não), 70% contêm a palavra “compre”:

$$P(\text{compre}) = 0{,}7$$

- Você acaba de receber um *email*. Antes de você abri-lo, qual a probabilidade de o *email* ser *spam*?
- Bem, na ausência de informação adicional, $P(\text{spam}) = 0{,}6$.
- Você abre o *email*. Ele contém a palavra “compre”.

- Agora, qual a probabilidade de ser *spam*?

$$P(\text{spam} \mid \text{compre}) = ?$$

- Lembre-se de que

$$P(\text{compre} \cap \text{spam}) = P(\text{spam} \mid \text{compre}) \cdot P(\text{compre})$$

- Mas também

$$P(\text{compre} \cap \text{spam}) = P(\text{compre} \mid \text{spam}) \cdot P(\text{spam})$$

- As duas expressões são iguais:

$$P(\text{spam} \mid \text{compre}) \cdot P(\text{compre}) = P(\text{compre} \mid \text{spam}) \cdot P(\text{spam})$$

- Isolando o termo que queremos descobrir:

$$P(\text{spam} \mid \text{compre}) = \frac{P(\text{compre} \mid \text{spam}) \cdot P(\text{spam})}{P(\text{compre})}$$

- Substituindo os valores:

$$P(\text{spam} \mid \text{compre}) = \frac{0{,}9 \cdot 0{,}6}{0{,}7} \approx 0{,}77$$

- Isto é inferência bayesiana:
  1. Começamos com uma probabilidade não-condicional: *a priori*, $P(\text{spam}) = 0{,}6$;
  2. Obtivemos nova informação: o *email* contém "compre";
  3. Usamos esta informação para calcular uma probabilidade condicional, *a posteriori*:

$$P(\text{spam} \mid \text{compre}) = \frac{P(\text{compre} \mid \text{spam}) \cdot P(\text{spam})}{P(\text{compre})}$$

- Perceba que, para isso, precisamos da probabilidade não-condicional $P(\text{compre})$ (no denominador).

7.18.2

No geral

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

- E se você não souber $P(B)$?
- Use o teorema da probabilidade total:

$$P(B) = P(B \cap A_1) + P(B \cap A_2) + \cdots + P(B \cap A_n)$$

  onde os A_i formam uma partição da população.

- Isto equivale a

$$P(B) = P(B \mid A_1)P(A_1) + P(B \mid A_2)P(A_2) + \cdots + P(B \mid A_n)P(A_n)$$

7.18.3

Outro exemplo

- Uma doença rara afeta 5 pessoas em 100.000.
- O exame que detecta a doença tem precisão de $99{,}9\%$; i.e., quando uma pessoa está doente, o exame dá positivo $99{,}9\%$ das vezes.
- Quando uma pessoa *não* está doente, o exame dá positivo 1% das vezes. Este caso é um falso positivo.
- Você faz o exame, e o resultado é positivo.
- Dado este resultado, qual a probabilidade de você ter a doença?
- Vamos nomear os eventos:
  - $D =$ você está doente;
  - $ND =$ você não está doente;
  - $+ =$ o exame deu positivo;
  - $- =$ o exame deu negativo.
- Vamos usar Bayes:

$$P(D \mid +) = \frac{P(+ \mid D)P(D)}{P(+)}$$

- $P(+ \mid D) = 0{,}999$, pelo enunciado.
- $P(D) = 0{,}00005$, pelo enunciado.

- Daí, $P(ND) = 1 - P(D) = 0{,}99995$.
- Não temos $P(+)$, mas podemos achar usando o teorema da probabilidade total, lembrando que o enunciado diz que $P(+ \mid ND) = 0{,}01$:

$$\begin{aligned} P(+) &= P(+ \cap D) \ + \ P(+ \cap ND) \\ &= P(+ \mid D)P(D) \ + \ P(+ \mid ND)P(ND) \\ &= 0{,}999 \cdot 0{,}00005 \ + \ 0{,}01 \cdot 0{,}99995 \\ &= 0{,}01004945 \end{aligned}$$

- Inserindo os valores no teorema de Bayes:

$$\begin{aligned} P(D \mid +) &= \frac{P(+ \mid D)P(D)}{P(+)} \\ &= \frac{0{,}999 \cdot 0{,}00005}{0{,}01004945} \\ &= 0{,}00497 \end{aligned}$$

- A probabilidade de estar doente é menos do que $0{,}5\%$!
- Você provavelmente esperava uma probabilidade maior.
- Qual das 3 probabilidades usadas no cálculo fez o resultado ser tão pequeno?
  - $P(+ \mid D)$?
  - $P(D)$?
  - $P(+)$?

CAPÍTULO 8

Variáveis aleatórias

8.1 Vídeo

https://youtu.be/8PI-rfsgNdE

8.2 O que é uma variável aleatória?

- Uma variável aleatória é uma maneira de associar a cada resultado do espaço amostral um número real.
- Dependendo do conjunto de números, a variável aleatória pode ser discreta ou contínua.
- Falamos sobre a probabilidade de uma variável aleatória assumir um valor (ou uma faixa de valores).

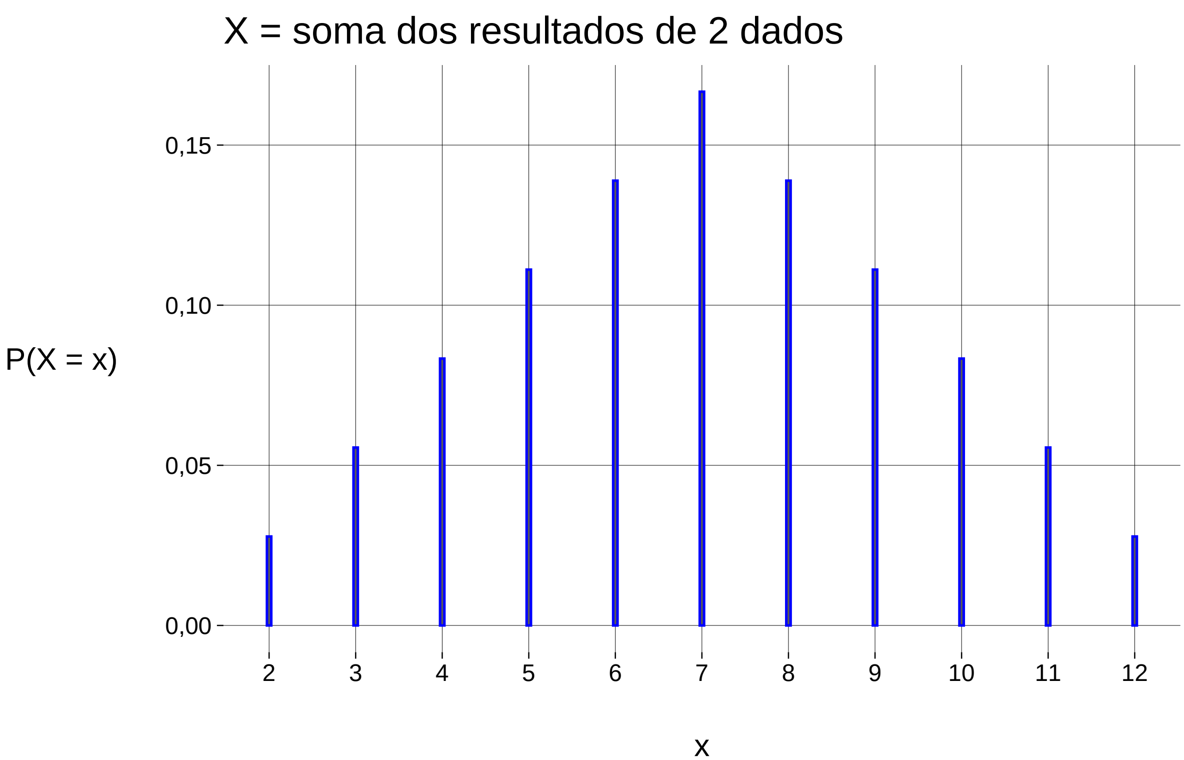
8.3 Exemplos

8.3.1 Lançar dois dados

- Definimos $X =$ soma dos resultados dos dois dados.
- Esta é uma variável aleatória discreta, com 11 valores possíveis.
- A tabela com todos os valores possíveis de X e suas probabilidades é chamada de distribuição de probabilidade:

x	P(X = x)
2	1/36
3	2/36
4	3/36
5	4/36
6	5/36
7	6/36
8	5/36
9	4/36
10	3/36
11	2/36
12	1/36

- Graficamente:



- Suponha que a distribuição de probabilidade de X esteja na seguinte *tibble*:

```
glimpse(dados_distr)
```

```
## Rows: 11
## Columns: 3
## $ x          <int> 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
## $ `P(X = x)` <chr> "1/36", "2/36", "3/36", "4/36", "5/36", "6/36", "5~
## $ num        <dbl> 0,02777778, 0,05555556, 0,08333333, 0,11111111, 0,~
```

A coluna `num` tem os valores numéricos das probabilidades.

- Qual a probabilidade de conseguir 10 ou mais?

  Basta somar as probabilidades de $X = 10$, $X = 11$ e $X = 12$:

```
dados_distr %>%
  filter(x >= 10) %>%
  pull(num) %>%
  sum()
```

```
## [1] 0,1666667
```

- Qual a probabilidade de conseguir entre 6 e 8, inclusive?

```
dados_distr %>%
  filter(x >= 6 & x <= 8) %>%
```

```
  pull() %>%
  sum()
```

```
## [1] 0,4444444
```

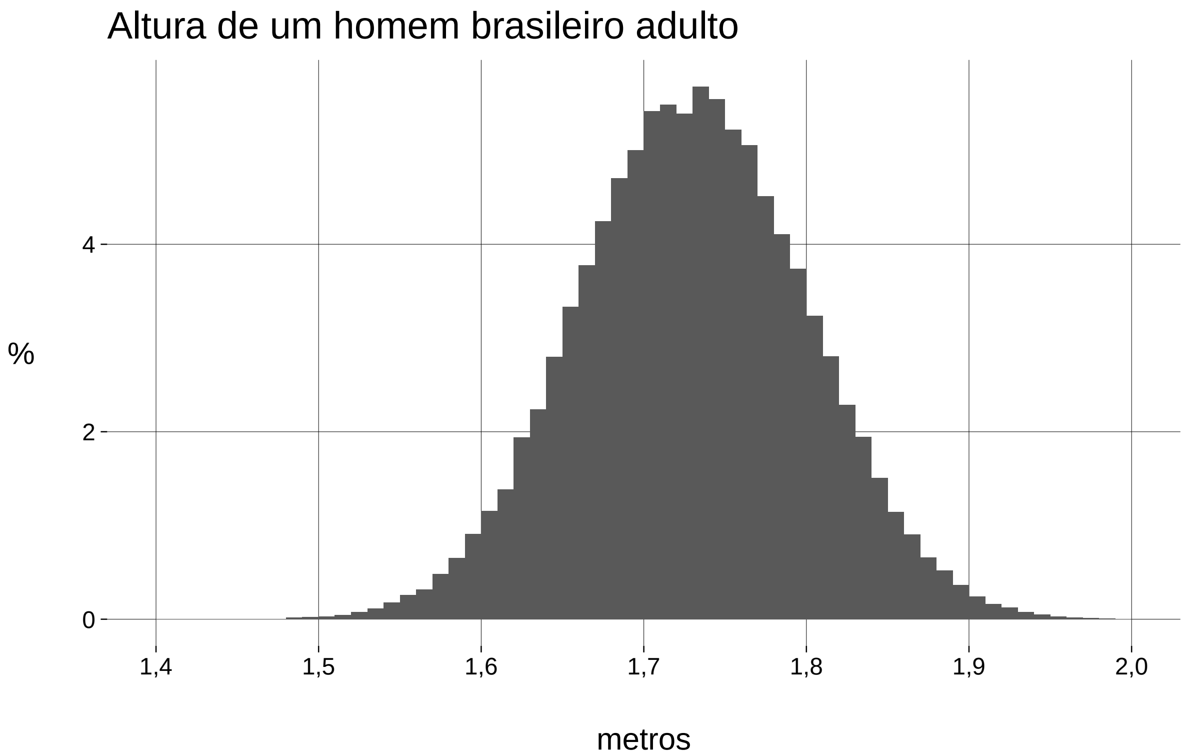
8.3.2 Altura de um homem adulto

- Definimos $X =$ estatura em metros de um homem brasileiro adulto, escolhido ao acaso.
- Esta é uma variável aleatória contínua, com um número infinito não-enumerável de valores.
- Segundo o Levantamento do Perfil Antropométrico da População Brasileira Usuária do Transporte Aéreo Nacional, em 2009, a estatura média do homem brasileiro adulto era de $1{,}73$m, com desvio-padrão de $0{,}07$m.
- Vamos simular uma amostra de muitos homens desta população:

```
media <- 1.73
desvio <- 0.07
homens <- tibble(
  altura = rnorm(1e5, mean = media, sd = desvio)
)
```

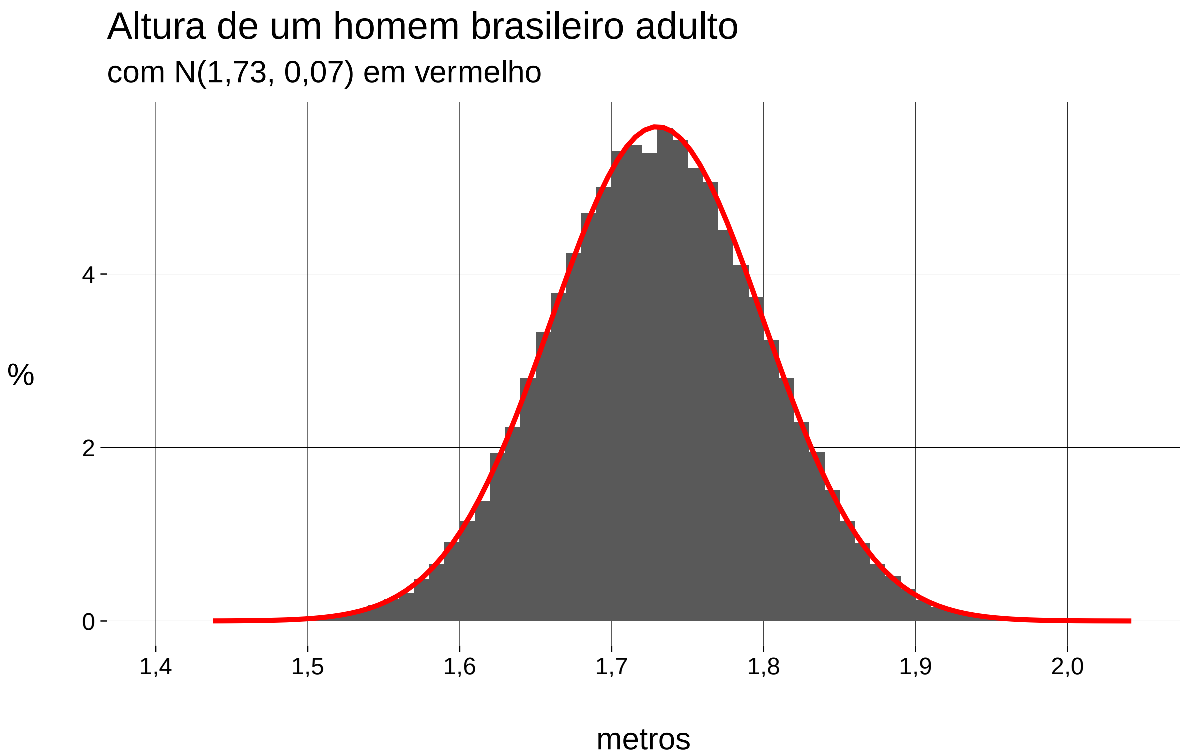
- Eis um histograma com as percentagens:

```
homens_plot <- homens %>%
  ggplot(aes(x = altura)) +
    geom_histogram(
      aes(y = after_stat(density)),
      breaks = seq(1.4, 2, 0.01)
    ) +
    scale_x_continuous(breaks = seq(1.4, 2.0, .1)) +
    labs(
      title = 'Altura de um homem brasileiro adulto',
      x = 'metros',
      y = '%'
    )

homens_plot
```



- Agora, sobrepomos o gráfico de uma distribuição normal com a mesma média e o mesmo desvio-padrão que a distribuição das alturas:

```
homens_normal <- homens_plot +
  stat_function(
    fun = dnorm,
    args = list(
      'mean' = media,
      'sd' = desvio
    ),
    geom = 'line',
    color = 'red',
    linewidth = 1
  ) +
  labs(
    subtitle = paste0('com N(', media, ', ', desvio,') em vermelho')
  )

homens_normal
```



- A curva vermelha no gráfico é a função de densidade de probabilidade da distribuição normal, dada por

$$\text{fdp}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

> Em uma distribuição contínua, não faz sentido perguntar o valor de $P(X = x)$. Como X pode assumir uma quantidade infinita não-enumerável de valores, esta probabilidade é igual a zero!
>
> Com uma distribuição contínua, vamos sempre perguntar sobre faixas de valores.

- Qual a probabilidade de um homem ter mais de $1{,}80$m?
  - Na amostra:

```
mean(homens$altura > 1.80)
```

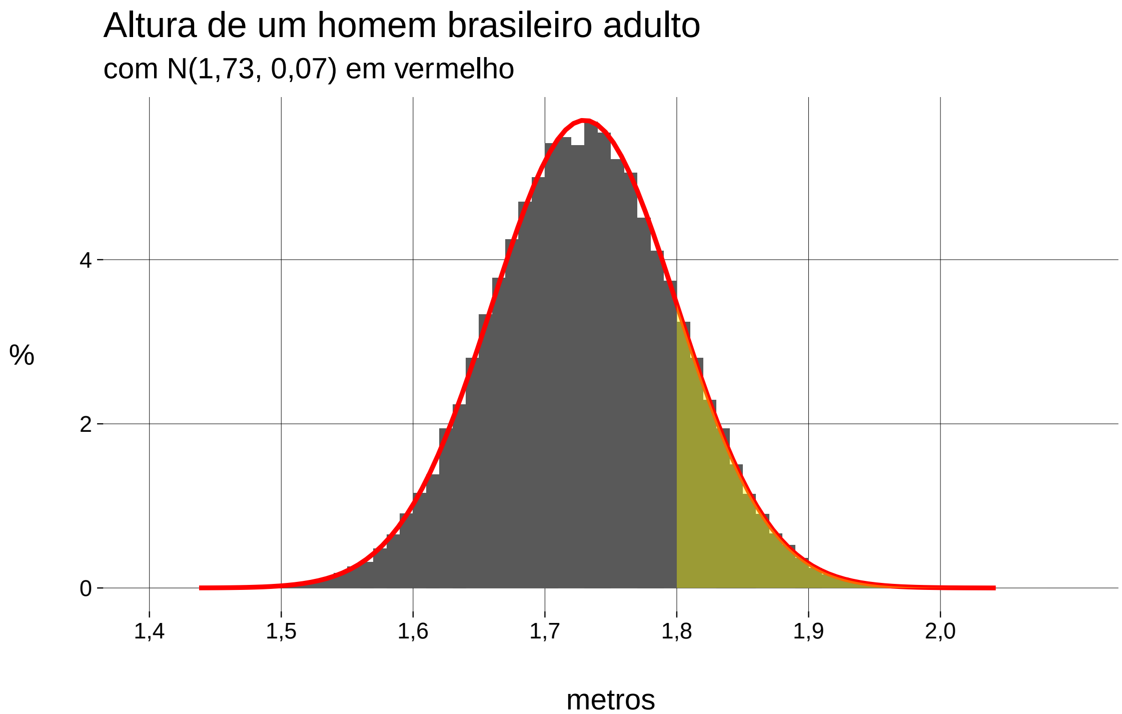
```
## [1] 0,16123
```

  - Na distribuição teórica:

```
pnorm(1.80, mean = media, sd = desvio, lower.tail = FALSE)
```

```
## [1] 0,1586553
```

- No gráfico:



- Qual a probabilidade de um homem ter entre $1{,}60$m e $1{,}70$m?
  - Na amostra:

```
mean(homens$altura > 1.60 & homens$altura < 1.70)
```

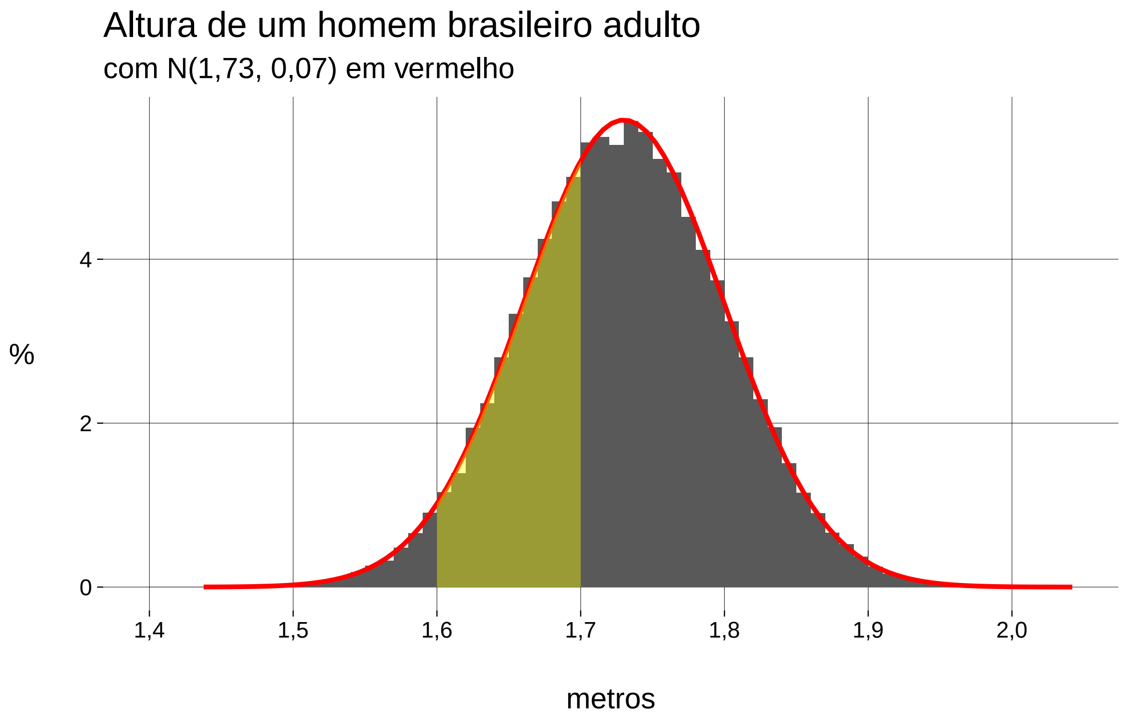
```
## [1] 0,30585
```

  - Na distribuição teórica:

```
pnorm(1.70, mean = media, sd = desvio) -
pnorm(1.60, mean = media, sd = desvio)
```

```
## [1] 0,3024722
```

  - No gráfico:



8.4 Valor esperado

- O valor esperado (ou esperança matemática) de uma variável aleatória é a média ponderada dos valores possíveis da variável, considerando as probabilidades (ou, no caso contínuo, a densidade de probabilidade) como pesos.
- No caso discreto:

$$\mu = E(X) = \sum_i x_i \cdot P(X = x_i)$$

- No caso contínuo:

$$\mu = E(X) = \int_{-\infty}^{+\infty} x \cdot \textsf{fdp}(x)\textsf{d}x$$

8.4.1

Lançar dois dados

- Lembrando que a *tibble* `dados_distr` contém a distribuição de probabilidades do valor da soma de dois dados, o valor esperado é

```
dados_distr %>%
  summarize(E = sum(x * num)) %>%
  pull(E)
```

```
## [1] 7
```

8.4.2

Lançar um dado

- O valor esperado do valor obtido em um lançamento de um dado não-viciado (onde cada valor tem a probabilidade $1/6$) é

```
lado <- 1:6
p <- 1/6
sum(lado * p)
```

```
## [1] 3,5
```

8.4.3

Altura de um homem adulto

- Estimamos o valor esperado da população simplesmente calculando a média da amostra:

```
mean(homens$altura)
```

```
## [1] 1,730044
```

- Se a variável aleatória X é normalmente distribuída, com média μ e desvio-padrão σ, i.e., $X \sim \mathcal{N}(\mu, \sigma)$, então o valor esperado $E(X)$ é igual a μ, que é o valor da integral

$$\int_{-\infty}^{+\infty} x \cdot \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \mathrm{d}x$$

8.5

Propriedades do valor esperado

- Vamos ver como o valor esperado se comporta.
- O valor esperado de uma constante é ela mesma:

$$E(c) = c$$

- Somar uma constante à variável X altera $E(X)$ pelo valor da constante:

$$E(X \pm c) = E(X) \pm c$$

- Multiplicar a variável X por uma constante multiplica $E(X)$ pelo valor da constante:

$$E(c \cdot X) = c \cdot E(X)$$

- O valor esperado da soma (resp. diferença) de duas variáveis aleatórias é a soma (resp. diferença) dos valores esperados:

$$E(X \pm Y) = E(X) \pm E(Y)$$

- O valor esperado de uma função $f(X)$ de uma variável aleatória X é
  - Caso discreto:

$$E(f(X)) = \sum_i f(x_i) \cdot P(X = x_i)$$

  - Caso contínuo:

$$E(f(X)) = \int_{-\infty}^{+\infty} f(x) \cdot \text{fdp}(x)\text{d}x$$

8.6

Variância

- A variância de uma variável aleatória X é a média ponderada dos desvios quadrados, com as probabilidades como peso.
  - Caso discreto:

$$\sigma^2(X) = \sum_i (x_i - \mu)^2 \cdot P(X = x_i)$$

- Caso contínuo:

$$\sigma^2(X) = \int_{-\infty}^{+\infty} (x-\mu)^2 \cdot \text{fdp}(x)\text{d}x$$

- Em qualquer caso,

$$\begin{aligned}\sigma^2(X) &= E[(X-\mu)^2] \\ &= E(X^2) - [E(X)]^2\end{aligned}$$

- Faça as contas, partindo de $E[(X-\mu)^2]$ e usando as propriedades do valor esperado para chegar a $E(X^2) - [E(X)]^2$.

8.6.1

Lançar dois dados

- A variância é

```
dados_distr %>%
  summarize(s2 = sum((x - 7)^2 * num)) %>%
  pull(s2)
```

```
## [1] 5,833333
```

8.6.2

Lançar um dado

- A variância é

```
lado <- 1:6
p <- 1/6
sum((lado - 3.5)^2 * p)
```

```
## [1] 2,916667
```

8.6.3

Altura de um homem adulto

- Estimando pela variância da amostra:

```
var(homens$altura)
```

```
## [1] 0,004917532
```

- Se X é normalmente distribuída com média μ e desvio-padrão σ, i.e., $X \sim \mathcal{N}(\mu, \sigma)$, então $\sigma^2(X) = \sigma^2$. De acordo com o estudo, $\sigma^2 = 0.07^2 = 0{,}0049$.

8.7 Propriedades da variância

- A variância de uma constante é zero:

$$\sigma^2(c) = 0$$

- Somar uma constante à variável X não altera a variância:

$$\sigma^2(X \pm c) = \sigma^2(X)$$

- Multiplicar a variável X por uma constante multiplica a variância pelo quadrado da constante:

$$\sigma^2(c \cdot X) = c^2 \cdot \sigma^2(X)$$

- A variância da soma ou diferença de duas variáveis aleatórias é a soma das variâncias das variáveis:

$$\sigma^2(X \pm Y) = \sigma^2(X) + \sigma^2(Y)$$

Por que a variância da diferença é a soma das variâncias?

Variância significa incerteza.

Considere o seguinte exemplo para entender por que $\sigma^2(X - Y) = \sigma^2(X) + \sigma^2(Y)$:

- Você compra uma caixa de 500g de cereal no mercado. Como o peso não é exato, considere que X é a variável aleatória que representa o peso do cereal na caixa, com $\mu_X = 500$g e uma variância qualquer σ_X^2 (que corresponde à incerteza do processo de embalagem do cereal na fábrica).
- Você decide comer 100g de cereal, despejando parte do conteúdo na caixa em uma tigela. Como sua capacidade de medir 100g não é exata, considere que Y é a variável aleatória que representa o peso do cereal na tigela, com $\mu_Y = 100$g e uma variância qualquer σ_Y^2 (que corresponde à incerteza do seu processo de pesar 100g).
- Considere a variável aleatória $Z = X - Y$, que representa a quantidade de cereal que sobrou na caixa.
    * Certamente a média $\mu_Z = \mu_X - \mu_Y = 400$g.
    * E a variância σ_Z^2?
    * O fato de Z ser o resultado da subtração de duas variáveis aleatórias diminui a incerteza?
    * Ou a composição de incertezas aumenta a incerteza?

8.8 Mais exemplos

8.8.1 Seguradora

- Você tem uma seguradora, com 1.000 segurados, cada um deles pagando 50 dinheiros por ano.
- Por ano, 1 dos 1.000 segurados morre. Neste caso, sua seguradora deve pagar 10.000 dinheiros.
- Por ano, 2 dos 1.000 segurados ficam inválidos. Neste caso, sua seguradora deve pagar 5.000 dinheiros.
- Quanto sua seguradora espera ter de lucro (ou prejuízo) por segurado, por ano?

- Chamando de X a variável aleatória que representa o dinheiro pago pela seguradora por apólice, por ano, temos

$$\begin{aligned} P(X = 10000) &= 1/1000 \\ P(X = 5000) &= 2/1000 \\ P(X = 0) &= 997/1000 \end{aligned}$$

- Daí,

$$\begin{aligned} E(X) &= 10000 \cdot \frac{1}{1000} \;+\; 5000 \cdot \frac{2}{1000} \;+\; 0 \cdot \frac{997}{1000} \\ &= 20 \end{aligned}$$

- Como cada segurado paga 50 dinheiros por ano, sua seguradora lucra, em média, 30 dinheiros por apólice, por ano.

- E o desvio-padrão?
  - Calculando a variância antes:

$$\begin{aligned} \sigma^2(X) &= (10000 - 20)^2 \cdot \frac{1}{1000} \;+\; (5000 - 20)^2 \cdot \frac{2}{1000} \;+\; (0 - 20)^2 \cdot \frac{997}{1000} \\ &= 9980^2 \cdot \frac{1}{1000} \;+\; 4980^2 \cdot \frac{2}{1000} \;+\; (-20)^2 \cdot \frac{997}{1000} \\ &= 149600 \end{aligned}$$

  - O desvio-padrão é a raiz quadrada de σ^2:

$$\sigma = 386{,}78$$

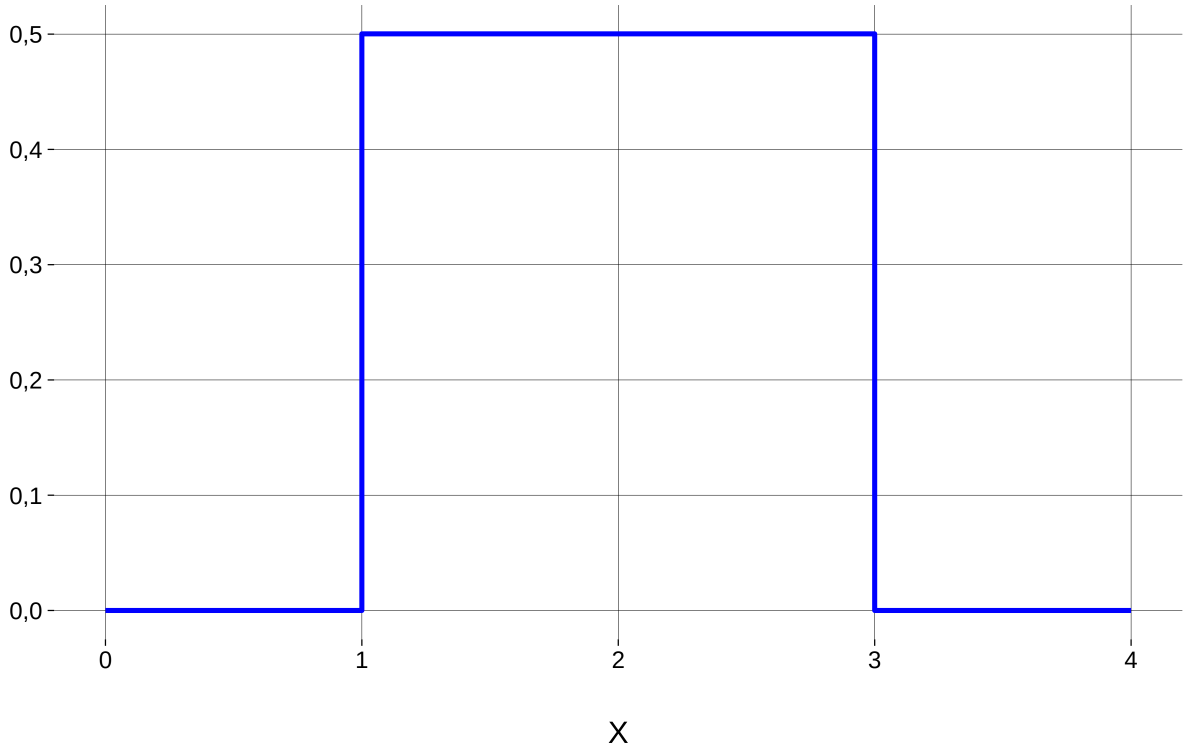
8.8.2

Gerador de números aleatórios

- Você programa um gerador de números aleatórios $x \in [1, 3] \subset \mathbb{R}$.
- Considere X a variável aleatória que representa os números gerados.
- X é uma variável aleatória contínua, com fdp

$$f(x) = \begin{cases} \frac{1}{2} & \text{se } x \in [1, 3] \\ 0 & \text{se } x \notin [1, 3] \end{cases}$$

cujo gráfico é



- Isto significa que a densidade de probabilidade está distribuída uniformemente no intervalo $[1, 3]$.
- Vamos calcular o valor esperado $E(X)$:

$$
\begin{aligned}
E(X) &= \int_{-\infty}^{+\infty} x \cdot f(x) \mathsf{d}x \\
&= \int_{1}^{3} x \cdot \frac{1}{2} \mathsf{d}x \\
&= \frac{1}{2} \cdot \frac{x^2}{2} \bigg|_{1}^{3} \\
&= 2
\end{aligned}
$$

- Ou seja, a média dos números gerados, depois de muitas execuções, vai ser 2.
- Vamos calcular a variância $\sigma^2(X)$:

$$
\begin{aligned}
\sigma^2(X) &= \int_{-\infty}^{+\infty} (x-2)^2 \cdot f(x)\mathsf{d}x \\
&= \int_{1}^{3} (x-2)^2 \cdot \frac{1}{2}\mathsf{d}x \\
&= \frac{1}{3}
\end{aligned}
$$

- Isto vai dar um desvio-padrão $\sigma = \sqrt{\sigma^2} = \frac{\sqrt{3}}{3} \approx 0{,}58$.
- Mas R tem este gerador! Vamos gerar muitos valores e calcular a média e o desvio-padrão:

```
valores <- runif(n = 1e6, min = 1, max = 3)
mean(valores)
```

```
## [1] 1,999605
```

```
sd(valores)
```

```
## [1] 0,5771331
```

- Como exercício, verifique que, para qualquer variável aleatória contínua X distribuída uniformemente entre a e b, i.e., com fdp

$$
f(x) = \begin{cases} \dfrac{1}{b-a} & \text{se } x \in [a,b] \\ 0 & \text{se } x \notin [a,b] \end{cases}
$$

ocorre que

  - $E(X) = \frac{a+b}{2}$, e
  - $\sigma^2(X) = \frac{(a-b)^2}{12}$.

CAPÍTULO 9

Distribuições discretas

9.1 Vídeo 1

https://youtu.be/jKSRgZdlTEM

9.2 Distribuição uniforme discreta

9.2.1 Exemplo: um dado

- Cada resultado de 1 a 6 tem a mesma probabilidade de ocorrer.
- A variável aleatória X representa o número que resulta de um lançamento.
- O suporte (conjunto de valores possíveis) de X é o conjunto $\{1, 2, 3, 4, 5, 6\}$.
- A distribuição de probabilidade de X é uniforme discreta com 6 valores, escrita como UnifDiscr$(X \mid n = 6)$:

$$P(X = x) = \begin{cases} 1/6 & \text{se } x \in \{1, 2, 3, 4, 5, 6\} \\ 0 & \text{senão} \end{cases}$$

9.2.2

No geral

- Com n valores possíveis, a distribuição é UnifDiscr$(X = x \mid n)$.
- $n \in \mathbb{N}^+$.
- Probabilidades:

$$P(X = x) = \begin{cases} 1/n & \text{se } x \in \{1, \dots, n\} \\ 0 & \text{senão} \end{cases}$$

- Valor esperado:

$$\begin{aligned} E(X) &= \frac{1}{n} \cdot (1 + \cdots + n) \\ &= \frac{1}{n} \cdot \frac{n(n+1)}{2} \\ &= \frac{n+1}{2} \end{aligned}$$

- Variância:

$$
\begin{aligned}
\sigma^2(X) &= \sum\left[(x_i-\mu)^2\cdot\frac{1}{n}\right] && \text{(definição de variância)}\\
&= \frac{1}{n}\sum(x_i-\mu)^2\\
&= \frac{1}{n}\left[\sum(x_i^2-2\mu x_i+\mu^2)\right]\\
&= \frac{1}{n}\left[\sum x_i^2-2\mu\sum x_i+n\mu^2\right]\\
&= \frac{1}{n}\sum x_i^2-2\mu\frac{\sum x_i}{n}+\frac{n\mu^2}{n} && \left(\text{mas } \frac{\sum x_i}{n}=\mu\right)\\
&= \frac{1}{n}\sum x_i^2-2\mu^2+\mu^2\\
&= \frac{1}{n}\sum x_i^2-\mu^2 && \left(\text{vamos usar a fórmula para } \sum x_i^2\right)\\
&= \frac{1}{n}\cdot\frac{n(n+1)(2n+1))}{6}-\mu^2 && \text{(vamos substituir } \mu \text{ pela fórmula)}\\
&= \frac{1}{n}\cdot\frac{n(n+1)(2n+1))}{6}-\left(\frac{n+1}{2}\right)^2 && \text{(o resto é contarada)}\\
&= (n+1)\left(\frac{2n+1}{6}-\frac{n+1}{4}\right)\\
&= (n+1)\left(\frac{4n+2-3n-3}{12}\right)\\
&= \frac{(n+1)(n-1)}{12}\\
&= \frac{n^2-1}{12}
\end{aligned}
$$

9.2.3 Em R

> ⚠ As funções `dunif`, `punif`, `qunif` e `runif` trabalham com a distribuição uniforme contínua. Não servem para a distribuição uniforme discreta.

- Para definir os valores possíveis da variável aleatória X, use um vetor.

```
x <- 1:6
```

- Todas as probabilidades são iguais a $\frac{1}{n}$:

```
probs <- 1 / length(x)
```

- Distribuição:

```
distr <- tibble(
  X = x,
  p = probs
)

distr %>% gt()
```

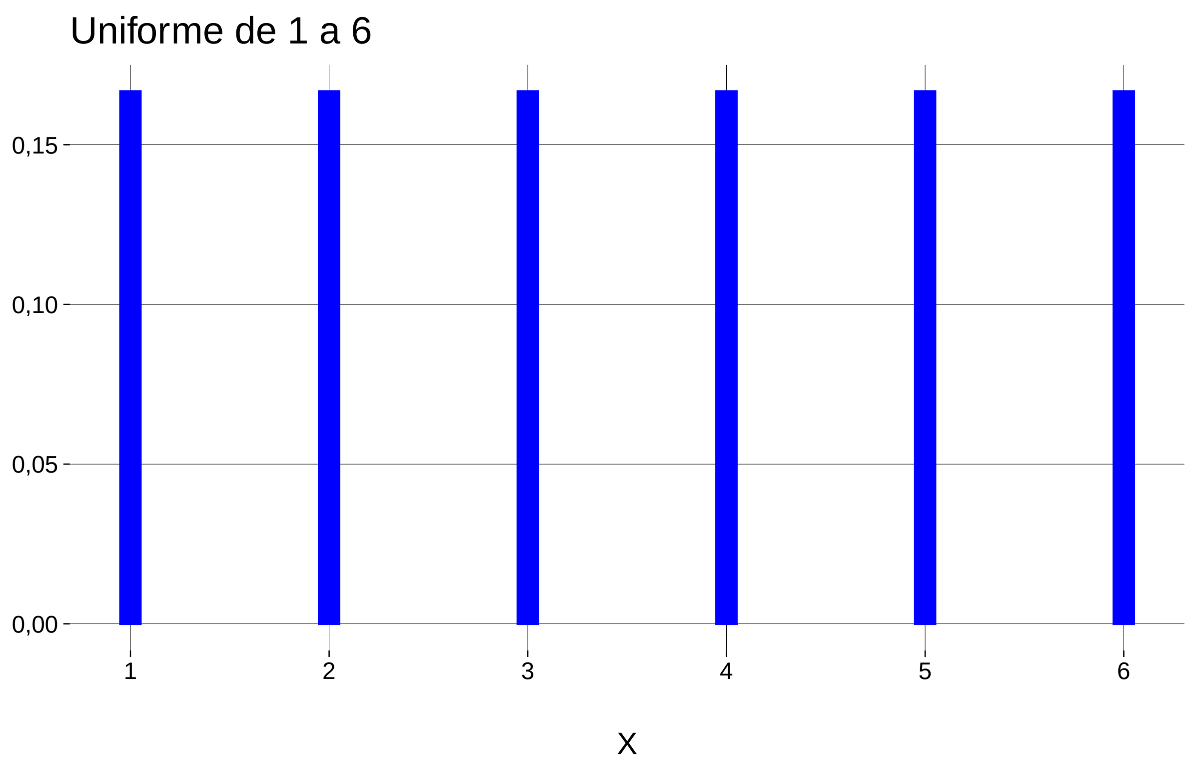
X	p
1	0,1666667
2	0,1666667
3	0,1666667
4	0,1666667
5	0,1666667
6	0,1666667

- Exemplo: $P(X \leq 2)$:

```
distr %>%
  filter(X <=2) %>%
  pull(p) %>%
  sum()
```

```
## [1] 0,3333333
```

- Gráfico:



- Para gerar amostras, use `sample`, que, por *default*, trabalha com a distribuição uniforme discreta.
- Simulando dez lançamentos de um dado:

```
sample(1:6, size = 10, replace = TRUE)
```

```
##  [1] 6 3 2 2 6 3 2 2 3 3
```

9.3 Distribuição de Bernoulli

9.3.1 Exemplo: uma moeda

- O experimento tem exatamente dois resultados possíveis: coroa ou cara.
- A variável aleatória X representa numericamente os dois resultados possíveis. É comum usar 0 para um resultado, 1 para o o outro.

$$X = \begin{cases} 0 & \text{se coroa} \\ 1 & \text{se cara} \end{cases}$$

- O suporte é o conjunto $\{0, 1\}$.

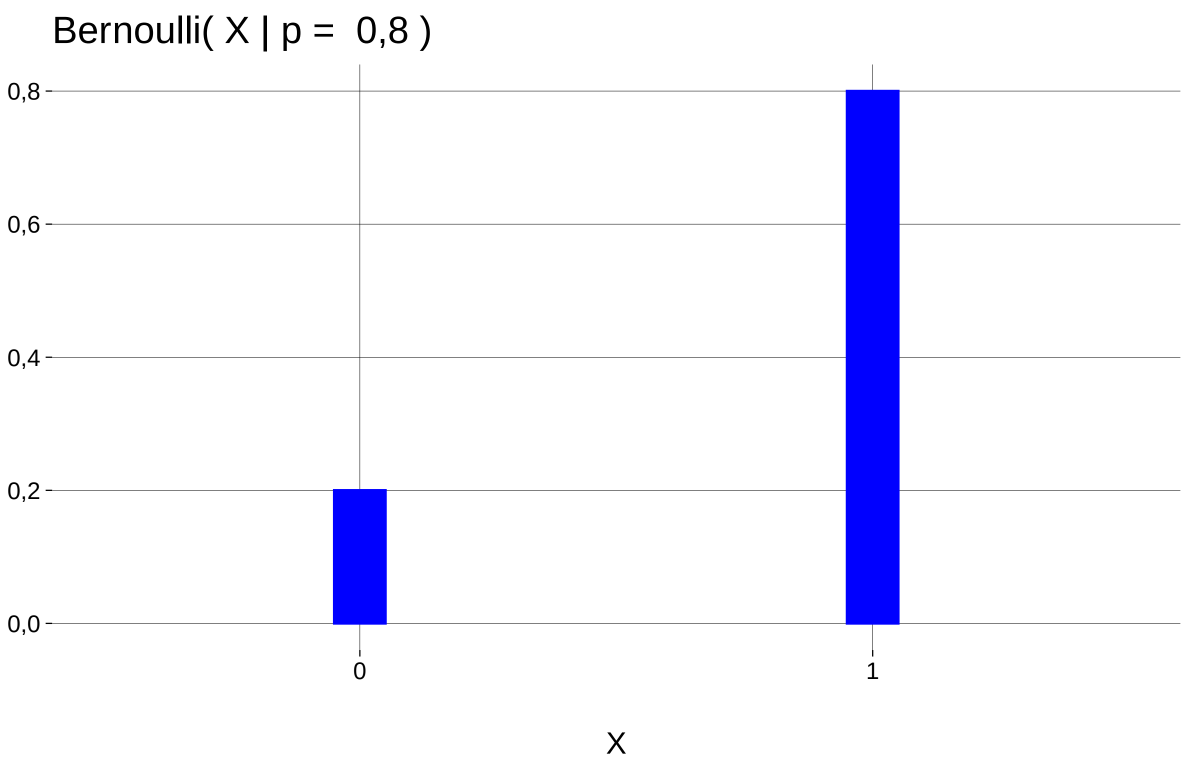
- Vamos chamar de p a probabilidade de cara (o caso $X = 1$, comumente chamado de sucesso)[1].
- Para uma moeda justa, a distribuição de X é a distribuição de Bernoulli com $p = 0{,}5$, escrita como Bernoulli$(X \mid p = 0{,}5)$:

$$P(X = 0) = 1 - 0{,}5$$
$$P(X = 1) = 0{,}5$$

- Podemos escrever $P(X = x)$ de forma mais compacta:

$$P(X = x) = 0{,}5^x \cdot (1 - 0{,}5)^{1-x}$$

- Cada valor do parâmetro p dá uma distribuição de Bernoulli diferente.
- Gráfico com $p = 0{,}8$ (uma moeda muito viciada, para a qual a probabilidade de cara é de 80%):



[1] Aqui, chamar cara de sucesso é uma escolha totalmente arbitrária.

9.3.2 No geral

- Para probabilidade de sucesso p, a distribuição é escrita como Bernoulli$(X = x \mid p)$.
- $x \in \{0, 1\}$.
- $p \in [0, 1]$.
- Probabilidades:

$$P(X = x) = p^x \cdot (1-p)^{1-x}$$

- É comum chamar de q a probabilidade de fracasso. Ou seja,

$$q = 1 - p$$

- As probabilidades ficam

$$P(X = x) = p^x \cdot q^{1-x}$$

- Valor esperado:

$$\begin{aligned} E(X) &= 0 \cdot (1-p) \ + \ 1 \cdot p \\ &= p \end{aligned}$$

- Variância:

$$\begin{aligned} \sigma^2(X) &= (0-p)^2 \cdot (1-p) \ + \ (1-p)^2 \cdot p \\ &= p^2 - p^3 + p - 2p^2 + p^3 \\ &= -p^2 + p \\ &= p(1-p) \\ &= pq \end{aligned}$$

9.3.3 Em R

> Em R, a distribuição de Bernoulli é um caso especial ($n = 1$) da distribuição binomial, que nós vamos ver mais adiante neste capítulo.

- Se você não quiser usar as funções da distribuição binomial, pode definir um vetor com os dois valores possíveis e um vetor com as duas probabilidades, e então usar `sample` com o argumento `prob`.

- Por exemplo, para simular 10 lançamentos de uma moeda viciada, onde $p = 0{,}8$:

```
moeda <- 0:1
probs <- c(.2, .8)
sample(moeda, size = 10, prob = probs, replace = TRUE)
```

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

9.4 Distribuição geométrica

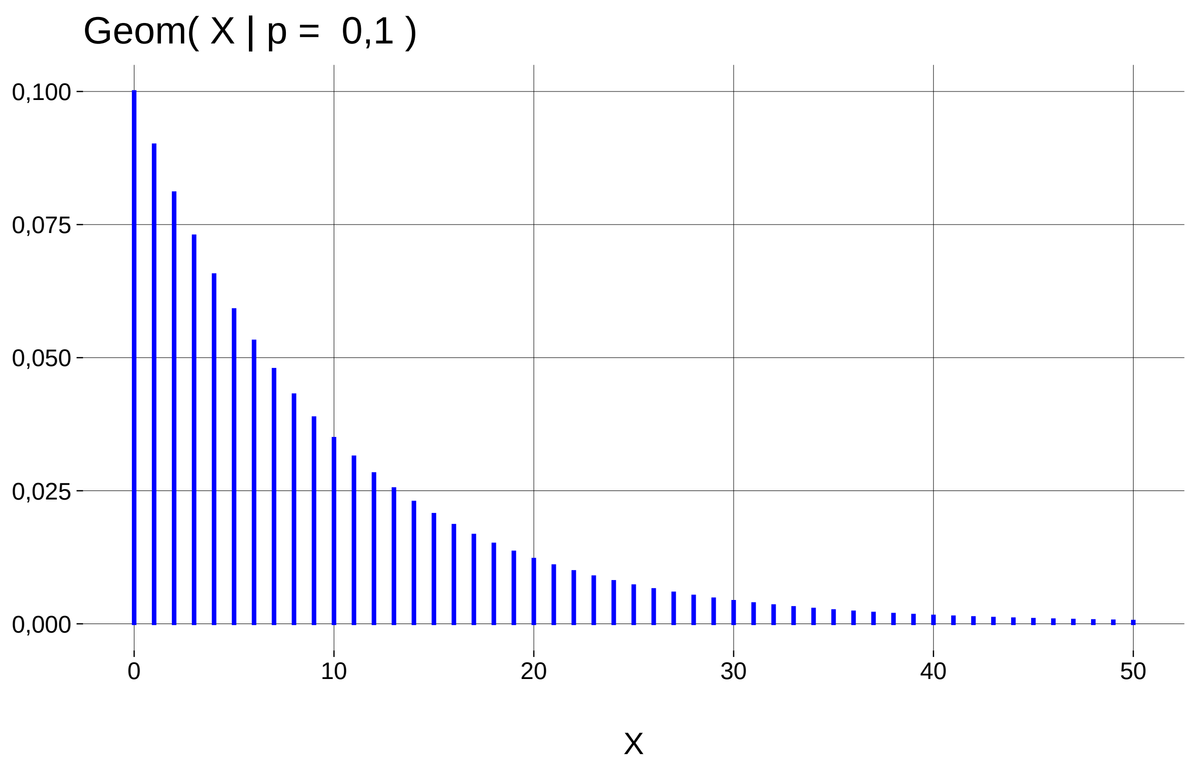
9.4.1 Exemplo: *spam*

- Cada e-mail tem probabilidade $0{,}1$ de não ser *spam*, e $0{,}9$ de ser *spam*.
- Considere que cada e-mail é independente de cada outro.
- Você abre sua *inbox* (sem filtro *antispam*) e começa a ler as mensagens sequencialmente.
- A variável aleatória X representa o número de mensagens *spam* que você precisa abrir até chegar à primeira mensagem que não é *spam*.
- O suporte é $\{0, 1, 2, 3, \ldots\}$ (nossa primeira distribuição com suporte infinito!).[2]
- Vamos chamar de p a probabilidade de sucesso (a mensagem não ser *spam*).
- A distribuição de X é a distribuição geométrica com $p = 0{,}1$, escrita como Geom$(X \mid p = 0{,}1)$.
- Qual a probabilidade de que a primeira mensagem não-*spam* seja a décima, por exemplo?

$$\begin{aligned}\text{Geom}(X = 9 \mid p = 0{,}1) &= 0{,}9^9 \cdot 0{,}1 \\ &\approx 0{,}039\end{aligned}$$

- Gráfico:

[2]Mas como o conjunto é enumerável, a variável aleatória X é discreta.



- Cada barra corresponde à probabilidade de que seja necessário abrir exatamente x mensagens *spam* antes de chegar à primeira mensagem não-*spam*.
- Mais adiante, vamos ver como calcular a probabilidade de que seja necessário abrir no mínimo (ou no máximo) x mensagens *spam* antes de chegar à primeira mensagem não-*spam*.
- Cada valor de p dá uma distribuição geométrica diferente.

9.4.2 No geral

- Para probabilidade de sucesso p, a distribuição é escrita como Geom$(X = x \mid p)$.
- $x \in \{0, 1, 2, 3, ...\}$.
- $p \in [0, 1]$.
- X conta a quantidade de provas de Bernoulli que têm resultado fracasso antes do primeiro sucesso.
- As provas de Bernoulli são independentes e têm probabilidade fixa de sucesso p.
- Probabilidades:

$$\text{Geom}(X = x \mid p) = (1 - p)^x \cdot p$$

- Valor esperado:

Vamos chamar $1 - p$ de q.

Então, $P(X = x) = q^x \cdot p$.

Daí,

$$
\begin{aligned}
E(X) &= 0p + 1qp + 2q^2p + 3q^3p + \cdots \\
&= 0(1-q) + 1q(1-q) + 2q^2(1-q) + 3q^3(1-q) + \cdots \\
&= q - q^2 + 2q^2 - 2q^3 + 3q^3 - 3q^4 + \cdots \\
&= q + q^2 + q^3 + \cdots \\
&= \frac{q}{1-q} \\
&= \frac{1-p}{p} \\
&= \frac{q}{p}
\end{aligned}
$$

No exemplo dos e-mails,

$$E(X) = \frac{q}{p} = \frac{0{,}9}{0{,}1} = 9$$

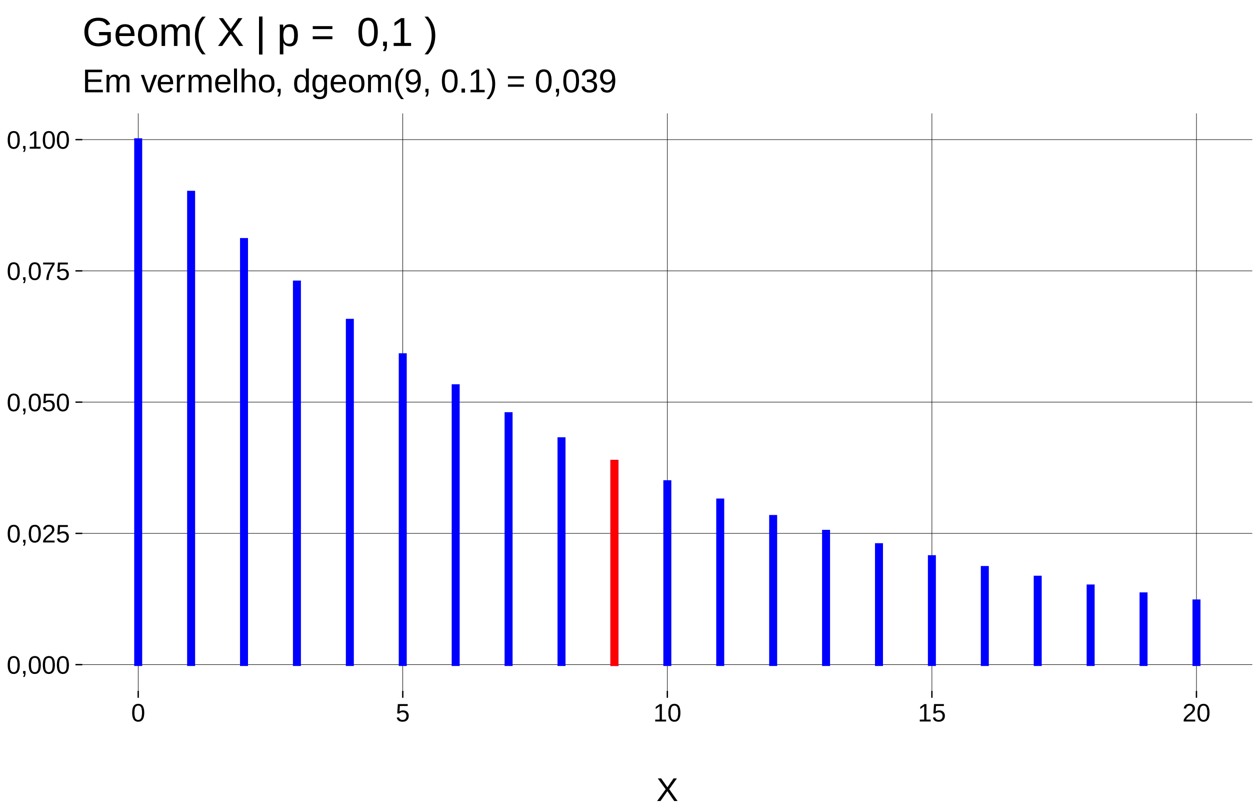
- Variância:

$$\sigma^2(X) = \frac{1-p}{p^2} = \frac{q}{p^2}$$

- Exercício: derive esta fórmula da variância.

9.4.3 Em R

Função de distribuição de probabilidade: Geom$(X = x \mid p)$



- Para calcular Geom$(X = x \mid p)$, use `dgeom(x, prob=p)`.
- Tanto `x` quanto `prob` podem ser vetores.
- No exemplo do *spam*, vamos computar as probabilidades de X ser $0, 1, 2, \ldots, 10$:

```
dgeom(x = 0:10, prob = .1)
```

```
##  [1] 0,10000000 0,09000000 0,08100000 0,07290000 0,06561000 0,05904900
##  [7] 0,05314410 0,04782969 0,04304672 0,03874205 0,03486784
```

- Vamos computar Geom$(X = 10 \mid p)$ para vários valores de p:

```
dgeom(10, prob = seq(0.1, 1.0, 0.1))
```

```
##  [1] 0,03486784401 0,02147483648 0,00847425747 0,00241864704
##  [5] 0,00048828125 0,00006291456 0,00000413343 0,00000008192
##  [9] 0,00000000009 0,00000000000
```

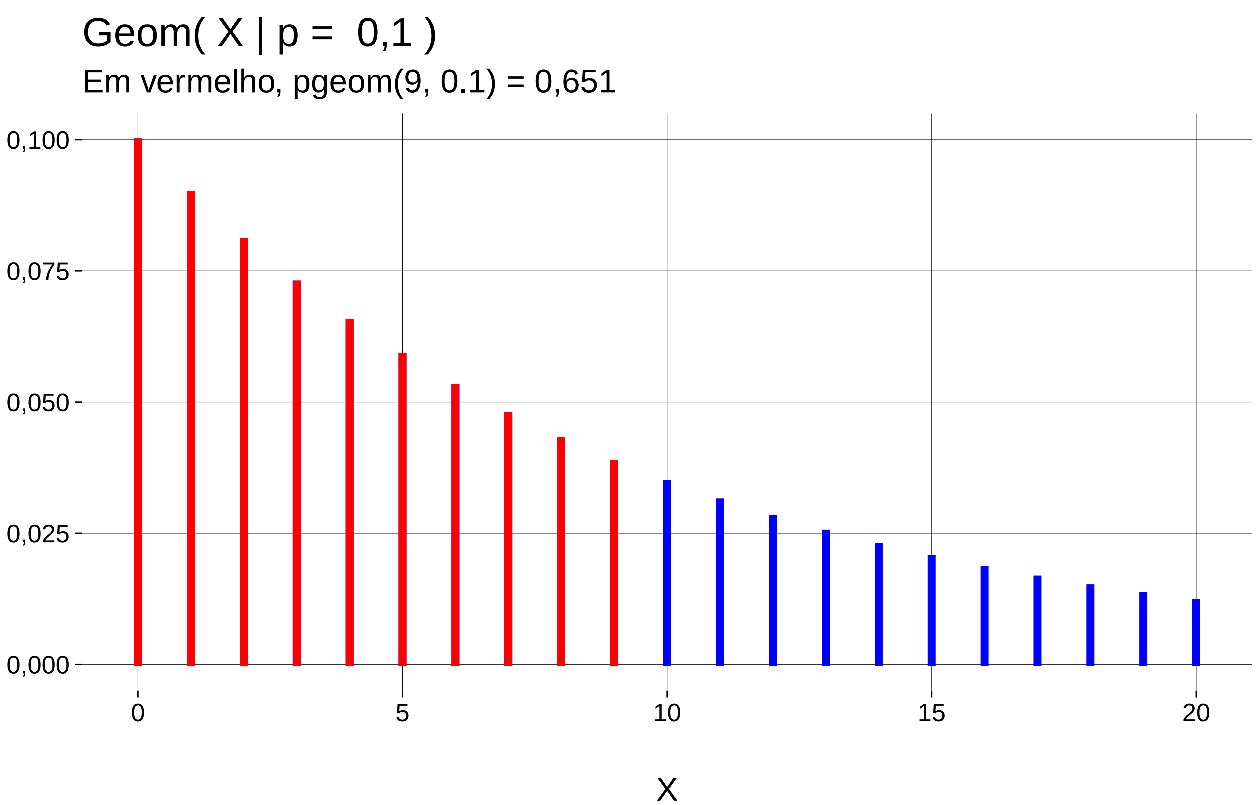
- Mas cuidado: se os dois argumentos forem vetores, o resultado é um vetor com o comprimento do maior argumento, com os valores de `x` pareados um a um com os valores de `prob` (lembrando que R recicla o vetor mais curto):

```r
dgeom(c(10, 11, 12), c(0.1, 0.2))
```

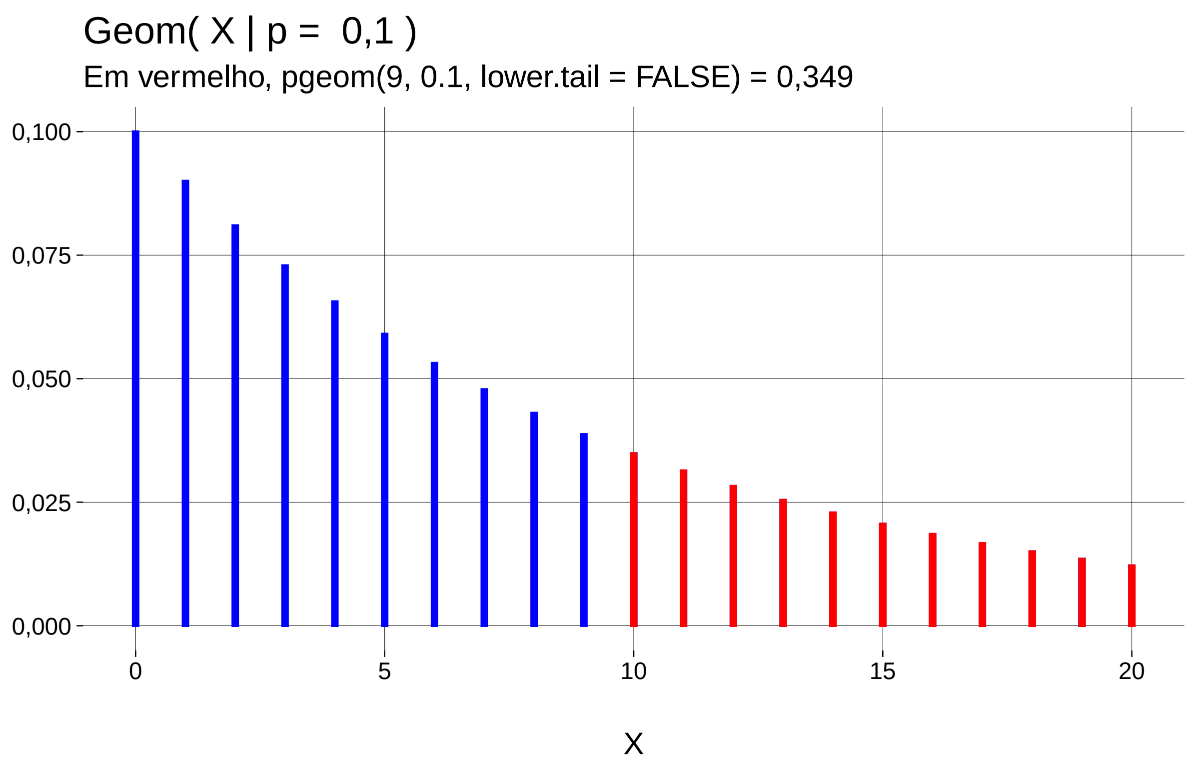
```
## [1] 0,03486784 0,01717987 0,02824295
```

Os valores acima são, respectivamente, $P(X = 10 \mid p = 0{,}1)$, $P(X = 11 \mid p = 0{,}2)$ e $P(X = 12 \mid p = 0{,}1)$.

Função de distribuição acumulada: Geom$(X \leq q \mid p)$



- Para calcular Geom$(X \leq q \mid p)$, use `pgeom(q, prob=p)`.
- Se você passar, como argumento, `lower.tail = FALSE`, a probabilidade calculada é $P(X > q \mid p)$ (a probabilidade acumulada à direita do valor q).



- Tanto `q` quanto `prob` podem ser vetores.
- No exemplo do *spam*, vamos computar a probabilidade de precisar abrir no máximo 10 mensagens *spam* para então abrir a primeira mensagem não-*spam*:

```
pgeom(q = 10, prob = .1)
```

```
## [1] 0,6861894
```

Na verdade, isto é o mesmo que somar as probabilidades de $X = 0$, $X = 1$, etc., até $X = 10$:

```
sum(dgeom(0:10, .1))
```

```
## [1] 0,6861894
```

- Um exemplo mais realista: cada vez que você joga 6 números na Mega-Sena, a probabilidade de você acertar a sena é de 1 em $50.063.860$, segundo http://loterias.caixa.gov.br/wps/portal/loterias/landing/megasena.[3]
- Qual a probabilidade de você acertar a sena em alguma das primeiras 1.000 vezes que você jogar? 10.000 vezes? 100.000 vezes (o que equivale a cerca de 962 anos, jogando 2 vezes por semana)?

[3]Esta probabilidade é computada usando a distribuição hipergeométrica, que não vamos cobrir neste curso. Veja mais informações aqui.

```r
p <- 1/50063860
pgeom(c(1e3, 1e4, 1e5), p)
```

```
## [1] 0,00001999426 0,00019974491 0,00199547524
```

- Qual a probabilidade de você jogar duas vezes por semana, durante 100 anos, sem acertar a sena? Considerando 52 semanas por ano:

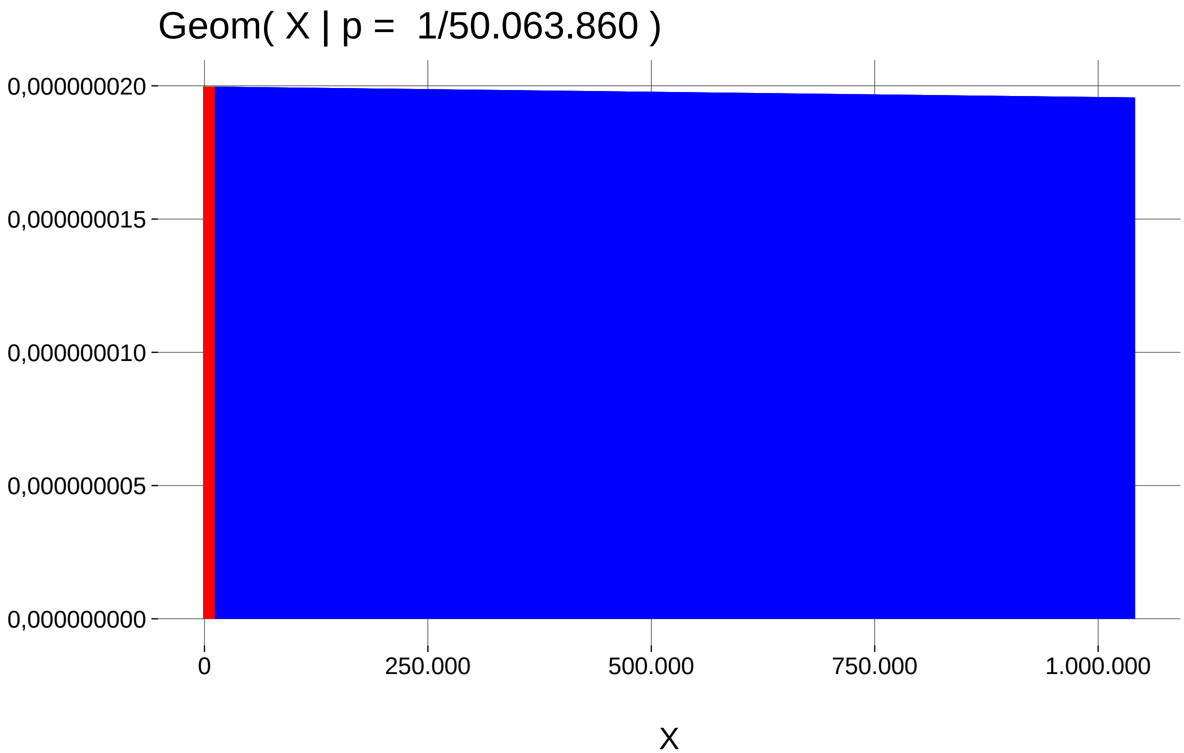
```r
vezes <- 2 * 52 * 100
vezes
```

```
## [1] 10400
```

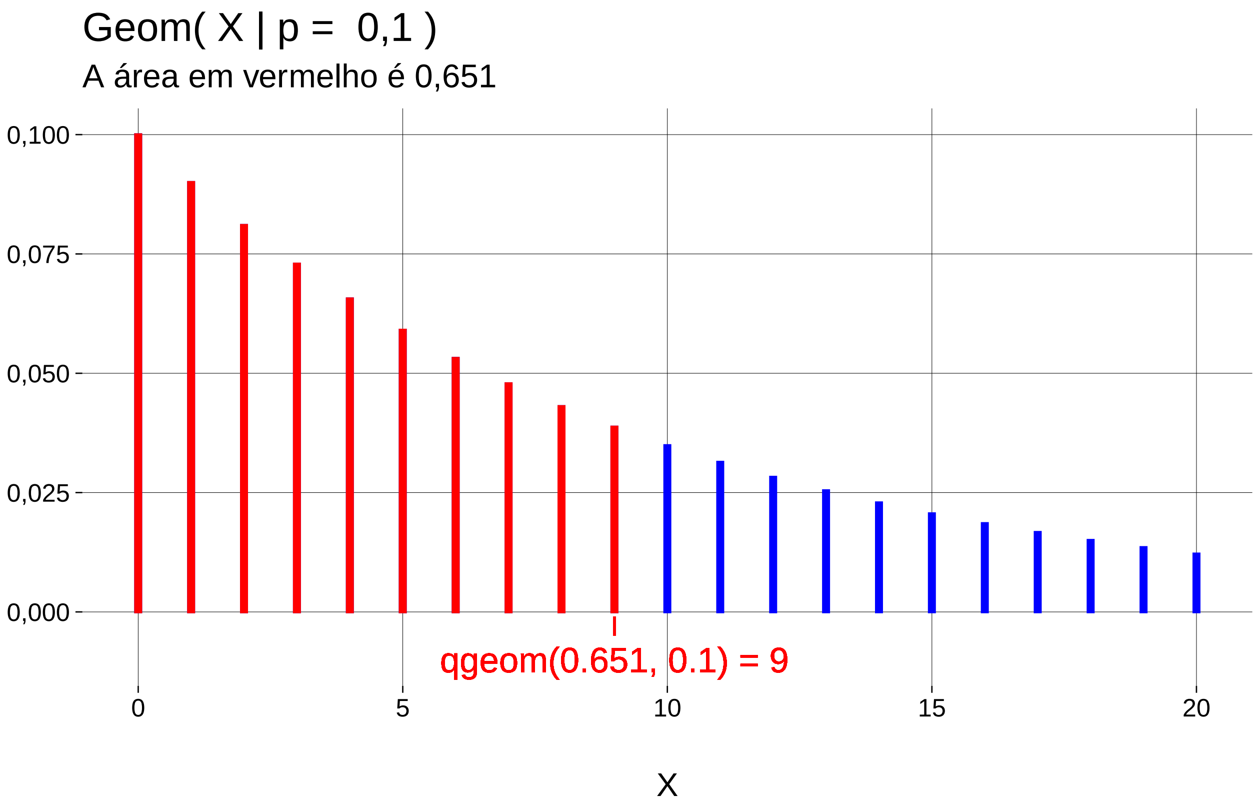
```r
pgeom(vezes, p, lower.tail = FALSE)
```

```
## [1] 0,9997923
```

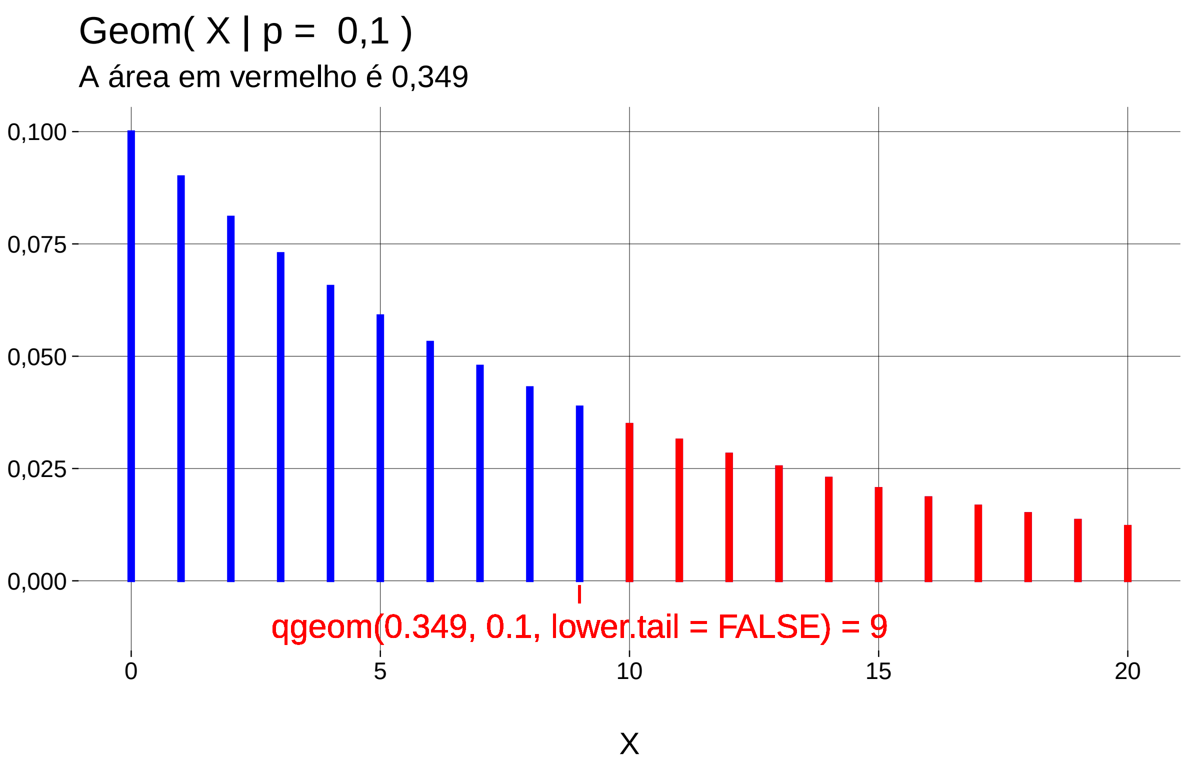
- O problema é que, com uma probabilidade de sucesso tão baixa, a distribuição geométrica começa em um valor baixo e decresce muito lentamente. Isto significa que a maior parte da probabilidade vai estar à direita de valores muito altos.
- O gráfico abaixo vai até $X = 1$ milhão. A área da faixa vermelha é a probabilidade de você acertar a sena jogando no máximo 10 mil vezes. Esta probabilidade é $0{,}00019974491$.
- Isto equivale a dizer que a probabilidade de você precisar de mais de 10 mil jogos para acertar a sena é de $1 - 0{,}00019974491 = 0{,}99980025509$, que é a área em azul no gráfico, mais a área restante à direita, de 1 milhão até o infinito, que não aparece no gráfico!


Geom(X | p = 1/50.063.860)
0,000000020
0,000000015
0,000000010
0,000000005
0,000000000
0
250.000
500.000
750.000
1.000.000
X

Função quantil: dado um valor de Geom$(X \leq x \mid p)$, então $x = ?$



- O objetivo é achar x tal que Geom$(X \leq x \mid p) = m$.
- Em palavras: achar o valor x à esquerda do qual — incluindo x — existe a probabilidade acumulada de m.
- Para isto, use `qgeom(m, prob=p)`.
- Se você passar, como argumento, `lower.tail = FALSE`, o valor calculado é x tal que $P(X > x \mid p) = m$ (o valor à direita do qual — excluindo x — existe a probabilidade acumulada de m):



- Quantas vezes você precisa apostar 6 números na Mega-Sena para ter 50% de chance de acertar a sena alguma vez?

```
p <- 1/50063860
qgeom(.5, p)
```

```
## [1] 34701623
```

- Isto equivale a 333.669 anos, jogando duas vezes por semana, toda semana.

Função para gerar números aleatórios

- Para gerar um vetor com n valores aleatórios a partir de uma distribuição Geom$(X \mid p)$, use `rgeom(n, prob=p)`.
- Voltando ao exemplo do *spam*, vamos simular muitos experimentos.
- Os resultados são os valores de X, i.e., as quantidades de mensagens *spam* que precisaram ser abertas antes de chegarmos à primeira mensagem não-*spam*:

```
amostra <- rgeom(1000, .1)
head(amostra)
```

```
## [1]  1 11  6  7  5  2
```

- O valor esperado teórico é $\frac{1-p}{p} = 9$. Vamos comparar com a média da amostra gerada:

```
mean(amostra)
```

```
## [1] 9,165
```

- A variância teórica é $\frac{1-p}{p^2} = 90$. Vamos comparar com a variância da amostra gerada:

```
var(amostra)
```

```
## [1] 97,85763
```

- Vamos simular 100 mil pessoas jogando na Mega-Sena e ver se alguma acertou a sena antes de jogar mil vezes:

```
p <- 1/50063860
resultados <- rgeom(1e5, p)
resultados[resultados <= 1e3]
```

```
## [1] 660 869
```

- Qual a média da nossa amostra? O valor esperado teórico é $50.063.859$.

```
mean(resultados)
```

```
## [1] 50058162
```

- E a mediana?

```
median(resultados)
```

```
## [1] 34844436
```

- E o máximo?

```
max(resultados)
```

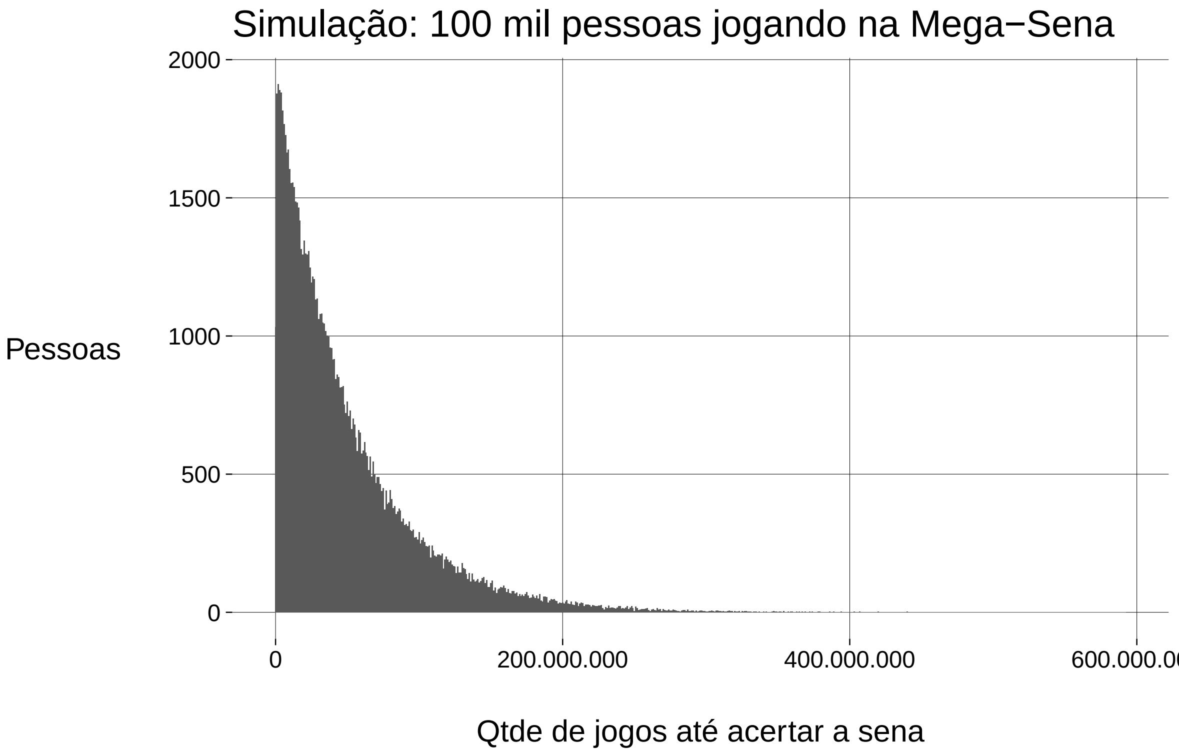
```
## [1] 592400919
```

- E o mínimo?

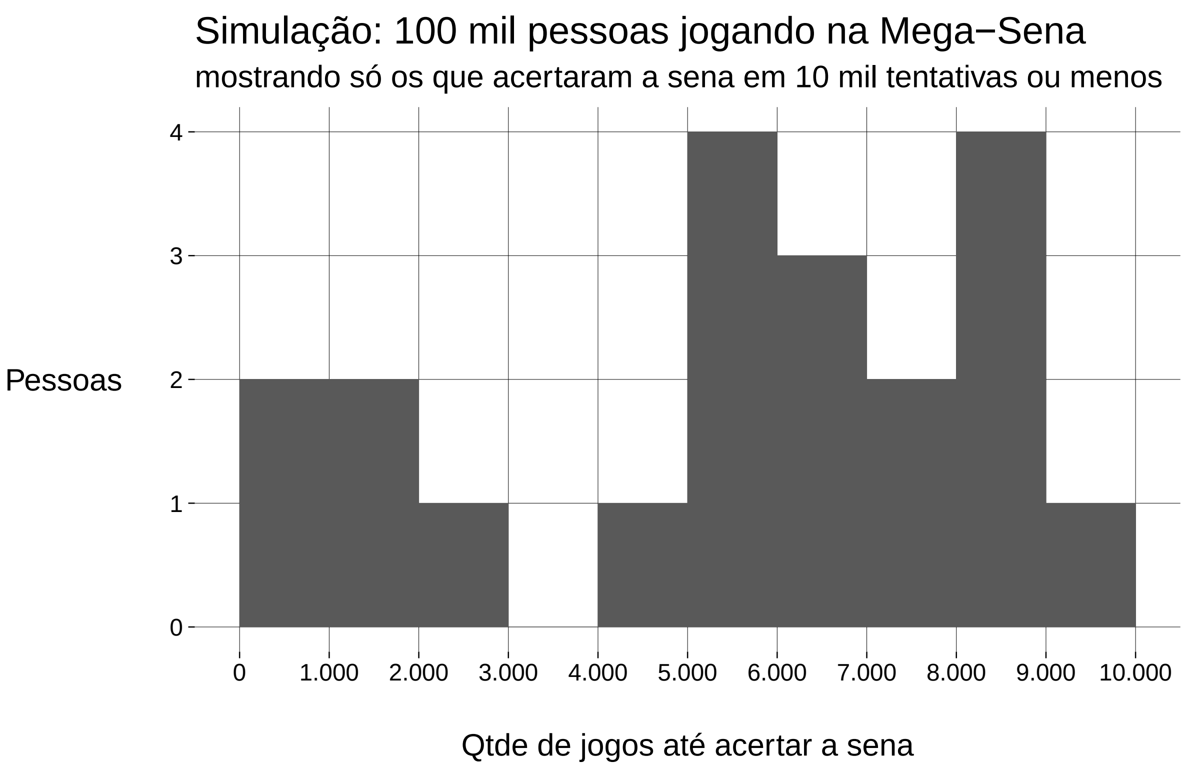
```
min(resultados)
```

```
## [1] 660
```

- Gráfico da nossa amostra:



- Pode parecer que muitas pessoas acertaram a sena jogando poucos jogos, mas as aparências enganam: cada barra do histograma acima corresponde a 1 milhão de jogos.
- Os sortudos que ganharam antes de jogar 10 mil jogos:



9.5 Vídeo 2

https://youtu.be/F6OEoEaYrCw

9.6 Distribuição binomial

9.6.1 Exemplo: *spam* novamente

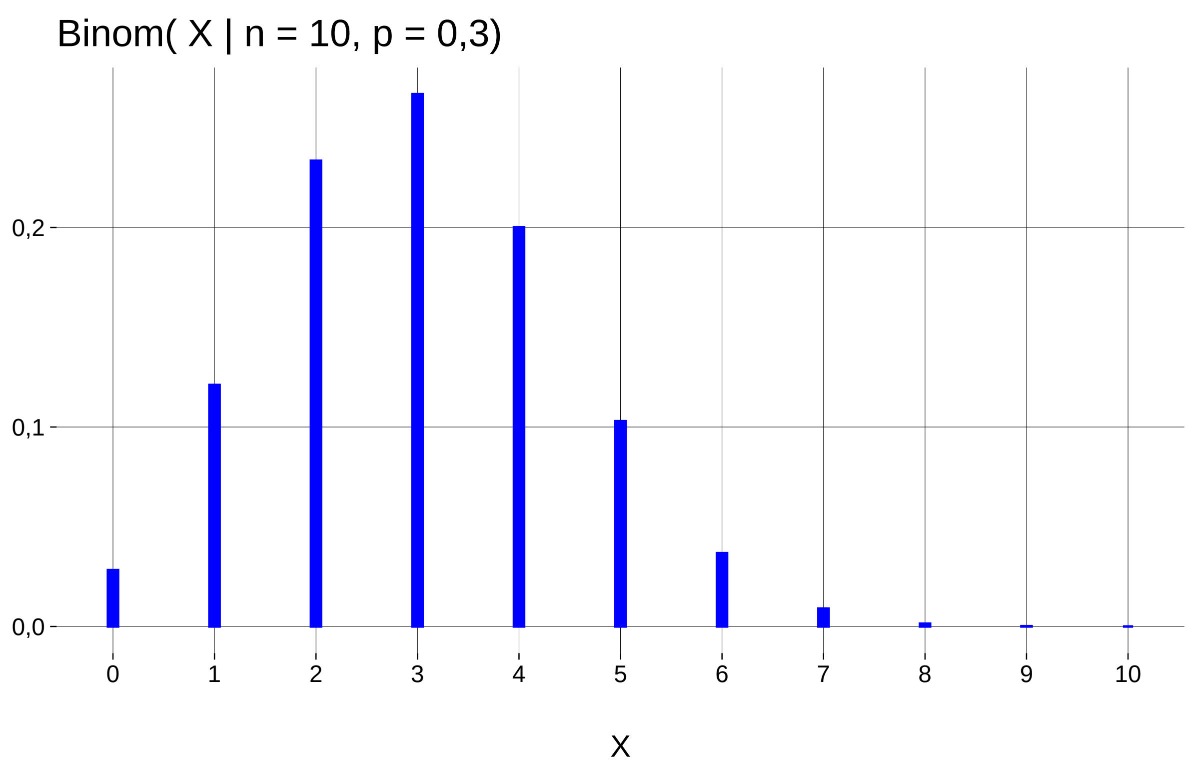
- Vamos mudar para um mundo onde a probabilidade de uma mensagem qualquer não ser *spam* é $p = 0{,}3$.
- Agora, vamos mudar o experimento: ao recebermos $n = 10$ mensagens, quantas não são *spam*?
- A variável aleatória X representa o número de sucessos (não *spam*) em $n = 10$ mensagens.
- O suporte de X é $\{0, 1, 2, \dots, 10\}$.
- Os parâmetros são:

- p: a probabilidade de sucesso (uma mensagem não ser *spam*).
- n: o número total de mensagens.

- A distribuição de probabilidade de X é binomial, com estes parâmetros, escrita como Binom$(X \mid n = 10, p = 0{,}3)$.
- Qual a probabilidade de haver, dentre as 10 mensagens, 4 mensagens não *spam*? As mensagens são independentes. O resultado vai envolver:
  - A probabilidade de 4 mensagens não serem *spam*: $(0{,}3)^4$.
  - A probabilidade de 6 mensagens serem *spam*: $(1 - 0{,}3)^6$.
  - A quantidade de ordenações diferentes destas mensagens: $\binom{10}{4} = \frac{10!}{4!6!} = 210$.
  - O resultado é

$$\text{Binom}(X = 4 \mid n = 10,\ p = 0{,}3) = \binom{10}{4} \cdot (0{,}3)^4 \cdot (1 - 0{,}3)^6$$
$$= 0{,}20$$

- Gráfico:



9.6.2

No geral

- X conta a quantidade de sucessos em um número fixo n de provas de Bernoulli.
- X pode assumir os valores $x \in \{0, 1, 2, \dots, n\}$.
- As provas de Bernoulli são independentes e têm probabilidade fixa de sucesso p.
- A probabilidade de X assumir o valor x é

$$\text{Binom}(X = x \mid n, p) = \binom{n}{x} \cdot p^x \cdot (1-p)^{n-x}$$

- Valor esperado:
  - Na verdade, X é a soma de n variáveis aleatórias independentes $X_1, \dots, X_n$, cada uma delas com distribuição de Bernoulli com probabilidade de sucesso p (e valor esperado p):

$$\begin{aligned} E(X) &= E(X_1 + \dots + X_n) \\ &= E(X_1) + \dots + E(X_n) \\ &= p + \dots + p \\ &= np \end{aligned}$$

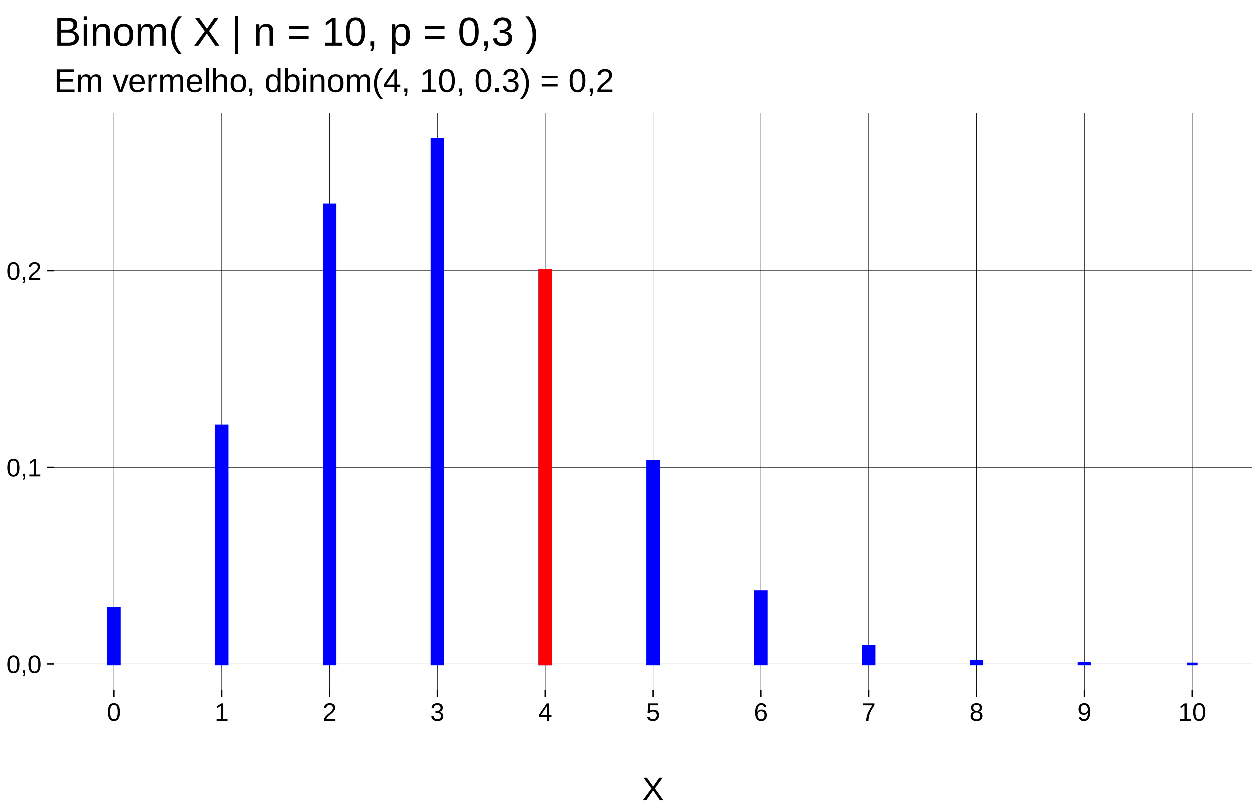
- Variância:
  - Como as variáveis $X_1, \dots, X_n$ são independentes, a variância da soma é a soma das variâncias:

$$\begin{aligned} \sigma^2(X) &= \sigma^2(X_1 + \dots + X_n) \\ &= \sigma^2(X_1) + \dots + \sigma^2(X_n) \\ &= p(1-p) + \dots + p(1-p) \\ &= np(1-p) \end{aligned}$$

9.6.3

Em R

Função de distribuição de probabilidade: Binom$(X = x \mid n, p)$

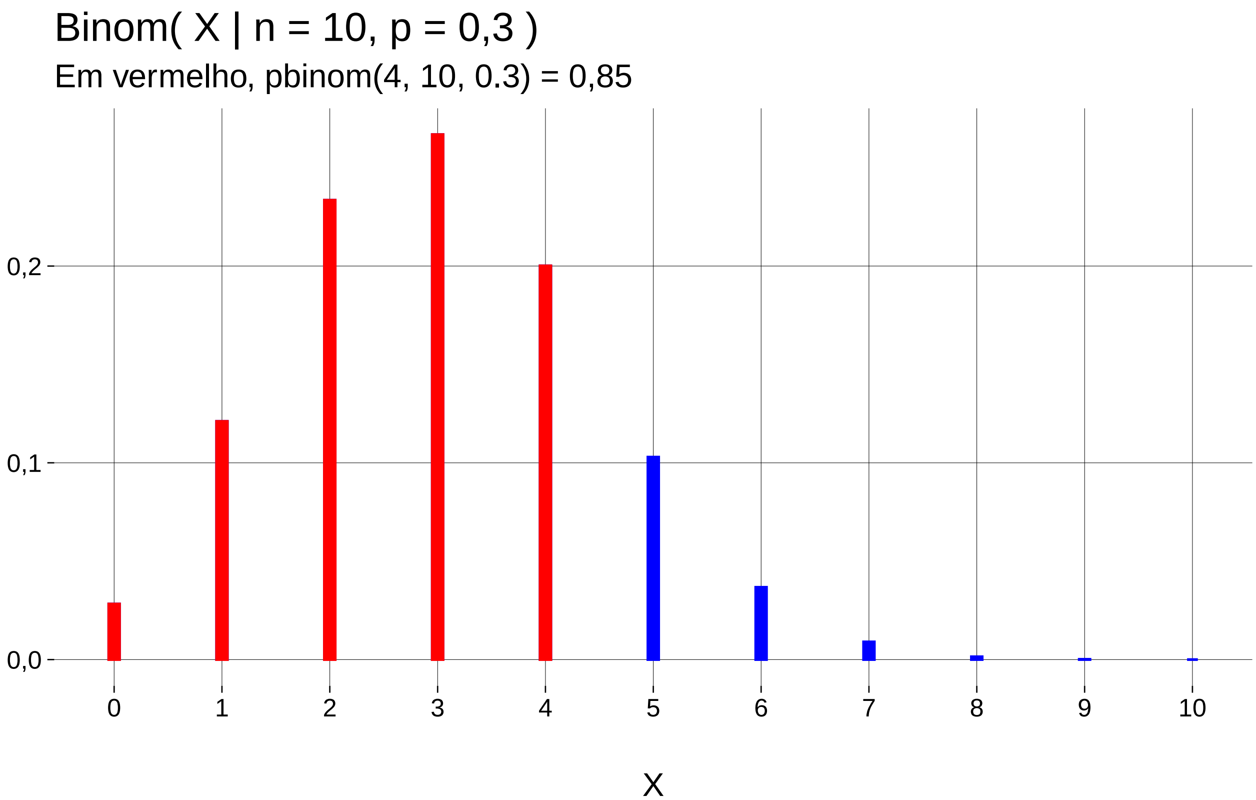


- Para calcular Binom$(X = x \mid n, p)$, use `dbinom(x, size=n, prob=p)`.
- Voltando à Mega-Sena: para uma única pessoa, jogar 6 números uma vez é uma prova de Bernoulli, com probabilidade de sucesso de $1/50.063.860 = 0{,}00000001997$.
- Se você jogar 6 números 10 mil vezes — duas vezes por semana, durante quase 100 anos — sua probabilidade de obter zero sucessos é

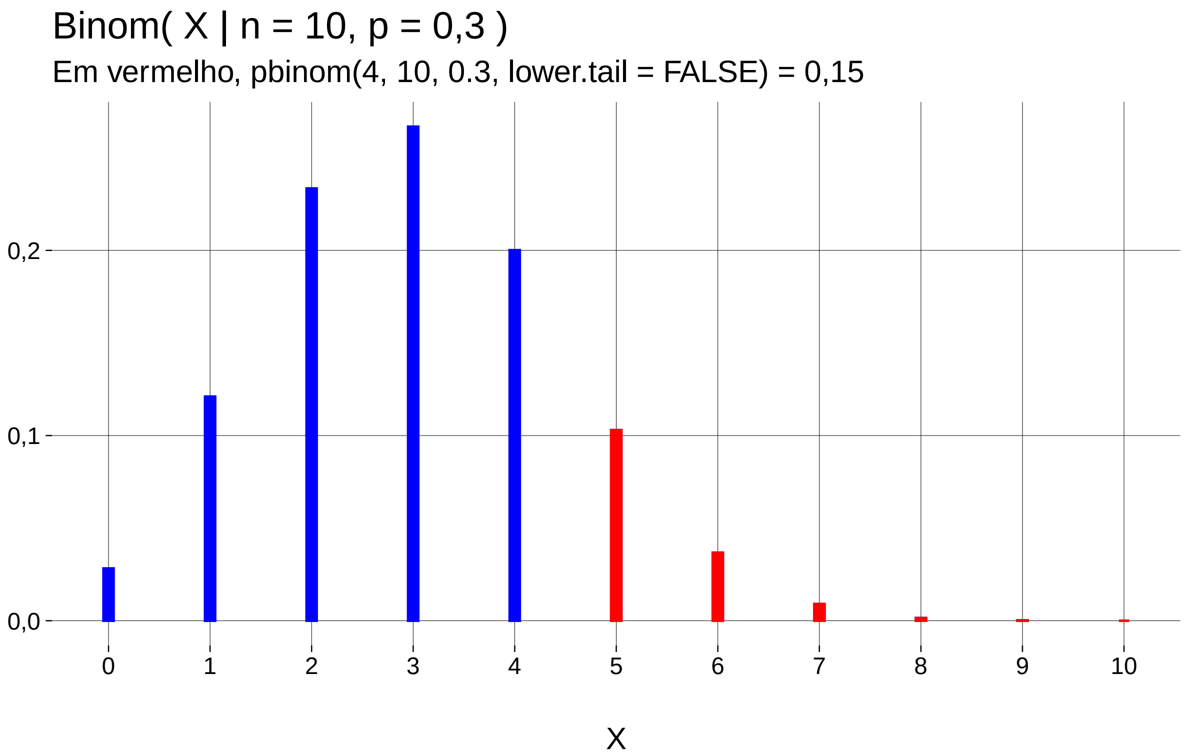
```
p <- 1/50063860
vezes <- 1e4
dbinom(0, n, p)
```

```
## [1] 0,9999998
```

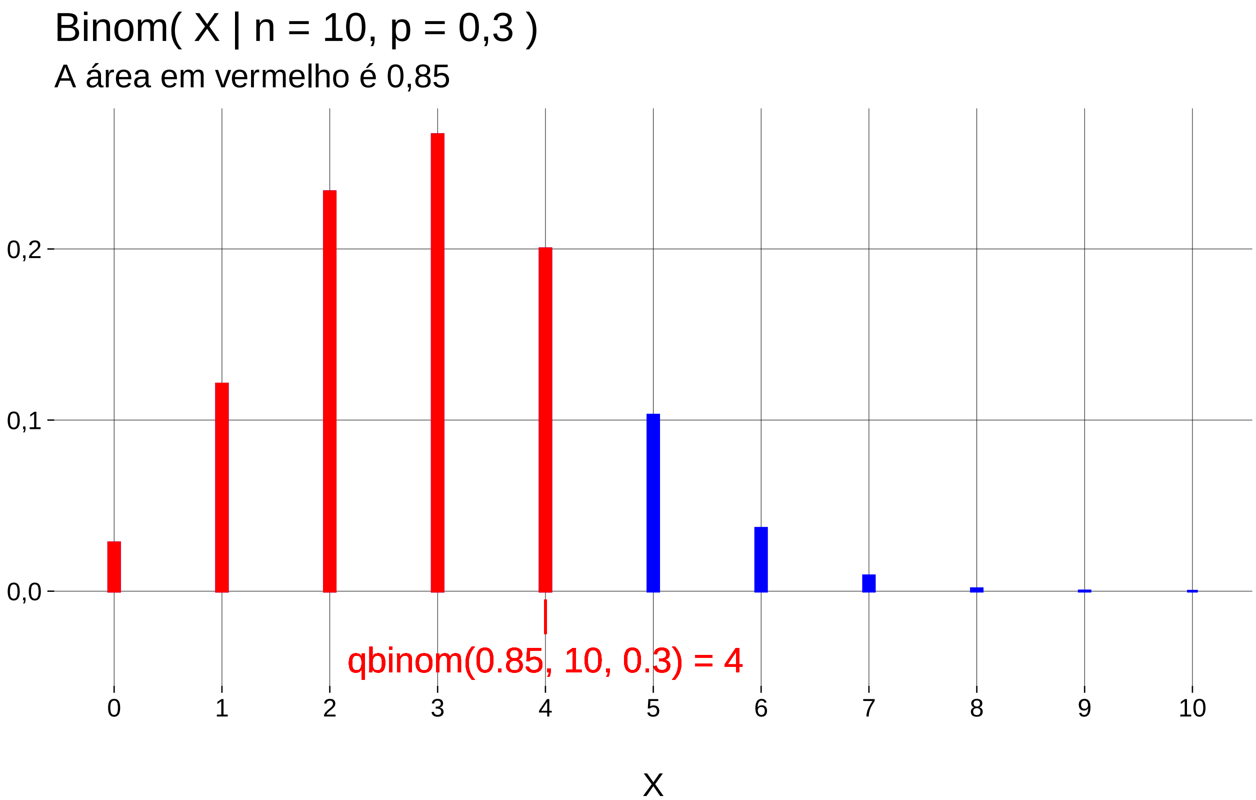
Função de distribuição acumulada: Binom$(X \leq q \mid n, p)$



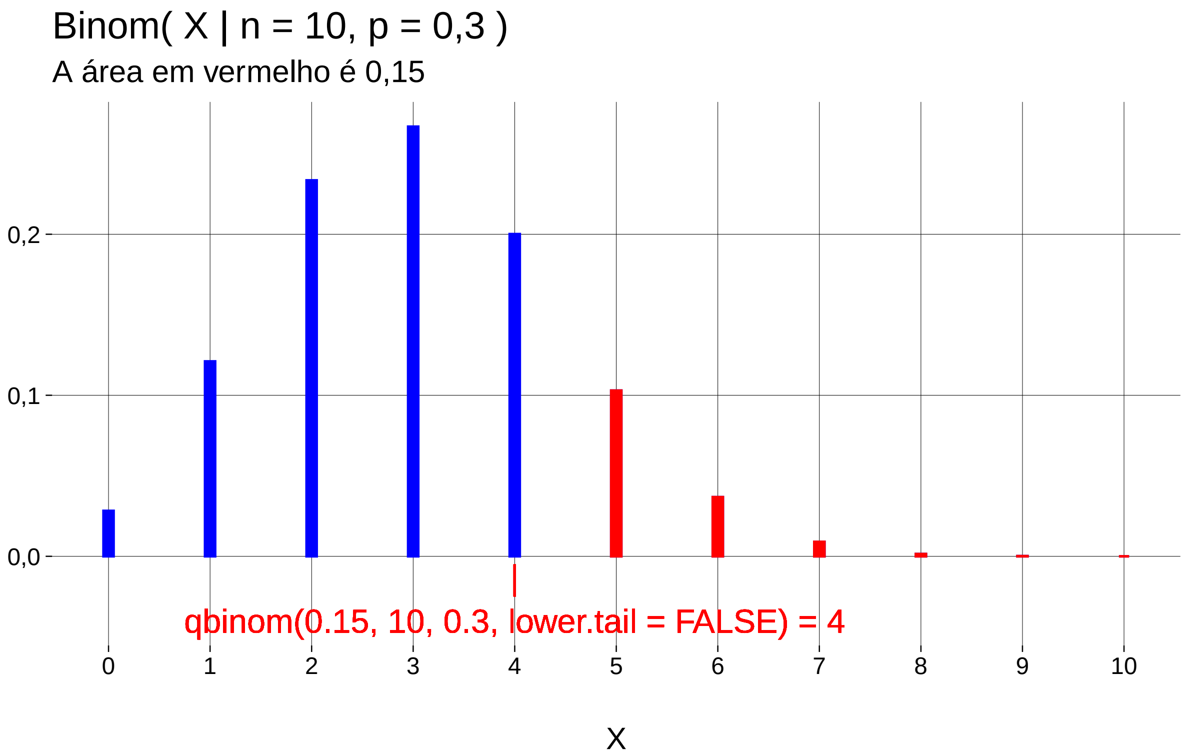
- Para calcular Binom$(X \leq q \mid n, p)$, use `pbinom(q, size=n, prob=p)`.
- Se você passar, como argumento, `lower.tail = FALSE`, a probabilidade calculada é $P(X > q \mid n, p)$ (a probabilidade acumulada à direita do valor q.


Binom(X | n = 10, p = 0,3)
Em vermelho, pbinom(4, 10, 0.3, lower.tail = FALSE) = 0,15
0,2
0,1
0,0
0
1
2
3
4
5
6
7
8
9
10
X

Função quantil: dado um valor de Binom$(X \leq x \mid n, p)$, então $x = ?$



- O objetivo é achar x tal que Binom$(X \leq x \mid n, p) = m$.
- Em palavras: achar o valor x à esquerda do qual — incluindo x — existe a probabilidade acumulada de m.
- Para isto, use `qbinom(m, size=n, prob=p)`.
- Se você passar, como argumento, `lower.tail = FALSE`, o valor calculado é x tal que $P(X > x \mid n, p) = m$ (o valor à direita do qual — excluindo x — existe a probabilidade acumulada de m):



Função para gerar números aleatórios

- Para gerar um vetor com v valores aleatórios a partir de uma distribuição Binom$(X \mid n, p)$, use `rbinom(v, size=n, prob=p)`.
- Vamos simular 100 vezes o experimento de abrir 10 mensagens e contar quantas delas não são *spam*:

```
amostra <- rbinom(100, 10, .3)
amostra
```

```
##   [1] 3 1 3 2 3 3 4 4 5 1 2 4 1 4 2 3 4 7 5 1 1 3 1 4 1 3 4 2 5 5 2 6 1
##  [34] 4 3 3 2 4 3 3 5 3 4 5 3 4 1 4 0 2 3 2 4 3 2 6 4 2 2 6 7 2 3 2 3 2
##  [67] 2 3 5 4 3 2 2 2 4 4 2 2 1 3 2 2 3 3 4 2 1 5 1 5 3 3 4 0 3 1 2 5 2
## [100] 1
```

- Média:

```
mean(amostra)
```

```
## [1] 2,97
```

- Finalmente, vamos simular 100 mil pessoas, cada uma jogando 10 mil jogos da Mega-Sena e ver quantas ganharam pelo menos uma vez:

```r
n <- 1e5
p <- 1/50063860
size <- 1e4

resultados <- rbinom(n, size, p)
head(resultados, 1000)
```

```
##    [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [33] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [65] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [97] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [129] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [161] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [193] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [225] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [257] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [289] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [321] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [353] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [385] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [417] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [449] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [481] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [513] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [545] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [577] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [609] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [641] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [673] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [705] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [737] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [769] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [801] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [833] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [865] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [897] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [929] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [961] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [993] 0 0 0 0 0 0 0 0
```

```r
ganhadores <- resultados[resultados > 0]
ganhadores
```

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

```
length(ganhadores)
```

```
## [1] 16
```

9.7 Distribuição de Poisson

9.7.1 Exemplo

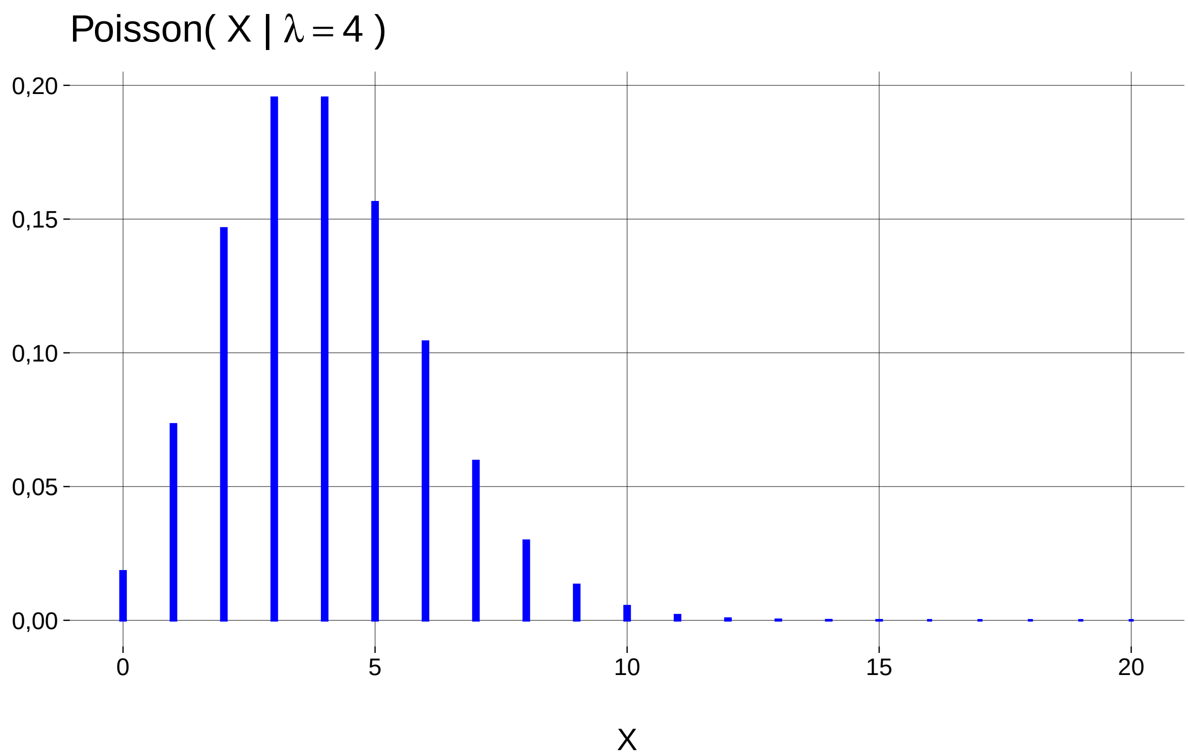
- Em um caixa de um pequeno mercado, o número de clientes que chegam por minuto é, em média, 4.
- Vamos representar este número pela variável aleatória X, com suporte $\{0, 1, 2, 3, ...\}$.
- As chegadas dos clientes são independentes umas das outras.
- Se você dividir o tempo em intervalos menores do que um minuto, a média se mantém: 2 clientes a cada 30 segundos, 1 cliente a cada 15 segundos etc.
- Clientes diferentes não chegam ao caixa no mesmo instante.
- Com estas condições, e chamando o número médio de clientes por minuto de $\lambda = 4$, a probabilidade de que 10 clientes cheguem ao caixa em um período de um minuto é dada por

$$\begin{aligned}
P(X = 10) &= \text{Poisson}(X = 10 \mid \lambda = 4) \\
&= \frac{\lambda^{10}}{10!} e^{-\lambda} \\
&= \frac{4^{10}}{10!} e^{-4} \\
&= 0{,}01
\end{aligned}$$

- A probabilidade de que o caixa fique vazio durante um minuto inteiro é

$$\begin{aligned}
P(X = 0) &= \text{Poisson}(X = 0 \mid \lambda = 4) \\
&= \frac{\lambda^{0}}{0!} e^{-\lambda} \\
&= \frac{4^{0}}{0!} e^{-4} \\
&= 0{,}02
\end{aligned}$$

- O gráfico desta distribuição (até $X = 20$) é



9.7.2 No geral

- A distribuição de Poisson é um bom modelo para situações em que a variável aleatória X conta o número de ocorrências de algum fenômeno em um intervalo de tempo fixo (ou em uma área fixa, ou em um volume fixo de espaço):
  - Carros passando em um cruzamento em uma hora,
  - Chamadas telefônicas chegando por minuto a uma central,
  - Partículas alfa emitidas por minuto por 1Kg de um material radioativo,
  - Casos de uma doença detectados em cada Km^2 de uma cidade,
  - Erros de impressão por página em livros produzidos por uma editora.
- Teoricamente, $P(X = x)$ é maior que zero para qualquer x natural.
- No mundo real, existem valores máximos que estas variáveis aleatórias podem assumir. Ainda assim, esta distribuição é uma boa aproximação para os valores que podem surgir na prática. Os valores não-realistas têm probabilidade muito baixa.
- Por exemplo, a probabilidade de 40 ou mais clientes chegarem ao nosso caixa de mercado em um minuto é de $0{,}000000000000000000000000003$.
- A probabilidade de X assumir o valor x quando a quantidade média de ocorrências por intervalo é λ é igual a

$$\text{Poisson}(X = x \mid \lambda) = \frac{\lambda^x}{x!} e^{-\lambda}$$

- O valor esperado $E(X)$ e a variância $\sigma^2(X)$ são iguais a λ.

9.7.3

Em R

Exercício

É você quem vai escrever esta seção.

Seguindo o estilo das explicações sobre a distribuição geométrica e a distribuição binomial, faça o seguinte:

1. Gere gráficos com exemplos do uso de `dpois`.
2. Gere gráficos com exemplos do uso de `ppois`.
3. Gere gráficos com exemplos do uso de `qpois`.
4. Faça uma simulação usando `rpois`. Detalhe o exemplo do nosso pequeno mercado, que atende 4 clientes por minuto:
    a. Gere uma amostra de números correspondendo a 1 hora de funcionamento do caixa.
    b. Escreva uma função em R para processar esta amostra e retornar um vetor com a quantidade de clientes na fila ao final de cada minuto (de 1 a 60). Lembre-se de que o caixa processa 4 clientes por minuto.
    c. Agora gere uma amostra de números correspondendo a 1 hora de funcionamento do caixa, com uma média de 5 clientes chegando por minuto.
    d. Use a função que você escreveu para processar esta amostra e retornar um vetor com a quantidade de clientes na fila a cada minuto (de 1 a 60), supondo que o caixa ainda processa só 4 clientes por minuto.

9.7.4

Aproximação da Binomial pela Poisson

- Um fabricante de carros descobre que 1 em cada 2.500 carros que ele produz tem um defeito.
- Qual a probabilidade de achar 4 carros com defeito em uma amostra de 6.000 carros?
- Vamos usar Binom $\left(X \mid n = 6.000, p = \frac{1}{2.500}\right)$:

```r
prob <- 1/2500
size <- 6000
x <- 4
dbinom(x, size, prob)
```

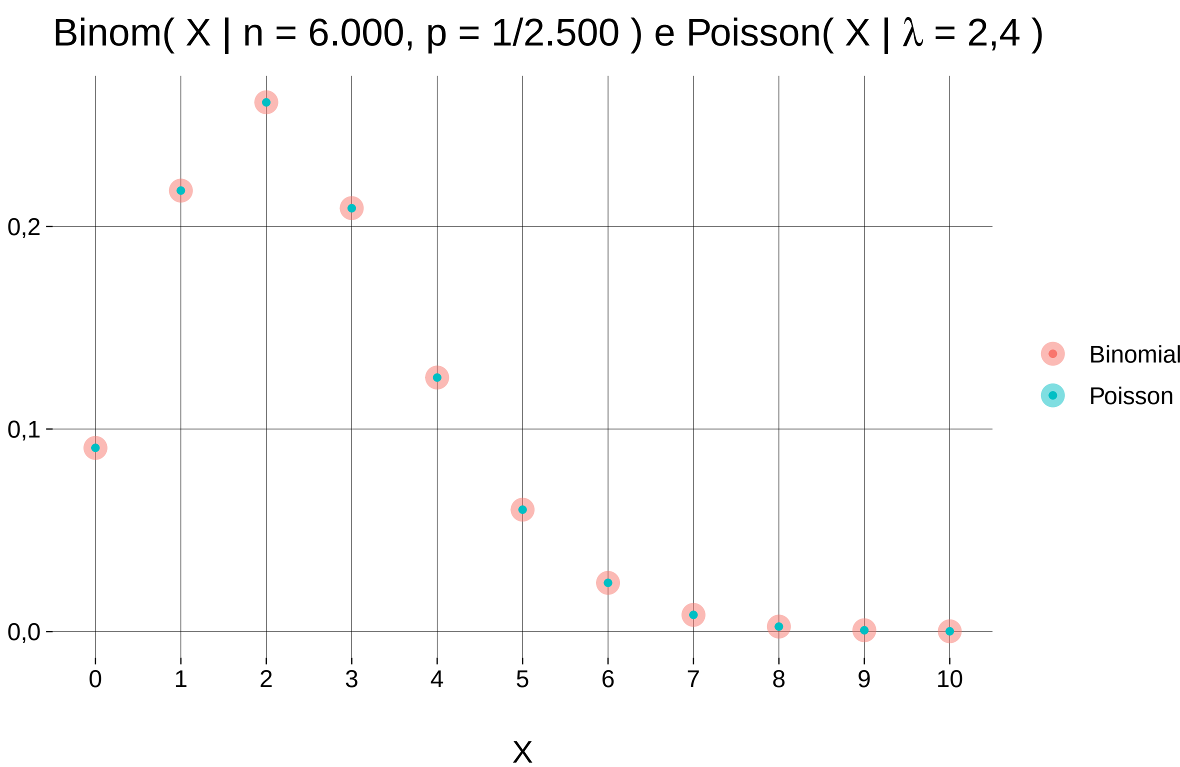
```
## [1] 0,1254235
```

- Mas, e se modelarmos o problema com Poisson em vez da Binomial?
- Uma das condições para usar Poisson é que a média seja constante. Para 6.000 carros, a média de defeitos é $\frac{6.000}{2.500} = 2{,}40$.
- Vamos usar, então, Poisson $(X \mid \lambda = 2{,}40)$:

```r
lambda <- 6000/2500
x <- 4
dpois(x, lambda)
```

```
## [1] 0,1254085
```

- Eis um gráfico comparando as duas distribuições:



- Para valores grandes de n (6.000) e pequenos de p $(1/2.500)$, a Binomial pode ser aproximada pela Poisson — desde que você calcule a média λ, como fizemos.
- A vantagem é que, diferente da Binomial, a Poisson não exige o cálculo do valor de $\binom{n}{x}$, que pode ser inviável mesmo com um computador.

Por quê?

O exemplo da quantidade de emissões de partículas alfa por minuto por uma massa fixa de material radioativo mostra uma relação fundamental entre as duas distribuições:

- Do ponto de vista de Poisson, sabemos a média de emissões por minuto do material, sem levar em consideração o comportamento de cada átomo.
- Do ponto de vista da Binomial, o material é composto por um número enorme de átomos (n é grande!), cada átomo com uma probabilidade fixa e muito baixa (p pequeno!) de emitir uma partícula alfa no período de um minuto.
- Matematicamente, a função de probabilidade de Poisson é o limite da função de probabilidade da Binomial quando n tende a infinito e p tende a zero, com o valor de np fixo em λ.

9.8 Funções para distribuições em R

- A esta altura, você já entendeu como funcionam as funções do R para distribuições discretas de probabilidade.
- Para uma distribuição de nome `DISTRIB`, as funções são as seguintes (`...` representam os parâmetros da distribuição):
  - `dDISTRIB(x, ...)`: retorna a probabilidade $P(X = x \mid \dots)$.
- `pDISTRIB(q, ...)`: retorna a probabilidade $P(X \leq q \mid \dots)$ — ou a probabilidade $P(X > q \mid \dots)$ se `lower.tail = FALSE`.
- `qDISTRIB(p, ...)`: retorna o maior valor q tal que $P(X = q \mid \dots) \leq p$ — ou o menor valor q tal que $P(X = q \mid \dots) > p$ se `lower.tail = FALSE`.
- `rDISTRIB(n, ...)`: retorna um vetor com n valores sorteados de acordo com a distribuição.

9.9 Jardim zoológico de distribuições

- Para sua diversão: https://ben18785.shinyapps.io/distribution-zoo/

CAPÍTULO 10

Distribuições contínuas

10.1 Vídeo

https://youtu.be/TrmDPVYTpHc

10.2 Distribuição uniforme

10.2.1 Exemplo: gerador de números aleatórios

- Um programa para gerar números reais aleatórios no intervalo $[0, 10]$.
- A probabilidade é distribuída uniformemente neste intervalo.
- A variável aleatória X representa os números gerados.

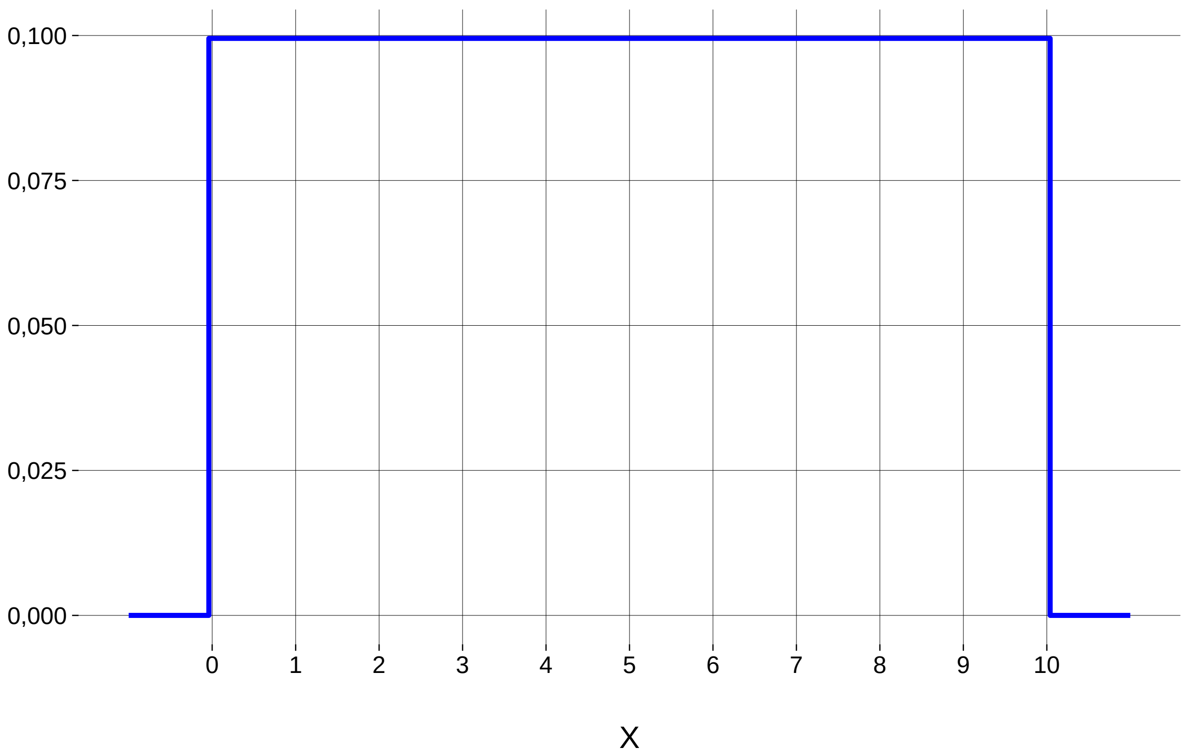
$P(X = x) = 0!$

Como X é uma variável aleatória contínua, as probabilidades $P(X = 1)$, $P(X = 3{,}1416)$, ou qualquer outra probabilidade pontual $P(X = x)$, são iguais a zero!

- A fdp (função de densidade de probabilidade) de X é

$$f(x) = \begin{cases} \frac{1}{10} & \text{se } x \in [0, 10] \\ 0 & \text{se } x \notin [0, 10] \end{cases}$$

cujo gráfico é



Densidade não é probabilidade!

Perceba que a probabilidade $P(X = 5)$ é 0, mas a densidade $f(5)$ é $1/10$.

- A probabilidade de um número gerado estar entre 5 e 7 é de

$$P(5 < X < 7) = P(5 \leq X \leq 7) = \int_5^7 f(x)\ \mathrm{d}x = \int_5^7 \frac{1}{10}\ \mathrm{d}x = \frac{1}{5}$$

- O valor esperado de X é

$$E(X) = \int_{-\infty}^{\infty} x f(x) \, \mathsf{d}x = \int_{0}^{10} \frac{1}{10} x \, \mathsf{d}x = 5$$

- A variância de X é

$$\sigma^2(X) = \int_{-\infty}^{+\infty} (x-5)^2 \cdot f(x) \, \mathsf{d}x = \int_{0}^{10} (x-5)^2 \cdot \frac{1}{10} \, \mathsf{d}x = \frac{25}{3} = 8{,}33$$

- **Exercício:** calcule $\sigma^2(X)$ usando a fórmula $\sigma^2(X) = E(X^2) - [E(X)]^2$.

10.2.2 No geral

- Escrevemos $X \sim \text{Unif}(a, b)$.
- Os extremos do intervalo são $a, b \in \mathbb{R}, a < b$.
- O suporte é o intervalo $[a, b]$.
- A função de densidade de probabilidade é

$$f(x \mid a, b) = \begin{cases} \frac{1}{b-a} & \text{se } x \in [a, b] \\ 0 & \text{se } x \notin [a, b] \end{cases}$$

- Valor esperado:

$$E(X) = \frac{a+b}{2}$$

- Variância:

$$\sigma^2(X) = \frac{(a-b)^2}{12}$$

10.2.3 Em R

Função de densidade de probabilidade: $f(x \mid a, b)$

- `dunif(x, min = a, max = b)`.

- No exemplo do gerador de números aleatórios em $[0, 10]$, todos os valores no intervalo têm a mesma densidade: $\frac{1}{10}$. Valores fora do intervalo têm densidade 0:

```
dunif(
  c(-1, 1, 2, 10, 11),
  min = 0,
  max = 10
)
```

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

Função de probabilidade acumulada: Unif$(X \leq q \mid a, b)$

- `punif(q, min = a, max = b)`.
- No exemplo do gerador de números aleatórios no intervalo $[0, 10]$, qual a probabilidade de obter um número menor que 4?

```
punif(4, min = 0, max = 10)
```

```
## [1] 0,4
```

- E um número maior que 4?

```
punif(4, min = 0, max = 10, lower.tail = FALSE)
```

```
## [1] 0,6
```

Função quantil: dado um valor de Unif$(X \leq x \mid a, b)$, então $x = ?$

- O objetivo é achar x tal que Unif$(X \leq x \mid a, b) = m$.
- `qunif(p, min = a, max = b)`.
- No exemplo do gerador de números aleatórios no intervalo $[0, 10]$, qual o número x tal que existe uma probabilidade de 80% de que um número menor ou igual a x seja gerado?

```
qunif(.8, min = 0, max = 10)
```

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

- E qual o número x tal que existe uma probabilidade de 80% de que um número maior ou igual a x seja gerado?

```
qunif(.8, min = 0, max = 10, lower.tail = FALSE)
```

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

Função para gerar números aleatórios

- `runif(n, min = a, max = b)`
- Os números gerados são de ponto flutuante:

```
runif(10, min = 0, max = 10)
```

```
##  [1] 6,195134 9,606586 3,167580 3,720381 1,325058 2,860137 7,216193
##  [8] 7,550518 6,066161 7,476152
```

- **Exercício:** e se você quiser gerar apenas números inteiros no intervalo dado?
- **Exercício:** os extremos do intervalo (`min` e `max`) podem ser gerados?

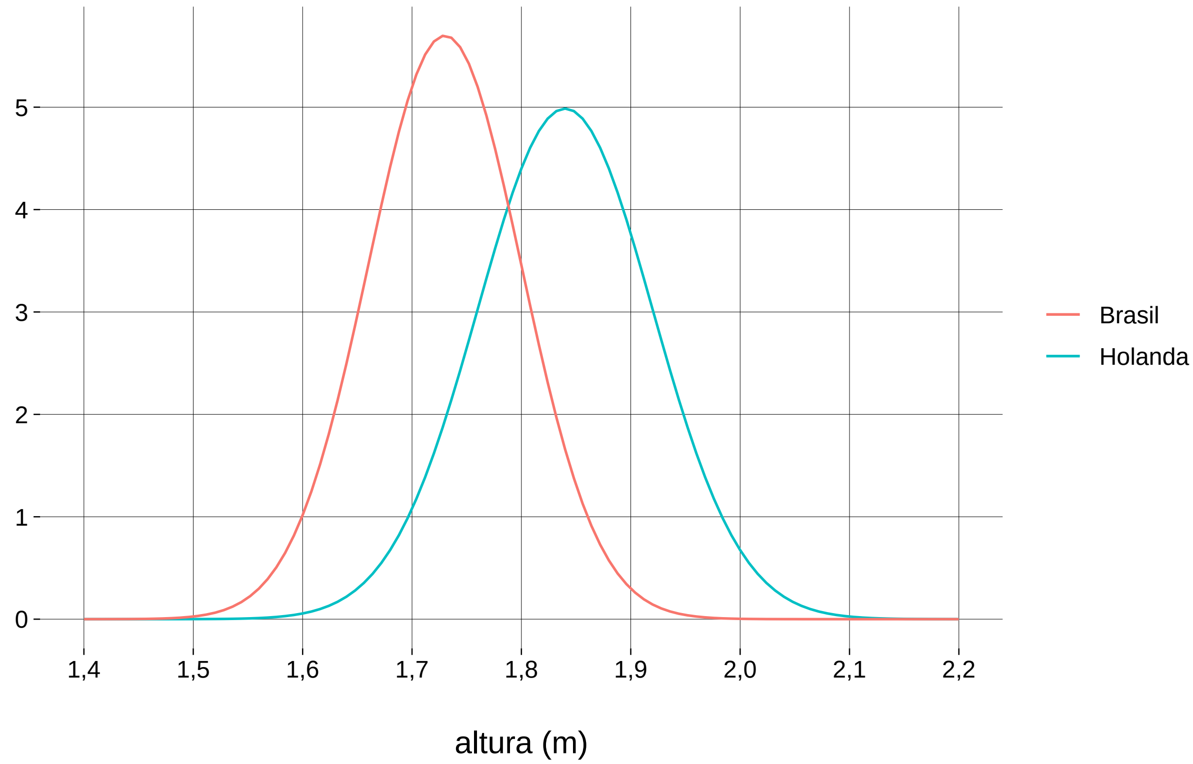
10.3 Distribuição normal

10.3.1 Exemplo

- Segundo o Levantamento do Perfil Antropométrico da População Brasileira Usuária do Transporte Aéreo Nacional, em 2009, a estatura do homem brasileiro adulto era distribuída normalmente, com média $1{,}73$m e desvio-padrão $0{,}07$m.
- Segundo este livro, a estatura do homem holandês é distribuída normalmente, com média $1{,}84$m e desvio-padrão $0{,}08$m.

```
mu_brasil <- 1.73
sigma_brasil <- .07

mu_holanda <- 1.84
sigma_holanda <- .08
```

- Gráfico:



- **Exercício:** o que são os números no eixo vertical?
- Por que a curva do Brasil é mais alta?

10.3.2
No geral

- Escrevemos $X \sim \mathcal{N}(\mu, \sigma)$.
- O valor esperado é $\mu \in \mathbb{R}$.
- O desvio-padrão é $\sigma \in \mathbb{R}_{\geq 0}$.
- O suporte é o intervalo$(-\infty, \infty)$.
- A função de densidade de probabilidade é:

$$f(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

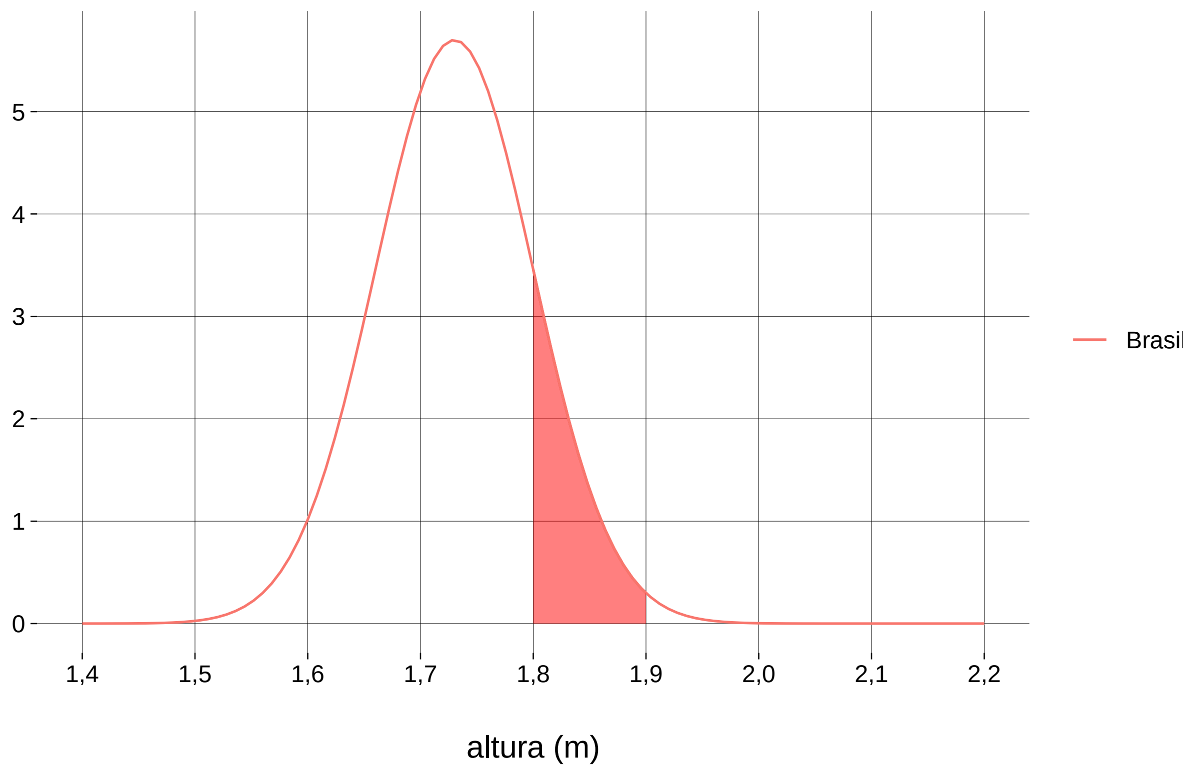
10.3.3 Em R

Função de densidade de probabilidade: $f(x \mid \mu, \sigma)$

- `dnorm(x, mean =  , sd =  )`.

Função de probabilidade acumulada: Norm$(X \leq q \mid \mu, \sigma)$

- `pnorm(q, mean =  , sd =  )`.
- Qual a probabilidade de um homem brasileiro ter entre $1{,}80$m e $1{,}90$m de altura?
- Gráfico:



- Menos que $1{,}80$m:

```
pnorm(1.80, mu_brasil, sigma_brasil)
```

```
## [1] 0,8413447
```

- Menos que $1{,}90$m:

```
pnorm(1.90, mu_brasil, sigma_brasil)
```

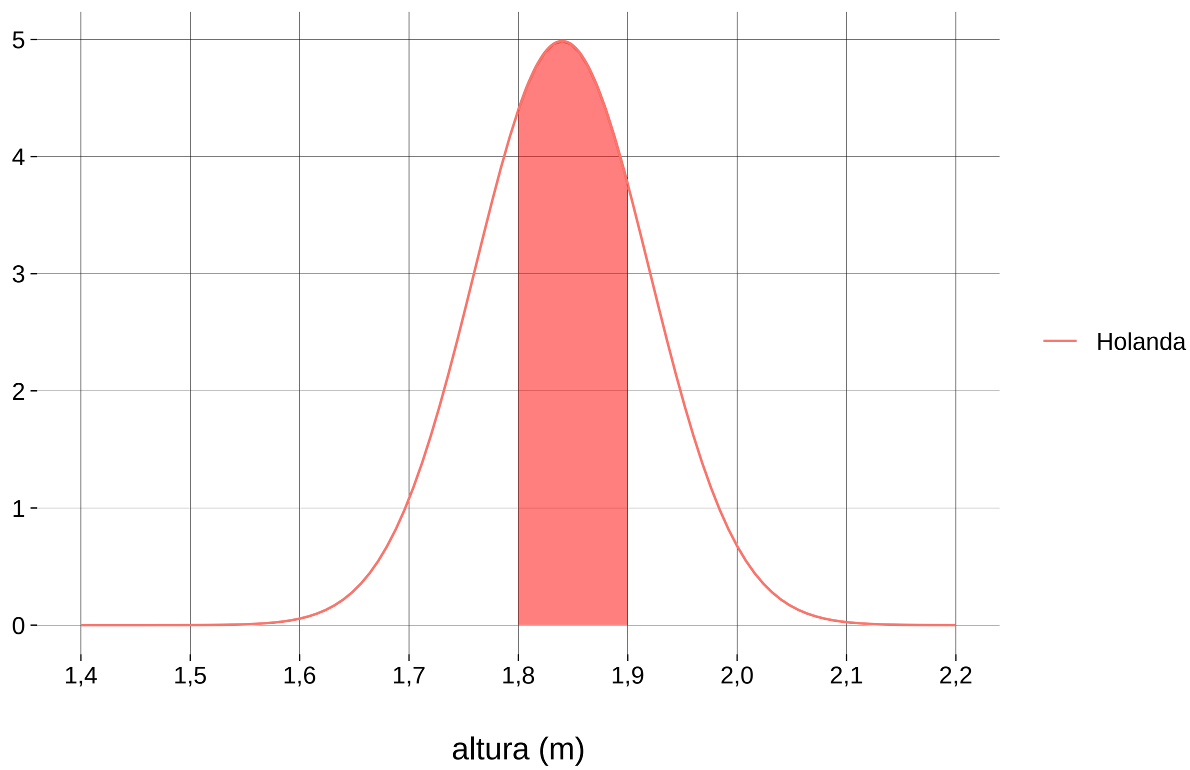
```
## [1] 0,9924208
```

- Entre $1{,}80$m e $1{,}90$m:

```
pnorm(1.90, mu_brasil, sigma_brasil) -
  pnorm(1.80, mu_brasil, sigma_brasil)
```

```
## [1] 0,151076
```

- Qual a probabilidade de um homem holandês ter entre $1{,}80$m e $1{,}90$m de altura?
- Gráfico:



- Menos que $1{,}80$m:

```
pnorm(1.80, mu_holanda, sigma_holanda)
```

```
## [1] 0,3085375
```

- Menos que $1{,}90$m:

```
pnorm(1.90, mu_holanda, sigma_holanda)
```

```
## [1] 0,7733726
```

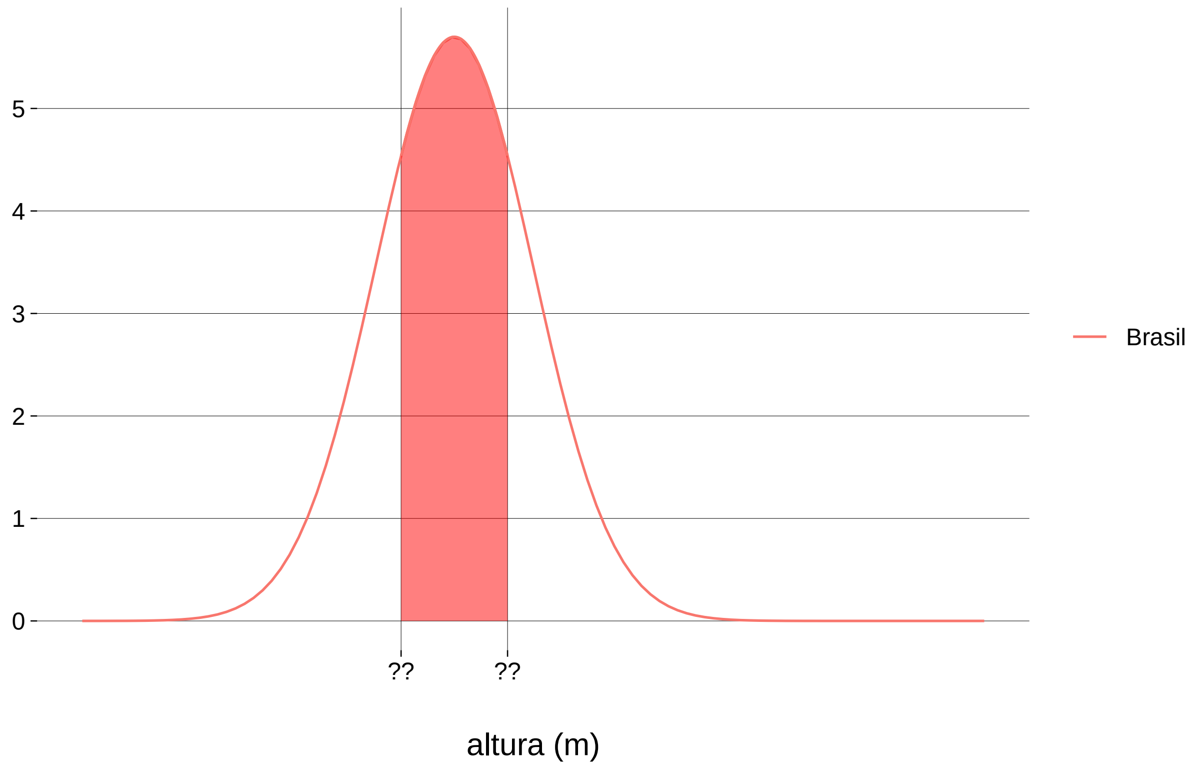
- Entre $1{,}80$m e $1{,}90$m:

```
pnorm(1.90, mu_holanda, sigma_holanda) -
  pnorm(1.80, mu_holanda, sigma_holanda)
```

```
## [1] 0,4648351
```

Função quantil: dado um valor de Norm$(X \leq x \mid \mu, \sigma)$, então $x = ?$

- `qnorm(p, mean = , sd = ).`
- Quais as alturas mínima e máxima dos homens que estão nos 50% da população em torno da média?
- Gráfico:



- O limite inferior é o valor à esquerda do qual existem 25% de probabilidade:

```
qnorm(.25, mu_brasil, sigma_brasil)
```

```
## [1] 1,682786
```

- O limite superior é o valor à esquerda do qual existem 75% de probabilidade:

```
qnorm(.75, mu_brasil, sigma_brasil)
```

```
## [1] 1,777214
```

- Ou, equivalentemente, o valor à direita do qual existem 25% de probabilidade:

```
qnorm(.25, mu_brasil, sigma_brasil, lower.tail = FALSE)
```

```
## [1] 1,777214
```

- As alturas mínima e máxima dos homens que estão nos 50% da população em torno da média são $1{,}68$m e $1{,}78$m.

Função para gerar números aleatórios

- `rnorm(n, mean = , sd = ).`
- Qual é a probabilidade de um homem holandês escolhido ao acaso ser mais baixo do que um homem brasileiro escolhido ao acaso?
- Vamos responder fazendo uma simulação.
- Vamos repetir muitas vezes o experimento de escolher um holandês ao acaso e um brasileiro ao acaso:

```
n <- 1e6
holandeses <- rnorm(n, mu_holanda, sigma_holanda)
brasileiros <- rnorm(n, mu_brasil, sigma_brasil)
```

- Qual a proporção de vezes em que o holandês é mais baixo?

```
mean(holandeses < brasileiros)
```

```
## [1] 0,150608
```

- Segundo nossa simulação, esta é a resposta.

10.3.4 *QQplots*

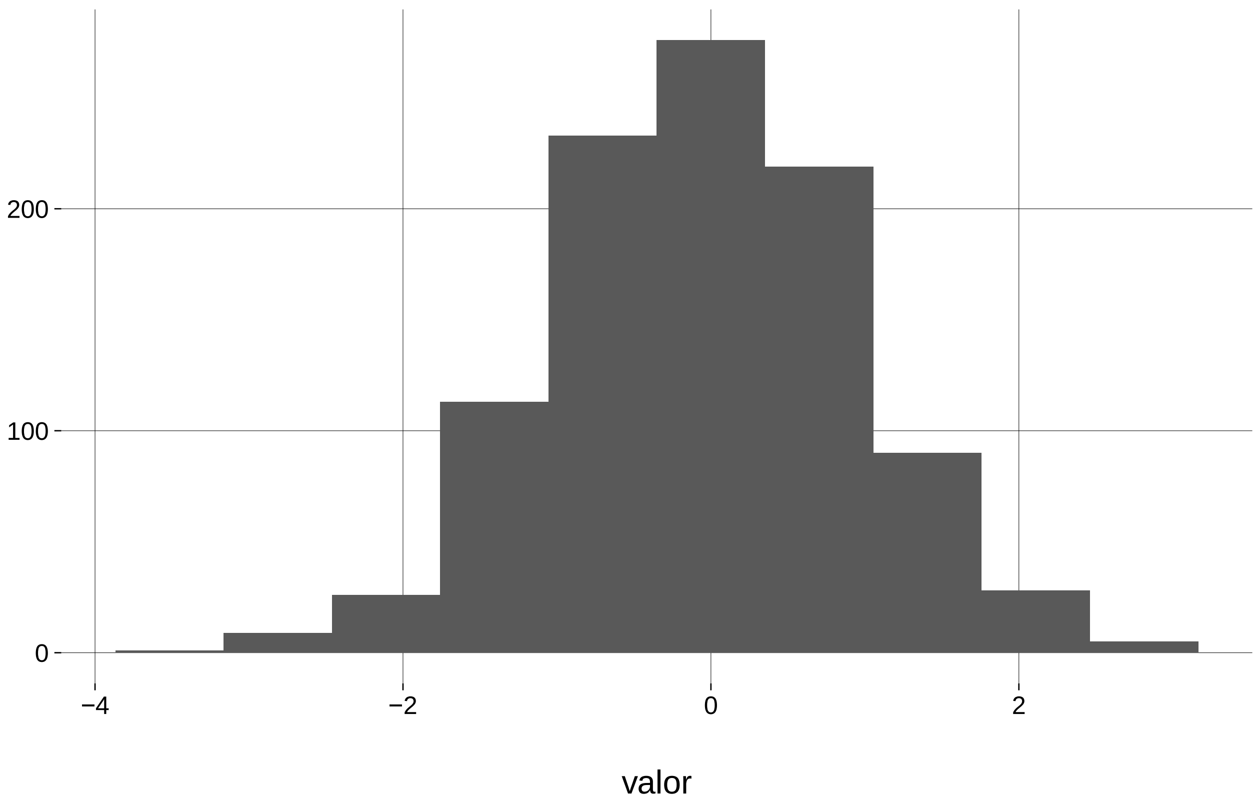
- Como saber se um conjunto de dados numéricos segue a distribuição normal?
- Vamos gerar dados aproximadamente normais (média 0, desvio padrão 1):

```
valores <- rnorm(1000)
head(valores)
```

```
## [1] -0,05378437  0,12248840  1,34226789  1,07468061 -0,99897604
## [6] -0,48099153
```

- Histograma:

```r
valores %>%
  as_tibble() %>%
  ggplot() +
    geom_histogram(aes(x = value), bins = 10) +
    labs(
      x = 'valor',
      y = NULL
    )
```



- Para estes dados, os quartis são

```r
quantile(valores)
```

```
##          0%         25%         50%         75%        100% 
## -3,35177281 -0,74657613 -0,02370021  0,62391244  2,97670459
```

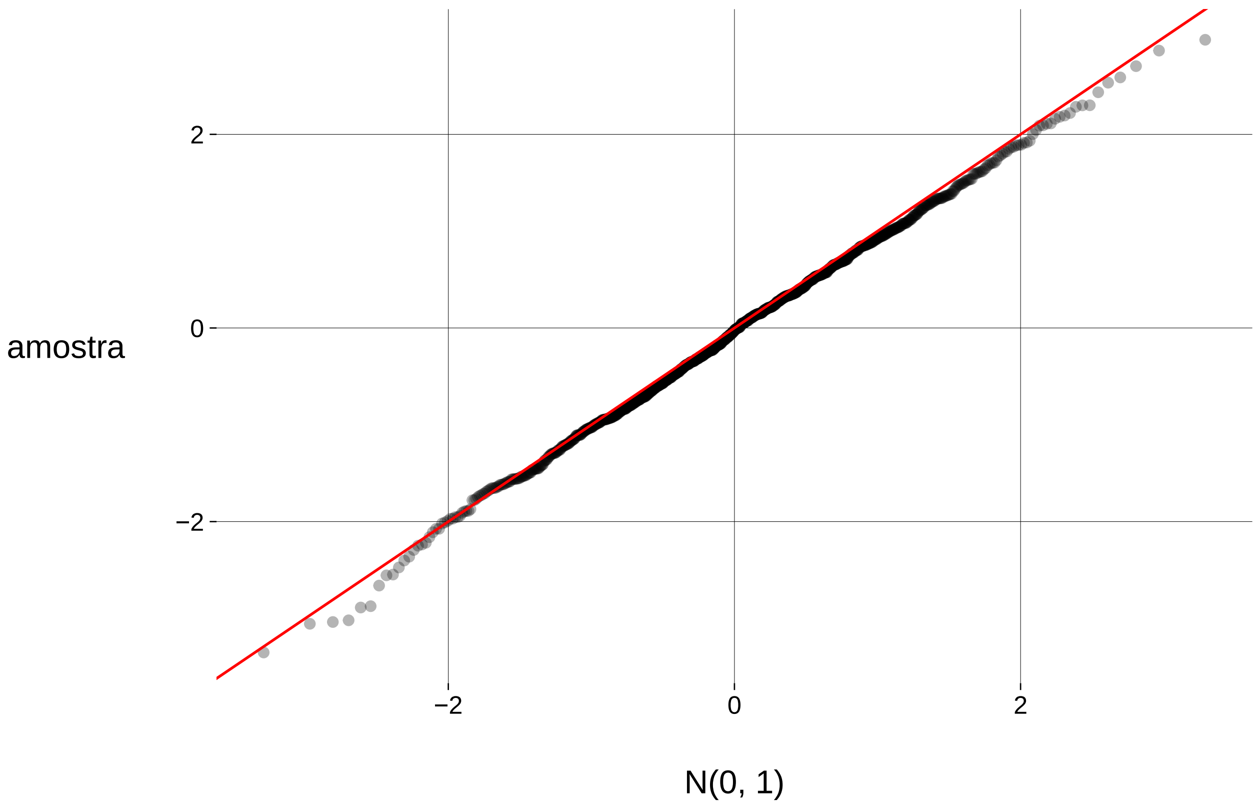
- Na distribuição $\mathcal{N}(0, 1)$ teórica, os quartis seriam

```r
qnorm(c(0, .25, .5, .75, 1))
```

```
## [1]       -Inf -0,6744898  0,0000000  0,6744898        Inf
```

- O gráfico QQ (quantil-quantil) mostra os quantis da amostra no eixo y e os quantis teóricos no eixo x. A reta vermelha serve para referência; se as duas distribuições fossem idênticas, todos os pontos estariam sobre a reta.

```
valores %>%
  as_tibble() %>%
  ggplot(aes(sample = value)) +
    geom_qq(alpha = .3) +
    labs(
      y = 'amostra',
      x = 'N(0, 1)'
    ) +
    geom_abline(color = 'red')
```

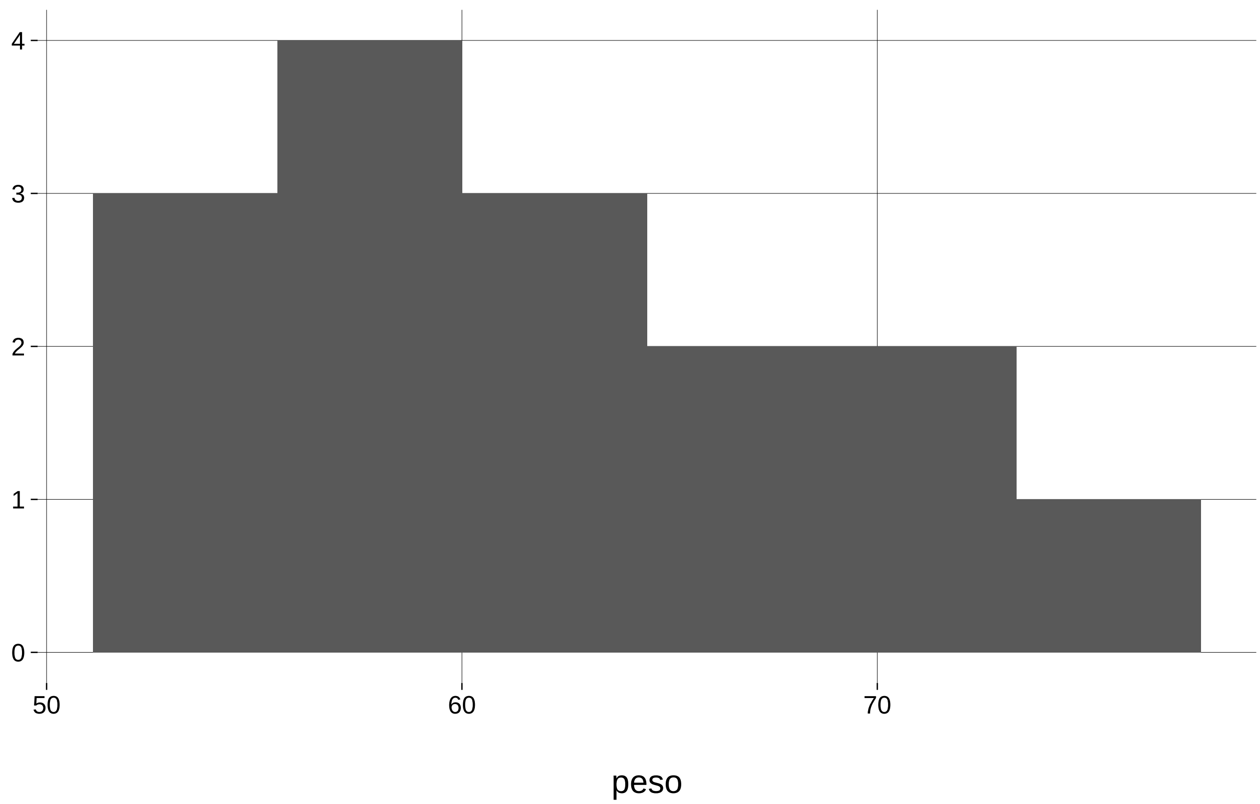


- Um exemplo concreto: pesos de 15 mulheres com idades entre 30 e 39 anos:

```
pesos <- women %>%
  select(weight) %>%
  transmute(peso = weight * .4535924)
```

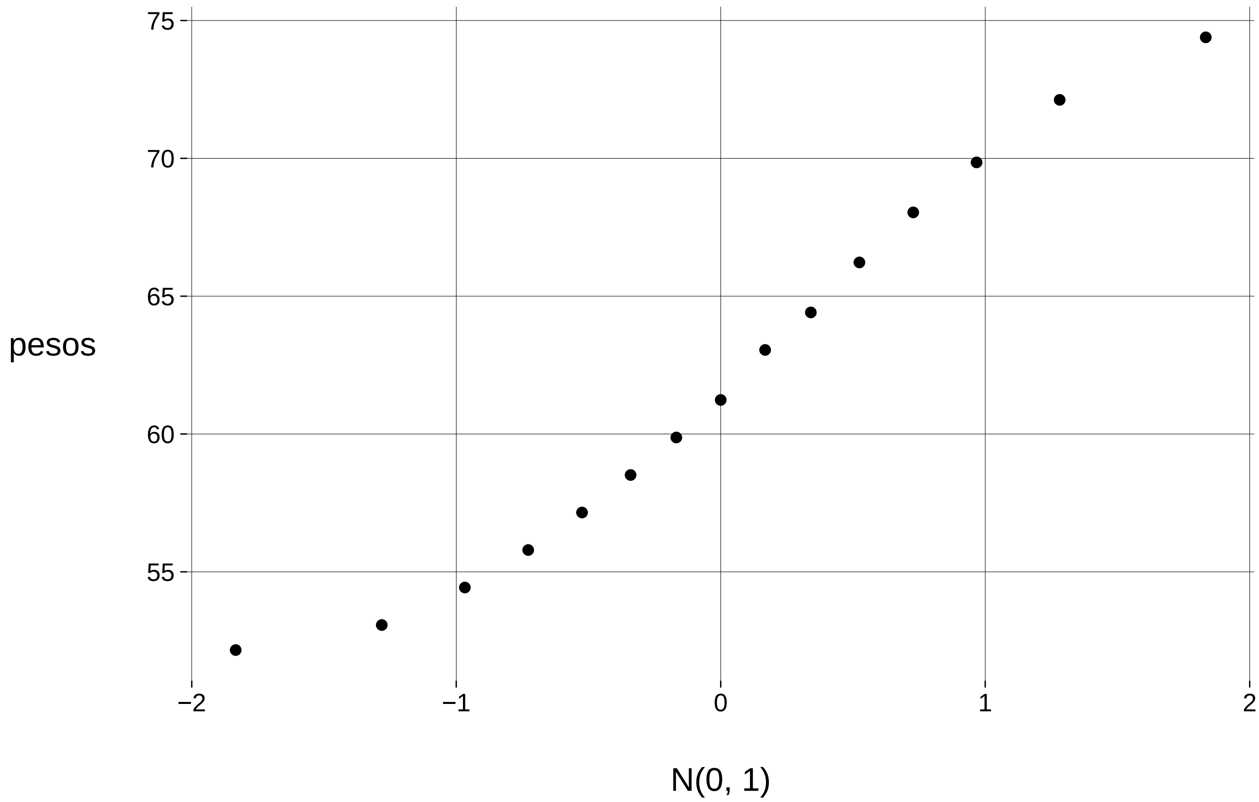
- Histograma:

```r
pesos %>%
  ggplot() +
    geom_histogram(aes(x = peso), bins = 6) +
    labs(y = NULL)
```



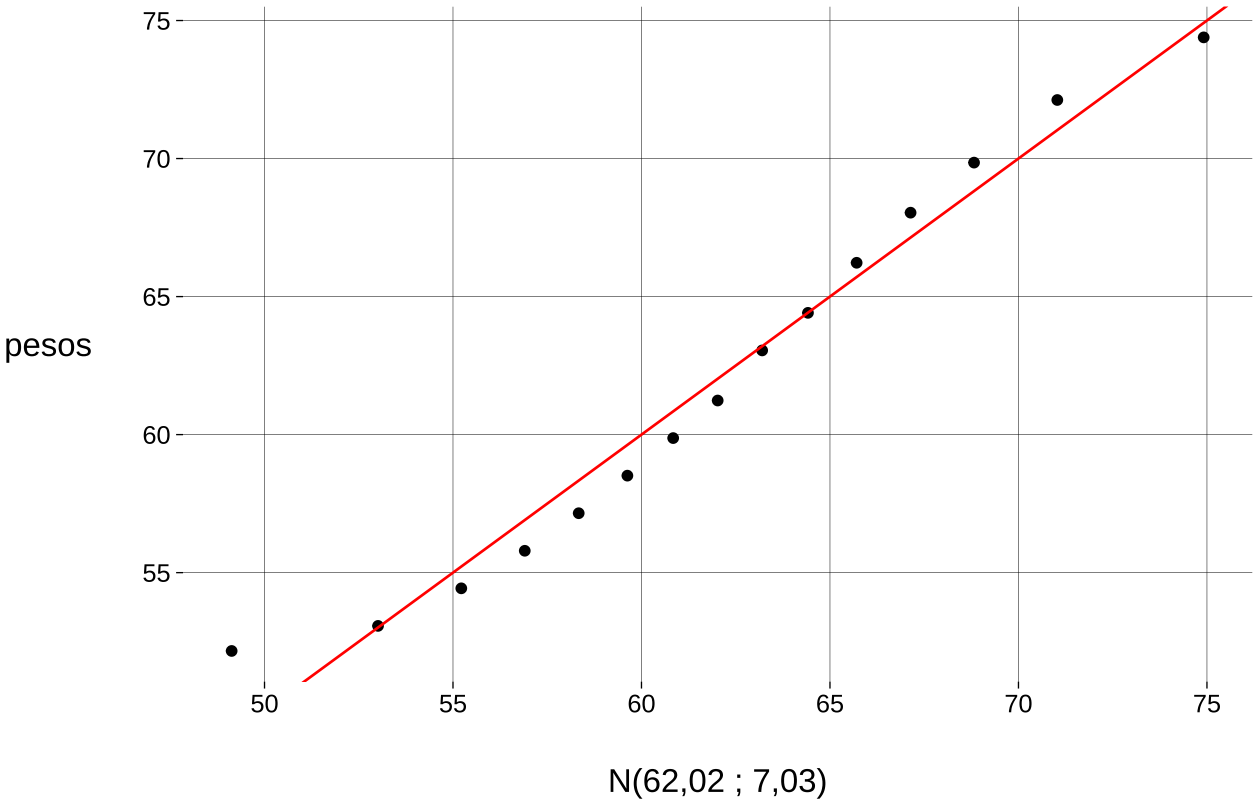
- O gráfico quantil-quantil fica

```r
pesos %>%
  ggplot(aes(sample = peso)) +
    geom_qq() +
    labs(
      y = 'pesos',
      x = 'N(0, 1)'
    )
```



- Quanto mais próximos de uma reta com inclinação de 45 graus os pontos ficarem, mais próximos da distribuição normal os dados estão.
- Podemos usar, como distribuição teórica no eixo x, a distribuição normal com média e desvio padrão iguais à média e ao desvio padrão dos dados:

```
m <- mean(pesos$peso)
s <- sd(pesos$peso)

pesos %>%
  ggplot(aes(sample = peso)) +
    geom_qq(
      dparams = list(
        mean = m,
        sd = s
      )
    ) +
    geom_abline(color = 'red') +
    labs(
      y = 'pesos',
      x = paste0('N(', m %>% round(2), ' ; ', s %>% round(2), ')')
    )
```



- Ou podemos padronizar os dados no eixo y, fazendo com que a média deles seja 0 e o desvio padrão deles seja 1.
- Para fazer isso manualmente, basta subtrair a média e dividir pelo desvio padrão:

```
pesos %>%
  mutate(peso = (peso - mean(peso)) / sd(peso)) %>%
  kbl(
    format.args = list(big.mark = '.')
  ) %>%
  kable_paper(
    c('striped', 'hover'),
    full_width = FALSE
  )
```

- Ou podemos usar a função `scale`, que faz o mesmo:

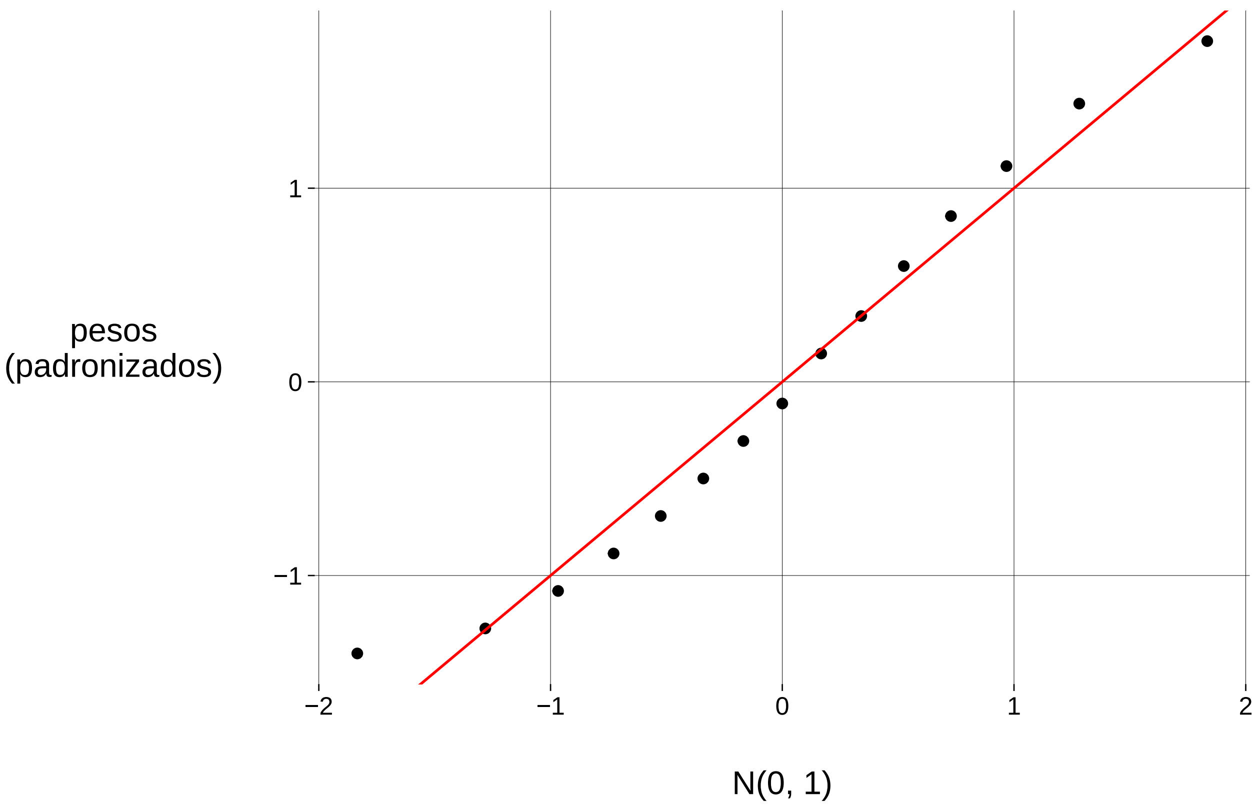
```
pesos %>%
  mutate(peso = scale(peso)) %>%
  kbl(
    format.args = list(big.mark = '.')
  ) %>%
  kable_paper(
    c('striped', 'hover'),
```

peso
-1,4022687
-1,2732255
-1,0796608
-0,8860962
-0,6925315
-0,4989668
-0,3054021
-0,1118374
0,1462489
0,3398136
0,5978998
0,8559861
1,1140723
1,4366802
1,7592880

```
    full_width = FALSE
  )
```

```
pesos %>%
  mutate(peso = scale(peso)) %>%
  ggplot(aes(sample = peso)) +
    geom_qq() +
    labs(
      y = 'pesos\n(padronizados)',
      x = 'N(0, 1)'
    ) +
    geom_abline(color = 'red')
```

peso
-1,4022687
-1,2732255
-1,0796608
-0,8860962
-0,6925315
-0,4989668
-0,3054021
-0,1118374
0,1462489
0,3398136
0,5978998
0,8559861
1,1140723
1,4366802
1,7592880

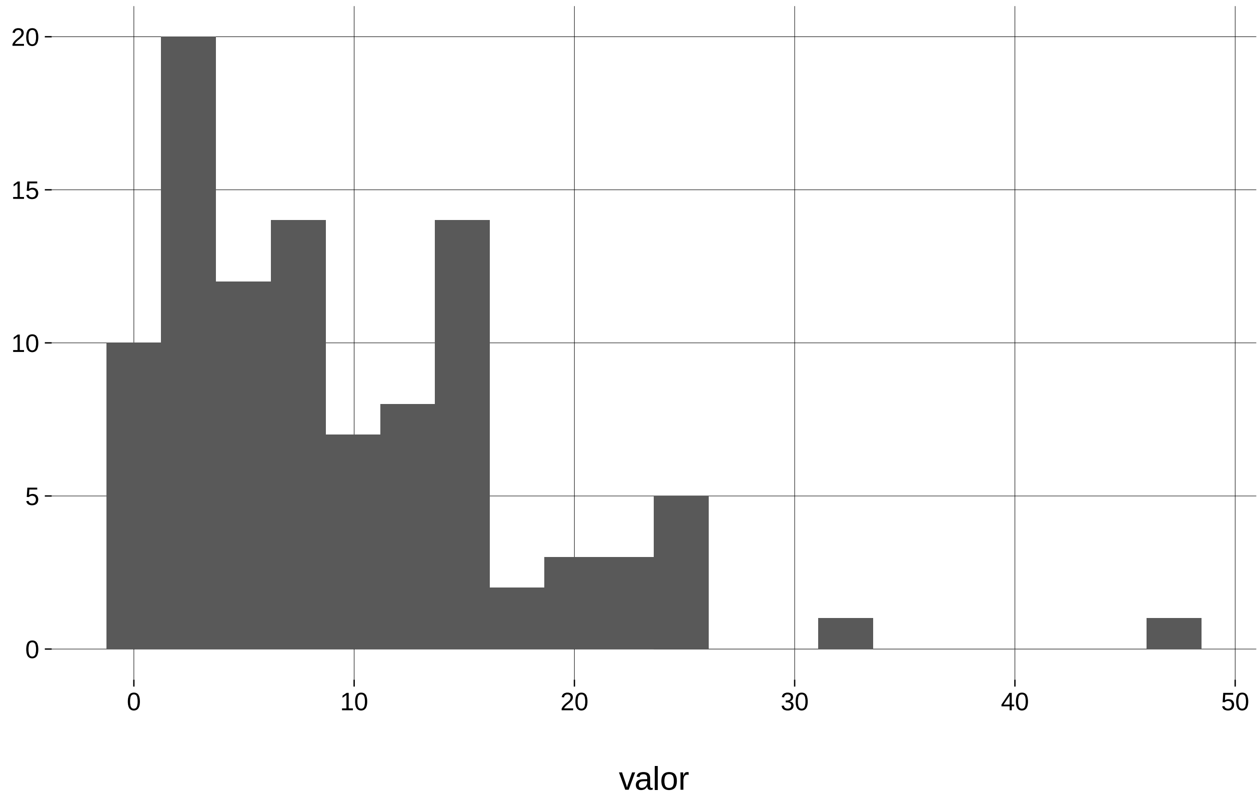


- Um exemplo de dados não normais:

```
dados <- rexp(100, rate = .1)
m <- mean(dados)
s <- sd(dados)
```

- Histograma:

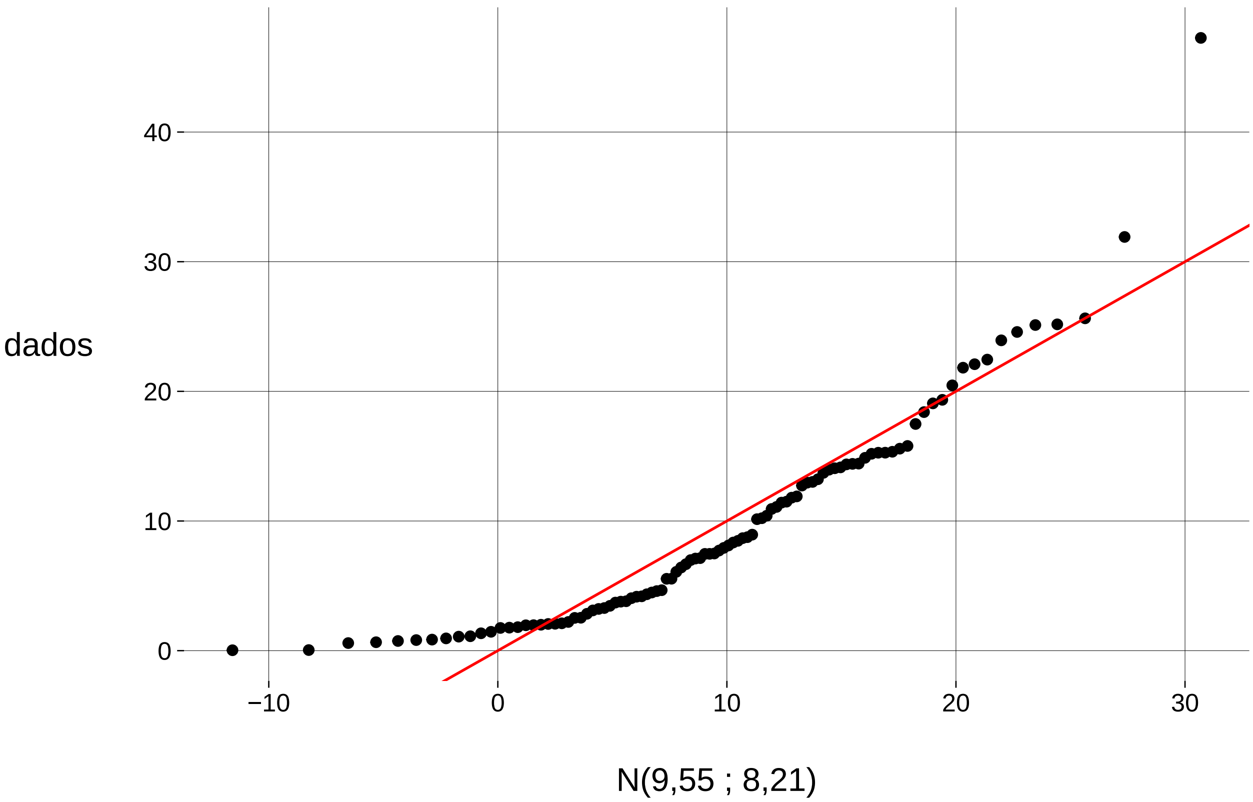
```
dados %>%
  as_tibble() %>%
  ggplot() +
    geom_histogram(aes(x = value), bins = 20) +
    labs(
      x = 'valor',
      y = NULL
    )
```



- **QQ:**

```
dados %>%
  as_tibble() %>%
  ggplot(aes(sample = value)) +
    geom_qq(
      dparams = list(
        mean = m,
        sd = s
      )
    ) +
    geom_abline(color = 'red') +
    labs(
      y = 'dados',
```

```r
    x = paste0('N(', m %>% round(2), ' ; ', s %>% round(2), ')')
  )
```



- **Exercício:** gere um gráfico quantil-quantil para estes dados, usando a distribuição exponencial com média 10 como distribuição teórica no eixo x.

10.3.5

Aproximação da binomial pela normal

- Exemplo:
  - Uma organização vai receber sangue de 32 mil doadores escolhidos ao acaso.
  - A probabilidade de um doador ter sangue tipo O- é $p = 0{,}06$.
  - Qual a probabilidade de a organização conseguir no mínimo 1.850 doadores com sangue do tipo O-?
- Cada doador é uma prova de Bernoulli, com probabilidade de sucesso $p = 0{,}06$. Supondo que os doadores são independentes (por exemplo, não são todos parentes) e usando a distribuição binomial, a resposta é

$$\text{Binom}(X \geq 1850 \mid n = 32000, p = 0.06)$$

- Em R

```
pbinom(
  q = 1849,
  size = 32000,
  prob = 0.06,
  lower.tail = FALSE
)
```

```
## [1] 0,9521106
```

- Este resultado é o valor da longa e trabalhosa expressão

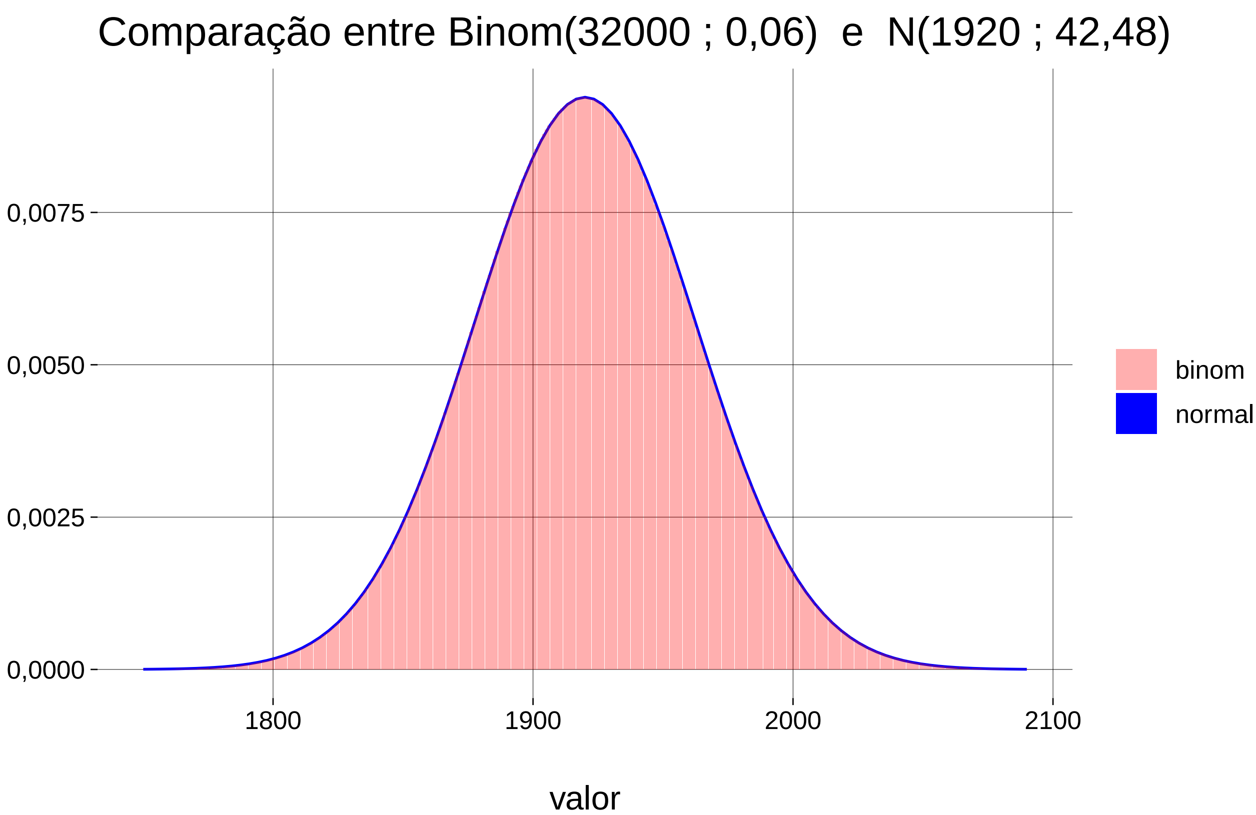
$$\binom{32000}{1850}0{,}06^{1850}0{,}94^{30150}+\binom{32000}{1851}0{,}06^{1851}0{,}94^{30149}+\cdots+\binom{32000}{32000}0{,}06^{32000}0{,}94^{0}$$

- Felizmente, podemos usar a distribuição normal para achar uma aproximação:
  - A média da distribuição binomial é $np = 32.000 \cdot 0{,}06 = 1920$.
  - O desvio padrão da distribuição binomial é $\sqrt{np(1-p)} \approx 42{,}48$.
  - Usando uma normal com esta média e este desvio padrão, temos

```
pnorm(
  q = 1849,
  mean = 1920,
  sd = 42.48,
  lower.tail = FALSE
)
```

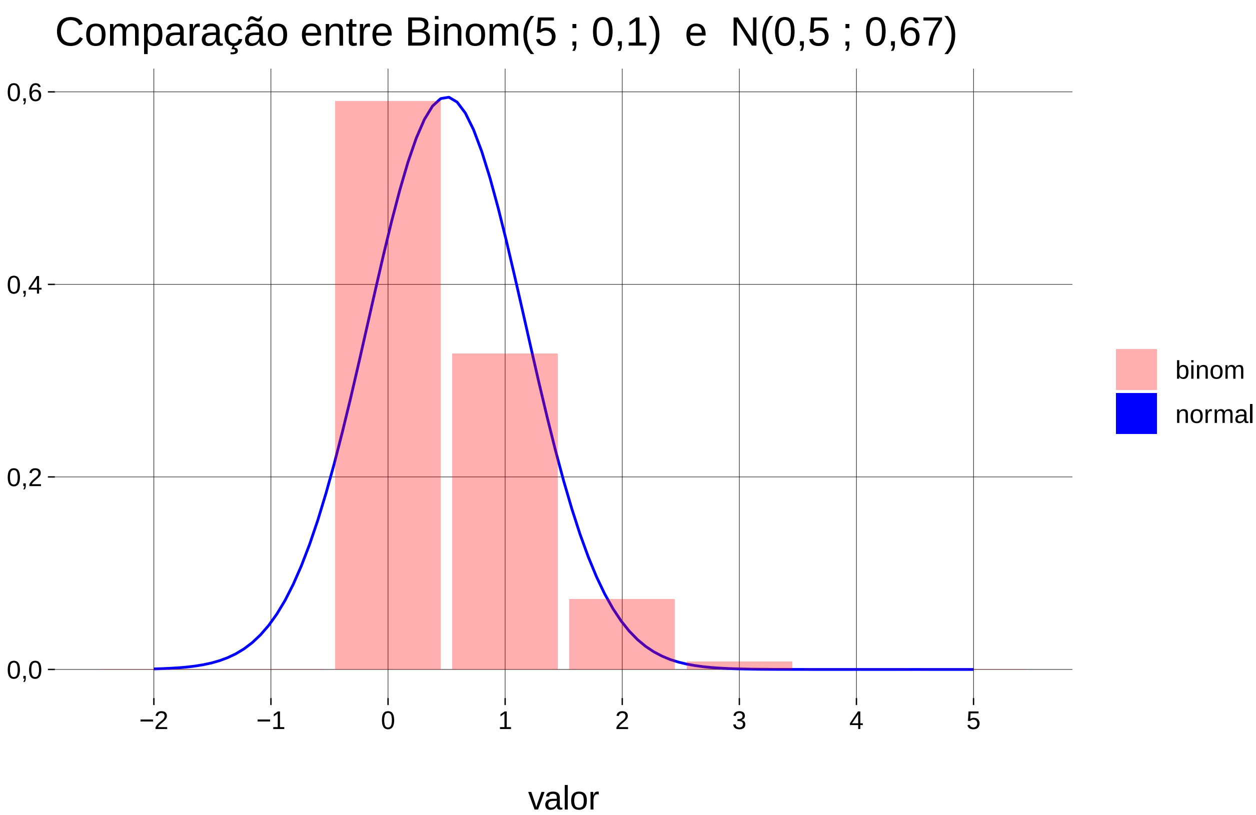
```
## [1] 0,9526762
```

- As distribuições são mesmo parecidas:



- Mas nem toda binomial se parece com uma normal:

```
n <- 5
p <- .1
normal_binom(n, p, limites = c(-2, 5)) +
  scale_x_continuous(breaks = -2:5)
```



- Neste exemplo, quanto da probabilidade da normal está em valores abaixo de -0.5 (valores que não fazem sentido para a binomial!)?

```
pnorm(-.5, n * p, sqrt(n * p * (1 - p)))
```

```
## [1] 0,06801856
```

- Como resolver isto?
- Lembrando que $99{,}7\%$ da probabilidade da distribuição normal está a 3 desvios padrão de distância da média, podemos exigir que a distribuição binomial esteja dentro destes limites.
- Então, a normal deve ser $\mathcal{N}(\mu, \sigma)$ tal que o intervalo de $\mu - 3\sigma$ até $\mu + 3\sigma$ só tenha valores positivos.
- Ou seja, queremos

$$\mu - 3\sigma > 0$$

- O que equivale a

$$\mu > 3\sigma$$

- Como a normal é calculada a partir de Binom$(X \mid n, p)$, isto equivale a

$$np > 3\sqrt{np(1-p)}$$

- Elevando os dois lados ao quadrado:

$$n^2p^2 > 9np(1-p)$$

- Dividindo ambos os lados por np (que é positivo):

$$np > 9(1-p)$$

- Como $(1-p) \leq 1$, podemos nos satisfazer com

$$np > 9$$

- **Conclusão:**
  - Para aproximar a binomial Binom$(X \mid n, p)$ por uma normal, exigimos que np seja pelo menos 10.
  - Como a normal é simétrica, também exigimos que $n(1-p)$ também seja pelo menos 10.
  - Em outras palavras, o número esperado de sucessos e o número esperado de fracassos precisam ser, ambos, maiores ou iguais a 10.

10.3.6 Correção de continuidade

- E se, no exemplo dos doadores de sangue, quiséssemos calcular a probabilidade de a organização obter *exatamente* 1.850 doadores com sangue O-?

```
dbinom(1850, 32000, .06)
```

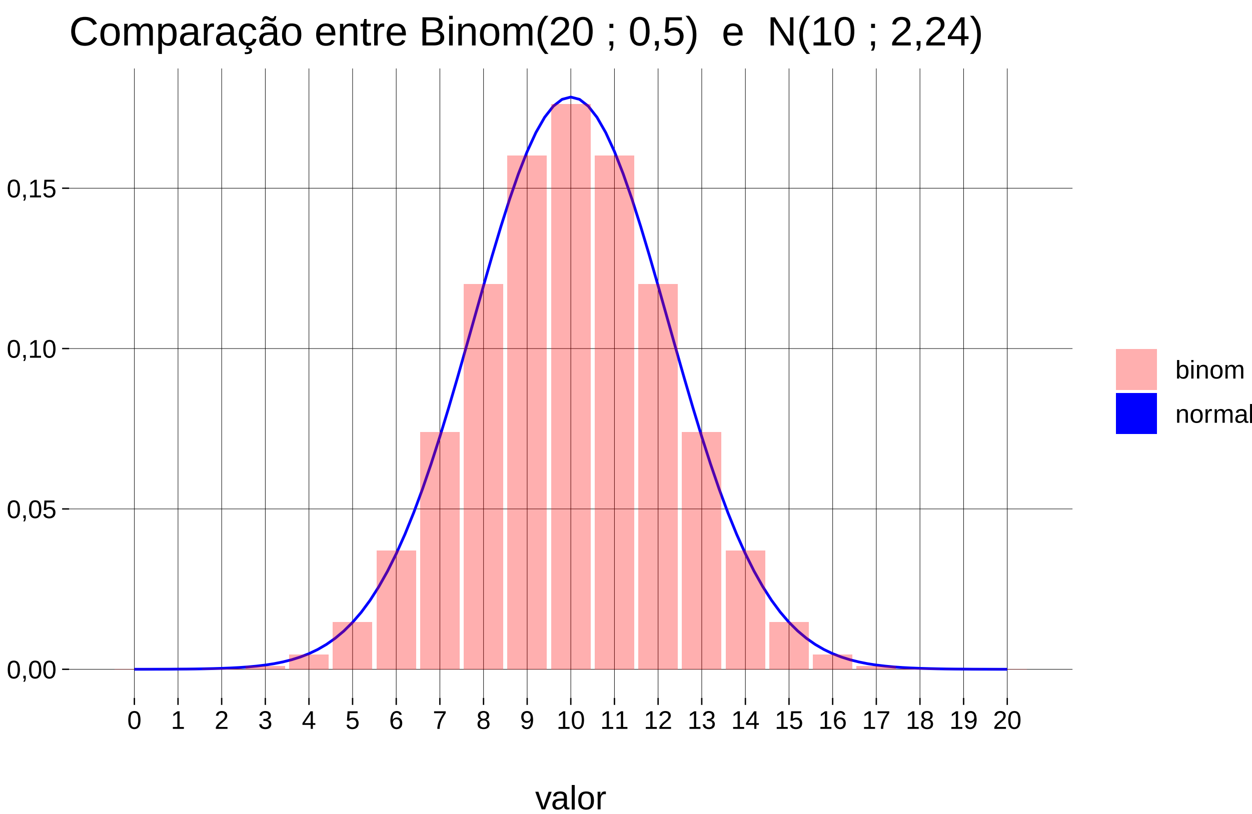
```
## [1] 0,002420205
```

- Usando a aproximação normal, como calcularíamos esta probabilidade? Como a normal é uma distribuição contínua, $P(X = 1850)$ é igual a zero (como qualquer probabilidade pontual)!
- A solução é calcular $P(1849{,}5 \leq X \leq 1850{,}5)$:

```
probs <- pnorm(
  c(1849.5, 1850.5),
  mean = 32000 * 0.06,
  sd = sqrt(32000 * 0.06 * (1 - 0.06))
)

probs[2] - probs[1]
```

```
## [1] 0,002416361
```

- Trabalhando com uma distribuição Binom$(X \mid n = 20, p = 0{,}5)$, isto fica claro no gráfico. Cada barra centrada no valor x compreende uma região de $x - 0{,}5$ até $x + 0{,}5$:



10.4 Exponencial

10.4.1 Exemplo

- Lembre-se de que a distribuição de Poisson modela o número de ocorrências de um fenômeno que tem uma média de λ ocorrências por período de tempo.
- Imagine que o números de visitas por minuto à sua página web é $\lambda = 4$.
- Então, o *tempo* entre visitas é uma variável aleatória contínua X, que pode ser modelada pela distribuição exponencial com média $1/\lambda = 1/4 = 0{,}25$.
- Qual a probabilidade de que o tempo entre uma visita e a próxima seja menor do

que $0{,}33$ minuto?

$$
\begin{aligned}
P(X \leq 0{,}33 \mid \lambda = 4) &= \int_0^{0{,}33} \lambda e^{-\lambda \cdot 0{,}33} dx \\
&= 1 - e^{-\lambda \cdot 0{,}33} \\
&= 1 - e^{-1{,}32} \\
&\approx 0{,}73
\end{aligned}
$$

10.4.2 No geral

- $X \sim$ Exp(λ)
- Média: $1/\lambda$
- Desvio padrão: $1/\lambda^2$
- Suporte: $[0, \infty)$
- Fdp:

$$f(x \mid \lambda) = \lambda e^{-\lambda x}$$

10.4.3 Em R

- Densidade: `dexp(x, rate = 1)`
- Probabilidade acumulada: `pexp(q, rate = 1, lower.tail = TRUE)`
- Quantil: `qexp(p, rate = 1, lower.tail = TRUE)`
- Geração de números aleatórios: `rexp(n, rate = 1)`

10.5 Outras distribuições contínuas importantes

- Distribuição t de Student
- Distribuição χ^2
- Distribuição F

10.6

Jardim zoológico de distribuições

Para sua diversão: https://ben18785.shinyapps.io/distribution-zoo/

Referências

Morgado, Augusto César de Oliveira, João Bosco Pitombeira de Carvalho, Paulo Cezar Pinto Carvalho, e Pedro Jesus Fernandez. 2004. *Análise Combinatória e Probabilidade*. Rio de Janeiro: Impa / Vitae.