Café com estatística e R

Treinamento 2 - Importação e manipulação de dados no R e estatística descritiva: parte 1

Marcelo Teixeira Paiva

2025-10-28

Abstract

Relatório do segundo treinamento onde foi apresentado como importar dados e manipulá-los no R, bem como as principais estatísticas descritivas univariadas e multivariadas.

Índice

1 Importação de dados no R **5**
1.1 Pacotes necessários . . . 5
1.2 Leitura de datasets externos ao R . . . 9
1.2.1 Importando dados do Excel . . . 9
1.2.2 Importando dados do Stata . . . 12
1.2.3 Verificação e diagnóstico dos dados importados . . . 14

2 Manipulação de dados com os pacotes do `tidyverse` **16**
2.1 O pipe (%>% e |>) . . . 17
2.2 Manipulação dos dados com o `dplyr` . . . 18
2.3 Reestruturação dos dados com o `tidyr` . . . 23
2.4 Unindo diferentes tabelas com `dplyr` . . . 27
2.5 Manipulando texto, datas e fatores com `stringr`, `lubridate` e `forcats` . . . 28

3 Estatística descritiva **32**
3.1 Tabela de distribuição de frequências (Univariada) . . . 35
3.2 Representação gráfica dos dados de uma variável . . . 38
3.2.1 Criação de gráficos no `ggplot2` . . . 39
3.2.2 Representação gráfica de variáveis qualitativas (Univariada) . . . 57
3.2.2.1 Gráfico de barras . . . 57
3.2.3 Gráfico de setores ou pizza . . . 62
3.2.4 Diagrama de Pareto . . . 64
3.2.5 Outras opções gráficas para variáveis qualitativas . . . 67
3.2.6 Representação gráfica de variáveis quantitativas (Univariada) . . . 77
3.2.6.1 Gráfico de linhas e pontos . . . 77
3.2.6.2 Histograma . . . 79
3.2.6.3 Boxplot ou diagrama de caixa . . . 81
3.3 Medidas-resumo . . . 88
3.4 Medidas de posição ou localização . . . 88
3.4.0.1 Média . . . 88
3.4.0.2 Mediana . . . 93
3.4.0.3 Moda . . . 96
3.4.0.4 Medidas separatrizes . . . 100
3.5 Identificação de outliers . . . 103
3.5.1 Método de Tukey (boxplot) . . . 103
3.5.2 Método Z-Score . . . 104
3.5.3 Método MAD (Median Absolute Deviation) . . . 105
3.5.4 Teste de Grubbs . . . 106

3.6 Medidas de dispersão ou variabilidade . . . 109
3.6.1 Amplitude ou *range* . . . 109
3.6.2 Desvio-médio . . . 113
3.6.3 Variância . . . 114
3.6.4 Desvio-padrão . . . 116
3.6.5 Erro-padrão . . . 118
3.6.6 Coeficiente de variação . . . 121
3.7 Medidas de forma . . . 121
3.7.1 Assimetria . . . 122
3.7.2 Curtose . . . 124
3.7.3 Teste combinado de normalidade para assimetria e curtose . . . 127
3.7.4 Desigualdade Fundamental de Pearson . . . 128

Lista de Figuras

2.1 Tipos de joins do `dplyr` . . . 27

3.1 Chamada de `ggplot` somente com argumento data . . . 40
3.10 Múltiplos painéis no `ggplot2` com `facet_wrap`. . . . 43
3.11 Alterando os *headers* de painéis do `facet_*()`. . . . 44
3.12 Múltiplos painéis no `ggplot2` com `facet_grid`. . . . 45
3.13 Manipulando escalas no `ggplot2` com `scale_*_continuous`. . . . 46
3.14 Manipulando escalas no `ggplot2` com `scale_*_discrete`. . . . 47
3.17 Alterando as escalas dos eixos com `coord_cartesian`. . . . 48
3.24 `scale_fill_brewer(type = "qual")` . . . 52
3.25 `scale_fill_brewer(palette = "Pastel1")` . . . 52
3.28 Alterando títulos e legendas em `labs()`. . . . 54
3.37 Personalizando temas em `theme()`. . . . 57
3.46 Exemplo de gráfico de setores . . . 64
3.47 Exemplo de gráfico de Pareto . . . 67
3.48 Exemplo de gráfico de waffle . . . 71
3.49 Exemplos de gráfico de Cleveland: Contagem de uma variável qualitativa . . . 72
3.50 Exemplos de gráfico de Cleveland: Distribuição das contagens em relação à média . . . 73
3.51 Exemplos de gráfico treemap . . . 74
3.52 Exemplo de gráfico de empacotamento circular . . . 77
3.53 Exemplo de gráfico de empacotamento circular . . . 78
3.58 Exemplo de gráfico boxplot . . . 83
3.59 Exemplo de gráfico composto histograma + boxplot . . . 86
3.60 Exemplo de gráfico violino + boxplot + histograma . . . 88
3.63 Variação do SE conforme o tamanho da amostra . . . 121
3.64 Comparação dos diferentes tipos de assimetria . . . 124
3.65 Comparação dos diferentes tipos de assimetria . . . 127
3.66 Posição da assimetria e curtose de distribuições reais em relação à desigualdade de Pearson 130

Lista de Tabelas

3.1 Distribuição de frequências de uma variável qualitativa 37
3.2 Distribuição de frequências de uma variável quantitativa discreta 38
3.3 Distribuição de frequências de uma variável quantitativa contínua 38
3.11 Quartis do ganho de peso diário segundo o estado vacinal 103

```r
# lapply(c(
#   "tidyverse",
#   "gridExtra",
#   "tidyverse",
#   "kableExtra",
#   "ggtext",
#   "paletteer"
# ), install.packages, character.only = TRUE)

options(repos = c(CRAN = "https://cloud.r-project.org"))

# Pacotes
library(tidyverse)
library(gridExtra)
library(tidyverse)
library(kableExtra)

# Tema personalizado para gráficos
tema_didatico <- theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 14),
    plot.subtitle = element_text(face = "italic", size = 11),
    axis.title = element_text(size = 12),
    legend.position = "top",
    panel.grid.minor = element_blank()
  )

cores <- c("#FF6B6B", "#4ECDC4", "#45B7D1", "#96CEB4", "#FFEAA7")
```

Chapter 1

Importação de dados no R

1.1 Pacotes necessários

Pacotes (**package**) são coleções de funções, dados e documentação que estendem as capacidades do R base (aquele que você recebe na instalação padrão). São como "caixas de ferramentas" especializadas que você adiciona ao R para realizar tarefas específicas, então você tem pacotes para elaboração de gráficos, para certos tipos de análises, para manipulação de dados, para leitura (importação) de dados. Em https://cran.r-project.org/web/views/ há uma "breve" lista de pacotes conforme a sua finalidade.

```r
# funções no R base
length(ls("package:base"))
```

```
[1] 1283
```

```r
# funções especializadas no pacote dplyr
length(ls("package:dplyr"))
```

```
[1] 297
```

```r
# um pacote possui um conjunto de arquivos associados
system.file(package = "ggplot2") %>% list.files()
```

```
 [1] "CITATION"    "data"        "DESCRIPTION" "doc"         "help"
 [6] "html"        "INDEX"       "LICENSE"     "Meta"        "NAMESPACE"
[11] "NEWS.md"     "R"
```

Por padrão, ao iniciar uma sessão no R, serão carregados os pacotes e funções associados ao R base. Os demais devem ser instalados primeiramente, e depois carregados na seção para serem usados.

```r
# pacotes carregados no seu ambiente
search()
```

```
 [1] ".GlobalEnv"         "package:kableExtra" "package:gridExtra"
 [4] "package:lubridate"  "package:forcats"    "package:stringr"
 [7] "package:dplyr"      "package:purrr"      "package:readr"
[10] "package:tidyr"      "package:tibble"     "package:ggplot2"
[13] "package:tidyverse"  "package:stats"      "package:graphics"
[16] "package:grDevices"  "package:utils"      "package:datasets"
[19] "package:methods"    "Autoloads"          "package:base"
```

```r
# pacotes instalados
instalados <- installed.packages()[, "Package"]
instalados[1:4]
```

```
      abind     askpass   backports   base64enc
    "abind"   "askpass" "backports" "base64enc"
```

```r
length(instalados)
```

```
[1] 240
```

```r
# verificando se um pacote já está instalado
sum(installed.packages()[, "Package"] == 'dplyr')
```

```
[1] 1
```

```r
any(installed.packages()[, "Package"] == 'dplyr')
```

```
[1] TRUE
```

```r
"ggplot2" %in% rownames(installed.packages())
```

```
[1] TRUE
```

A instalação de pacotes no R é feita usando a função `install.packages` ou `devtools::install_github` para pacotes que estão no github e não em um repositório de pacotes.

```r
# pelo repositório oficial (na web)
install.packages("ggplot2")
install.packages(c("dplyr", "tidyr", "readr")) # instalando vários pacotes de uma vez

# Instalar o pacote e todas dependências relacionadas a ele
install.packages("ggplot2", dependencies = TRUE)

# instalar de um arquivo local
install.packages("caminho/para/pacote.tar.gz", repos = NULL, type = "source")

# Instalar pacote mantido no GitHub
install.packages("devtools")
devtools::install_github("tidyverse/ggplot2")

# Usar outros repositórios para instalação
install.packages("ggplot2", repos = "https://cloud.r-project.org/")
```

Para carregar um pacote em uma sessão usamos `library()` ou `require()`. A diferença entre os dois é que, na ausência do pacote que você pretende carregar, `library` gera um erro, enquanto o `require` retorna um valor `FALSE` invisível, o qual pode ser usado, por exemplo, para criar uma lógica em seu script para instalar o pacote caso o mesmo não possa ser carregado ou, então, para gerar uma mensagem no terminal indicando essa ausência do pacote.

```r
library(ggplot2)

# Não exibir mensagens de carregamento do pacote
suppressPackageStartupMessages(library(ggplot2))
```

```r
# criando uma lógica simples com require para instalar pacotes que
# não possam ser carregados
if(!require(ggplot2)) {
  install.packages("ggplot2")
  require(ggplot2)
}

# usando uma função do pacote sem o carregar (namespace qualification)
head(dplyr::filter(mtcars, mpg > 20), 2)
```

```
              mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4      21   6  160 110  3.9 2.620 16.46  0  1    4    4
Mazda RX4 Wag  21   6  160 110  3.9 2.875 17.02  0  1    4    4
```

```r
# carregando vários pacotes de uma lista de nomes
pacotes <- c("ggplot2", "dplyr", "tidyr")
x <- lapply(pacotes, library, character.only = TRUE, quietly = TRUE)
```

Além dessas funções para instalação e carregamento de pacotes, também outras funções que devem ser conhecidas na rotina são as de atualização (`update.packages()`) e remoção (`remove.packages()`) de pacotes, descrição (`packageDescription()`), versão (`packageVersion()`) e forma recomendada pelo seus autores de citação (`citation()`) quando usada em uma publicação.

```r
# Atualização de pacotes
update.packages()  # todos, mas exige confirmação
update.packages(ask = FALSE) # todos

# apagar um pacote
remove.packages("nome_pacote")
```

```r
# descrição e versão
packageDescription("ggplot2")
```

```
Package: ggplot2
Title: Create Elegant Data Visualisations Using the Grammar of Graphics
Version: 4.0.0
Authors@R: c( person("Hadley", "Wickham", , "hadley@posit.co", role =
        "aut", comment = c(ORCID = "0000-0003-4757-117X")),
        person("Winston", "Chang", role = "aut", comment = c(ORCID =
        "0000-0002-1576-2126")), person("Lionel", "Henry", role =
        "aut"), person("Thomas Lin", "Pedersen", ,
        "thomas.pedersen@posit.co", role = c("aut", "cre"), comment =
        c(ORCID = "0000-0002-5147-4711")), person("Kohske",
        "Takahashi", role = "aut"), person("Claus", "Wilke", role =
        "aut", comment = c(ORCID = "0000-0002-7470-9261")),
        person("Kara", "Woo", role = "aut", comment = c(ORCID =
        "0000-0002-5125-4188")), person("Hiroaki", "Yutani", role =
        "aut", comment = c(ORCID = "0000-0002-3385-7233")),
        person("Dewey", "Dunnington", role = "aut", comment = c(ORCID =
        "0000-0002-9415-4582")), person("Teun", "van den Brand", role =
        "aut", comment = c(ORCID = "0000-0002-9335-7468")),
        person("Posit, PBC", role = c("cph", "fnd"), comment = c(ROR =
```

```
        "03wc8by49")) )
Description: A system for 'declaratively' creating graphics, based on
        "The Grammar of Graphics". You provide the data, tell 'ggplot2'
        how to map variables to aesthetics, what graphical primitives
        to use, and it takes care of the details.
License: MIT + file LICENSE
URL: https://ggplot2.tidyverse.org,
        https://github.com/tidyverse/ggplot2
BugReports: https://github.com/tidyverse/ggplot2/issues
Depends: R (>= 4.1)
Imports: cli, grDevices, grid, gtable (>= 0.3.6), isoband, lifecycle (>
        1.0.1), rlang (>= 1.1.0), S7, scales (>= 1.4.0), stats, vctrs
        (>= 0.6.0), withr (>= 2.5.0)
Suggests: broom, covr, dplyr, ggplot2movies, hexbin, Hmisc, knitr,
        mapproj, maps, MASS, mgcv, multcomp, munsell, nlme, profvis,
        quantreg, ragg (>= 1.2.6), RColorBrewer, rmarkdown, roxygen2,
        rpart, sf (>= 0.7-3), svglite (>= 2.1.2), testthat (>= 3.1.5),
        tibble, vdiffr (>= 1.0.6), xml2
Enhances: sp
VignetteBuilder: knitr
Config/Needs/website: ggtext, tidyr, forcats, tidyverse/tidytemplate
Config/testthat/edition: 3
Config/usethis/last-upkeep: 2025-04-23
Encoding: UTF-8
LazyData: true
RoxygenNote: 7.3.2
Collate: 'ggproto.R' 'ggplot-global.R' 'aaa-.R'
        'aes-colour-fill-alpha.R' .....
NeedsCompilation: no
Packaged: 2025-08-19 08:21:45 UTC; thomas
Author: Hadley Wickham [aut] (ORCID:
        <https://orcid.org/0000-0003-4757-117X>), Winston Chang [aut]
        (ORCID: <https://orcid.org/0000-0002-1576-2126>), Lionel Henry
        [aut], Thomas Lin Pedersen [aut, cre] (ORCID:
        <https://orcid.org/0000-0002-5147-4711>), Kohske Takahashi
        [aut], Claus Wilke [aut] (ORCID:
        <https://orcid.org/0000-0002-7470-9261>), Kara Woo [aut]
        (ORCID: <https://orcid.org/0000-0002-5125-4188>), Hiroaki
        Yutani [aut] (ORCID: <https://orcid.org/0000-0002-3385-7233>),
        Dewey Dunnington [aut] (ORCID:
        <https://orcid.org/0000-0002-9415-4582>), Teun van den Brand
        [aut] (ORCID: <https://orcid.org/0000-0002-9335-7468>), Posit,
        PBC [cph, fnd] (ROR: <https://ror.org/03wc8by49>)
Maintainer: Thomas Lin Pedersen <thomas.pedersen@posit.co>
Repository: CRAN
Date/Publication: 2025-09-11 07:10:02 UTC
Built: R 4.5.1; ; 2025-10-17 13:11:49 UTC; unix

-- File: /home/marcelo/R/x86_64-pc-linux-gnu-library/4.5/ggplot2/Meta/package.rds
```

```r
packageVersion("ggplot2")
```

```
[1] '4.0.0'
```

```r
# forma de citação
citation("ggplot2")
```

```
To cite ggplot2 in publications, please use

  H. Wickham. ggplot2: Elegant Graphics for Data Analysis.
  Springer-Verlag New York, 2016.

Uma entrada BibTeX para usuários(as) de LaTeX é

  @Book{,
    author = {Hadley Wickham},
    title = {ggplot2: Elegant Graphics for Data Analysis},
    publisher = {Springer-Verlag New York},
    year = {2016},
    isbn = {978-3-319-24277-4},
    url = {https://ggplot2.tidyverse.org},
  }
```

Algo a se ter em mente é que nada impede de vários pacotes terem o mesmo nome para funções com finalidades diferentes. Nesse caso, ao carregar esses pacotes, o último a ser carregado irá mascarar o nome da anterior no seu ambiente. Assim, para evitar conflitos, ou o uso da função errada, recomenda-se usar a função seguindo o padrão `nome_do_pacote::nome_da_função`.

1.2 Leitura de datasets externos ao R

A importação de dados é o primeiro passo em qualquer análise. O R oferece múltiplos pacotes especializados para diferentes formatos de arquivos, mas iremos focar nos pacotes de leitura dos arquivos provenientes dos softwares Excel, SAS, Stata e SPSS. Para isso, utilizaremos os pacotes `readxl` e `haven`.

```r
# mini rotina para instalar um pacote se ainda não estiver instalado
instala_se_nao_existe <- function(nome_do_pacote){
  if(nome_do_pacote %in% rownames(installed.packages())) return()
  install.packages(nome_do_pacote, quiet = TRUE)
  return()
}
lapply(c("readxl", "haven"), instala_se_nao_existe)

# Carregar pacotes
library(readxl)    # Excel
library(haven)     # SAS, SPSS, STATA
```

1.2.1 Importando dados do Excel

Para leitura de arquivos do Excel nos formatos .xls e .xlsx usaremos o pacote `readxl`, o qual faz parte do conjunto de pacotes do `tidyverse`. Dele podemos usar as funções `read_excel()`, `read_xls()` ou `read_xlsx()`, os quais recebem argumentos semelhantes, com a diferença que os dois últimos são específicos ao formato do arquivo.

O primeiro e mais importante argumento a ser fornecido para essa função é o `path`, o local onde o arquivo se encontra no seu computador. Esse caminho pode ser absoluto (desde a raiz, normalmente `/` no linux ou `C:` no windows, até o local) ou relativo ao diretório de trabalho (que pode ser verificado usando a função `getwd()`).

Como os arquivos do Excel aceitam múltiplas planilhas (em diferentes abas), o argumento de `sheet` do `read_excel()` permite escolher qual aba se pretende carregar. Caso seja necessário verificar primeiro o nome das abas disponíveis no arquivo, use `excel_sheets(path)`.

Outro problema comum em arquivos do Excel são planilhas que não iniciam na linha 1 ou que apresentam um conjunto de colunas que não pretendemos usar (sem conteúdo ou preenchido com informações que não fazem parte do dateset). Para contornar esses obstáculos, podemos usar o argumento `skip` com o número de linhas iniciais que não devem ser lidas, ou usar o `range` com um `character` indicando a primeira e última células que delimitam seus dados (por exemplo, `range = "B2:D20"` indica que devem ser lidas as colunas B, C e D, das linhas 2 até a 20).

Por padrão, essas funções buscam adivinhar o tipo de dados presente em cada coluna da planilha, mas é possível declarar o tipo usando o argumento col_types com um vetor com comprimento igual ao número de colunas que irá importar. Esse vetor deve, para cada coluna, usar uma das opções:

- "skip": remove a coluna do dataset
- "guess": deixa para a função escolher o tipo
- "logical": booleano
- "numeric": numérico
- "date": data
- "text": character
- "list": lista

Também por padrão, a primeira linha é usada para obter os nomes de cada coluna. Se você não possui nomes das colunas na sua planilha use `col_names = FALSE` na função ou passe um vetor dos nomes das colunas para o argumento `col_names`.

Um aspecto importante de qualquer conjunto de dados é saber como foram codificados os dados ausentes. O argumento `na` permite passar um vetor de `character` com os códigos usados na planilha para declarar um dado ausente, o qual será convertido para `NA` no R.

```r
excel_sheets("../datasets/excel/ap2.xlsx")
```

```
[1] "Data"
```

```r
dados_excel <- read_excel("../datasets/excel/ap2.xlsx")
head(dados_excel)
```

```
# A tibble: 6 x 21
  farm_id batch_id litt_id pig_id parity vacc_mp seas_fin age_t w_age_t age_t6
    <dbl>    <dbl>   <dbl>  <dbl>  <dbl>   <dbl>    <dbl> <dbl>   <dbl>  <dbl>
1       1        1       1      1      8       1        1    70    33.8    116
2       1        1       1      2      8       1        1    70    32.9    116
3       1        1       1      3      8       1        1    70    29.4    116
4       1        1       2      4      8       1        1    60    19.8    106
5       1        1       2      5      8       1        1    60    20.4    106
6       1        1       2      6      8       1        1    60    20.3    106
# i 11 more variables: w_age_t6 <dbl>, dwg_fin <dbl>, ap2_t <dbl>, mp_t <dbl>,
#   infl_t <dbl>, prrs_t <dbl>, ap2_t6 <dbl>, mp_t6 <dbl>, infl_t6 <dbl>,
#   prrs_t6 <dbl>, ap2_sc <dbl>
```

```r
# definir a planilha por nome ou índice
dados_pela_aba <- read_excel("../datasets/excel/ap2.xlsx", sheet = "Data")
dados_pela_aba <- read_excel("../datasets/excel/ap2.xlsx", sheet = 1)

# carregar somente um intervalo de células, em que a linha 1 não é header
dados_pelo_range <- read_excel(
  "../datasets/excel/ap2.xlsx",
  range = "A2:B100",
  sheet = "Data",
  col_names = FALSE
)
```

```
New names:
* `` -> `...1`
* `` -> `...2`
```

```r
head(dados_pelo_range)
```

```
# A tibble: 6 x 2
   ...1  ...2
  <dbl> <dbl>
1     1     1
2     1     1
3     1     1
4     1     1
5     1     1
6     1     1
```

```r
# mesmo exemplo, mas definindo os nomes das colunas
dados_pelo_range <- read_excel(
  "../datasets/excel/ap2.xlsx",
  range = "A2:B100",
  sheet = "Data",
  col_names = c('fazenda', 'lote')
)
head(dados_pelo_range)
```

```
# A tibble: 6 x 2
  fazenda  lote
    <dbl> <dbl>
1       1     1
2       1     1
3       1     1
4       1     1
5       1     1
6       1     1
```

```r
# declarando os tipos de colunas
dados_tipos <- read_excel(
  "../datasets/excel/ap2.xlsx",
  col_types = c("text", "numeric", rep("text", 19))
)
```

```r
head(dados_tipos)
```

```
# A tibble: 6 x 21
  farm_id batch_id litt_id pig_id parity vacc_mp seas_fin age_t w_age_t age_t6
  <chr>      <dbl> <chr>   <chr>  <chr>  <chr>   <chr>    <chr> <chr>   <chr>
1 1              1 1       1      8      1       1        70    33.8    116
2 1              1 1       2      8      1       1        70    32.9    116
3 1              1 1       3      8      1       1        70    29.4    116
4 1              1 2       4      8      1       1        60    19.8    106
5 1              1 2       5      8      1       1        60    20.4    106
6 1              1 2       6      8      1       1        60    20.3    106
# i 11 more variables: w_age_t6 <chr>, dwg_fin <chr>, ap2_t <chr>, mp_t <chr>,
#   infl_t <chr>, prrs_t <chr>, ap2_t6 <chr>, mp_t6 <chr>, infl_t6 <chr>,
#   prrs_t6 <chr>, ap2_sc <chr>
```

```r
# definir os códigos usados na planilha para dados ausentes
dados_na <- read_excel(
  "../datasets/excel/ap2.xlsx",
  na = c("", "NA", "N/A", "-")
)
```

1.2.2 Importando dados do Stata

Para leitura de arquivos do Stata no formato .dta usaremos o pacote `heaven`, o qual possui funções para leitura de arquivos do Stata, SPSS e SAS. Nesse treinamento vamos focar na função `read_dta()` para leitura dos arquivos do Stata (superiores a versão 13.0).

Assim como no `read_excel()`, o primeiro argumento de `read_dta()` deve ser a localização do arquivo. Além disso, a função aceita como argumentos `encoding`, a codificação de carácteres usada, `skip` para remover um certo número de linhas, `col_select` para definir quais colunas serão selecionadas e `n_max` para declarar o número máximo de linhas que devem ser importadas.

Um diferença importante entre arquivos do Excel e do Stata é que no segundo o dataset e as suas variáveis podem conter metadados ("notes" e "labels") com informações sobre esses dados. Essas informações podem ser acessadas na função `attr()`.

```r
dados_stata <- read_dta("../datasets/stata/ap2.dta")
head(dados_stata)
```

```
# A tibble: 6 x 21
  farm_id batch_id litt_id pig_id parity vacc_mp   seas_fin age_t w_age_t age_t6
    <dbl>    <dbl>   <dbl>  <dbl>  <dbl> <dbl+lbl> <dbl+lb> <dbl>   <dbl>  <dbl>
1       1        1       1      1      8 1 [vac]   1 [wint~    70    33.8    116
2       1        1       1      2      8 1 [vac]   1 [wint~    70    32.9    116
3       1        1       1      3      8 1 [vac]   1 [wint~    70    29.4    116
4       1        1       2      4      8 1 [vac]   1 [wint~    60    19.8    106
5       1        1       2      5      8 1 [vac]   1 [wint~    60    20.4    106
6       1        1       2      6      8 1 [vac]   1 [wint~    60    20.3    106
# i 11 more variables: w_age_t6 <dbl>, dwg_fin <dbl>, ap2_t <dbl+lbl>,
#   mp_t <dbl+lbl>, infl_t <dbl+lbl>, prrs_t <dbl+lbl>, ap2_t6 <dbl+lbl>,
#   mp_t6 <dbl+lbl>, infl_t6 <dbl+lbl>, prrs_t6 <dbl+lbl>, ap2_sc <dbl+lbl>
```

```r
dados_stata_com_encoding <- read_dta(
  "../datasets/stata/ap2.dta",
  encoding = "UTF-8"
)

# tranformar colunas labelled em factor
dados_stata_como_factor <- read_dta(
  "../datasets/stata/ap2.dta",
  encoding = "UTF-8"
) |> as_factor()
head(dados_stata_como_factor)
```

```
# A tibble: 6 x 21
  farm_id batch_id litt_id pig_id parity vacc_mp seas_fin age_t w_age_t age_t6
    <dbl>    <dbl>   <dbl>  <dbl>  <dbl> <fct>   <fct>    <dbl>   <dbl>  <dbl>
1       1        1       1      1      8 vac     winter      70    33.8    116
2       1        1       1      2      8 vac     winter      70    32.9    116
3       1        1       1      3      8 vac     winter      70    29.4    116
4       1        1       2      4      8 vac     winter      60    19.8    106
5       1        1       2      5      8 vac     winter      60    20.4    106
6       1        1       2      6      8 vac     winter      60    20.3    106
# i 11 more variables: w_age_t6 <dbl>, dwg_fin <dbl>, ap2_t <fct>, mp_t <fct>,
#   infl_t <fct>, prrs_t <fct>, ap2_t6 <fct>, mp_t6 <fct>, infl_t6 <fct>,
#   prrs_t6 <fct>, ap2_sc <fct>
```

```r
# Notas do Stata
attr(dados_stata, "notes")
```

```
[1] "5 Aug 2002 16:24 data provided by Dr. Haakan Vigre, Denmark"
[2] "1"
```

```r
# Labels das variáveis
labels <- sapply(dados_stata, function(x) attr(x, "label"))
kable(
  tibble(var=names(labels), metadata=labels),
  col.names = c("Variável", "Label")
)
```

Variável	Label
farm_id	farm identification
batch_id	batch identifiaction number
litt_id	litter identification number
pig_id	pig identification
parity	the farrowing no. of the sow
vacc_mp	the batch vaccinated against M.hyop yes=1
seas_fin	prod. season in finishing unit: winther=1
age_t	pig-age transfer from weaning to finishing unit
w_age_t	weight in kg. at age_t
age_t6	age_tra plus approx. 6 weeks
w_age_t6	weight in kg. at age_t6
dwg_fin	dwg in g. between age_t and age_t6

Variável	Label
ap2_t	serological reac. against A.pleuropneumoniae serotype 2 at age_t
mp_t	serological reac. against M.hyopneumoniae at age_t
infl_t	serological reac. against Influenza virus at age_t
prrs_t	serological reac. against PRRS virus at age_t
ap2_t6	serological reac. against A.pleuropneumoniae serotype 2 at age_t6
mp_t6	serological reac. against M.hyopneumoniae at age_t6
infl_t6	serological reac. against Influenza virus at age_t6
prrs_t6	serological reac. against PRRS virus at age_t6
ap2_sc	seroconversion to ap2 during the finishing period

1.2.3 Verificação e diagnóstico dos dados importados

Uma vez carregados os dados, é importante avaliar a estrutura desse conjunto de dados importado. Para uma exploração inicial, será interessante avaliar, no mínimo, as dimensões desses dados (número de observações e variáveis), quais os tipos das variáveis no R, resumos estatísticos simples, quantidade de valores ausentes por variável.

```
verificar_dados <- function(dados) {
  cat("Dimensões:", dim(dados), "\n")
  cat("Tipos de variáveis:\n")
  print(sapply(dados, class))
  cat("\nPrimeiras linhas:\n")
  print(head(dados, 3))
  cat("\nResumo estatístico:\n")
  print(summary(dados))
  cat("\nValores missing por coluna:\n")
  print(colSums(is.na(dados)))
  cat("\nStructura dos dados:\n")
  str(dados)
}

# Aplicar a qualquer dataset importado
verificar_dados(dados_stata)
```

Por fim, em grandes datasets é comum que os dados sejam registrados em múltiplos arquivos (principalmente no Excel, por causa do limite de linhas). Nesse caso, para não ser necessário carregar cada um desses arquivos e depois construir um data.frame que uni todos, podemos usar recursos de programação funcional do pacote `purrr` para importar diretamente todos os arquivos em um único data.frame.

```
library(purrr)
# obter uma lista dos arquivos que serão importados e
# mapear todos os arquivos para um unico data.frame
dados <- list.files("datasets/csv", pattern = "\\.csv$", full.names = TRUE) |> map_df(read_csv2)
```

Quadro Resumo das funções que podem ser usadas na importação de arquivos externos ao R

Formato	Pacote	Função
CSV	`readr`	`read_delim()`, `read_csv()`, `read_csv2()`
Excel	`readxl`	`read_excel()`, `read_xls()`, `read_xlsx()`

Formato	Pacote	Função
SAS	`haven`	`read_sas()`
SPSS	`haven`	`read_sav()`
Stata	`haven`	`read_stata()`, `read_dta()`
Múltiplos	`rio`	`import()`

Chapter 2

Manipulação de dados com os pacotes do `tidyverse`

O **tidyverse** é uma coleção de pacotes R voltados para a ciência de dados, que compatilham uma mesma filosofia, gramática e estruturas de dados. Ele é composto dos seguintes pacotes:

- tibble: extensão do `data.frame`;
- dplyr: funções na forma de verbos que fornece a gramática para a manipulação dos dados;
- tidyr: funções para obtenção dos dados que seguem a filosia dos "dados arrumados";
- readr: importação de dados tabulares (csv, tsv, fwf);
- purrr: programação funcional;
- stringr: manipulação de strings (`character`);
- forcats: manipulação de fatores (`factor`);
- lubridate: manipulação de datas (`date`);
- ggplot2: criação de gráficos.

```
# Carregar todo o conjunto de pacotes
library(tidyverse)

# Ou carregar pacotes individuais
library(dplyr)
library(tidyr)
library(readr)
```

Como já mencionado, o `tidyverse` segue a filosofia de "dados arrumados" (*tidy data*), o que basicamente significa que:

- **cada variável forma uma coluna;**
- **cada unidade observacional (unidade amostral) forma uma linha;**
- **cada célula é uma observação e um único valor**

Esse padrão facilita análises posteriores:

```
set.seed(42)
dados_tidy <- tibble(
  animal_id = 1:6,
  especie = rep(c("cão", "gato"), 3),
```

```
  peso_kg = round(rnorm(6, mean = 15, sd = 5), 1),
  idade_anos = sample(1:10, 6, replace = TRUE)
)
dados_tidy
```

```
# A tibble: 6 x 4
  animal_id especie peso_kg idade_anos
      <int> <chr>     <dbl>      <int>
1         1 cão        21.9          7
2         2 gato       12.2          4
3         3 cão        16.8          9
4         4 gato       18.2          5
5         5 cão        17            4
6         6 gato       14.5         10
```

O `tibble` facilita a compreensão dos seus dados, uma vez que sua impressão (com `print`) apresenta o tipo de cada variável, não imprime o conjunto completo (somente as primeiras linhas) e não faz conversões automáticas de variáveis `character` para `factor`. Além disso, ele aceita nomes não sintáticos do R para as variáveis (usando ").

Uma diferença importante entre `tibble` e `data.frame` está na forma como você extrai uma variável do conjunto. No `data.frame` usamos os padrões `nome_do_dataframe["nome_da_coluna"]`, `nome_do_dataframe[indice_da_coluna]` ou `nome_do_dataframe$nome_da_coluna`. No `tibble` usamos os padrões `nome_do_tibble$nome_da_coluna`, `nome_do_tibble[[indice_da_coluna]]`, `nome_do_tibble[["nome_da_coluna"]]` ou, ainda, extrair por meio do pipe com `nome_do_tibble %>% .$nome_da_coluna`, `nome_do_tibble %>% .[["nome_da_coluna"]]` ou `nome_do_tibble |> pull("nome_da_coluna")`.

```
# extraindo uma variável do tibble
dados_tidy %>% .$idade_anos
```

```
[1]  7  4  9  5  4 10
```

```
dados_tidy %>% .[["idade_anos"]]
```

```
[1]  7  4  9  5  4 10
```

```
dados_tidy |> pull("idade_anos")
```

```
[1]  7  4  9  5  4 10
```

2.1 O pipe (%>% e |>)

O pipe permite agrupar em um código que parece ser uma única operação múltiplas operações (chamadas de funções), em que o resultado de uma operação e fornecido como o primeiro argumento da subsequente. Isso torna o código mais legível.

```
# Sem pipe, com funções aninhadas
resultado <- summarise(
  group_by(
    filter(dados_tidy, peso_kg > 10),
    especie
  ),
```

```r
  peso_medio = mean(peso_kg)
)
# ou criando várias etapas
dados_filtrados <- filter(dados_tidy, peso_kg > 10)
dados_agrupados <- group_by(dados_filtrados, especie)
resultado <- summarise(dados_agrupados, peso_medio = mean(peso_kg))

# Com pipe do tidyverse (%>%)
resultado <- dados_tidy %>%
  filter(peso_kg > 10) %>%
  group_by(especie) %>%
  summarise(peso_medio = mean(peso_kg))

# com pipe nativo do R 4.1+ (|>)
resultado <- dados_tidy |>
  filter(peso_kg > 10) |>
  group_by(especie) |>
  summarise(peso_medio = mean(peso_kg))
```

2.2 Manipulação dos dados com o `dplyr`

Usamos o `select()` para obter um subconjunto do nosso dataset com somente as variáveis de interesse.

```r
df_exemplo <- tibble(
  id = 1:100,
  especie = sample(c("cão", "gato", "coelho"), 100, replace = TRUE),
  idade = round(runif(100, 1, 15), 1),
  peso = round(rnorm(100, 15, 5), 2),
  vacinado = sample(c(TRUE, FALSE), 100, replace = TRUE),
  data_consulta = seq(as.Date("2023-01-01"), by = "day", length.out = 100),
  temperatura = round(rnorm(100, 38.5, 0.5), 1)
)
```

```r
# seleção de colunas pelo nome ou com vetor de caractéres
df_exemplo %>%
  select(id, especie, peso)
df_exemplo %>%
  select(c("id", "especie", "peso"))

# Seleção com funções helpers
# starts_with para as colunas que iniciam com certo valor
df_exemplo %>%
  select(starts_with("data"))
# ends_with para as colunas que terminam com certo valor
df_exemplo %>%
  select(ends_with("do"))
# contains para as colunas que possuem um certo valor
df_exemplo %>%
  select(contains("ac"))
# where para colunas que correspondem a TRUE para alguma função de retorno lógico
```

```r
df_exemplo %>%
  select(where(is.numeric))
# usando padrão de fórmula para múltiplas condições
df_exemplo %>%
  select(where(~ is.numeric(.x) && min(.x) > 10))

# Seleção pela exclusão de determinadas colunas
df_exemplo %>%
  select(-id, -data_consulta)
df_exemplo %>%
  select(-c("id", "data_consulta"))

# Seleção com renomeação de determinadas colunas
df_exemplo %>%
  select(
    identificador = id,
    tipo_animal = especie,
    everything()
  )
```

Usamos o `filter()` para obter um subconjunto do nosso dataset com somente as observações que atendem a uma determinada condição.

```r
# Filtro básico
df_exemplo %>%
  filter(especie == "cão")

# Múltiplas condições
# AND - , ou &
df_exemplo %>%
  filter(especie == "gato", peso > 10, vacinado == TRUE)
df_exemplo %>%
  filter(especie == "gato" & peso > 10 & vacinado == TRUE)

# OR - |
df_exemplo %>%
  filter(especie == "cão" | especie == "gato")

# agrupando os OR de == com %in%
df_exemplo %>%
  filter(especie %in% c("cão", "gato"))

# Filtros com funções
# between para min <= x <= max
df_exemplo %>%
  filter(between(idade, 5, 10))

df_exemplo %>%
  filter(!is.na(temperatura))
```

Usamos o `mutate()` para criar ou modificar variáveis.

```r
# criar colunas
# case_when para construir uma variável baseado em condições das demais
df_exemplo %>%
  mutate(
    score_inventado = peso / (idade ^ 0.5),
    categoria_idade = case_when(
      idade < 1 ~ "Filhote",
      idade < 7 ~ "Adulto",
      TRUE ~ "Idoso"
    )
  ) %>%
  select(id, especie, idade, categoria_idade, score_inventado)

# modificar colunas
df_exemplo %>%
  mutate(
    peso = round(peso, 0),
    temperatura = temperatura * 9/5 + 32
  )

# transformações em várias colunas
# scale centraliza a variável (desvio / desvio-padrão)
# cuidado para multiplos across no mutate, a ordem importa
df_exemplo %>%
  mutate(
    across(where(is.numeric), ~round(.x, 1)),
    across(c(peso, temperatura), ~scale(.x)[,1], .names = "{.col}_z")
  )
df_exemplo %>%
  mutate(
    across(c(peso, temperatura), ~scale(.x)[,1], .names = "{.col}_z"),
    across(where(is.numeric), ~round(.x, 1))
  )

# transmute() - mantém apenas as colunas criadas
df_exemplo %>%
  transmute(
    id,
    peso_libras = peso * 2.205,
    idade_meses = idade * 12
  )
```

Usamos o `arrange()` para ordenar as variáveis por uma ou mais variáveis.

```r
# Ordenação crescente por uma variável
df_exemplo %>%
  arrange(peso)

# Ordenação decrescente por uma variável
df_exemplo %>%
  arrange(desc(peso))
```

```r
# Ordenação múltipla
df_exemplo %>%
  arrange(especie, desc(idade), peso)

# Ordenação com NA
df_exemplo %>%
  arrange(desc(is.na(temperatura)), temperatura)
```

Para criar agregações (resumos estatíticos) usamos `summarise()` e o `group_by()` caso esse resumo deva ser calculado para cada categoria de determinada variável.

```r
# Resumo simples
df_exemplo %>%
  summarise(
    n_animais = n(),
    peso_medio = mean(peso, na.rm = TRUE),
    peso_dp = sd(peso, na.rm = TRUE),
    peso_mediana = median(peso, na.rm = TRUE),
    peso_min = min(peso, na.rm = TRUE),
    peso_max = max(peso, na.rm = TRUE)
  )

# Agrupamento e resumo
df_exemplo %>%
  group_by(especie) %>%
  summarise(
    n = n(),
    idade_media = mean(idade, na.rm = TRUE),
    prop_vacinados = mean(vacinado, na.rm = TRUE),
    .groups = "drop"
  )

# Agrupamento por múltiplas variáveis
df_exemplo %>%
  group_by(especie, vacinado) %>%
  summarise(
    n = n(),
    peso_medio = mean(peso, na.rm = TRUE),
    temp_media = mean(temperatura, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  arrange(especie, vacinado)

# Criando colunas baseado nas informações do grupo (categoria)
df_exemplo %>%
  group_by(especie) %>%
  transmute(
    especie,
    peso,
    peso_padronizado = (peso - mean(peso)) / sd(peso),
```

```
    peso_centralizado = scale(peso)[,1],
    rank_peso = rank(peso)
  ) %>%
  ungroup() %>%
  arrange(rank_peso)
```

Para obter um subconjunto de observações também podemos usar funções da família `slice_*`.

```
# Primeiras ou últimas n observações
df_exemplo %>% slice_head(n = 5)
df_exemplo %>% slice_tail(n = 5)

# Linhas específicas
df_exemplo %>% slice(c(1, 5, 10))

# Amostragem "aleatória"
df_exemplo %>% slice_sample(n = 10)
df_exemplo %>% slice_sample(prop = 0.1)

# Extremos por grupo
df_exemplo %>%
  group_by(especie) %>%
  slice_max(peso, n = 3) # 3 maiores valores
df_exemplo %>%
  group_by(especie) %>%
  slice_min(idade, n = 2) # 2 menores valores
# retornando exatamente o valor n
df_exemplo %>%
  group_by(especie) %>%
  slice_min(idade, n = 2, with_ties = FALSE)
```

Para obter os valores únicos de uma ou mais variáveis usamos `distinct()`.

```
# Valores únicos de uma coluna
df_exemplo %>%
  distinct(especie)

# Combinações únicas
# .keep_all = TRUE para manter as outras colunas
df_exemplo %>%
  distinct(especie, vacinado, .keep_all = TRUE)

# Remover duplicatas
df_exemplo %>%
  distinct()
```

Para contagem de ocorrências usamos `count()` e `add_count()`.

```
# quantidade de observações por categoria
df_exemplo %>%
  count(especie, sort = TRUE, name = "Amostra")
```

```r
# Contagem com peso
df_exemplo %>%
  count(especie, wt = peso, name = "peso_total")
# mesmo que agrupar e agregar para a soma
df_exemplo %>%
  group_by(especie) %>%
  summarise(
    peso_total = sum(peso)
  )

# Adicionar contagem sem agregar
df_exemplo %>%
  add_count(especie, name = "n_por_especie") %>%
  add_count(especie, wt = peso, name = "peso_por_especie") %>%
  select(id, especie, n_por_especie, peso_por_especie)
```

2.3 Reestruturação dos dados com o `tidyr`

Algumas vezes os dados importados apresentam um conjunto de colunas que precisam ser transformadas em uma única com seu valor e outra com a categoria, ou o contrário. Para conseguir isso usamos as funções `pivot_longer()` e `pivot_wider()`. `pivot_longer()` retorna um dataset com mais observações e menos colunas, usando os nomes das colunas alvo para construir uma variável e o valor de cada coluna para construir outra. `pivot_wider()` retorna um dataset com mais colunas e menos observações, usando os valores de uma ou mais colunas alvo para criar novas variáveis e outra coluna para extrair os valores.

```r
dados_wide <- tibble(
  id = 1:100,
  especie = sample(c("cão", "gato"), 100, replace = TRUE),
  peso_2021 = round(rnorm(100, 13, 5), 2),
  peso_2022 = round(rnorm(100, 15.1, 2), 2),
  peso_2023 = round(rnorm(100, 11.3, 5), 2),
)

# Converter para long
dados_long <- dados_wide %>%
  pivot_longer(
    cols = starts_with("peso"),
    names_to = "ano",
    values_to = "peso",
    names_prefix = "peso_",
    names_transform = list(ano = as.integer)
  )

dados_long %>% slice_sample(n = 4)
```

```
# A tibble: 4 x 4
     id especie   ano  peso
  <int> <chr>   <int> <dbl>
1    30 gato     2021 10.8
2    19 gato     2023 11.6
```

```
3    17 gato     2021  8.12
4    27 gato     2023 17.6
```

```r
dados_wide_novo <- dados_long %>%
  pivot_wider(
    names_from = ano,
    values_from = peso,
    names_prefix = "peso_"
  )

# exemplo mais complexo
medidas_long <- tibble(
  animal_id = rep(1:3, each = 6),
  momento = rep(c("antes", "depois"), each = 3, times = 3),
  parametro = rep(c("peso", "temperatura", "frequencia"), times = 6),
  valor = round(runif(18, 10, 40), 1)
)
medidas_wide <- medidas_long %>%
  pivot_wider(
    names_from = c(parametro, momento),
    values_from = valor,
    names_sep = "_"
  ) %>%
  select(animal_id, starts_with("peso"), starts_with("temperatura"), starts_with("frequencia"))
nomes_tabela <- c("Animal", "Peso (antes)", "Peso (depois)", "Temp (antes)", "Temp (depois)", "Freq (antes
kable(medidas_wide, col.names = nomes_tabela)
```

Animal	Peso (antes)	Peso (depois)	Temp (antes)	Temp (depois)	Freq (antes)	Freq (depois)
1	17.1	25.0	26.7	13.5	29.9	29.3
2	28.8	20.6	17.7	19.2	17.1	33.1
3	16.9	17.7	23.8	23.8	35.1	35.0

Também é possível separar os valores de uma variável em novas variáveis usando o `separate()` ou unir os valores de diferentes variáveis em uma única usando o `unite()`.

```r
df_exemplo_2 <- tibble(
  id = 1:5,
  info = c(
    "cão_macho_5anos",
    "gato_femea_3anos",
    "cão_femea_8anos",
    "gato_macho_2anos",
    "coelho_macho_1anos"
  )
)

# Separar em múltiplas colunas
dados_separados <- df_exemplo_2 %>%
  separate(
    info,
```

```r
    into = c("especie", "sexo", "idade"),
    sep = "_",
    convert = TRUE
  ) %>%
  mutate(idade = as.integer(str_remove(idade, "anos")))

# Unir colunas
dados_unidos <- dados_separados %>%
  unite("descricao", especie, sexo, sep = ", ", remove = FALSE)

# separate_rows() separa em múltiplas linhas
dados_unidos <- tibble(
    veterinario = c("Dr. Fulano", "Dra. Sicrano"),
    especialidades = c("cirurgia,clinica", "clinica,dermatologia,cardiologia")
  ) %>%
  separate_rows(especialidades, sep = ",")
```

Quando temos dados ausentes (`NA`) em variáveis do nosso conjunto de dados, normalmente, algum tratamento será necessário para lidar com essas observações, seja removendo elas ou substituindo por outro valor. `fill()` substitui os valores `NA` das variáveis alvo por valores anteriores e posteriores àquela observação no conjunto de dados, enquanto `replace_na()` substitui os valores `NA` por um valor padrão. Já o `drop_na()` remove a unidade observacional inteira que apresente um `NA` nas variáveis alvo.

```r
dados_na <- tibble(
  dia = 1:7,
  temperatura = c(38.5, NA, 38.7, NA, NA, 39.0, 38.6),
  medicacao = c("A", NA, NA, "B", NA, NA, "C")
)

# fill() - mais útil para dados temporais
dados_na %>%
  fill(temperatura, .direction = "down")
dados_na %>%
  fill(everything(), .direction = "up")
dados_na %>%
  fill(everything(), .direction = "updown")

# replace_na()
dados_na %>%
  replace_na(list(
    temperatura = 38.5,
    medicacao = "Nenhuma"
  ))

# drop_na()
dados_na %>%
  drop_na()
dados_na %>%
  drop_na(temperatura)
```

Por vezes em nossa análise chegamos em um ponto onde é necessário dividir nosso dataset em multiplos

conjuntos, baseado em alguma categoria, seja para aplicar algum teste ou ajustar um modelo. Esse processo, normalmente, envolveria criar várias variáveis cada uma com o dataset filtrado para a categoria de interesse e depois aplicar o teste a cada uma desses subconjuntos. Entretanto, o `tidyr` oferece uma estratégia mais elegante para esse processo por meio do aninhamento usando o `nest()`. O `nest()` permite criar um novo `tibble` em que, em uma coluna do tipo lista, cada observação se torna um `tibble` filtrado para a categoria de interesse, assim, podemos aplicar o teste de forma iterativa em cada um, por meio de funções de programação funcional do `purrr`, mantendo também o resultado em formato de lista para cada observação.

```
dados_aninhados <- df_exemplo %>%
  group_by(especie) %>%
  nest() %>%
  mutate(
    n_observacoes = map_int(data, nrow),
    modelo = map(data, ~lm(peso ~ idade, data = .x)),
    teste_media_peso_vacinado = map(data, ~t.test(peso ~ vacinado, data = .x))
  )

dados_desaninhados <- dados_aninhados %>%
  select(-modelo) %>%
  unnest(data) %>%
  ungroup()

# resultados de gato
summary(dados_aninhados$modelo[[1]])
```

```
Call:
lm(formula = peso ~ idade, data = .x)

Residuals:
     Min       1Q   Median       3Q      Max 
-10.5686  -3.9560   0.4428   2.6257  12.5723 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  12.1732     1.9516   6.237    3e-07 ***
idade         0.3133     0.2401   1.305      0.2    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.364 on 37 degrees of freedom
Multiple R-squared:  0.04399,	Adjusted R-squared:  0.01816 
F-statistic: 1.703 on 1 and 37 DF,  p-value: 0.2
```

```
dados_aninhados$teste_media_peso_vacinado[[1]]
```

```
    Welch Two Sample t-test

data:  peso by vacinado
t = 0.0048531, df = 34.736, p-value = 0.9962
alternative hypothesis: true difference in means between group FALSE and group TRUE is not equal to 0
```

```
95 percent confidence interval:
 -3.539016  3.555972
sample estimates:
mean in group FALSE  mean in group TRUE
           14.46500            14.45652
```

2.4 Unindo diferentes tabelas com `dplyr`

Outra situação com que se depara na análise de dados é a necessidade de importar dados de diferentes planilhas (tabelas), as quais depois precisamos unir em um único dataset, baseado na informação de alguma variável que ocorre em ambas as tabelas. Isso é particularmente comum quando trabalhamos com dados importados de bancos de dados relacionais, onde temos uma tabela que possui uma coluna de chave primária (valores únicos) e outra com chave estrangeira que indica que aquele registro se refere à observação única daquela outra tabela.

Para unir essas tabelas usamos `joins`, uma operação derivada do SQL, uma linguagem de consulta de banco de dados relacionais. `joins` guardam uma semelhança com a teoria de conjuntos (união, diferença, intersercção).

Figura 2.1: Tipos de joins do `dplyr`

```r
animais <- tibble(
  id = 1:5,
  nome = c("Rex", "Mia", "Bob", "Luna", "Max"),
  especie = c("cão", "gato", "cão", "gato", "coelho")
)

consultas <- tibble(
  id_animal = c(1, 2, 1, 3, 6),  # id 6 não existe no tibble animais
  data = as.Date(c(
    "2023-01-10", "2023-01-15", "2023-02-01",
    "2023-02-10", "2023-02-15"
  )),
  motivo = c("vacina", "checkup", "checkup", "vacina", "emergência")
```

```r
)

# inner_join - mantém apenas registros com correspondência
inner <- animais %>%
  inner_join(consultas, by = c("id" = "id_animal"))

# left_join - mantém todos da esquerda
left <- animais %>%
  left_join(consultas, by = c("id" = "id_animal"))

# right_join - mantém todos da direita
right <- animais %>%
  right_join(consultas, by = c("id" = "id_animal"))

# full_join - mantém todos
full <- animais %>%
  full_join(consultas, by = c("id" = "id_animal"))

# semi_join - mantém linhas de L que têm match em R
animais_com_consulta <- animais %>%
  semi_join(consultas, by = c("id" = "id_animal"))

# anti_join - mantém linhas de L que NÃO têm match em R
animais_sem_consulta <- animais %>%
  anti_join(consultas, by = c("id" = "id_animal"))
```

2.5 Manipulando texto, datas e fatores com `stringr`, `lubridate` e `forcats`

Quando nosso dataset possui variáveis que representam um texto, por exemplo, uma resposta aberta, é normal ter que manipular essa variável para padronizá-la antes de uma análise. O `stringr` oferece um conjunto de funções que permitem manipular dados do tipo `character`, que inclui meios de tranformar o texto em maiúsculas ou minúsculas, detectar certas palavras, contar o número de ocorrências de uma palavra, remover ou substituir uma palavra por outra ou, ainda, usar expressões regulares (*REGEX*) para tranformar seus dados iniciais.

```r
exemplo <- tibble(
  id = 1:5,
  descricao = c(
    "  Infecção respiratória AGUDA  ",
    "dermatite alérgica crônica",
    "FRATURA do fêmur esquerdo",
    "gastroenterite viral",
    "Otite média bilateral"
  )
)

# Funções básicas
diagnosticos_limpos <- exemplo %>%
```

```r
  mutate(
    # Remover espaços extras no inicio e fim do texto
    descricao_limpa = str_trim(descricao),

    # Converter para minúsculas
    descricao_lower = str_to_lower(descricao_limpa),

    # Converter para título ("Um Texto Dessa Forma")
    descricao_title = str_to_title(descricao_limpa),

    # Detectar padrões
    tem_infeccao = str_detect(descricao_lower, "infec"),

    # Extrair palavras
    primeira_palavra = str_extract(descricao_limpa, "^\\w+"),

    # Substituir
    descricao_mod = str_replace(descricao_lower, "aguda|crônica", "***"),

    # Contar palavras
    n_palavras = str_count(descricao_limpa, "\\w+")
  )

diagnosticos_limpos

# Expressões regulares
telefones <- c("(11) 1234-5678", "11-98765.4321", "1112345678", "11 1234 5678")
telefones_limpos <- telefones %>%
  str_remove_all("[^0-9]") %>%  # Remove tudo exceto números
  str_replace("^(\\d{2})(\\d{4,5})(\\d{4})$", "(\\1) \\2-\\3")

telefones_limpos
```

O `lubridate` por sua vez permite trabalhar com dados temporais (data e tempo). Com ele você converter dados para os tipos `Date` e `POSIXct` (*date-time*), usar aritméticas entre datas, adcionar ou remover *timezones*, além de criar períodos, durações e intervalos.

```r
datas_texto <- c("01/03/2023", "15-06-2023", "2023-12-25")

datas <- tibble(
  texto = datas_texto,
  data_dmy = dmy(c("01/03/2023", "15/06/2023", "25/12/2023")),
  data_dmy_hm = dmy_hm(c("01/03/2023 15:20", "15/06/2023 08:12", "25/12/2023 17:30")),
  data_ymd = ymd("2023-12-25")
)

class(df_exemplo$data_consulta)

# obter os componentes da data
consultas_datas <- df_exemplo %>%
  mutate(
```

```r
    ano = year(data_consulta),
    mes = month(data_consulta, label = TRUE),
    dia = day(data_consulta),
    dia_semana = wday(data_consulta, label = TRUE),
    semana_epidemio = epiweek(data_consulta),
    trimestre = quarter(data_consulta),
    dia_ano = yday(data_consulta)
  )

intervencoes <- tibble(
  inicio = ymd(c("2023-01-01", "2023-03-15", "2023-06-01")),
  fim = ymd(c("2023-01-15", "2023-04-01", "2023-06-30"))
) %>%
  mutate(
    duracao_dias = as.numeric(fim - inicio),
    duracao_semanas = as.numeric(difftime(fim, inicio, units = "weeks")),
    meio_periodo = inicio + days(as.integer(duracao_dias / 2))
  )

# criando periodos
intervencoes$inicio + years(1)
# criando durações
intervencoes$inicio + dyears(1)
# criando intervalos
intevalo_um_ano <- interval(
  intervencoes$inicio, intervencoes$inicio + years(1)
)
# testando se uma data está dentro de um intervalo
(intervencoes$inicio[1] + days(20)) %within% intevalo_um_ano

# Sequências de datas
calendario_vacinacao <- tibble(
  data = seq(ymd("2023-01-01"), ymd("2023-12-31"), by = "month"),
  tipo = "Vacinação mensal"
)

# Arredondar datas
agora <- now()
tibble(
  original = agora,
  hora = floor_date(agora, "hour"), # floor arredonda para baixo
  dia = round_date(agora, "day"), # round arredonda para o mais próximo
  semana = ceiling_date(agora, "week"), # ceiling arredonda para cima
  mes = floor_date(agora, "month")
)
```

Para finalizar nossa seção sobre o `tidyverse` temos o pacote `forcats` para manipulação de fatores. Ele possui ferramentas para alterar ou reordenar os fatores de uma variável de forma simples.

```r
dados_fatores <- df_exemplo %>%
  mutate(
```

```r
    especie_fator = factor(especie),
    especie_freq = fct_infreq(especie_fator), # Reordenar níveis por frequência
    especie_peso = fct_reorder(especie_fator, peso, mean), # Reordenar por outra variável

    # Recodificar níveis
    especie_pt = fct_recode(
      especie_fator,
      "Canino" = "cão", # padrão "Novo nome" = "Antigo nome"
      "Felino" = "gato",
      "Lagomorfo" = "coelho"
    ),

    # Agrupar níveis raros
    especie_agrup = fct_lump_min(
      especie_fator, min = 30, other_level = "Outros"
    )
  )

levels(dados_fatores$especie_fator)
levels(dados_fatores$especie_freq)

dados_fatores %>%
  group_by(especie) %>%
  summarise(media = mean(peso)) %>%
  arrange(media)
levels(dados_fatores$especie_peso)

levels(dados_fatores$especie_pt)
levels(dados_fatores$especie_agrup)
```

Os pacotes apresentados possuem uma grande gama de ferramentas, e nem todas serão apresentadas no nosso treinamento. Caso queira aprender mais sobre elas, recomendo o livro R para Ciência de Dados, o qual possui uma versão online de acesso aberto ou a documentação dos pacotes do `tidyverse` que podem ser encontrados em https://www.tidyverse.org/packages/.

Chapter 3

Estatística descritiva

Estatísticas descritivas fazem parte de todo trabalho de análise de dados. Podemos ver ela como um passo inicial, em que você utiliza técnicas de análise exploratória para entender melhor o comportamento dos seus dados considerando o todo, por meio de tabelas, gráficos e medidas-resumo. Na estatística descritiva univariada buscamos o comportamento de uma variável isolada das demais do conjunto de dados, enquanto nas estatísticas bivariada ou multivariada exploramos o comportamento em conjunto de duas ou mais variáveis.

Para exemplificar as estatísticas descritivas em nosso treinamento, usaremos o dataset `ap2` (mais informações desse conjunto de dados podem ser verificadas no arquivo dicionário de dados.pdf).

O tipo de variável em estudo irá influenciar na técnica que você escolhe para analisá-la, assim, podemos sintetizar as possibilidades de análise da seguinte forma para estatísticas univariadas:

Estatísticas descritivas univariadas para variáveis qualitativas. Fonte: adaptado de Favaro *et al.* (2017)

Tipo de Variável
Quantitativa
Tabelas
Medidas-resumo
Distribuição de frequências
Forma
Dispersão ou Variabilidade
Forma
Assimetria
Curtose
Dispersão ou Variabilidade
Amplitude
Desvio-médio
Variância

Estatísticas descritivas univariadas para variáveis quantitativas (tabelas e medidas-resumo). Fonte: adaptado de Favaro *et al.* (2017)

Estatísticas descritivas univariadas para variáveis quantitativas (gráficos). Fonte: adaptado de Favaro *et al.* (2017)

3.1 Tabela de distribuição de frequências (Univariada)

Tabelas de frequências representam o número absoluto ou relativo de ocorrências de uma categoria (variáveis qualitativas), valor (variáveis quantitativas discretas) ou intervalo de valores (variáveis quantitativas contínuas).

Assim, a tabela será composta dos possíveis valores:

- Frequência absoluta (F_i): contagem das observações na classe i;
- Frequência relativa (Fr_i): contagem das observações na classe i em relação ao total de observações da variável;
- Frequência acumulada (F_{ac_i}): soma das observações na classe i e nas anteriores a ela (só faz sentido se há noção de ordem nos dados);
- Frequência acumulada relativa (Fr_{ac_i}): soma das frequências relativas até a classe i (inclusive);

No caso de variáveis contínuas, para criarmos tabelas de frequências delas, primeiro criamos classes representado intervalos de valores, depois contamos o número de observações em cada intervalo. O número de classes e o intervalo de cada classe é arbitrário, entretanto, Bussab e Morettin (2011) sugerem o seguinte algoritmo para construção da tabela:

1. Ordenar os dados do menor ao maior;
2. Calcular o número de classes (k) por

- $k = 1 + 3,322 \times \log_{10} n$ (Equação de Sturges)
- $k = \sqrt{n}$
- n é o número de observações e k deve ser arredondado para o inteiro mais próximo;

3. Calcular o intervalo das classes, Δ_h, como $\Delta_h = \frac{A}{k}$, em que A é a amplitude (máximo valor menos o mínimo);
4. Construa os intervalos iniciando pelo menor valor como primeiro limite inferior e somando Δ_h ao limite inferior de cada classe para obter os limites superiores. Cada intervalo, exceto o primeiro, será aberto (exclui) no limite inferior e fechado (inclui) no superior, e o limite inferior das classes após a primeira serão o limite superior da anterior (ex.: {[1, 11],]11, 21],]21, 31],]31, 41]});
5. Conte o número de observações que possuem valores dentro de cada intervalo para construir a tabela de frequências.

No R, as tabelas de frequências podem ser obtidas da seguinte forma:

```
ap2 <- haven::read_dta(
  "../datasets/stata/ap2.dta",
  encoding = "UTF-8"
  ) %>%
  as_factor() %>%
  mutate(
    vacc_mp = fct_recode(
      vacc_mp,
      "Vacinado" = "vac",
      "Não vacinado" = "not vac."
    )
  )

# Qualitativa
tab_frequencias_qualitativa <- ap2 %>%
  count(vacc_mp, name = "Fi") %>%
  mutate(
    Fri = scales::percent(Fi / sum(Fi), accuracy = .1),
    Fac = cumsum(Fi),
    Frac = scales::percent(cumsum(Fi / sum(Fi)), accuracy = .1)
  )

# Quantitativa discreta
tab_frequencias_discreta <- ap2 %>%
  count(parity, name = "Fi") %>%
  mutate(
    Fri = scales::percent(Fi / sum(Fi), accuracy = .1),
    Fac = cumsum(Fi),
    Frac = scales::percent(cumsum(Fi / sum(Fi)), accuracy = .1)
  )

# Quantitativa contínua
cria_classes_continua <- function(variavel, tipo = "sqrt") {
  if(!(tipo %in% c("sqrt", "sturges"))){
    cat("O tipo informado não está disponível\n")
    return(NULL)
  }
```

```r
  if(tipo == "sqrt") {
    classes_k <- round(sqrt(length(variavel)))
    breaks <- seq(
      min(variavel),
      max(variavel),
      length.out = classes_k + 1
    )
    breaks[length(breaks)] <- ceiling(max(variavel))

    return(list(k = classes_k, breaks = breaks))
  }

  classes_k <- round(1 + 3.322 * log10(length(variavel)))
  breaks <- seq(
    min(variavel),
    max(variavel),
    length.out = classes_k + 1
  )
  breaks[length(breaks)] <- ceiling(max(variavel))

  return(list(k = classes_k, breaks = breaks))
}

k_breaks <- cria_classes_continua(ap2$w_age_t, tipo="sturges")
breaks <- k_breaks[["breaks"]]

tab_frequencias_continua <- ap2 %>%
  mutate(
    w_age_t_classificado = cut(w_age_t, breaks = breaks, right = TRUE, include.lowest = TRUE)
  ) %>%
  count(w_age_t_classificado, name = "Fi") %>%
  mutate(
    Fri = scales::percent(Fi / sum(Fi), accuracy = .1),
    Fac = cumsum(Fi),
    Frac = scales::percent(cumsum(Fi / sum(Fi)), accuracy = .1)
  )
```

```r
tabela_headers <- c("Classe", "$F_i$", "$Fr_i$", "$F_{ac_i}$", "$Fr_{ac_i}$")

kable(tab_frequencias_qualitativa, col.names = tabela_headers)
```

Tabela 3.1: Distribuição de frequências de uma variável qualitativa

Classe	F_i	Fr_i	F_{ac_i}	Fr_{ac_i}
Não vacinado	419	37.6%	419	37.6%
Vacinado	695	62.4%	1114	100.0%

```r
kable(tab_frequencias_discreta, col.names = tabela_headers)
```

Tabela 3.2: Distribuição de frequências de uma variável quantitativa discreta

Classe	F_i	Fr_i	F_{ac_i}	Fr_{ac_i}
1	199	17.9%	199	17.9%
2	194	17.4%	393	35.3%
3	158	14.2%	551	49.5%
4	157	14.1%	708	63.6%
5	162	14.5%	870	78.1%
6	127	11.4%	997	89.5%
7	50	4.5%	1047	94.0%
8	36	3.2%	1083	97.2%
9	16	1.4%	1099	98.7%
10	9	0.8%	1108	99.5%
11	6	0.5%	1114	100.0%

```r
kable(tab_frequencias_continua, col.names = tabela_headers)
```

Tabela 3.3: Distribuição de frequências de uma variável quantitativa contínua

Classe	F_i	Fr_i	F_{ac_i}	Fr_{ac_i}
[12.1,15.3]	15	1.3%	15	1.3%
(15.3,18.5]	65	5.8%	80	7.2%
(18.5,21.7]	155	13.9%	235	21.1%
(21.7,24.9]	201	18.0%	436	39.1%
(24.9,28.1]	189	17.0%	625	56.1%
(28.1,31.2]	169	15.2%	794	71.3%
(31.2,34.4]	137	12.3%	931	83.6%
(34.4,37.6]	97	8.7%	1028	92.3%
(37.6,40.8]	64	5.7%	1092	98.0%
(40.8,44]	17	1.5%	1109	99.6%
(44,48]	5	0.4%	1114	100.0%

3.2 Representação gráfica dos dados de uma variável

Os gráficos hoje são insdispensáveis na estatística e análise de dados, servindo como uma ponte entre os dados brutos e seu comportamento em conjunto. Normalmente os usamos para detectar padrões, tendências, anomalias e relações que não seriam percebidos em tabelas ou medidas resumo.

Na análise exploratória de dados (EDA), histogramas, boxplots e gráficos de dispersão permitem identificar a forma da distribuição, detectar outliers e avaliar a simetria. Também usamos gráficos para auxiliar na verificação de suposições de modelos estatísticos, por exemplo, quando criamos um gráfico Q-Q para identificar desvios da normalidade.

Além disso, os gráficos são ferramentas para a comunicação estatística, os quais usamos para traduzir conceitos complexos em representações visuais mais intuitivas e didáticas que facilitem a compreensão por um público que não necessariamente possui um conhecimento formal de estatística, mas que podem ser

o alvo dos nossos resultados, por exemplo, um gestor público ou de uma empresa cujas nossas análises poderiam (ou deveriam) trazer uma informações relevantes para aumentar a eficácia de sua administração.

Em contextos como epidemiologia veterinária, um gráfico de série temporal pode sugerir surtos epidêmicos ou sazonalidade de doenças, mapas de calor espaciais podem revelar clusters de casos, indicando os locais para intensificar as ações de defesa sanitária.

Em nossos treinamentos, a maioria dos gráficos serão construídos usando o pacote `ggplot2` do `tidyverse`. Existem muitas fontes de informação para o aprendizado do `ggplot2` e da criação de gráficos no R, porém, deixo aqui duas recomendações que considero como um "guia de bolso" para criação de gráficos no R, que são o livro *R Graphics Cookbook* do Winston Chang (2025), um livro online de acesso aberto com várias explicações sobre a criação de gráficos com o `ggplot2`, e o site The R Graph Gallery onde você encontra explicações e exemplos de código para a criação dos mais variados gráficos no R. Agora, se você quer um conteúdo avançado sobre o `ggplot2`, o livro online ggplot2: Elegant Graphics for Data Analysis é o que você procura.

3.2.1 Criação de gráficos no `ggplot2`

O ggplot2 implementa os conceitos da *Grammar of Graphics* de Leland Wilkinson, onde gráficos são construídos através de **camadas semânticas**, as quais são combinadas de forma modular e sistemática. Seus principais aspectos são:

- **Decomposição**: Todo gráfico pode ser decomposto em componentes independentes;
- **Composição**: Componentes são combinados usando o operador `+`;
- **Declarativo**: Você descreve O **QUE** quer, não **COMO** desenhar.

Todo gráfico criado com o `ggplot2` vai depender da seguinte estrutura mínima:

```
ggplot(
  data = <DATA>,            # 1. Dados (fonte da informação para o gráfico)
  mapping = aes(<MAPPINGS>) # 2. Mapeamentos estéticos (quais variáveis serão usadas e onde)
  ) +
  <GEOM_FUNCTION>()         # 3. Geometria (o que será desenhado)
```

Assim, `ggplot()` inicia a construção do gráfico, indicando às camadas posteriores de construção qual o dataset fonte das variáveis (argumento `data`) e, em geral, como serão mapeadas as variáveis para o gráfico (quem será o eixo das ordenadas? e das abscissas? exitem variáveis categóricas que gostaríamos de usar como fonte para as cores do que será apresentado, ou seja, criar subconjuntos?). Iniciado o `ggplot()`, que podemos pensar como um quadro em branco, passamos a de fato desenhar nosso gráfico com `geom_*()` e outras camadas de estilização.

Além desses componentes primários, outros que costumam ser usados na construção de gráficos são:

- **Facets**: Divisão em subgráficos
- **Statistics (stat)**: Transformações estatísticas
- **Coordinates (coord)**: Sistema de coordenadas
- **Themes**: Aparência não relacionada aos dados
- **Scales**: Controle de mapeamentos

Vamos então observar na prática como é construção do gráfico no `ggplot2`. Iniciamos com `ggplot(data=dados)`, o que gera somente o nosso painel, sem qualquer escala ainda.

```
set.seed(42)

dados <- data.frame(
```

```r
  x = rnorm(100, mean = 10, sd = 2),
  y = rnorm(100, mean = 15, sd = 3),
  grupo = sample(c("A", "B", "C"), 100, replace = TRUE),
  tamanho = runif(100, 1, 10)
)

p_base <- ggplot(dados)
p_base  # Produz apenas o painel vazio
```


Figura 3.1: Chamada de `ggplot` somente com argumento data

O `aes()` pode ser informado no próprio `ggplot`, caso esse mapeamento vá ser usado da mesma forma por todas as demais camadas, ou pode ser informado a cada `geom_*()`, caso seja algo específico daquela geometria. Os principais elementos estéticos definidos no `aes()` são: `x` (eixo das abscissas), `y` (eixo das ordenadas), `color` (cor de delimitação), `fill` (cor de preenchimento), `size` (tamanho), `alpha` (transparência), `shape` (forma), `linetype` (tipo de linha).

```r
p_base <- ggplot(dados, aes(x = x, y = y))

# Mapeamento geral
p1 <- ggplot(dados, aes(x = x, y = y)) +
  geom_point()

# Mapeamento local (na geometria)
p2 <- ggplot(dados) +
  geom_point(aes(x = x, y = y))

# Mapeamentos múltiplos
```

```r
p3 <- ggplot(dados, aes(x = x, y = y)) +
  geom_point(aes(color = grupo, size = tamanho), alpha = 0.6)  # alpha fixo (não mapeado)
```

```r
p_base
p1
p2
p3
```

(a) Chamada de `ggplot(dados, aes(x = x, y = y))` sem uma geometria

(a) Chamada de `ggplot(dados, aes(x = x, y = y))` com uma geometria de pontos

(a) `aes()` definidos na geometria

(a) `aes()` em diferentes camadas

Uso do `aes()`

Como já falado, o `ggplot2` segue um sistema de camadas, então cada operação após o `+` adiciona uma nova feição sobre o gráfico e pode inclusive sobrescrever geometrias definidas em operações anteriores, então é importante verificar a ordem em que são declarados os aspectos visuais do gráfico. Observe abaixo, como alterar a ordem de declaração das geometrias "esconde" certos pontos no gráfico da esquerda em relação ao da direita.

```r
p_ordem1 <- ggplot(dados, aes(x, y)) +
  geom_smooth(method = "lm", se = TRUE, color = "blue", linewidth = 4) +
  geom_point(size = 4, color = "black")  # Pontos sobre a linha
```

```r
p_ordem2 <- ggplot(dados, aes(x, y)) +
  geom_point(size = 4, color = "black") +
  geom_smooth(method = "lm", se = TRUE, color = "blue", linewidth = 4)  # Linha sobre pontos

p_ordem1
```

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

```r
p_ordem2
```

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

(a) Chamada de `ggplot() + geom_smooth() + geom_point()`

(a) Chamada de `ggplot() + geom_point() + geom_smooth()`

Importância de definir a ordem correta das geometrias no seu gráfico.

Quando definimos um `aes()` na chamada de `ggplot()`, ele será herdado pelas demais camadas, porém, ao usarmos ele em uma geometria, ele só será usado nela.

```r
p_heranca <- ggplot(dados, aes(x, y, color = grupo)) +  # color global
  geom_smooth(method = "lm", se = FALSE) +
  geom_point(size = 3)

p_sem_heranca <- ggplot(dados, aes(x, y)) +
  geom_smooth(method = "lm", se = FALSE) +
  geom_point(size = 3, aes(color = grupo))

p_heranca
p_sem_heranca
```

Quando queremos criar múltiplos gráficos da mesma geometria, segundo uma variável alvo, podemos usar o `facet_*()` para criação de vários subpainéis.

Com `facet_wrap(~ variavel_alvo)` os painéis são dispostos horizontalmente considerando o número de categorias. Por padrão, os painéis serão dispostos em uma tabela quadrada, mas essa disposição pode ser alterada usando os argumentos `nrow` para o número de linhas e `ncol` para o número de colunas.

(a) `ggplot() + geom_smooth() + geom_point()` com herança de `aes()` do `ggplot()`

(a) Chamada de `ggplot() + geom_point() + geom_smooth()` com `aes()` específico para `geom_point()`

Herança de `aes()` no `ggplot2`.

```
p_wrap <- ggplot(dados, aes(x, y)) +
  geom_point(color = "steelblue") +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  facet_wrap(~ grupo, ncol = 2)

p_wrap
```

Figura 3.10: Múltiplos painéis no `ggplot2` com `facet_wrap`.

Observe na figura acima que o nome das categorias são exibidas como *headers* de cada painel, assim, se queremos alterar o título da categoria de cada painel, seria necessário alterar a própria variável (usando por exemplo o `recode()` se for um dado do tipo `factor`). O argumento `labeller = label_both` faz com que o título do painel também apresente o nome da variável. Aspectos visuais do título serão alterados na camada de `theme()` do gráfico no `ggplot2`, a qual será apresentada mais adiante no treinamento, mas podemos alterar tanto características do texto (`strip.text`) quanto do fundo do título (`strip.background`).

```
p_wrap <- dados %>%
  mutate(rcd_grupo = recode(grupo, "A" = "Tratamento 1", "B" = "Tratamento 2", "C" = "Tratamento 3")) %>%
  ggplot(aes(x, y)) +
  geom_point(color = "steelblue") +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  facet_wrap(~ rcd_grupo, labeller = label_both) +
  theme(
    strip.text = element_text(face = "italic", size = rel(1.2)),
    strip.background = element_rect(fill = "#7688d6bb", colour = "black", linewidth = .7)
  )

p_wrap
```

Figura 3.11: Alterando os *headers* de painéis do `facet_*()`.

Já o `facet_grid(variavel_alvo_linha ~ variavel_alvo_coluna)` permite criar painéis a partir de subconjuntos de uma variável para as colunas e outra para as linhas do gráfico. As mesmas técnicas de estilização do `facet_wrap` se aplicam ao `facet_grid`.

```
# facet_grid: duas variáveis
dados$categoria <- sample(c("Categoria 1", "Categoria 2"), 100, replace = TRUE)
```

```r
p_grid <- ggplot(dados, aes(x, y)) +
  geom_point(aes(color = tamanho)) +
  facet_grid(categoria ~ grupo) +
  theme(
    strip.text = element_text(face = "italic", size = rel(1.2)),
    strip.background = element_rect(fill = "#7688d6bb", colour = "black", linewidth = .7)
  )

p_grid
```

Figura 3.12: Múltiplos painéis no `ggplot2` com `facet_grid`.

Para controlar a escala das estéticas mapeadas no gráfico do `ggplot2` usamos as funções da família `scale_*`. `scale_*` segue um padrão de nome onde primeiro declaramos qual estética queremos controlar (x, y, size...) e em seguida a forma ou transformação que consideramos para a variável mapeada.

Para escalas de posição (eixos x e y), usamos `scale_*_continuous` para controlar a escala que varia em um intervalo contínuo e `scale_*_discrete` se queremos que cada valor único da variável mapeada seja tratada como uma categoria.

Com `scale_*_continuous` podemos controlar os limites do eixo alvo com `limits = c(minimo, maximo)` e suas marcações com `breaks` e `minor_breaks`. `breaks` e `minor_breaks` recebe um vetor das posições das marcações ou uma função que, a partir dos limites do eixo, calculará as posições das marcações. Além disso, podemos usar o argumento `transform` para transformações dos dados naquela escala, por exemplo, aplicando a raiz quadrada ou o log, e podemos usar `labels` para alterar o texto exibido em cada marcação do eixo.

```r
p_scales <- ggplot(dados, aes(x, y, color = tamanho)) +
  geom_point(size = 3) +
```

```
  scale_x_continuous(
    name = "X",
    limits = c(5, 15),
    breaks = seq(5, 15, length.out = 3),
    labels = c("pequeno", "grande", "muito grande")
  ) +
  scale_y_continuous(
    name = "sqrt(Y)",
    transform = "sqrt"
  )

p_scales
```

Figura 3.13: Manipulando escalas no `ggplot2` com `scale_*_continuous`.

Com `scale_*_discrete` podemos também definir o nome do eixo, o texto das marcações e outras características como no `scale_*_continuous`.

```
ggplot(dados, aes(x = grupo)) +
  geom_bar(aes(y = after_stat(prop), group = 1), fill = "#ac565693", color= "black") +
  scale_y_continuous(
    name = "Frequência relativa (%)",
    labels = scales::label_percent(),
    limits = c(.0, .65),
    breaks = scales::breaks_width(width = .05),
    minor_breaks =  scales::breaks_width(width = .025),
  ) +
  scale_x_discrete(
```

```
    name = "Grupos",
    labels = c("Tratamento 1", "Tratamento 2", "Controle")
  )
```

Figura 3.14: Manipulando escalas no `ggplot2` com `scale_*_discrete`.

Em ambos, `limits` define o próprio limite para os dados mapeados, então, devemos tomar cuidado, principalmente em gráficos que calculam estatísticas para sua construção, já que dados fora desses limites serão desconsiderados

```
p_cat <- ggplot(dados, aes(grupo, y)) +
  geom_boxplot()

p_cat_scales <- ggplot(dados, aes(grupo, y)) +
  geom_boxplot() +
  scale_y_continuous(limits = c(NA, 20)) +
  scale_x_discrete(limits = c("A", "B"))

p_cat
p_cat_scales
```

```
Warning: Removed 29 rows containing missing values or values outside the scale range
(`stat_boxplot()`).
```

```
Warning: Removed 2 rows containing non-finite outside the scale range
(`stat_boxplot()`).
```

Se queremos limitar nossa escala no gráfico, sem alterar esse mapeamento e, consequentemente, haver perda de dados, podemos usar `coord_cartesian` e definir nossos limites para as ordenadas com `ylim` e as abscissas

(a) Original sem perda

(a) Usando limites nas ordenadas e nas abscissas

Perda de dados ao usar `limits` no `scale_*_`.

com `xlim`.

```
p_cat <- ggplot(dados, aes(grupo, y)) +
  geom_boxplot() +
  coord_cartesian(ylim = c(11, 20), xlim = c(1, 2))

p_cat
```

Figura 3.17: Alterando as escalas dos eixos com `coord_cartesian`.

Para alterar escalas de cores e preenchimentos usamos `scale_color_*` e `scale_fill_*`. Para criar um

gradiente contínuo podemos usar `scale_*_gradient`, o qual aceita os parâmetros `low` e `high` como os valores das cores a usar para criar seu gradiente de cores. Então, se usados ambos, será criado uma escala com um gradiente de cores variando entre a cor em `low` (menor valor dos dados e maior intensidade da cor declara em `low`) e a cor em `high` (maior valor dos dados e maior intensidade da cor declara em `high`). Se declarado somente `low`, teremos um gradiente criado a partir de uma única cor, com a maior intensidade no menor valor.

Para criar uma escala de cor divergente (gradiente com três cores, com um delas indicando um ponto central), podemos usar `scale_*_gradient2`. Além disso, podemos criar um gradiente com n cores com `scale_*_gradientn` (nesse caso, deve ser fornecido um vetor dos nomes de cores a usar no gradiente).

```
ggplot(dados, aes(x, y, color = tamanho, size = tamanho)) +
  geom_point() +
  scale_color_gradient(low = "blue")

ggplot(dados, aes(x, y, color = tamanho, size = tamanho)) +
  geom_point() +
  scale_color_gradient(low = "blue", high = "red")

ggplot(dados, aes(x, y, color = tamanho, size = tamanho)) +
  geom_point() +
  scale_color_gradient2(
    low = "blue",
    mid = "white",
    high = "red",
    midpoint = mean(dados$tamanho)
  )

ggplot(dados, aes(x, y, color = tamanho, size = tamanho)) +
  geom_point() +
  scale_color_gradientn(colors = c("blue", "red", "grey", "green"))
```

Para escalas de cores discretas, por exemplo, para diferenciar categorias, podemos usar `scale_*_manual` com o vetor de cores de cada categoria.

Para escalas de tamanho (`size`) podemos alterar o comportamento com `scale_radius()` se queremos uma escala contínua de variação, ou `scale_size_binned()` se queremos criar quebras.

```
ggplot(dados, aes(x, y, color = grupo, size = tamanho)) +
  geom_point() +
  scale_color_manual(values = c("red", "blue", "black")) +
  scale_radius(limits = c(0, NA))

ggplot(dados, aes(x, y, color = grupo, size = tamanho)) +
  geom_point() +
  scale_color_manual(values = c("red", "blue", "black")) +
  scale_size_binned() +
  guides(
    size = guide_bins(
      show.limits = TRUE,
      title = "Tamanho",
      axis.colour = "blue",
      axis.arrow = arrow(
```

(a) `scale_color_gradient` definindo somente `low`

(a) `scale_color_gradient` definindo `low` e `high`

(a) `scale_color_gradient2`

(a) `scale_color_gradientn` com 4 cores

Alterando as escalas de cores com `scale_*_gradient`.

```
        length = unit(.05, "inches"),
        ends = "first",
        type = "closed"
      )
    )
  )
```

(a) `scale_radius`

(a) `scale_size_binned`

Alterando as escalas de cores discretas com `scale_*_discrete` e tamanhos com `scale_radius` e `scale_size_binned`.

Embora seja possível definir manualmente as cores que queremos usar nas escalas, a escolha de cores interfere em muito em como o público conseguirá identificar padrões no gráfico. Por isso, recomendo usar funções que já fornecem paletas de cores criadas com uma finalidade específica como as do ColorBrewer com `scale_*_brewer` ou do pacote `paletteer`. ColorBrewer possui um site onde é possível testar a paleta de cores que queremos usar. Usando o `scale_*_brewer`, definimos o tipo de paleta (seq" para sequencial, "div" para divergente ou "qual" para qualitativo) e o nome da paleta.

```
ggplot(dados, aes(grupo, fill = grupo)) +
  geom_bar() +
  scale_fill_brewer(type = "qual")

ggplot(dados, aes(grupo, fill = grupo)) +
  geom_bar() +
  scale_fill_brewer(palette = "Pastel1")
```

Figura 3.24: `scale_fill_brewer(type = "qual")`

Figura 3.25: `scale_fill_brewer(palette = "Pastel1")`

Alterando as escalas de cores com `scale_*_brewer`.

O pacote `paletter` também possui um site para escolher sua paleta. Para usá-lo precisamos instalar o pacote e depois utilizar as funções adequadas ao tipo de dados (`scale_*_paletteer_c` para contínuas e `scale_*_paletteer_d` para discretas).

```
require(paletteer)
instala_se_nao_existe("nbapalettes")
instala_se_nao_existe("pals")
ggplot(dados, aes(x, y, color = tamanho)) +
  geom_point() +
  scale_color_paletteer_c("pals::coolwarm")

ggplot(dados, aes(grupo, fill = grupo)) +
  geom_bar() +
  scale_fill_paletteer_d("nbapalettes::supersonics_holiday")
```

```
NULL
```

(a) `scale_color_paletteer_c` para paletas contínuas

```
NULL
```

(a) `scale_fill_paletteer_d` para paletas discretas

Alterando as escalas de cores com `scale_*_paletteer`.

Para finalizar nossa introdução à criação de gráficos com o `ggplot2` vamos ver como personalizar nosso gráfico. A camada de legendas e títulos do gráfico podem ser alteradas com `labs()`. Os argumentos `title`, `subtitle` e `caption` criam textos para o título, subtítulo e nota de rodapé do gráfico, respectivamente. Outros elementos como as legendas e nomes dos eixos são representados pelo nome do argumento usado no `aes()`, por exemplo, o argumento `color` declara o título da legenda que representa as categorias dessa estética no gráfico. Caso queira remover o título de uma legenda ou eixo atribua o valor `element_blank()`, que indica ao `ggplot2` para não desenhar nada para aquele elemento.

```
p_base <- ggplot(
  dados,
  aes(x, y, color = grupo, size = tamanho, shape = categoria)
  ) +
  geom_point() +
  labs(
    title = "Gráfico de dispersão de X por Y",
    subtitle = "Um subtítulo para esse gráfico",
    x = "Título do eixo X",
    y = "Título do eixo Y",
```

```
    shape = "Legenda de shape",
    color = "Legenda de color",
    size = "Legenda de size",
    caption = "Uma nota explicativa ao conteúdo do gráfico"
  )
p_base
```

Figura 3.28: Alterando títulos e legendas em `labs()`.

Além do padrão temático inicial contruído no `ggplot2`, o pacote fornece alguns temas já contruídos que podemos usar em nosso gráfico como modelo de estilo e ir adicionando novas personalizações conforme a necessidade. Os seguintes temas estão disponíveis: `theme_gray`, `theme_bw`, `theme_linedraw`, `theme_light`, `theme_dark`, `theme_minimal`, `theme_classic` e `theme_void`.

```
p_base + theme_gray()
p_base + theme_bw()
p_base + theme_linedraw()
p_base + theme_light()
p_base + theme_dark()
p_base + theme_minimal()
p_base + theme_classic()
p_base + theme_void()
```

Cada um desses temas predefinidos possuem argumentos para certo controle de sua aparência (verifique os argumentos na documentação deles com `help(nome_do_tema)`), porém, quando queremos alterar algum elemento de aparência do gráfico, normalmente usaremos `theme()`. `theme()` permite personalizar todos componentes não definidos pelos seus dados. Sua utilização envolve declarar um argumento com o nome do componente que queremos modificar e fornecer um objeto do tipo `element_*`. `element_*`

(a) `theme_gray`

(a) `theme_bw`

(a) `theme_linedraw`

(a) `theme_light`

(a) `theme_dark`

(a) `theme_minimal`

(a) `theme_classic`

(a) `theme_void`

representam elementos temáticos: `element_blank` para declarar que o componente não deve ser desenhado, `element_rect` para bordas e planos de fundo, `element_line` para linhas, `element_text` para componentes de texto, `element_polygon` para polígonos e `element_point` para pontos.

Então, por exemplo, para personalizar o texto do título e subtítulo podemos usar em `theme()` o `plot.title = element_text(args)` e `plot.subtitle = element_text(args)`, para alterar a legenda dos eixos usar o `axis.title = element_text(args)`, para alterar as linhas de guia no gráfico usar `panel.grid.major = element_rect()` e `panel.grid.minor = element_rect()`. Além disso, alguns argumentos de `theme` não recebem `element_`, como o `legend.position`, no qual você declara onde as legendas serão alocadas (topo, abaixo, esquerda ou direita).

```
instala_se_nao_existe("showtext")
```

```
NULL
```

```
require(showtext)
font_add_google("Alegreya")
font_add_google("IBM Plex Serif")
showtext_auto()

p_base +
  theme_minimal() +
  theme(
    plot.title = element_text(
      family = "Alegreya", face = "bold", size = 12
    ),
    plot.subtitle = element_text(
      family = "Alegreya", face = "italic", size = 9
    ),
    axis.title = element_text(
      family = "IBM Plex Serif", color = "#e60e3dff", size = 9
    ),
    panel.grid.major = element_line(
      color = "black", linetype = "solid", linewidth = .5
    ),
    panel.grid.minor = element_line(
      color = "black", linetype = "dashed", linewidth = .2
    ),
    legend.position = "top"
  )
```

Figura 3.37: Personalizando temas em `theme()`.

Embora o `ggplot2` traga muitas possibilidades de criação de gráficos, há sempre espaço para inovações e especializações na comunidade do R, então recomendo o site ggplot2 extensions que traz uma série de pacotes baseados no `ggplot2`, mas com um "temperinho a mais" para determinados objetivos que poderiam ser particularmente difíceis ou tediosos de alcançar usando somente o `ggplot2` ("não reinvente a roda se ela já foi criada").

3.2.2 Representação gráfica de variáveis qualitativas (Univariada)

3.2.2.1 Gráfico de barras

Apresenta por meio de barras as frequências absolutas ou relativas de cada possível categoria (ou valor). Nesse caso o comprimento ou altura da barra representa sua frequência, enquanto cada barra representa uma classe. Embora seja mais usual para dados qualitativos, também é possível usar o gráfico de barras para representar variáveis discretas (desde que o número de possíveis valores não seja alto), ou para variáveis contínuas agrupadas em intervalos.

```
# paletteer_d("nationalparkcolors::Acadia")
# ap2 %>% count(vacc_mp)

library(ggtext)

plot_theme <- theme_minimal() +
  theme(
    plot.title = element_markdown(size = 16),
    axis.text.x = element_text(size = 12, face = "bold"),
    axis.text.y = element_text(size = 12),
    axis.title.x = element_markdown(size = 14),
```

```r
    axis.title.y = element_markdown(size = 14),
  )

ggplot(ap2) +
  geom_bar(aes(vacc_mp), fill = "#72874EFF", color = "black") +
  labs(
    title = "Frequência absoluta dos porcos vacinados contra *M. hyopneumoniae*",
    x = "Estado vacinal contra *M. hyopneumoniae*",
    y = "Frequência absoluta"
  ) +
  scale_y_continuous(
    breaks = seq(0, 750, 50)
  ) +
  plot_theme

ggplot(ap2) +
  geom_bar(
    aes(vacc_mp, y = after_stat(prop), group = 1),
    fill = "#72874EFF",
    color = "black"
  ) +
  labs(
    title = "Frequência relativa dos porcos vacinados contra *M. hyopneumoniae*",
    x = "Estado vacinal contra *M. hyopneumoniae*",
    y = "Frequência relativa (%)"
  ) +
  scale_y_continuous(
    labels = scales::label_percent(),
    breaks = scales::breaks_width(width = .05),
  ) +
  plot_theme +
  coord_flip()
```

```r
ggplot(ap2) +
  geom_bar(aes(age_t), fill = "#476F84FF", color = "black") +
  scale_x_continuous(breaks = 1:6) +
  labs(
    title = "Frequência absoluta das idades dos porcos na transferência do desmame para a unidade de acaba
    x = "Idade (dias)",
    y = "Frequência absoluta"
  ) +
  plot_theme

ggplot(ap2) +
  geom_bar(
    aes(x = age_t, y = after_stat(prop), group = 1),
    fill = "#476F84FF",
    color = "black"
  ) +
  scale_x_continuous(breaks = 1:6) +
```

(a) Frequência absoluta em gráfico de barras vertical de uma variável qualitativa

(a) Frequência relativa em gráfico de barras horizontal de uma variável qualitativa

Exemplos de gráficos de barras para variáveis qualitativas

```
  scale_y_continuous(
    labels = scales::label_percent(),
    breaks = scales::breaks_width(width = .05),
  ) +
  labs(
    title = "Frequência relativa das idades dos porcos na transferência do desmame para a unidade de acaba
    x = "Idade (dias)",
    y = "Frequência relativa (%)"
  ) +
  plot_theme
```

```
classes_k <- round(1 + 3.322 * log10(length(ap2$dwg_fin)))
breaks <- floor(seq(min(ap2$dwg_fin), max(ap2$dwg_fin), length.out = classes_k + 1))
breaks[length(breaks)] <- ceiling(max(ap2$dwg_fin))

frequencias_dwg_fin <- ap2 %>%
  mutate(
    dwg_fin_classificado = cut(dwg_fin, breaks = breaks, right = TRUE, include.lowest = TRUE, dig.lab = 4)
  ) %>%
  count(dwg_fin_classificado, name = "Fi") %>%
  mutate(
    Fri = Fi / sum(Fi),
    Fac = cumsum(Fi),
    Frac = cumsum(Fi / sum(Fi))
  )

ggplot(frequencias_dwg_fin) +
  geom_bar(aes(x = dwg_fin_classificado, y = Fi), stat = "identity", fill = "#FED789FF", color = "black", 
  labs(
    title = "Ganho de peso diário entre *age_t* e *age_t6*",
    x = "Intervalos de ganho de peso",
    y = "Frequência absoluta"
  ) +
  plot_theme

ggplot(frequencias_dwg_fin) +
  geom_bar(aes(x = dwg_fin_classificado, y = Fri), stat = "identity", fill = "#FED789FF", color = "black",
  labs(
    title = "Ganho de peso diário entre *age_t* e *age_t6*",
    x = "Intervalos de ganho de peso",
    y = "Frequência relativa"
  ) +
  scale_y_continuous(
    labels = scales::label_percent(),
    breaks = scales::breaks_width(width = .05),
  ) +
  plot_theme

ggplot(frequencias_dwg_fin) +
  geom_bar(aes(x = dwg_fin_classificado, y = Fac), stat = "identity", fill = "#FED789FF", color = "black",
```

(a) Frequência absoluta em gráfico de barras vertical de uma variável discreta

(a) Frequência relativa em gráfico de barras vertical de uma variável discreta

Exemplos de gráficos de barras para variáveis discretas

```
  labs(
    title = "Ganho de peso diário entre *age_t* e *age_t6*",
    x = "Intervalos de ganho de peso",
    y = "Frequência acumulada"
  ) +
  plot_theme

ggplot(frequencias_dwg_fin) +
  geom_bar(aes(x = dwg_fin_classificado, y = Frac), stat = "identity", fill = "#FED789FF", color = "black"
  labs(
    title = "Ganho de peso diário entre *age_t* e *age_t6*",
    x = "Intervalos de ganho de peso",
    y = "Frequência relativa acumulada"
  ) +
  scale_y_continuous(
    labels = scales::label_percent(),
    breaks = scales::breaks_width(width = .05),
  ) +
  plot_theme
```

3.2.3 Gráfico de setores ou pizza

O gráfico de setores divide um círculo de raio arbitrário em n partes (n igual ao número de categorias), sendo a área de cada parte correspondente à frequência relativa da categoria. Em geral, não é um gráfico indicado quando temos muitas categorias ou diferenças muito pequenas entre as frequências de cada categoria, já que visualmente torna-se difícil detectar essas diferenças.

```
contagem <- ap2 %>%
  count(vacc_mp, name = "quantidade") %>%
  mutate(prop = quantidade / sum(quantidade))


ggplot(contagem, aes(x="", y=quantidade, fill=vacc_mp)) +
  geom_bar(stat="identity", width=1) +
  coord_polar("y", start=0) +
  theme_void() +
  geom_text(aes(label = scales::percent(prop, accuracy = 0.01)), position = position_stack(vjust = 0.5)) +
  scale_fill_brewer(palette="Pastel1") +
  labs(fill = "Estado Vacinal")
```

(a) Frequência absoluta em gráfico de barras vertical de uma variável contínua agrupada

(a) Frequência relativa em gráfico de barras vertical de uma variável contínua agrupada

Figura 3.46: Exemplo de gráfico de setores

3.2.4 Diagrama de Pareto

O diagrama de Pareto é um gráfico de barras sobreposto por um gráfico de linhas e pontos. Nele as barras representam as frequências das categorias em ordem decrescente, enquanto as linhas representam a frequência acumulada. É uma ferramenta visual, normalmente, utilizada para avaliar as principais causas de problemas em linhas de produção.

```
mortalidade_producao <- tibble(
  causa = c(
    "Mortalidade neonatal",
    "Pneumonia",
    "Diarreia",
    "Problemas de parto",
    "Esmagamento",
    "Síndrome MMA",
    "Defeitos congênitos",
    "Canibalismo",
    "Problemas locomotores",
    "Outras causas"
  ),
  frequencia = c(
    rpois(1, 45),
    rpois(1, 38),
    rpois(1, 32),
    rpois(1, 25),
    rpois(1, 20),
    rpois(1, 15),
```

```r
    rpois(1, 10),
    rpois(1, 8),
    rpois(1, 5),
    rpois(1, 12)
  )
  ) %>%
  arrange(desc(frequencia)) %>%
  mutate(
    freq_acumulada = cumsum(frequencia),
    freq_total = sum(frequencia),
    percentual = (frequencia / freq_total) * 100,
    percentual_acumulado = (freq_acumulada / freq_total) * 100,
    causa = factor(causa, levels = causa)
  )

max_freq <- max(mortalidade_producao$frequencia)
fator_escala <- max_freq / 100

ggplot(mortalidade_producao, aes(x = causa)) +
  geom_col(
    aes(y = frequencia),
    fill = "blue",
    alpha = 0.8,
    color = "red",
    linewidth = 0.5
  ) +
  geom_line(
    aes(y = percentual_acumulado * fator_escala, group = 1),
    color = "red",
    linewidth = 1.2
  ) +
  geom_point(
    aes(y = percentual_acumulado * fator_escala),
    color = "red",
    size = 3
  ) +
  geom_text(
    aes(y = frequencia, label = frequencia),
    vjust = -0.5,
    size = 3
  ) +
  geom_text(
    aes(
      y = percentual_acumulado * fator_escala,
      label = paste0(round(percentual_acumulado, 0), "%")
    ),
    vjust = -2,
    size = 3,
    color = "red"
  ) +
```

```r
  geom_hline(
    yintercept = 80 * fator_escala,
    linetype = "dashed",
    color = "gray40",
    alpha = 0.7
  ) +
  scale_y_continuous(
    name = "Frequência",
    sec.axis =
    sec_axis(
      ~ . / fator_escala,
      name = "Porcentagem Acumulada (%)",
      breaks = seq(0, 100, 20)
    )
  ) +
  labs(
    title = "Principais causas de mortalidade na granja",
    x = ""
  ) +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1, size = 10),
    axis.title.y.right = element_text(color = "red"),
    axis.text.y.right = element_text(color = "red"),
    plot.title = element_text(
      size = 14, face = "bold", hjust = 0.5
    ),
    panel.grid.major.x = element_blank(),
    panel.grid.minor = element_blank()
  )
```

Figura 3.47: Exemplo de gráfico de Pareto

3.2.5 Outras opções gráficas para variáveis qualitativas

As opções a seguir, embora não sejam tão comuns para análise exploratória e estatística descritiva em relatórios científicos, podem ser interessantes em apresentações e divulgações científicas, pelo possibilibilidade de conseguir um visual mais atrativo para o público, principalmente quando trabalhamos com mais de uma variável no mesmo gráfico (como veremos mais a frente).

O gráfico de waffle utiliza uma grade de pequenos quadrados de tamanhos iguais para representar partes de um todo, semelhante a um gráfico de pizza, com cada quadrado representando uma determinada porcentagem ou valor. Esse gráfico permite demonstrar proporções e o progresso em direção a uma meta, e são fáceis de entender devido à sua estrutura de grade simples. Eles podem ser criados com um número fixo de células, por exemplo, em uma grade de 10×10 cada célula 1%.

```
bvd_test <- read_dta(
  "../datasets/stata/bvd_test.dta",
  encoding = "UTF-8"
  ) %>%
  as_factor() %>%
  mutate(
    season = fct_recode(
      season,
      "Inverno" = "Winter",
      "Primavera" = "Spring",
      "Verão" = "Summer",
      "Outono" = "Autumn"
    )
  )
```

```r
bvd_test %>%
  count(season)
```

```
# A tibble: 4 x 2
  season        n
  <fct>     <int>
1 Inverno     826
2 Primavera   824
3 Verão       151
4 Outono      361
```

```r
criar_waffle_data <- function(
  data,
  var,
  n_linhas = 10,
  total = 100) {

  contagem <- data %>%
    count(.data[[var]])

  valores <- contagem$n
  names(valores) <- contagem[[var]]
  nomes <- names(valores)

  # Normalizar valores para o total especificado
  if(is.null(nomes)) nomes <- paste0("Cat", seq_along(valores))
  proporcoes <- round(valores / sum(valores) * total)

  # Ajustar para garantir que soma seja exatamente igual ao total
  diferenca <- total - sum(proporcoes)
  if(diferenca != 0) {
    idx_max <- which.max(proporcoes)
    proporcoes[idx_max] <- proporcoes[idx_max] + diferenca
  }

  # Criar sequência de categorias
  categorias <- rep(nomes, times = proporcoes)

  # Criar grid de coordenadas
  n_colunas <- ceiling(total / n_linhas)

  dados_waffle <- expand.grid(
    y = 1:n_linhas,
    x = 1:n_colunas
  ) %>%
    mutate(
      posicao = row_number(),
      categoria = c(categorias, rep(NA, n() - length(categorias)))[1:n()]
    ) %>%
    filter(!is.na(categoria))
```

```r
  return(dados_waffle)
}

plotar_waffle <- function(
  dados_waffle,
  titulo = "",
  cores = NULL,
  palette = NULL) {

  p <- ggplot(
    dados_waffle,
    aes(x = x, y = y, fill = categoria)
    ) +
    geom_tile(
      color = "white",
      size = 0.5,
      width = 0.95,
      height = 0.95
    ) +
    coord_equal() +
    scale_y_reverse() +
    theme_void() +
    theme(
      legend.position = "bottom",
      legend.title = element_blank(),
      plot.title = element_text(face = "bold", hjust = 0.5, size = 14),
      plot.margin = margin(10, 10, 10, 10)
    ) +
    labs(title = titulo)

  # Adicionar cores personalizadas se fornecidas
  if(!is.null(cores)) {
    p <- p + scale_fill_manual(values = cores)
  }
  if(!is.null(palette)) {
    p <- p + scale_fill_brewer(palette = palette)
  }

  return(p)
}

criar_waffle_multiplo <- function(
  data,
  var,
  grupo_var = NULL,
  titulo_geral = "",
  n_linhas = 10,
  total = 100,
  cores = NULL,
  palette = NULL,
```

```r
ncol_facet = NULL) {

# Sem variável de grupo - apenas um waffle
if(is.null(grupo_var)) {
  dados_waffle <- criar_waffle_data(data, var, n_linhas, total)
  p <- plotar_waffle(dados_waffle, titulo_geral, cores, palette)
  return(p)
}

# Com variável de grupo - múltiplos waffles (facetas)
todos_dados <- data %>%
  group_by(.data[[grupo_var]]) %>%
  group_modify(~ {
    criar_waffle_data(.x, var, n_linhas, total)
  }) %>%
  ungroup() %>%
  rename(painel = all_of(grupo_var))

# número de colunas para facet
n_paineis <- n_distinct(todos_dados$painel)
if(is.null(ncol_facet)) {
  ncol_facet <- ceiling(sqrt(n_paineis))
}

p <- ggplot(todos_dados, aes(x = x, y = y, fill = categoria)) +
  geom_tile(
    color = "white",
    size = 0.3,
    width = 0.9,
    height = 0.9
  ) +
  facet_wrap(~painel, ncol = ncol_facet) +
  coord_equal() +
  scale_y_reverse() +
  theme_void() +
  theme(
    legend.position = "bottom",
    strip.text = element_text(face = "bold", size = 11),
    plot.title = element_text(face = "bold", hjust = 0.5, size = 14, vjust = 1.5),
    plot.margin = margin(10, 10, 10, 10),
    panel.spacing = unit(1, "lines")
  ) +
  labs(title = titulo_geral, fill = "")

if(!is.null(cores)) {
  p <- p + scale_fill_manual(values = cores)
}
if(!is.null(palette)) {
  p <- p + scale_fill_brewer(palette = palette)
}
```

```
  return(p)
}

criar_waffle_multiplo(
  data = bvd_test,
  var = "season",
  titulo_geral = "Estação de parição das vacas.",
  palette = "Dark2",
  n_linhas = 10, total = 10 * 15
)
```

Figura 3.48: Exemplo de gráfico de waffle

O gráfico de pontos de Cleveland é um tipo de visualização de dados que utiliza pontos para representar valores de diferentes categorias, servindo como uma alternativa ao gráfico de barras. Ele exibe um único ponto para cada categoria no ponto correspondente à sua frequência, facilitando a comparação de valores e a identificação de padrões. Por vezes, ele é ligado ao eixo das abscissas (ou ordenadas para gráficos na horizontal) por uma linha, de modo que a altura represente também a frequência. Outra utilização interessante para esse gráfico seria para mostrar variações das frequências (ou outro valor) em relação à outro valor de referência, como no segundo exemplo abaixo.

```
bvd_test %>%
  count(breed) %>%
  ggplot(aes(x = n, y = reorder(breed, n))) +
  geom_segment(aes(yend = breed), xend = 0, colour = "grey50") +
  geom_point(size = 3, color = "blue") +
  theme_bw() +
  theme(
```

```
    panel.grid.major.y = element_blank(),
    axis.text = element_text(size = 12),
    axis.title = element_text(size = 14),
  ) +
  labs(
    title = "Vacas avaliadas segundo a raça",
    x = "Frequência absoluta",
    y = "Raça",
  )
```

Figura 3.49: Exemplos de gráfico de Cleveland: Contagem de uma variável qualitativa

```
exemplo_df <- tibble(
  date = seq(ymd("2025-01-01"), by="day", length.out = 365),
  ocorrencias = round(runif(365, 0, 100))
) %>%
mutate(
  mes = month(date, label = TRUE, abbr = FALSE)
) %>%
group_by(mes) %>%
summarise(
  total = sum(ocorrencias)
)

valor_referencia <- round(mean(exemplo_df$total))

ggplot(exemplo_df, aes(x=mes, y=total)) +
  geom_hline(yintercept = valor_referencia, color = "red", linewidth = 1) +
```

```r
geom_segment(
  aes(x=mes, xend=mes, y=valor_referencia, yend=total),
  color="grey"
) +
geom_point(color="steelblue", size=4) +
theme_minimal() +
theme(
  panel.grid.major.x = element_blank(),
  panel.border = element_blank(),
  axis.ticks.x = element_blank(),
  axis.text = element_text(size = 12),
  axis.title = element_text(size = 14),
) +
labs(
  x = element_blank(),
  y = "Frequência mensal em relação a média anual"
) +
annotate(
  "text",
  label = str_c("Média anual: ", valor_referencia),
  x = 5,
  y = valor_referencia,
  vjust = -1,
  size = 7,
  color = "black"
)
```

Figura 3.50: Exemplos de gráfico de Cleveland: Distribuição das contagens em relação à média

O treemap representa dados hierárquicos através de retângulos aninhados, em que o tamanho do retângulo é proporcional a uma variável quantitativa (frequências, por exemplo), a cor representa categorias ou outra variável quantitativa e a hierarquia é observada por retângulos dentro de retângulos, demontrando os níveis de agrupamento.

```
instala_se_nao_existe("treemapify")
```

```
NULL
```

```
library(treemapify)

bvd_test %>%
  count(breed) %>%
  ggplot(aes(area = n, fill = breed, label = breed)) +
  geom_treemap() +
  geom_treemap_text(
    colour = "white", place = "centre", grow = FALSE
  ) +
  labs(
    title = "Distribuição de vacas por raça",
    subtitle = "Tamanho proporcional à frequência relativa de cada raça na amostragem"
  ) +
  theme(legend.position = "none")
```

Figura 3.51: Exemplos de gráfico treemap

Isso pode ser particularmente interessante em apresentações, quando queremos apresentar contagens de variáveis em múltiplos níveis (considerando várias variáveis), ou quando temos muitas categorias e queremos deixar evidente a importância de determinada categoria em relação às demais.

```r
instala_se_nao_existe("treemap")
```

```
NULL
```

```r
instala_se_nao_existe("plotly")
```

```
NULL
```

```r
treemap_to_plotly <- function(
  data,
  index_cols,
  size_col,
  title = "") {
  tm <- treemap::treemap(
    data,
    index = index_cols,
    vSize = size_col,
    type = "index",
    draw = FALSE
  )

  tm_data <- tm$tm %>%
    as_tibble() %>%
    mutate(
      # labels hierárquicos para o plotly
      label = case_when(
        level == 1 ~ as.character(get(index_cols[1])),
        level == 2 ~ as.character(get(index_cols[2])),
        level == 3 ~ as.character(get(index_cols[3])),
        TRUE ~ ""
      ),
      parent = case_when(
        level == 1 ~ "",
        level == 2 ~ as.character(get(index_cols[1])),
        level == 3 ~ paste(get(index_cols[1]), get(index_cols[2]), sep = " - "),
        TRUE ~ ""
      ),
      ids = case_when(
        level == 1 ~ as.character(get(index_cols[1])),
        level == 2 ~ paste(get(index_cols[1]), get(index_cols[2]), sep = " - "),
        level == 3 ~ paste(
          get(index_cols[1]),
          get(index_cols[2]),
          get(index_cols[3]),
          sep = " - "
        ),
        TRUE ~ ""
      )
    )

  plotly::plot_ly(
```

```
    type = "treemap",
    labels = tm_data$label,
    parents = tm_data$parent,
    values = tm_data$vSize,
    ids = tm_data$ids,
    marker = list(
      colors = tm_data$color,
      line = list(width = 2, color = "white")
    ),
    textinfo = "label+value+percent parent",
    hovertemplate = paste(
      '<b>%{label}</b>',
      '<br>Valor: %{value}',
      '<br>%{percentParent:.1%} do grupo pai',
      '<extra></extra>'
    ),
    domain = list(x = c(0, 1), y = c(0, 1))
  ) %>%
  plotly::layout(
    title = title,
    margin = list(l = 0, r = 0, t = 30, b = 0)
  )
}
```

```
bvd_test %>%
  count(spec, breed, season) %>%
  treemap_to_plotly(
    index_cols = c("spec", "breed", "season"),
    size_col = "n",
    title = "Treemap Interativo - Wrapper Function"
  )
```

Por fim temos o empacotamento circular ou treemap circular, que assim como treemap permite visualizar uma organização hierárquica. Nele, cada nó da árvore é representado como um círculo e seus subnós são representados como círculos dentro dele. O tamanho de cada círculo pode ser proporcional a um valor específico (como a frequência da distribuição).

```
instala_se_nao_existe("packcircles")
```

```
NULL
```

```
data <- bvd_test %>%
  count(season)

# obtém as coordenadas do centro de cada categoria
# e o raio proporcional ao valor
packing <- packcircles::circleProgressiveLayout(
  data$n,
  sizetype='area'
)
data <- cbind(data, packing)
# obtém os vertices do circulo
```

```r
dat.gg <- packcircles::circleLayoutVertices(packing, npoints=50)

ggplot() +
  geom_polygon(
    data = dat.gg,
    aes(x, y, group = id, fill=as.factor(id)),
    colour = "black",
    alpha = 0.6
  ) +
  geom_text(data = data, aes(x, y, size=n, label = season)) +
  scale_size_continuous(range = c(5,10)) +
  theme_void() +
  theme(legend.position="none") +
  labs(title = "Estação de parto das vacas") +
  coord_equal()
```

Figura 3.52: Exemplo de gráfico de empacotamento circular

3.2.6 Representação gráfica de variáveis quantitativas (Univariada)

3.2.6.1 Gráfico de linhas e pontos

No gráficos de linhas e pontos, os pontos no gráfico, posicionados por pares de valores indicando a abcissa (X) e ordenadas (Y), são ligados por segmentos de reta. Apesar de sua construção depender de dois valores, o que indicaria a necessidade de duas variáveis, se uma delas varia em iontervalos regulares (por exemplo, meses do ano), então, estamos criando um gráfico para descrever o comportamento de uma única variável quantitativa ao longo desse intervalo regular.

```r
air_quality <- read_xlsx(
  "../datasets/excel/AirQualityUCI.xlsx"
)

air_quality <- air_quality %>%
  mutate(
    year = year(Date),
    month = month(Date)
  ) %>%
  group_by(year, month) %>%
  summarise(
    temperatura = mean(`T`, na.rm = TRUE)
  ) %>%
  mutate(
    data = ymd(str_c(year, month, 1, sep="-"))
  )

ggplot(air_quality, aes(x = data, y = temperatura)) +
  geom_line(color = "grey") +
  geom_point(shape=21, color="black", fill="#6b93dfb6", size=6) +
  plot_theme +
  labs(
    x = NULL,
    y = "temperatura (°C)",
    title = "Variação de temperatura ao longo do estudo"
  )
```

Figura 3.53: Exemplo de gráfico de empacotamento circular

3.2.6.2 Histograma

O histograma é uma representação gráfica da distribuição de frequências de uma variável contínua (ou discreta com muitos valores). Ele divide o intervalo de valores em classes (*bins*) e mostra quantas observações caem em cada classe através de barras adjacentes. Normalmente, construímos um histograma para visualizar a forma, centro e dispersão da distribuição dos dados.

Para construir um histograma precisamos calcular os seguintes elementos:

- **Classes (bins)**: Intervalos que irão agrupar nossa variável quantitativa (serão representadas como a largura de cada barra)
- **Frequência**: Contagem de observações em cada classe (serão representadas como a altura de cada barra)

O número de classes e largura do intervalo de cada classe, quando adotamos um intervalo igual para todas as classes, pode ser calculado da mesma forma que foi explicado na tabela de frequências de variáveis contínuas. Já o valor em cada classe pode ser calculado como a frequência absoluta (F_i, o número de observações na classe i), frequência relativa (F_i/n, em que n é o número de observações) ou como densidade ($F_i/(n \times h_i)$, em que n é o número de observações e h_i é a amplitude da classe). O interessante de usar a densidade no histograma é que obtemos um estimador não paramétrico da fdp (função densidade de probabilidade) da distribuição, considerando que $F_i/n \to P(X \in [a, b))$ quando $n \to \infty$.

```r
tema <- theme_minimal() +
  theme(
    plot.title = element_text(size = 16),
    axis.title = element_text(size = 14),
    axis.text = element_text(size = 12)
  )

breaks <- cria_classes_continua(ap2$w_age_t, "sqrt")[["breaks"]]

ggplot(ap2) +
  geom_histogram(
    aes(x = w_age_t),
    breaks = breaks,
    color = "black",
    fill = "steelblue"
  ) +
  scale_x_continuous(
    breaks = round(breaks, 1)
  ) +
  tema +
  labs(
    title = expression("Histograma de" ~ F[i] ~ "usando" ~ k == sqrt(n)),
    x = "Peso do porco à transferência do desmame para a unidade de terminação (kg)",
    y = expression("Frequência absoluta (" ~ F[i] ~ "%)")
  )

breaks <- cria_classes_continua(ap2$w_age_t, "sturges")[["breaks"]]

ggplot(ap2) +
  geom_histogram(
    aes(x = w_age_t),
```

```
    breaks = breaks,
    color = "black",
    fill = "steelblue"
  ) +
  scale_x_continuous(
    breaks = round(breaks, 2)
  ) +
  tema +
  labs(
    title = expression("Histograma de" ~ F[i] ~ "usando" ~ k == 1 + 1.33 %*% log[10](n) ~ "(Equação de Stu
    x = "Peso do porco à transferência do desmame para a unidade de terminação (kg)",
    y = expression("Frequência absoluta (" ~ F[i] ~ "%)")
  )

ggplot(ap2) +
  geom_histogram(
    aes(x = w_age_t, y = after_stat(count / sum(count))),
    breaks = breaks,
    color = "black",
    fill = "steelblue"
  ) +
  scale_x_continuous(
    breaks = round(breaks, 2)
  ) +
  scale_y_continuous(
    labels = scales::label_percent(),
    limits = c(.0, .2),
    breaks = scales::breaks_width(width = .025),
  ) +
  tema +
  labs(
    title = expression("Histograma de" ~ F[ri] ~ "usando" ~ k == 1 + 1.33 %*% log[10](n) ~ "(Equação de St
    x = "Peso do porco à transferência do desmame para a unidade de terminação (kg)",
    y = expression("Frequência relativa (" ~ F[ri] ~ "%)")
  )

ggplot(ap2) +
  geom_histogram(
    aes(x = w_age_t, y = after_stat(density)),
    breaks = breaks,
    color = "black",
    fill = "steelblue"
  ) +
  scale_x_continuous(
    breaks = round(breaks, 2)
  ) +
  tema +
  labs(
    title = expression("Histograma de" ~ F[i] / n %*% h[i] ~ "usando" ~ k == 1 + 1.33 %*% log[10](n) ~ "(E
    x = "Peso do porco à transferência do desmame para a unidade de terminação (kg)",
```

```
    y = expression("Densidade (" ~ F[i] / n %*% h[i] ~ ")")
  )
```

Gráficos de histograma também costumam ser associados a estimativas Kernel de densidade (KDE), um método de suavização não paramétrico da fdp de variáveis aleatórias contínuas a partir de uma amostra de dados. Entretanto, como ainda não introduzimos os conceitos básicos de probabilidade e distribuições de probabilidade, deixarei para explicar melhor a KDE após esses assuntos.

3.2.6.3 Boxplot ou diagrama de caixa

O boxplot (diagrama de caixa) é uma representação gráfica que resume a distribuição de uma variável quantitativa através de cinco estatísticas de ordem, permitindo visualizar simultaneamente medidas de posição, dispersão e identificação de valores atípicos.

Seja $X_1, X_2, ..., X_n$ uma amostra aleatória. As estatísticas de ordem $X_{(1)} \leq X_{(2)} \leq ... \leq X_{(n)}$ são os valores amostrais ordenados. Para essa amostra ordenada, definimos:

- **Primeiro Quartil** (Q_1): $Q_1 = X_{(\lceil 0.25n \rceil)}$
- **Mediana** (Q_2 ou $\tilde{x}$):

$$\tilde{x} = \begin{cases} X_{(\frac{n+1}{2})} & \text{se } n \text{ é ímpar} \\ \frac{X_{(\frac{n}{2})} + X_{(\frac{n}{2}+1)}}{2} & \text{se } n \text{ é par} \end{cases}$$

- **Terceiro Quartil** (Q_3): $Q_3 = X_{(\lceil 0.75n \rceil)}$
- **Amplitude Interquartílica** (IQR): $\text{IQR} = Q_3 - Q_1$. O IQR representa a dispersão dos 50% dados centrais, sendo uma medida robusta de variabilidade.

Para a construção do boxplot usamos os seguintes elementos:

1. **Caixa**: Retângulo delimitado por Q_1 e Q_3
2. **Linha mediana**: Segmento em Q_2 dentro da caixa
3. **Whiskers** (bigodes): Estendem-se até os valores extremos não-outliers
4. **Outliers**: Pontos além dos whiskers

Existem várias formas de definir o limite dos whiskers, mas, em geral usamos o critério de Tukey:

- **Limite Inferior**: $L_I = Q_1 - 1.5 \times \text{IQR}$
- **Limite Superior**: $L_S = Q_3 + 1.5 \times \text{IQR}$

Calculados os limites inferior e superior, o whisker inferior será $\min\{x_i : x_i \geq L_I\}$, ou seja, o menor valor entre o mínimo dos dados e o L_I, enquanto o whisker superior será $\max\{x_i : x_i \leq L_S\}$, o maior valor entre o máximo dos dados e o L_S. Valores superiores ao whisker superior ou inferiores ao whisker inferior serão considerados *outliers* e representados como pontos no gráfico.

```
ggplot(ap2, aes(y = w_age_t, x = 1)) +
  geom_boxplot(
    color = "black",
    fill = "steelblue",
    outlier.size = 4,
    outlier.shape = 23,
    outlier.color = "black",
    outlier.fill = "red",
  ) +
```

(a) Histograma de frequência absoluta com k obtido pela sqrt(n)

(a) Histograma de frequência absoluta com k obtido pela equação de Sturges

```
  geom_jitter(
    color="black",
    size=0.6,
    alpha=0.9,
    width = .1,
    height = 0
  ) +
  scale_x_continuous(
    breaks = NULL
  ) +
  scale_y_continuous(
    breaks = seq(0, 50, 5)
  ) +
  tema +
  labs(
    title = "Boxplot do peso dos porcos à transferência do desmame para a unidade de terminação",
    x = NULL,
    y = "Peso do porco à transferência do desmame para a unidade de terminação (kg)"
  )
```

Figura 3.58: Exemplo de gráfico boxplot

Com algumas manipulações podemos combinar os gráficos de histograma, densidade estimada e boxplot, para uma visualização ainda mais completa do comportamento dos nossos dados.

```
instala_se_nao_existe("patchwork")
```

```
NULL
```

```r
require(patchwork)

criar_hist_boxplot <- function(
  data,
  variavel,
  x_caption = NULL,
  title = NULL,
  metodo_agrupamento = "sturges",
  color_hist = "black",
  fill_hist = "steelblue",
  color_box = "black",
  fill_box = "lightgray") {

  valores <- data[[variavel]]

  breaks <- cria_classes_continua(
    valores,
    metodo_agrupamento
  )[["breaks"]]

  # Calcular limites comuns para os eixos x
  x_limits <- range(valores, na.rm = TRUE)
  x_expand <- diff(x_limits) * 0.05  # 5% de margem
  x_limits <- x_limits + c(-x_expand, x_expand)

  escala_x <- scale_x_continuous(
    limits = x_limits,
    expand = c(0, 0),
    breaks = round(breaks, 2)
  )

  # Histograma principal
  p_hist <- ggplot(
    data,
    aes_string(x = variavel)
    ) +
    geom_histogram(
      aes(y = after_stat(density)),
      breaks = breaks,
      fill = fill_hist,
      color = color_hist,
      alpha = 0.8
    ) +
    geom_density(
      color = "darkred",
      linewidth = 1,
      alpha = 0.8
    ) +
    escala_x +
    labs(
```

```
      title = title,
      x = NULL,
      y = expression("Densidade (" ~ F[i] / n %*% h[i] ~ ")")
    ) +
    theme_minimal() +
    theme(
      axis.text.x = element_blank(),
      axis.ticks.x = element_blank(),
      plot.margin = margin(5, 5, 0, 5)
    )

  # Boxplot marginal
  p_box <- ggplot(data, aes_string(x = variavel)) +
    geom_boxplot(
      aes(y = 0),
      width = 0.25,
      fill = fill_box,
      color = color_box,
      outlier.color = "red",
      outlier.shape = 21,
      outlier.fill = "red",
      outlier.alpha = 0.5
    ) +
    escala_x +
    coord_cartesian(
      ylim = c(-0.15, 0.15)
    ) +
    labs(
      x = x_caption,
      y = NULL
    ) +
    theme_minimal() +
    theme(
      axis.text.y = element_blank(),
      axis.ticks.y = element_blank(),
      panel.grid = element_blank(),
      plot.margin = margin(0, 5, 5, 5)
    )

  plot <- p_hist / p_box + plot_layout(heights = c(4, 1))

  return(plot)
}

criar_hist_boxplot(
  ap2,
  "w_age_t6",
  x_caption = "Peso à 6ª semana na terminação",
  title = "Distribuição do peso dos porcos após 6 semanas na terminação"
)
```

Figura 3.59: Exemplo de gráfico composto histograma + boxplot

Uma propriedade importante do boxplot é que ele usa a mediana como ponto central, uma medida mais robusta que a média para dados com outliers muito extremos. Além disso, em dados com distribuição normal $X \sim N(\mu, \sigma^2) : P(L_I \leq X \leq L_S) \approx 0.9930$, ou seja, a proporção esperada de outliers nos dados é de aproximadamente 0,7%.

Além disso, o boxplot permite avaliar assimetrias na distribuição em relação à mediana (pelo tamanho das caixas):

- **Simétrica**: $Q_2 - Q_1 \approx Q_3 - Q_2$, caixas de mesmo tamanho;
- **Assimétrica à direita**: $Q_3 - Q_2 > Q_2 - Q_1$, caixa superior maior;
- **Assimétrica à esquerda**: $Q_2 - Q_1 > Q_3 - Q_2$, caixa inferior maior.

Alguns aspectos que devem ser levados em consideração na análise e uso do boxplot são o fato dele não demonstrar possíveis multimodalidades (mais de uma moda), ser instável para amostras pequenas (< 20) e o fato de que o critério de Tukey ($1.5 \times$ IQR) nem sempre será adequado para definir o que é um outlier (é sempre importante avaliar cada um desses dados considerados outliers de forma individual e seu real impacto no teste ou modelo que será utilizado posteriormente).

Uma alternativa ao boxplot é o gráfico de violino. Esse gráfico nada mais é que um gráfico de densidade espelhado, obtido pela KDE, que dá essa forma parecida com um violino. Como ainda não apresentamos formalmente os conceitos da KDE, nesse momento irei apresentar somente a maneira de obter esse gráfico.

```
ggplot(ap2, aes(y = w_age_t, x = 0)) +
  geom_violin(
    scale = "width",
    trim = FALSE,
    linewidth = 1,
    fill = "#FF7F50",
```

```r
    alpha = .2
  ) +
  stat_bin(
    aes(x = after_stat(density/max(density) * 0.4)),
    geom = "bar",
    alpha = 0.7,
    bins = 11,
    orientation = "y",
    color = "#DC143C",
    fill = "#DC143C",
    linewidth = 1
  ) +
  geom_boxplot(
    color = "#00008B",
    fill = "steelblue",
    outlier.size = 5,
    outlier.shape = 23,
    outlier.color = "orange",
    outlier.fill = "orange",
    linewidth = 1.1,
    width = .2
  ) +
  scale_x_continuous(
    breaks = NULL,
    limits = c(-0.5, 0.5)
  ) +
  scale_y_continuous(
    breaks = seq(0, 50, 5)
  ) +
  tema +
  labs(
    title = "Densidade estimada e boxplot do peso dos porcos à transferência do desmame para a unidade de t
    x = NULL,
    y = "Peso do porco à transferência do desmame para a unidade de terminação (kg)"
  )
```

Figura 3.60: Exemplo de gráfico violino + boxplot + histograma

3.3 Medidas-resumo

Como o próprio nome indica, medidas-resumo tentam representar o comportamento de uma variável em estudo por meio de um valor. Com exceção da moda (valor mais frequente de uma variável), as medidas-resumo são calculadas normalmente para variáveis quantitativas.

3.4 Medidas de posição ou localização

Medidas de posição indicam a localização dos dados em relação ao intervalos de valores que a variável em estudo assume. Elas podem ser agrupadas em medidas de tendência central (média, mediana e moda) e medidas separatrizes (quartis, decis e percentis).

3.4.0.1 Média

A média aritmética ou média simples representa o valor central no intervalo de valores assumidos pela variável. Em uma população de tamanho N representamos a média simples como μ, enquanto, em uma amostra de tamanho n, indicamos a média amostral como $\bar{X}$.

A média aritmética de uma variável é a soma de todos os valores registrados no estudo dividido pelo total de observações. Ou seja,

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

em que x_i é o valor da observação i, com $i = 1, 2, ..., n$ e n é o número de observações.

Pensando em dados de uma população de uma variável aleatória X com função probabilidade (variável discreta) ou função densidade de probabilidade (variável contínua), μ é definido como

- Variável aleatória discreta: $\mu = E[X] = \sum_x x \cdot P(X = x)$;
- Variável aleatória contínua: $\mu = E[X] = \int_{-\infty}^{\infty} x \cdot f(x)dx$.

A média simples é uma medida de posição altamente utilizada na estatística por apresentar uma série de propriedades interessantes, que serão apresentadas a seguir:

1 - **Linearidade**

A esperança (média populacional) preserva operações lineares, ou seja, $E[aX + b] = aE[X] + b$. Se você multiplicar uma variável por uma constante ou somar uma constante, a média será transformada da mesma forma.

Exemplo para uma variável discreta:

$E[aX+b] = \sum_x (ax+b) \cdot P(X = x) = \sum_x ax \cdot P(X = x) + \sum_x b \cdot P(X = x) = a \sum_x x \cdot P(X = x) + b \sum_x P(X = x) = aE[X] + b \cdot 1 = aE[X] + b = a\mu + b$

Essa propriedade facilita cálculos quando alteramos a escala da variável (em uma transformação linear) e permite a padronização de variáveis para o valor Z, definido como $Z = \frac{X-\mu}{\sigma}$, que tem $E[Z] = 0$.

2 - **Minimização de erro quadrático**:

A média é o valor que minimiza a soma dos quadrados dos desvios ($\bar{X} = \arg\min_c \sum_{i=1}^n (X_i - c)^2$). Se você imaginar os dados como pontos em uma reta, a média é o ponto que representa a menor soma das distâncias (as quadrado) entre os dados e esse ponto.

Demonstrando essa propriedade, buscamos um valor c que minimize $g(c) = \sum_{i=1}^n (X_i - c)^2$, ou seja, que apresente o menor desvio quadrático. Desenvolvendo essa equação:

Passo 1: Expandir o quadrado

$g(c) = \sum_{i=1}^n (X_i^2 - 2X_i c + c^2)$

$= \sum_{i=1}^n X_i^2 - 2c \sum_{i=1}^n X_i + nc^2$

Passo 2: Derivar em relação a c

Para obter essa derivada vamos aplicar algumas regras de derivadas

1. **Derivada de uma constante**: Se k é constante, $\frac{d}{dc}(k) = 0$
2. **Regra da potência**: $\frac{d}{dc}(c^n) = nc^{n-1}$
3. **Regra da constante multiplicativa**: $\frac{d}{dc}(k \cdot c) = k$
4. **Linearidade da derivada**: $\frac{d}{dc}(f + g) = \frac{df}{dc} + \frac{dg}{dc}$

nosso $g(c)$ possui 3 termos,

$$g(c) = \underbrace{\sum_{i=1}^n X_i^2}_{\text{Termo 1}} - \underbrace{2c \sum_{i=1}^n X_i}_{\text{Termo 2}} + \underbrace{nc^2}_{\text{Termo 3}}$$

então, para o Termo 1, $\sum_{i=1}^n X_i^2$, como ele não depende de c, o tratamos como uma constante, logo, $\frac{d}{dc}\left(\sum_{i=1}^n X_i^2\right) = 0$. Para o Termo 2, $-2c \sum_{i=1}^n X_i$, podemos reescrevê-lo como $-2\left(\sum_{i=1}^n X_i\right) \cdot c$, então $\sum_{i=1}^n X_i$ novamente é uma constante em relação ao c (que trataremos como $K = \sum_{i=1}^n X_i$), portanto, buscamos a derivada em relação ao c de $-2Kc$, assim, usando a regra da constante multiplicativa, $\frac{d}{dc}(-2Kc) = -2K = -2 \sum_{i=1}^n X_i$. Para o Termo 3, nc^2, n é uma constante e aplicamos a regra da potência, logo, $\frac{d}{dc}(nc^2) = n \cdot 2c^{2-1} = 2nc$. Combinando todos esses termos (linearidade da deriva ou regra da soma), obtemos

$\frac{dg}{dc} = -2\sum_{i=1}^{n} X_i + 2nc$

Passo 3: Igualar a zero e resolver

$-2\sum_{i=1}^{n} X_i + 2nc = 0$

$nc = \sum_{i=1}^{n} X_i$

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

3 - **Propriedade dos desvios**

A soma dos desvios em relação à média é sempre zero ($\sum_{i=1}^{n}(X_i - \bar{X}) = 0$), logo, os desvios positivos e negativos se cancelam. Assim, a média aritmética é o centro do intervalo dos valores da variável. A consequência dessa propriedade ocorre no momento de calcularmos a variabilidade dos dados (o quanto variam em relação a média). Como os desvios se cancelam, a estratégia adotada para conseguir calcular medidas de dispersão seria somar os desvios absolutos ($\sum_{i=1}^{n} |X_i - \bar{X}|$) ou os desvios quadráticos ($(\sum_{i=1}^{n}(X_i - \bar{X})^2)$). A segunda opção acaba sendo a mais adotada, pela sua relação com os momentos (que aprenderemos em treinamentos futuros) e outras características que serão explicadas no tópico das medidas de dispersão.

4 - **Teorema do Limite Central**

A distribuição da média amostral se aproxima de uma normal quando n aumenta. Se $X_1, ..., X_n$ são i.i.d. com $E[X_i] = \mu$ e $\text{Var}(X_i) = \sigma^2 < \infty$, então, $\frac{\bar{X}-\mu}{\sigma/\sqrt{n}} \xrightarrow{d} N(0,1)$ quando $n \to \infty$, ou, de forma equivalente, $\bar{X} \xrightarrow{d} N\left(\mu, \frac{\sigma^2}{n}\right)$.

5 - **Média aritmética como um estimador**

A média simples é um estimador não enviesado da média populacional ($E[\bar{X}] = \mu$). Ou seja, se repetimos um experimento aleatório várias vezes e para cada tentativa registramos a média amostral, conforme o número de tentativas aumenta, a média das médias amostrais de cada experimento se aproxima cada vez mais da média populacional.

$E[\bar{X}] = E\left[\frac{1}{n}\sum_{i=1}^{n} X_i\right]$

$= \frac{1}{n}\sum_{i=1}^{n} E[X_i]$

$= \frac{1}{n}\sum_{i=1}^{n} \mu = \frac{n\mu}{n} = \mu$

A média é também um estimador consistente ($\bar{X} \xrightarrow{p} \mu$), ou seja, conforme o n amostral aumenta, a probabilidade de $\bar{X}$ estar próximo de μ tende a 1 ($P(|\bar{X} - \mu| \leq \epsilon) \to 1$ quando $n \to \infty$).

Além disso, entre todos os estimadores não-viesados que são combinações lineares dos dados, a média tem a menor variância.

$\text{Var}(\bar{X}) = \text{Var}\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right)$

$= \frac{1}{n^2}\text{Var}\left(\sum_{i=1}^{n} X_i\right)$

$= \frac{1}{n^2}\sum_{i=1}^{n} \text{Var}(X_i)$ (em condição de i.i.d.)

$= \frac{1}{n^2} \cdot n\sigma^2 = \frac{\sigma^2}{n}$

Ou seja, $\text{Var}(\hat{\mu}) \geq \text{Var}(\bar{X}) = \sigma^2/n$.

Na média simples, todas as observações possui o mesmo peso ou importância no seu cálculo, porém, quando queremos atribuir diferentes pesos (w_i) para cada observação, podemos calcular a média ponderada como

$\bar{X}_w = \frac{\sum_{i=1}^{n} w_i X_i}{\sum_{i=1}^{n} w_i}$, onde $w_i > 0$, ou $\bar{X}_w = \sum_{i=1}^{n} w_{r_i} X_i$, onde w_{r_i} é o peso relativo ($\frac{w_i}{\sum_{i=1}^{n} w_i}$).

A média ponderada é particularmente útil quando precisamos obter a média amostral, mas nosso conjunto de dados foi registrado de uma forma em que uma váriavel apresenta os valores de interesse e outra apresenta o número de vezes em que foi observado aquele determinado valor.

No R calculamos a média simples com a função `mean()`. Um aspecto importante a se considerar quando usando essa função (e qualquer outra função do R que realiza operações matemáticas) é que valores ausentes (`NA`) contaminam seus resultados, ou seja, se houver uma observação `NA` na sua variável, o resultado da operação por padrão também é `NA`. Para contornar esse problema, usamos o argumento `na.rm = TRUE`, assim, esses valores são removidos antes de ser calculada a média.

```
ap2 %>%
  summarise(
    across(starts_with("w_age"), ~ mean(.x, na.rm = TRUE), .names = "mean_{.col}")
  ) %>%
  pivot_longer(
    cols = everything(),
    names_to = "Variável",
    names_prefix = "mean_",
    values_to = "Média",
  ) %>%
  kable()
```

Variável	Média
w_age_t	27.42792
w_age_t6	69.97244

Já a média ponderada pode ser calculada com `weighted.mean()`.

```
dataset_alunos <- tibble(
  n_alunos = round(runif(10, 1, 10)),
  notas = round(runif(10, 0, 100), 2),
)

m1 <- dataset_alunos %>%
  summarise(
    media = round(weighted.mean(notas, n_alunos), 3)
  ) %>% pull(media)

m2 <- round(sum(dataset_alunos$notas * dataset_alunos$n_alunos) / sum(dataset_alunos$n_alunos), 3)
m3 <- round(sum(dataset_alunos$notas * (dataset_alunos$n_alunos / sum(dataset_alunos$n_alunos))), 3)

m1 == m2 & m2 == m3
```

```
[1] TRUE
```

Para dados agrupados, no caso de variáveis discretas calculamos a média usando a média ponderada, em que cada peso representará a frequência observada daquele valor. Já para variáveis contínuas agrupadas, cada classe não representa um valor, mas sim um intervalo, então estimamos a média usando o ponto médio de cada classe (cada intervalo) como o o valor observado e a frequência de observações daquela classe como o peso. Ou seja,

$\bar{X} = \frac{\sum_{i=1}^{k} x_i \times F_i}{\sum_{i=1}^{k} F_i} = \frac{\sum_{i=1}^{k} x_i \times F_i}{n}$, em que k é o número de classes, x_i é o ponto médio do intervalo da classe i e F_i é a frequência observada na classe i.

Um aspecto importante da média amostral é que ela é "puxada" na direção das observações extremas, ou seja, valores muito discrepantes em relação aos demais. Para tentar diluir esse efeito de *outliers* da amostra na hora de avaliar a posição central dos dados, podemos calcular a média aparada ou a média winsorizada.

A média aparada remove $\alpha\%$ das observações extremas (de forma igual nos limites inferior e superior) de uma variável ordenada antes de calcular a média. Ela é expressa como:

$\bar{X}_\alpha = \frac{1}{n-2k} \sum_{i=k+1}^{n-k} X_{(i)}$, em que α é a proporção a ser removida e k é o número efetivo de observações que serão removidas do cálculo da média aparada ($n \times \alpha/2$).

No R a média aparada também é obtida pela função `mean()` usando o argumento `trim = valor`, sendo que o valor passado ao `trim` corresponde ao $\alpha/2$, ou seja, a proporção a ser removida em cada extremo.

```
ap2 %>%
  summarise(
    across(starts_with("w_age"), ~ mean(.x, trim = 0.025, na.rm = TRUE), .names = "mean_{.col}")
  ) %>%
  pivot_longer(
    cols = everything(),
    names_to = "Variável",
    names_prefix = "mean_",
    values_to = "Média aparada (em 5%)",
  ) %>%
  kable()
```

Variável	Média aparada (em 5%)
w_age_t	27.36660
w_age_t6	69.94245

A média winsorizada, por sua vez, substitui os valores extremos pelos percentis correspondentes. Por exemplo, se queremos uma média winsorizada de $\alpha = 10\%$, iremos substituir todos os valores abaixo do percentil 10% pelo valor desse percentil e subtituir todos valores acima do percentil 90% pelo valor desse percentil.

No R não temos uma função, a não ser em pacotes específicos, da média winsorizada, porém podemos facilmente criar nossa própria função para calculá-la:

```
media_winsorizada <- function(x, p = 0.1) {
  # Remover NAs
  x <- x[!is.na(x)]

  x_sorted <- sort(x) # ordena a variavel
  n <- length(x)

  # obter o indice para os percentis
  k <- floor(n * p)

  # Se k = 0, então p = 0, retornar média comum
  if(k == 0) return(mean(x))
```

```r
  x_wins <- x

  # Valores dos percentis para substituição
  lower_val <- x_sorted[k + 1]
  upper_val <- x_sorted[n - k]

  # substitui valores extremos pelos percentis
  x_wins[x_wins < lower_val] <- lower_val
  x_wins[x_wins > upper_val] <- upper_val

  return(mean(x_wins))
}

ap2 %>%
  summarise(
    across(starts_with("w_age"), ~ media_winsorizada(.x, p = 0.025), .names = "mean_{.col}")
  ) %>%
  pivot_longer(
    cols = everything(),
    names_to = "Variável",
    names_prefix = "mean_",
    values_to = "Média winsorizada (em 5%)",
  ) %>%
  kable()
```

Variável	Média winsorizada (em 5%)
w_age_t	27.40700
w_age_t6	69.96948

3.4.0.2 Mediana

A mediana (Md) é uma medida de posição do centro da distribuição dos dados ordenados de forma crescente, de modo que $P(X \leq Md) \geq \frac{1}{2}$ e $P(X \geq m) \geq \frac{1}{2}$.

Seja $X_{(1)} \leq X_{(2)} \leq ... \leq X_{(n)}$ a nossa variável ordenada de forma crescente, definimos Md como

$$\text{Md} = \begin{cases} X_{(\frac{n+1}{2})} & \text{se } n \text{ é ímpar} \\ \frac{X_{(\frac{n}{2})}+X_{(\frac{n}{2}+1)}}{2} & \text{se } n \text{ é par} \end{cases},$$

em que n é o número de observações e $X_{(i)}$ é o valor da variável na posição i.

Algo importante para se compreender da mediana (e medidas separatrizes em geral) é que ela representa um valor para o qual 50% dos dados são menores ou iguais àquele valor, enquanto os outros 50% não necessariamente serão maiores que aquele valor (podem ser iguais). Para exemplificar, imagine uma amostra de 11 observações, cuja mediana será dada pelo 6º valor ordenado dessa avariável $((11+1)/2 = 6)$. Se nossa amostra ordenada for representada por

$$1 \quad 2 \quad 3 \quad 4 \quad 5 \quad *6* \quad 7 \quad 8 \quad 9 \quad 10 \quad 11$$

de fato, 50% das observações serão menores ou iguais à mediana, enquanto os demais foram maiores. Porém se nossa amostra fosse

1 2 3 4 5 $*6*$ 6 6 6 8 9

agora, 50% das observações continuam sendo menores ou iguais à mediana, porém não podemos 50% dos valores observados são maiores que 6. Conclusão, o correto a se afirmar sobre a mediana é que 50% dos valores são menores ou **iguais** à mediana ou 50% dos valores são maiores ou **iguais** à mediana.

Como propriedades da mediana, nós temos:

1 - **Minimização do desvio absoluto**

A mediana é o valor que minimiza a soma das distâncias absolutas de todos os pontos até ela.

$\text{Md} = \arg\min_c \sum_{i=1}^{n} |X_i - c|$

Demonstrando essa propriedade, buscamos um valor c que minimize $g(c) = \sum_{i=1}^{n} |X_i - c|$, ou seja, que apresente o menor desvio absoluto (lembrando que $|X_i - c|$ representa a distância de cada ponto X_i até c). Desenvolvendo essa equação:

Passo 1: Analisar a derivada de $|x|$

- Para $x > 0$: $\frac{d|x|}{dx} = 1$
- Para $x < 0$: $\frac{d|x|}{dx} = -1$
- Em $x = 0$: a derivada não existe (ponto de não-diferenciabilidade)

Passo 2: Calcular a derivada de $g(c)$ por partes

Seja $X_{(1)} \leq X_{(2)} \leq ... \leq X_{(n)}$ a nossa variável ordenada, para um c no intervalo $(X_{(k)}, X_{(k+1)})$:

- Temos exatamente k valores menores que c: $X_{(1)}, ..., X_{(k)}$
- Temos exatamente $n - k$ valores maiores que c: $X_{(k+1)}, ..., X_{(n)}$

Aplicando a regra da cadeia em cada parte

- Para $X_i < c$: $\frac{d|X_i - c|}{dc} = \frac{d|X_i - c|}{d(X_i - c)} \cdot \frac{d(X_i - c)}{dc} = (-1) \cdot (-1) = 1$
- Para $X_i > c$: $\frac{d|X_i - c|}{dc} = \frac{d|X_i - c|}{d(X_i - c)} \cdot \frac{d(X_i - c)}{dc} = (1) \cdot (-1) = -1$

Passo 3: Somar as derivadas

Então no intervalo $(X_{(k)}, X_{(k+1)})$ temos $g'(c) = \sum_{i:X_i<c} 1 + \sum_{i:X_i>c}(-1) = k - (n - k) = 2k - n$.

Passo 4: Encontrar o ponto crítico

$g'(c) = 0$ quando $2k - n = 0$, ou seja, quando $k = n/2$. Assim, obtemos o valor que minimiza o desvio absoluto quando usamos um valor c, tal que $X_{(n/2)} < c < X_{(\frac{n}{2}+1)}$, e a mediana é justamente a medida que cumpre esses requisitos.

2 - **Equivariância**

Se $Y = aX + b$ $(a > 0)$, então $\text{Med}(Y) = a \cdot \text{Med}(X) + b$. Observe que, sejam $X_1, X_2, ..., X_n$ as observações de X, então $Y_i = aX_i + b$ para $i = 1, 2, ..., n$. Como os X_i são deslocados por b no mesmo sentido e "esticados" ou "comprimidos" por a na mesma intensidade e direção (pois $a > 0$), então a ordem é preservada. Logo, se $X_i < X_j$ então $aX_i + b < aX_j + b$, portanto, $Y_i < Y_j$. Resumindo, a posição da mediana em Y é a mesma posição da mediana em X, com isso a $\text{Med}(Y) = Y_{(\frac{n+1}{2})} = aX_{(\frac{n+1}{2})} + b = a \cdot \text{Med}(X) + b$.

3 - **Robustez**

A mediana apresenta ponto de ruptura de 50%, o que, de forma bem simplista, significa que precisaríamos substituir cerca de metade das observações por valores arbitrários para que a variável não forneça informação suficiente sobre a verdadeira localização da mediana. Na prática, significa que ela sofre menos alterações

se decidirmos substituir outliers por uma medida de posição (como a própria mediana, a média, ou um percentil).

4 - **Distribuição assintótica da mediana** (para variáveis contínuas)

Uma distribuição assintótica descreve o **comportamento limite** da distribuição de um estimador (ou estatística) quando o tamanho da amostra, n, cresce indefinidamente, ou seja, quando $n \to \infty$. Quando nós calculamos uma estatística (por exemplo, a média, a variância ou a mediana amostral), sua distribuição exata depende do tamanho da amostra e da distribuição dos dados. À medida que o n cresce, muitas estatísticas passam a ter distribuições que semelhantes a alguma forma simples e bem conhecida, por exemplo, a distribuição normal. Essa distribuição limite é o que chamamos de **distribuição assintótica**.

No caso da mediana, temos o teorema que $\sqrt{n}(\text{Md} - \theta) \xrightarrow{d} N\left(0, \frac{1}{4f(\theta)^2}\right)$, onde θ é a mediana populacional e $f(\theta)$ é a densidade no ponto θ, ou seja, quando n é grande, a distribuição da variável transformada $\sqrt{n}(\text{Md} - \theta)$ se aproxima de uma normal $N(0,\, 1/(4f(\theta)^2))$.

Simplificando, a mediana amostral é aproximadamente normal quando a amostra é grande, com variância decrescendo a uma taxa de $1/n$, ou seja, a mediana, assim como a média, se torna cada vez mais precisa (mais próxima da mediana populacional) quando você tem mais dados. E essa precisão crescente se reflete matematicamente pela variância decrescendo a uma taxa de $1/n$.

Se definimos $Y_n = \sqrt{n}(\text{Md} - \theta)$, então, $\text{Md} - \theta = \frac{Y_n}{\sqrt{n}}$. Calculando a variância assintótica da Md teremos $\text{Var}(\text{Md}) = \text{Var}\left(\frac{Y_n}{\sqrt{n}}\right)$ (θ é constante e não altera a variância). Como multiplicar uma variável aleatória por uma constante $1/\sqrt{n}$ multiplica sua variância por $1/n$, então, $\text{Var}(\text{Md}) \approx \frac{1}{n}\,\text{Var}(Y_n)$. Agora, se $\text{Var}(Y_n) \to \frac{1}{4f(\theta)^2}$, então $\text{Var}(\text{Md}) \approx \frac{1}{4nf(\theta)^2}$, onde o único elemento que varia é n, por isso que a variância da distribuição amostral da mediana vai decrescendo a uma taxa de $1/n$ conforme n aumenta.

Para calcular a mediana de dados agrupados de uma variável discreta, usamos o mesmo método dos dados não agrupados, ou seja, calculamos a posição das observações que serão usadas para calcular a mediana, e utilizamos o valor da categoria em que cada posição está. Por exemplo, imagine a seguinte variável discreta agrupada:

Número de crias	F_i	F_{ac}
1	6	6
2	20	26
3	14	40
4	3	43
5	1	44

Como n é par, a mediana é a soma dos elementos na posição $n/2 = 22$ e $(n/2)+1 = 23$ divido por 2. Olhando a frequência acumulada, vemos que ambos estão na categoria 2, logo, a mediana é $Md = (2+2)/2 = 2$.

Para dados agrupados de uma variável contínua, primeiro calculamos a posição da mediana como $n/2$ e identificamos a classe em que contém a posição mediana, a partir da avaliação das frequências acumuladas. Essa classe, agora denominada classe mediana (C_{Md}) será a base para calcularmos a mediana segundo a expressão:

$$Md = LI_{C_{Md}} + \frac{\left(\frac{n}{2} - F_{ac(C_{Md}-1)}\right)}{F_{C_{Md}}} \times A_{C_{Md}}$$

em que

- $LI_{C_{Md}}$ é o limite inferior do intervalo da classe mediana
- $F_{ac(C_{Md}-1)}$ é a frequência acumulada da classe anterior à classe mediana

- $F_{C_{Md}}$ é a frequência absoluta da classe mediana
- n é o total de observações, e
- $A_{C_{Md}}$ é a amplitude da classe mediana (limite superior menos o inferior).

Como a mediana é uma medida separatriz, o segundo quartil (ou quantil de ordem 0,5), deixarei para apresentar o exemplo de sua obtenção no R no tópico de **medidas separatrizes**.

3.4.0.3 Moda

A moda é o valor de maior frequência (mais observado). De forma mais formal, para uma variável discreta a moda é $\text{Mo} = \arg\max_x P(X = x)$, enquanto para uma variável contínua a moda é $\text{Mo} = \arg\max_x f(x)$, ou seja, o valor com maior probabilidade de ocorrência daquela variável.

Para dados qualitativos ou quantitativos discretos, iremos calcular as frequências de cada valor/categoria e a classe (unimodal) ou classes (bi ou multimodal) de maior frequência serão a moda.

Já para dados contínuos, iremos primeiro agrupá-los (seguindo aquelas regras apresentadas anteriormente para tabelas de frequência) e depois estimar a moda usando o método de Czuber ou o método de King (raramente é usado atualmente).

No método de Czuber, primeiro identificamos a classe que contém a moda (aquela de maior frequência absoluta). Essa classe, agora denominada classe modal (C_{Mo}) será a base para calcularmos a moda segundo a expressão:

$$Mo = LI_{C_{Mo}} + \frac{(F_{C_{Mo}} - F_{C_{Mo-1}})}{2 \times F_{C_{Mo}} - (F_{C_{Mo-1}} + F_{C_{Mo+1}})} \times A_{C_{Mo}}$$

em que

- $LI_{C_{Mo}}$ é o limite inferior do intervalo da classe modal
- $F_{C_{Mo}}$ é a frequência absoluta da classe modal
- $F_{C_{Mo-1}}$ é a frequência absoluta da classe anterior à classe modal
- $F_{C_{Mo+1}}$ é a frequência absoluta da classe posterior à classe modal
- $A_{C_{Mo}}$ é a amplitude da classe modal (limite superior menos o inferior).

No método de King, também identificamos a classe que contém a moda e calculamos a moda segundo a expressão:

$$Mo = LI_{C_{Mo}} + \frac{F_{C_{Mo+1}}}{(F_{C_{Mo-1}} + F_{C_{Mo+1}})} \times A_{C_{Mo}}$$

em que

- $LI_{C_{Mo}}$ é o limite inferior do intervalo da classe modal
- $F_{C_{Mo-1}}$ é a frequência absoluta da classe anterior à classe modal
- $F_{C_{Mo+1}}$ é a frequência absoluta da classe posterior à classe modal
- $A_{C_{Mo}}$ é a amplitude da classe modal (limite superior menos o inferior).

No R não há uma função no `base` para obtenção da moda, entrentanto, iremos criar a nossa própria:

```
# Calcula a moda de uma variável categórica ou discreta
#
# Args:
#   data: data.frame ou tibble; fonte dos dados.
#   var: character; nome da variável para o qual será calculada a moda.
#
# Returns:
#   lista com dados originais, variável de interesse, método de agrupamento (NULL),
#   tabela de frequências e modas calculadas (list).
```

```r
moda_qualitativa_discreta <- function(data, var) {
  var_discreta <- is.numeric(data[[var]])

  if(var_discreta) {
    data <- data %>%
      arrange(.data[[var]])
    data[[var]] <- factor(data[[var]], ordered = TRUE)
  }

  tab_frequencias <- data %>%
    count(.data[[var]], name = "Fi") %>%
    mutate(
      Fri = scales::percent(Fi / sum(Fi), accuracy = .1),
      Fac = cumsum(Fi),
      Frac = scales::percent(cumsum(Fi / sum(Fi)), accuracy = .1)
    )

  freq_max <- max(tab_frequencias$Fi)
  classe_modal <- tab_frequencias %>%
    filter(Fi == freq_max)

  return(list(
    dados = data,
    variavel = var,
    metodo_agrupamento = NULL,
    metodo_moda = ifelse(var_discreta, "Dados discretos - max(frequencia)", "Dados qualitativos - max(freq
    tabela_de_frequencias = tab_frequencias,
    resultado = tibble(
      moda = as.character(classe_modal[[var]]),
      frequencia = classe_modal$Fi
    )
  ))
}

# Calcula a moda de uma variável contínua
#
# Args:
#   data: data.frame ou tibble; fonte dos dados.
#   var: character; nome da variável para o qual será calculada a moda.
#   tipo_agrupamento: character; método usado para criar as classes (intervalos) da variável.
#   metodo_moda: character; método que deverá usar para calcular a moda (aceita "czuber" e "king")
#
# Returns:
#   lista com dados originais, variável de interesse, método de agrupamento,
#   tabela de frequências e modas calculadas (list).
moda_continua <- function(data, var, tipo_agrupamento = "sturges", metodo_moda = "czuber") {
  metodos_disponiveis <- c("czuber", "king")
  if (!(metodo_moda %in% metodos_disponiveis)) {
    stop(
      sprintf(
```

```
      "Método '%s' não disponível. Métodos possíveis: %s.",
      metodo_moda,
      str_c(metodos_disponiveis, collapse = ", ")
    )
  )
}

n <- length(data[[var]])
if(n == 0) return(NULL)
if(n == 1) return(data[[var]])

k_breaks <- cria_classes_continua(data[[var]], tipo = tipo_agrupamento)
if(is.null(k_breaks)) return(NULL)

breaks <- k_breaks$breaks

tab_frequencias <- data %>%
  mutate(
    agrupamento = cut(.data[[var]], breaks = breaks, right = TRUE, include.lowest = TRUE),
  ) %>%
  count(agrupamento, name = "Fi") %>%
  mutate(
    idx = row_number(),
    Fri = scales::percent(Fi / sum(Fi), accuracy = .1),
    Fac = cumsum(Fi),
    Frac = scales::percent(cumsum(Fi / sum(Fi)), accuracy = .1),
    Ls = breaks[idx + 1],
    Li = breaks[idx],
    amplitude = Ls - Li
  )

freq_max <- max(tab_frequencias$Fi)
classe_modal <- tab_frequencias %>%
  filter(Fi == freq_max)

modas <- rep(NA_real_, nrow(classe_modal))

modas <- sapply(seq_along(classe_modal[[1]]), function(i) {
  indice = classe_modal$idx[i]
  Li_mo <- classe_modal$Li[i]
  A_mo <- classe_modal$amplitude[i]
  f_mo <- classe_modal$Fi[i]
  f_mo_anterior <- 0
  f_mo_posterior <- 0

  if(indice > 1) f_mo_anterior <- tab_frequencias$Fi[indice - 1]
  if(indice < max(tab_frequencias$idx)) f_mo_posterior <- tab_frequencias$Fi[indice + 1]

  if (metodo_moda == "czuber") {
    return(Li_mo + ( ( (f_mo - f_mo_anterior) / (2 * f_mo - (f_mo_anterior + f_mo_posterior)) ) * A_mo )
```

```r
    } else if (metodo_moda == "king") {
      return(Li_mo + ( ( (f_mo_posterior) / (f_mo_anterior + f_mo_posterior)) * A_mo ))
    }
  })

  return(list(
    dados = data,
    variavel = var,
    metodo_agrupamento = tipo_agrupamento,
    metodo_moda = metodo_moda,
    tabela_de_frequencias = tab_frequencias %>%
      rename_with(~ var, "agrupamento") %>%
      select(-c(idx, Ls, Li, amplitude)),
    resultado = tibble(
      classes_modais = as.character(classe_modal$agrupamento),
      moda = as.character(round(modas, 3))
    )
  ))
}

# Calcula a moda
#
# Args:
#   data: data.frame ou tibble; fonte dos dados.
#   var: character; nome da variável para o qual será calculada a moda.
#   tipo_var: character; tipo da variável (aceita os valores "cat" para variável qualitativa,
#   "cont" para variável contínua e "disc" para variável discreta).
#   ...: demais argumentos passados para moda_continua() quando tipo_var = "cont"
#
# Returns:
#   lista com dados originais, variável de interesse, método de agrupamento
#   (se variável contínua), tabela de frequências e modas calculadas (list).
moda <- function(data, var, tipo_var = "cat", ...) {
  tipos_aceitos <- c(cat="Categórico ou qualitativo", cont="Contínuo", disc="Discreto")
  if (!(tipo_var %in% names(tipos_aceitos))) {
    stop(
      sprintf(
        'O tipo de variável "%s" está incorreto. Tipos aceitos: %s.',
        tipo_var,
        str_c(
          str_c(paste0('"',names(tipos_aceitos),'"'), tipos_aceitos, sep = " para "),
          collapse = ", "
        )
      )
    )
  }

  if(tipo_var == "cont") return(moda_continua(data, var, ...))

  return(moda_qualitativa_discreta(data, var))
```

```r
}

# categorica
moda_cat <- moda(ap2, "vacc_mp", tipo_var = "cat")
# discreta
moda_disc <- moda(ap2, "age_t", tipo_var = "disc")
# continua
moda_cont <- moda(ap2, "w_age_t", tipo_var = "cont")

kable(tibble(
  `Tipo` = c("Qualitativa", "Quantitativa discreta", "Quantitativa contínua"),
  `Variável` = c(
    rep(moda_cat$variavel, nrow(moda_cat$resultado)),
    rep(moda_disc$variavel, nrow(moda_disc$resultado)),
    rep(moda_cont$variavel, nrow(moda_cont$resultado))
  ),
  `Moda` = c(
    moda_cat$resultado$moda,
    moda_disc$resultado$moda,
    moda_cont$resultado$moda
  )
))
```

Tipo	Variável	Moda
Qualitativa	vacc_mp	Vacinado
Quantitativa discreta	age_t	67
Quantitativa contínua	w_age_t	24.203

3.4.0.4 Medidas separatrizes

Medidas separatrizes são valores que dividem um conjunto de dados ordenados em partes iguais, ajudando a descrever como os dados se distribuem. Elas indicam a posição relativa de um valor dentro da amostra ou população. As principais são os **quartis** (dividem em 4 partes iguais, 25%, 50%, 75% e 100% ou Q1, Q2/Mediana, Q3 e Q4), **decis** (10 partes) e **percentis** (100 partes). Cada medida separatriz corresponde a um **quantil** (varia entre 0 e 1) específico, por exemplo, o 1º quartil (Q1) é o quantil de ordem 0,25, ou seja, 25% dos dados são menores ou iguais a ele. Essas medidas são úteis para resumir e interpretar a dispersão e a concentração dos dados de forma mais detalhada do que apenas a média ou a mediana.

A definição formal de quantis para uma população é a seguinte:

O quantil de ordem p, em que $0 < p < 1$, é o valor q_p tal que $P(X \leq q_p) \geq p$ e $P(X \geq q_p) \geq 1 - p$, ou seja, a probabilidade de uma observação daquela variável ser menor ou igual ao q_p é maior ou igual a p (ordem do quantil).

Para uma variável X, a sua função de distribuição acumulada (FDA) é denotada por $F(x) = P(X \leq x)$, então, $q_p = F^{-1}(p) = \inf\{x : F(x) \geq p\}$, ou seja, o quantil é a inversa da FDA para o valor p ou menor valor de x que ainda satisfaz a condição de que a probabilidade acumulada é maior ou igual a p (isso para garantir a validade da definição mesmo em distribuições discretas em que a $F(x)$ possui saltos).

Na prática, não teremos os dados populacionais para calcular o quantil, por isso iremos estimá-los a partir da nossa amostra. Então, para uma amostra de tamanho n, tal que nossas observações ordenadas sejam $X_{(1)} \leq X_{(2)} \leq ... \leq X_{(n)}$, iremos estimar o quantil de ordem p a partir desses dados.

No R, a função `quantile()` adota os métodos de estimação dos quantis propostos por Hyndman e Fan (1996), sendo que o método usado pela função é determinado no argumento `type`.

Ao estimar um quantil amostral, cada método o estima segundo a definição:

$Q_i(p) = (1-\gamma)x_{(j)} + \gamma x_{(j+1)}$,

onde:

- i é o método escolhido (representado na função do R por um valor discreto entre 1 e 9);
- p é a ordem do quantil estimado;
- $j = \lfloor np + m \rfloor$ (o menor valor inteiro do resultado de $np + m$, onde n é o tamanho da amostra);
- m é um **parâmetro de deslocamento** que muda conforme o método escolhido;
- γ é um valor calculado de acordo com o método escolhido e com g; e
- $g = np + m - j$.

Os métodos são dividos em dois grupos:

Discretos (tipos 1–3):

Tipo	Descrição	Parâmetro m	γ	Observação
1	Inverso da função de distribuição empírica (*ECDF*)	$m = 0$	$\gamma = 0$ se $g = 0$, senão 1	Método "Nearest Rank"
2	ECDF com média nos pontos de descontinuidade	$m = 0$	$\gamma = 0.5$ se $g = 0$, senão 1	Padrão antigo do SAS — suaviza empates
3	Valor de ordem mais próximo ("nearest even order statistic")	$m = -0.5$	$\gamma = 0$ se $g = 0$ e j par, senão 1	Escolhe o valor mais próximo de np; usado no SAS até 2010

Esses métodos retornam sempre um valor existente nos dados (ou média de valores adjacentes).

Contínuos (tipos 4–9):

Nesses métodos, o $Q(p)$ é tratado como uma função contínua de p e eles usam $\gamma = g$ (que é a parte decimal de $np + m$). Esses métodos são equivalentes a uma interpolação linear na forma:

$Q(p) = x_{(k)} + \frac{p - p_k}{p_{k+1} - p_k}(x_{(k+1)} - x_{(k)})$

- $x_{(k)}$ é o k-ésimo valor da amostra (depois de ordenada),
- p_k é a probabilidade acumulada teórica associada a esse valor, ou seja, o valor da função de distribuição empírica (ECDF) correspondente a $x_{(k)}$. Justamente por ser probabilidade teórica associada ao k-ésimo valor ordenado da amostra, ele nos permite avaliar o viés e a suavidade do estimador de quantil.

Tipo	Parâmetro m	Probabilidade associada p_k	Interpretação e uso
4	$m = 0$	$p_k = \frac{k}{n}$	Interpolação linear da ECDF (usada raramente; simples porém ligeiramente enviesada).
5	$m = 0.5$	$p_k = \frac{k-0.5}{n}$	Gera uma função linear definida por partes em que as "quebras" ocorrem no meio dos degraus da ECDF (popular entre hidrólogos).
6	$m = p$	$p_k = \frac{k}{n+1}$	Baseada no valor esperado das estatísticas de ordem (usada pelo SPSS e Minitab).
7 (**Padrão do R**)	$m = 1 - p$	$p_k = \frac{k-1}{n-1}$	Interpolação dos modas das estatísticas de ordem. Retorna valores suavizados entre os dados.
8	$m = (p + 1)/3$	$p_k = \frac{k-1/3}{n+1/3}$	Aproximadamente uma mediana não enviezada, adequada para qualquer distribuição. Recomendado por Hyndman e Fan (1996).
9	$m = p/4 + 3/8$	$p_k = \frac{k-3/8}{n+1/4}$	Aproximadamente não enviezados se x vem de uma distribuição Normal (usado em análises robustas).

Bom, de qualquer forma os próprios idealizadores dessa função indicam fortemente o uso do tipo 8 em seu artigo, então, quem sou eu para dizer o contrário . Vamos ver um exemplo de sua utilização:

```
quartis <- ap2 %>%
  group_by(vacc_mp) %>%
  summarise(
    q1 = quantile(dwg_fin, probs=.25, type=8),
    q2 = quantile(dwg_fin, probs=.5, type=8),
    q3 = quantile(dwg_fin, probs=.75, type=8),
```

```
    q4 = quantile(dwg_fin, probs=1, type=8)
  )

kable(
  quartis,
  col.names = c(
    "Estado vacinal",
    "$Q_8(25%)$ ou Q1",
    "$Q_8(50%)$ ou Q2 ou Mediana",
    "$Q_8(75%)$ ou Q3",
    "$Q_8(100%)$ ou máximo"
  )
)
```

Tabela 3.11: Quartis do ganho de peso diário segundo o estado vacinal

Estado vacinal	Q_8(25 ou Q1	Q_8(50 ou Q2 ou Mediana	Q_8(75 ou Q3	Q_8(100 ou máximo
Não vacinado	757.1667	857	950.6667	1188
Vacinado	600.0000	700	798.0000	1133

Quartis são considerados estimadores resistentes (não afetados por até 25% de valores extremos) e basicamente dividem os dados em 4 partes com aproximadamente $n/4$ observações cada. Além disso, são a base para o cálculo da amplitude interquartil (IQR), a medida usada na construção do boxplot, obtido por $\text{IQR} = Q_3 - Q_1$. O IQR em uma distribuição normal também apresenta uma característica interessante em relação ao desvio padrão, pois ele é $\approx 1.349\sigma$, o que permite usá-lo como um estimador robusto do desvio padrão ($\hat{\sigma}_{\text{robusto}} = \text{IQR}/1.349$) se consideramos que a amostra é de uma distribuição normal.

3.5 Identificação de outliers

Consideramos outliers as observações que apresentam medidas muito discrepantes em relação às demais do conjunto de observações em estudo (amostra). Essas observações discrepantes podem ter impacto nas medidas de resumo calculadas, o que mais a frente levará a distorções nos resultados das análises que dependem dessas medidas, por exemplo, ao tentar ajustar essa amostra a uma distribuição de probabilidade conhecida (como a normal) para a realização de algum teste de hipótese ou modelagem estatística.

Antes de decidir o que será feito com essas observações, precisamos identificar o que levou a tais valores discrepantes, se isso decorre de erros na medição ou na execução da seleção dos indivíduos amostrados, ou se realmente representa a variabilidade real da população em estudo.

Há vários métodos de detecção de outliers e nesse momento apresentarei alguns específicos para quando estamos avaliando somente uma variável, que são: o **método de Tukey** pela avaliação do boxplot, o **método do Z-Score**, o **método MAD** (*median absolute deviation*) e o **teste de Grubbs**.

3.5.1 Método de Tukey (boxplot)

Esse método já foi apresentado quando apresentei o boxplot na representação gráfica de dados univariados. Nele, calculamos o intervalo interquartil (IQR) e serão considerados outliers os valores X_i que são menores que um limite inferior ou maiores que um limite superior. Esses limites são calculados como:

Limite Inferior - LI $= Q_1 - k \cdot \text{IQR}$

Limite Superior - LS $= Q_3 + k \cdot \text{IQR}$

Onde Q_1 é o primeiro quartil (quantil de ordem 25%), Q_3 é o terceiro quartil (quantil de ordem 75%) e k assume os valores de $k = 1.5$ para identificação de outliers moderados ou $k = 3$ para outliers extremos. Então qualquer valor fora desses limites serão considerados outliers e deverão receber uma maior atenção nossa para verificar se há algum erro em sua caracterização ou se precisaremos adotar alguma abordagem para que não prejudiquem nossas análises.

A justificativa para a adoção desse método de detecção de outliers está na distribuição normal, uma vez que a probabilidade de uma observação estar fora desses limites na distribuição normal é de 0.35% para $k = 1.5$ e de 0.002% para $k = 3$, ou seja:

- $P(X < Q_1 - 1.5 \cdot \text{IQR}) \approx P(X > Q_3 + 1.5 \cdot \text{IQR}) \approx 0.0035$
- $P(X < Q_1 - 3 \cdot \text{IQR}) \approx P(X > Q_3 + 3 \cdot \text{IQR}) \approx 0.00002$

No R, isso podemos aplicar esse método da seguinte forma:

```
Q1 <- quantile(ap2$dwg_fin, probs = .25, type = 8)
Q3 <- quantile(ap2$dwg_fin, probs = .75, type = 8)
IQR <- Q3 - Q1

# filtrando do dataset os outliers moderados
outliers <- ap2 %>%
  filter(dwg_fin < (Q1 - (1.5 * IQR)) | dwg_fin > (Q3 + (1.5 * IQR)))

cat("Número de outliers moderados: ", length(outliers$dwg_fin), "\n")
```

```
Número de outliers moderados:  3
```

```
# filtrando do dataset os outliers extremos
outliers <- ap2 %>%
  filter(dwg_fin < (Q1 - (3 * IQR)) | dwg_fin > (Q3 + (3 * IQR)))

cat("Número de outliers extremos: ", length(outliers$dwg_fin), "\n")
```

```
Número de outliers extremos:  0
```

3.5.2 Método Z-Score

No método Z-Score, primeiro iremos padronizar (normalizar) nossa variável, o que é obtido subtraindo do valor observado a média amsotral e, em seguida, dividindo pelo desvio padrão amostral:

$Z_i = \frac{X_i - \bar{X}}{s}$

Após realizar esse escalonamento, adotamos o seguinte critério

- $|Z_i| > 2.5$: possível outlier ($P \approx 0.012$)
- $|Z_i| > 3$: outlier moderado ($P \approx 0.003$)
- $|Z_i| > 4$: outlier extremo ($P \approx 0.00006$)

Esse método assume que a variável possui uma distribuição normal e utiliza tanto a média quanto o desvio padrão amostral, porém, ambas as medidas são afetadas por valores extremos na nossa amostra, o que pode levar a um mascaramento de outliers e limita a sua utilização na prática.

No R:

```r
kable(
  ap2 %>%
  select(dwg_fin) %>%
  mutate(
    z_na_mao = (dwg_fin - mean(dwg_fin)) / sd(dwg_fin),
    z = scale(dwg_fin),
    outlier = case_when(
      abs(z) > 4 ~ "outlier extremo",
      abs(z) > 3 ~ "outlier moderado",
      abs(z) > 2.5 ~ "possível outlier",
      TRUE ~ "não outlier"
    )
  ) %>%
  count(outlier),
  col.names = c("", "Quantidade")
)
```

	Quantidade
não outlier	1105
outlier moderado	1
possível outlier	8

3.5.3 Método MAD (Median Absolute Deviation)

Para obter o MAD, primeiro subtraímos de cada observação a mediana e, então, calculamos a mediana desse desvio em módulo (desvio absoluto):

$\text{MAD} = \text{mediana}(|X_i - \text{Md}|)$

Em seguida, calculamos o Score MAD modificado de cada observação como:

$M_i = \frac{0.6745(X_i - \text{Md})}{\text{MAD}}$

Onde $0.6745 = \Phi^{-1}(0.75)$ para consistência com σ sob normalidade. Isso porque o $\Phi(z)$ representa a função de distribuição acumulada da distribuição normal padrão, o $Z \sim N(0,1)$, ou seja, a probabilidade acumulada até aquele valor z ($\Phi(z) = P(Z \leq z)$), logo, pela função inversa $\Phi^{-1}(0.75)$ buscamos justamente o valor z para o qual temos a probabilidade acumulada de 75% (o terceiro quartil da população Z). O MAD em uma população normal do tipo $N(0,\sigma)$ é $MAD = \sigma\Phi^{-1}(0.75)$, então, para fazer que nosso M_i se comporte de modo comparável a um z-score (unidades de desvio padrão sob normalidade) nós multiplicamos pelo valor de $0.6745 = \Phi^{-1}(0.75)$:

$MAD = \sigma\Phi^{-1}(0.75) \Rightarrow \frac{1}{\sigma} = \frac{\Phi^{-1}(0.75)}{MAD}$

$\frac{1}{\sigma} = \frac{0.6745}{MAD}$

$M_i = \frac{0.6745(X_i - \text{Md})}{\text{MAD}} = \frac{0.6745}{\text{MAD}} \cdot (X_i - \text{Md})$

$M_i \approx \frac{X_i - \text{Md}}{\sigma}$

Ou seja, M_i é aproximadamente o z-score robusto, expresso em unidades de "desvio padrão robusto" (robusto porque usa a medaiana).

Calculado o M_i, adotamos como critério de identificação de outliers, aquelas observações com valor absoluto de M_i superior a 3,5 ($|M_i| > 3.5$).

No R:

```
# median é o mesmo que quantile(x, .5)
mad <- median(abs(ap2$dwg_fin - quantile(ap2$dwg_fin, probs = .5, type = 8)))

kable(
  ap2 %>%
  select(dwg_fin) %>%
  mutate(
    m_i = (0.6745 * (dwg_fin - median(dwg_fin))) / mad,
    outlier = case_when(
      abs(m_i) > 3.5 ~ "outlier",
      TRUE ~ "não outlier"
    )
  ) %>%
  count(outlier),
  col.names = c("", "Quantidade")
)
```

	Quantidade
não outlier	1114

3.5.4 Teste de Grubbs

O teste de Grubbs é um método que utiliza de um teste de hipótese, conceito que ainda veremos em nossos treinamentos, mas que basicamente é um procedimento estatístico usado para decidir, com base em dados de uma amostra, se há evidências suficientes para aceitar ou rejeitar uma suposição (a hipótese) sobre uma característica da população. Mais precisamente, o teste de Grubbs é um teste paramétrico (assume que a variável é aproximadamente normal) utilizado para detectar um único valor atípico (outlier) em um conjunto de dados univariado.

Ele testa a hipótese de que um dos valores da amostra é um outlier, ou seja, de que ele pertence a uma população diferente ou do quão improvável é obter aquele valor extremo observado de uma população normal com as características da amostra em estudo.

Considere uma amostra

$x_1, x_2, \dots, x_n$

para a qual assumimos que ela vem de uma população normal ($X \sim N(\mu, \sigma^2)$). Para esses pressupostos, definimos as seguintes hipóteses:

- Hipótese nula (H_0): Não há outliers, ou seja, todos os dados vêm de uma mesma distribuição normal.
- Hipótese alternativa (H_1): Existe exatamente um outlier na amostra (uma observação de valor extremo que não é da mesma população em estudo).

Para esse teste calculamos a estatística de Grubbs, definida como:

$G = \max_i \frac{|x_i - \bar{x}|}{s}$

Ou seja, a observação com o maior desvio em relação à média em termos de desvios padrão (o desvio padronizado extremo).

Calculado o G da amostra, o comparamos com o valor crítico, o G_{crtico}. Se $G > G_{crtico}$, rejeitamos H_0, ou seja, a hipótese de que todos os dados são da mesma população normal, logo, consideramos o valor extremo

como um outlier. Caso contrário, aceitamos que não há evidências estatísticas o suficiente para considerar o valor extremo como um outlier.

Esse G_{crtico} é obtido a partir da distribuição t de Student com $n-2$ graus de liberdade da seguinte forma:

$G_{crtico} = \frac{(n-1)}{\sqrt{n}} \sqrt{\frac{t^2_{\alpha/(2n),\, n-2}}{n-2+t^2_{\alpha/(2n),\, n-2}}}$

onde $t_{\alpha/(2n),\, n-2}$ é o quantil da distribuição t com nível de significância ajustado para $\alpha/(2n)$ (teste bilateral com ajuste para múltiplas comparações). A limitação desse teste é que ele pressupõe que os dados são aproximadamente normais e que ele testa a hipótese de um único outlier, assim, se os dados apresentam mais de um, um outlier extremo pode mascarar os outros.

No R podemos aplicar esse teste da seguinte forma:

```
# testando a presença de um outlier extremo na
# variável w_age_t de ap2
# pelo teste de Grubbs com alfa de 5%
media_amostral <- mean(ap2$w_age_t)
dp_amostral <- sd(ap2$w_age_t)
n <- length(ap2$w_age_t)

idx_extremo <- which.max(abs(ap2$w_age_t - media_amostral))
valor_extremo <- ap2$w_age_t[idx_extremo]
cat("Observação potencialmente outlier: observação de nº", idx_extremo, "e valor", valor_extremo, "\n")
```

```
Observação potencialmente outlier: observação de nº 409 e valor 47.2
```

```
G <- max(abs(ap2$w_age_t - media_amostral)) / dp_amostral

alpha <- 0.05
t_crit <- qt(p = 1 - alpha/(2*n), df = n - 2)
G_crit <- ((n - 1) / sqrt(n)) * sqrt( t_crit^2 / (n - 2 + t_crit^2) )

if (G > G_crit) {
  cat("Há evidências de 1 outlier (Rejeita H0)\n")
} else {
  cat("Sem evidências de outlier (Não rejeita H0)\n")
}
```

```
Sem evidências de outlier (Não rejeita H0)
```

Também podemos aplicar esse teste no R calculando o p-valor. Nesse caso, estamos assumindo que a distribuição sob H_0 (ausência de outliers) está relacionada à distribuição t de Student. Dessa forma, o p-valor indica a probabilidade de observar um valor de G tão extremo (ou mais) quanto o obtido, caso realmente não existam outliers. Formalmente, nosso p-valor indica:

$p - valor = P(G \geq G_{obs} | H_0)$

Se o p-valor for inferior ao limite que estabelecemos (o α), rejeitamos H_0 e consideramos que existe evidência estatística de que há um outlier extremo. Para a obtenção do p-valor, usamos a relação entre as estatíticas G e t:

$t^2 = \frac{G^2 \cdot n \cdot (2-n)}{G^2 \cdot n - (n-1)^2}$ (isolando o t^2 da definição do G_{crit})

como $(2-n) = -(n-2)$

$t^2 = \frac{-G^2 \cdot n \cdot (n-2)}{G^2 \cdot n - (n-1)^2}$

$t^2 = \frac{G^2 \cdot n \cdot (n-2)}{(n-1)^2 - G^2 \cdot n}$

dividindo numerador e denominador por n

$t^2 = \frac{G^2 \cdot (n-2)}{\frac{(n-1)^2}{n} - G^2}$

como $\frac{(n-1)^2}{n} = \frac{n^2-2n+1}{n} = n - 2 + \frac{1}{n}$, então,

$t^2 = \frac{G^2 \cdot (n-2)}{(n-2+\frac{1}{n}) - G^2}$. Tomanda a raiz quadrada dos dois lados, chegamos à equação

$t = \frac{G\sqrt{n-2}}{\sqrt{(n-2+\frac{1}{n}) - G^2}}$, entretanto, a maioria dos livros adota a aproximação de $n - 2 + \frac{1}{n} \approx n - 1$ e apresenta essa relação como:

$t = \frac{G\sqrt{n-2}}{\sqrt{n-1-G^2}}$

com $n - 2$ graus de liberdade. Assim, o p-valor bilateral será calculado como:

$p = 2n[1 - F_t(|t|; n - 2)]$

em que $F_t(.; v)$ é a função de distribuição acumulada da t de Student com v graus de liberdade. No R:

```
# testando a presença de um outlier extremo na
# variável w_age_t de ap2
# pelo teste de Grubbs com alfa de 5%
media_amostral <- mean(ap2$w_age_t)
dp_amostral <- sd(ap2$w_age_t)
n <- length(ap2$w_age_t)

idx_extremo <- which.max(abs(ap2$w_age_t - media_amostral))
valor_extremo <- ap2$w_age_t[idx_extremo]
cat("Observação potencialmente outlier: observação de nº", idx_extremo, "e valor", valor_extremo, "\n")
```

```
Observação potencialmente outlier: observação de nº 409 e valor 47.2
```

```
G <- max(abs(ap2$w_age_t - media_amostral)) / dp_amostral

# usando a definição inicial (isolando o t)
s <- (G^2 * n * (2 - n)) / ((G^2 * n) - (n - 1)^2)
t_iso <- sqrt(s)

# usando a definição adotada em livros (aproximação de t)
t_aprox <- sqrt(n - 2) * G / sqrt(n - 1 - G^2)

cat("G =", round(G, 3), " | t (isolado) =", t_iso, " | t (aproximado) =", t_aprox, "\n")
```

```
G = 3.014  | t (isolado) = 3.026734  | t (aproximado) = 3.025364
```

```
p_valor <- 2 * n * (1 - pt(abs(t_iso), df = n - 2))
p_valor[p_valor > 1] <- 1

cat("p-valor =", round(p_valor, 5), "\n")
```

```
p-valor = 1
```

```r
if (p_valor < .05) {
  cat("Há evidências de 1 outlier (Rejeita H0)\n")
} else {
  cat("Sem evidências de outlier (Não rejeita H0)\n")
}
```

```
Sem evidências de outlier (Não rejeita H0)
```

3.6 Medidas de dispersão ou variabilidade

As medidas de dispersão ou de variabilidade quantificam o grau em que os dados se afastam do seu valor central (como a média, mediana ou moda). Enquanto as medidas de tendência central descrevem "onde" os dados se concentram, as medidas de dispersão informam "quanto" eles se espalham em torno desse centro.

Formalmente, se temos um conjunto de observações $X_1, X_2, \dots, X_n$, as medidas de dispersão descrevem o comportamento da variabilidade desses valores.

Além disso, essas medidas (principalmente a variância) são essenciais na inferência estatística, processo pelo qual extrapolamos um conhecimento sobre uma população a partir de dados de uma amostra dela, pois a variância é base de estimativas, testes de hipóteses e cálculo de intervalos de confiança.

3.6.1 Amplitude ou *range*

É a medida de dispersão mais simples e representa a diferença entre o maior e menor valor do conjunto de observações. Formalmente, definimos a amplitude (R) como:

$R = X_{(n)} - X_{(1)} = \max\{X_1, \dots, X_n\} - \min\{X_1, \dots, X_n\}$

Essa medida é extremamente sensível a outliers, afinal, basta um outlier para alterar seu valor.

Embora a amplitude não seja uma medida comum na epidemiologia veterinária, ela pode ser bem útil na área de controle de qualidade de processos, por exemplo, aplicado à indústria e laboratórios. Nesse contexto a amplitude é utilizada na construção dos gráficos $\bar{x}$ **e R**, ou gráfico de médias e amplitudes.

Em uma indústria devemos ter padronização entre produtos, por exemplo, em uma indústria farmacêutica, não esperamos, na verdade não é aceitável, uma grande variação na concentração de um fármaco entre comprimidos ou lotes. Então, para controlar o processo de produção desse nosso remédio hipotético é necessário analisar o comprimido, após sua produção. Mas essa análise levará a perda do produto e são muitos os comprimidos produzidos a cada dia, então nossa solução é coletar uma amostra. Porém, essa amostra deverá ser coletada regularmente pois podem ocorrer eventos que alterem a qualidade em um curto espaço de tempo, por exemplo, entre turnos talvez pela experiência do grupo de pessoas que trabalham em cada turno, ou até no mesmo turno, caso ocorra alguma falha em uma esteira de produção. Ou seja, precisamos de amostras coletadas regularmente em um intervalo não muito grande. Um tamanho de amostra grande seria inviável (economicamente) nesse caso, por isso são coletadas amostras pequenas ($1 < n \leq 10$) em um intervalo regular do processo produtivo, sob condições semelhantes e avalia-se o comportamento da distribuição da amplitude e das médias amostrais no gráfico $\bar{x}$ **e R**.

Esse gráfico é na verdade um composto de dois gráficos: o gráfico de variação do processo (gráfico R) e o gráfico de médias do processo (gráfico $\bar{x}$).

O gráfico R monitora a variabilidade dentro de cada subgrupo (amostra) pela amplitude ($R_i = max(x_{ij}) - min(x_{ij})$), e apresenta como linha central a amplitude média das amostras ($\bar{R} = \frac{1}{m}\sum_{i=1}^{m} R_i$, em que m é o número de amostragens realizadas). A partir de $\bar{R}$ calculamos os limites de controle do gráfico R como:

- *Lower control limit* - $LCL_R = D_3\bar{R}$
- *Upper control limit* - $UCL_R = D_4\bar{R}$

em que D_3 e D_4 são valores tabelados de manuais estatísticos que dependem do tamanho da amostra (n). Esse D_n deriva das propriedades estatísticas esperadas da distribuição de amplitudes amostrais sob a condição de normalidade dos dados.

Essas propriedades nos informam que, considerando uma variável normal $X \sim Normal(\mu, \sigma^2)$ e sua amplitude amostral $R = X_{(n)} - X_{(1)}$, podemos, a partir da padronização da variável, descrever a amplitude como:

$Z_i = \frac{X_i - \mu}{\sigma} \quad \Longrightarrow \quad R = \sigma\, R_Z = \sigma\,(Z_{(n)} - Z_{(1)})$

Então, a esperança da amplitude é $E[R] = \sigma E[R_Z]$, um valor que depende do desvio padrão da variável e da média da amplitude da amostra padronizada Z_i. Como essa esperança de Z_i vem de uma $Normal(0, 1)$, ela depende unicamente do tamanho da amostra (n), por isso convenciona-se denominá-la $d_2(n)$ ("2" por ser uma média da diferença entre os dois extremos da amostra padronizada).

Pela definição anterior, $E[R] = d_2(n)\sigma$, portanto, $\hat{\sigma} = \frac{\bar{R}}{d_2(n)}$, ou seja, uma estimativa do desvio padrão da variável baseado na média das amplitudes amostrais e no tamanho amostral (n). Além disso, $SD[R] = d_3(n)\sigma$, sendo $d_3(n)$ também um valor tabelado derivado da Z_i para um tamanho amostral n, então, definindo $\pm 3\sigma_R$ como limites do gráfico R chegamos à

$LCL_R = \bar{R} - 3\sigma_R \Rightarrow LCL_R = \bar{R} - 3d_3(n)\sigma,$

e substituindo σ por $\hat{\sigma} = \frac{\bar{R}}{d_2(n)}$

$LCL_R = \bar{R} - 3d_3(n)\frac{\bar{R}}{d_2(n)} \Rightarrow LCL_R = (1 - 3\frac{d_3(n)}{d_2(n)})\bar{R}.$

Usando a mesma lógica para o limite superior obtemos $UCL_R = \bar{R} + 3d_3(n)\frac{\bar{R}}{d_2(n)} \Rightarrow UCL_R = (1 + 3\frac{d_3(n)}{d_2(n)})\bar{R}.$

Ou seja, $D_3 = 1 - 3\frac{d_3(n)}{d_2(n)}$ e $D_4 = 1 + 3\frac{d_3(n)}{d_2(n)}$.

Então na avaliação do gráfico R, caso existam pontos fora dos limites, o processo é considerado fora de controle, mesmo que as médias das amostras pareçam estáveis. Caso estejam dentro do limite, a variabilidade do processo é considerado sob controle estatístico e então partimos para a análise do gráfico $\bar{x}$.

No gráfico $\bar{x}$ calculamos a média de cada subgrupo ($\bar{x_i} = \frac{1}{n}\sum_{j=1}^{n} x_{ij}$) para os pontos do gráfico, a linha central do gráfico como a média das médias amostrais ($\bar{x} = \frac{1}{m}\sum_{i=1}^{m} \bar{x_i}$) e os limites como:

- *Lower control limit* - $LCL_{\bar{x}} = \bar{x} - A_2\bar{R}$
- *Upper control limit* - $UCL_{\bar{x}} = \bar{x} + A_2\bar{R}$

em que A_2 também é um valor tabelado dependente do n amostral.

Se os dados são normais e o processo está sob controle, então, a distribuição das médias amostrais também é normal, na forma $\bar{X}_i \sim N(\mu, \frac{\sigma^2}{n})$. Nesse caso, 95,45% das médias amostrais estarão a $\pm 2\sigma/\sqrt{n}$ de μ e 99,73% delas estarão a $\pm 3\sigma/\sqrt{n}$ da média populacional.

Assim, nossos limites conceituais deveriam ser $\mathrm{CL}_{\bar{X}} = \mu \pm 3 \cdot \frac{\sigma}{\sqrt{n}}$, mas já que não sabemos o σ, usamos a relação anterior da amplitude para estimar o desvio padrão e calculamos os limites como $\mathrm{UCL}_{\bar{X}} = \bar{X} + 3 \cdot \frac{\bar{R}}{d_2\sqrt{n}}$ e $\mathrm{LCL}_{\bar{X}} = \bar{X} - 3 \cdot \frac{\bar{R}}{d_2\sqrt{n}}$. Ou seja, $A_2 = \frac{3}{d_2\sqrt{n}}$.

Nesse gráfico de $\bar{x}$, se os valores estão dentro dos limites, considera-se que o nível médio do processo está sob controle, mas em caso contrário, consideramos que existem evidências de alteracão sistemática ou deslocamento da média.

As limitações desse método derivam de seus pressupostos:

1. A característica de qualidade medida é bem modelada por uma variável de distribuição normal;
2. Os parâmetros de μ e σ são iguais para todas as amostras e independentes entre amostragens;
3. O procedimento de medição é consistente e padronizado entre amostras.

Além disso, esse método é menos eficiente que o **gráfico $\bar{x}$ e s**, que usa o desvio padrão amostral, principalmente para amostras maiores ($n > 10$) e também é menos sensível para pequenas mudanças no processo (alterações inferiores a $1,5\sigma$).

Abaixo segue um exemplo do gráfico $\bar{x}$ e R:

```
set.seed(123)
mu <- 100 # média da nossa pop simulada
sigma <- 2 # dp da nossa pop simulada
n <- 5 # tamanho de cada amostra
m <- 25 # número de amostras

dados <- tibble(
  subgrupo = rep(1:m, each = n),
  obs = rep(1:n, m),
  valor = rnorm(m * n, mean = mu, sd = sigma)
)

# media e amplitude de cada amostra
resumo <- dados %>%
  group_by(subgrupo) %>%
  summarise(
    xbar = mean(valor),
    R = max(valor) - min(valor)
  )

# medias da distribuicao amostral
xbar_bar <- mean(resumo$xbar)
R_bar <- mean(resumo$R)

A2 <- 0.577
D3 <- 0.000
D4 <- 2.114

# componentes dos gráficos
limites <- tibble(
  tipo = c("xbar", "R"),
  centro = c(xbar_bar, R_bar),
  LCL = c(xbar_bar - A2 * R_bar, D3 * R_bar),
  UCL = c(xbar_bar + A2 * R_bar, D4 * R_bar)
)

# gráfico R
g_R <- ggplot(resumo, aes(x = subgrupo, y = R)) +
  geom_line(color = "gray30") +
  geom_point(size = 2, color = "royalblue") +
```

```
  geom_hline(yintercept = limites$centro[limites$tipo=="R"], color="blue", lwd=0.8) +
  geom_hline(yintercept = limites$UCL[limites$tipo=="R"], color="red", lty=2, lwd=0.8) +
  geom_hline(yintercept = limites$LCL[limites$tipo=="R"], color="red", lty=2, lwd=0.8) +
  labs(
    title = "Gráfico R - Amplitude dos Subgrupos",
    x = "Subgrupo",
    y = "Amplitude (R)"
  ) +
  theme_minimal(base_size = 13)

g_xbar <- ggplot(resumo, aes(x = subgrupo, y = xbar)) +
  geom_line(color = "gray40") +
  geom_point(size = 2, color = "darkgreen") +
  geom_hline(yintercept = limites$centro[limites$tipo=="xbar"], color="blue", lwd=0.8) +
  geom_hline(yintercept = limites$UCL[limites$tipo=="xbar"], color="red", lty=2, lwd=0.8) +
  geom_hline(yintercept = limites$LCL[limites$tipo=="xbar"], color="red", lty=2, lwd=0.8) +
  labs(
    title = expression("Gráfico " ~ bar(x) ~ "- Médias dos Subgrupos" ),
    x = "Subgrupo",
    y = expression(bar(x))
  ) +
  theme_minimal(base_size = 13)
```

```
g_R
g_xbar
```

(a) Gráfico R

(a) Gráfico x

Exemplo de gráfico x e R

Uma outra medida de amplitude ou intervalo da qual já falamos é o Intervalo Interquartil ($IQR = Q_3 - Q_1$). Essa medida de dispersão é mais robusta que a amplitude tomada pelos extremos, uma vez que seriam necessários 25% dos dados em um dos seus limites compostos por outliers para afetar essa medida. Além disso, para uma distribuição normal $E[IQR] \approx 1.349\sigma$, e podemos pensar nele como uma medida robusta da amplitude de variação dos 50% dados centrais da população.

3.6.2 Desvio-médio

O desvio médio (ou desvio absoluto médio) mede o valor médio das distâncias absolutas dos dados em relação a um ponto de referência (c), que, geralmente, é uma medida de tendência central, como a média ($\bar{X}$) ou a mediana (Md), embora o primeiro seja o mais comumente apresentado em libros introdutórios de estatística.

A definição formal de desvio médio para uma amostra $X_1, X_2, \dots, X_n$ é:

$DM(c) = \frac{1}{n}\sum_{i=1}^{n} |X_i - c|$

onde $|X_i - c|$ representa a distância absoluta, ou seja, desconsiderando se é positiva ou negativa, entre cada observação e o ponto de referência c. Assim, para os desvios médios da média e mediana temos:

$DM_{\bar{X}} = \frac{1}{n}\sum |X_i - \bar{X}|$

$DM_{Md} = \frac{1}{n}\sum |X_i - Md|$

Então o desvio médio é uma medida de dispersão (variabilidade da variável), assim como o desvio padrão (σ) e a variância ($Var(X)$). Contudo, ele tem uma interpretação mais robusta e menos sensível a valores extremos, principalmente quando usamos a mediana como posição central, pois utiliza o valor absoluto, e não o quadrado das diferenças. Além disso, ele fornece uma métrica direta, na mesma unidade dos dados, o que facilita a interpretação (por exemplo, "em média, os valores diferem do centro em 3,5 kg").

Uma propriedade importante do desvio médio é que a medida que o minimiza é a mediana da distribuição:

$g(c) = E[|X - c|]$

Queremos encontrar $c^* = \arg\min_c g(c)$, ou seja, o valor de c para o qual se tem a menor esperança do desvio médio.

Note que para valores de c dentro de um intervalo entre observações $(x_{(k)}, x_{(k+1)})$:

$\frac{d}{dc}g(c) = P(X < c) - P(X > c)$, o que significa que a taxa imediata de variação desse esperança é igual a subtração da probabilidade da variável ser menor e maior que o valor c. Portanto, o ponto de mínimo ocorre onde $g'(c) = 0$, ou seja, quando $P(X < c) = P(X > c)$, e isso é a definição de mediana. Assim, $\min_c E[|X - c|]$ é atingido quando $c = Md(X)$.

Isso nos indica que se nosso objetivo for resumir a dispersão média das observações utilizando a soma dos desvios absolutos, apresentando o menor desvio médio possível, a medida de centro mais adequada é a mediana, não a média.

Outras propriedades do desvio médio são:

1. Homogeneidade escalar - $DM(aX) = |a|DM(X)$;
2. Invariância translacional - $DM(X + b) = DM(X)$, ou seja, deslocar os dados em b unidades não alterará o desvio médio; e
3. Subaditividade - $DM(X + Y) \leq DM(X) + DM(Y)$

Na prática, o desvio médio (pela mediana) acaba sendo usada como uma medida de dispersão robusta quando na presença de outliers na amostra, e também como uma alternativa ao desvio padrão quando temos uma distribuição assimétrica ou com caudas pesadas.

Isso é particularmente interessante para estratégias de modelagem, uma vez que na regressão linear por OLS (*Ordinary Least Square*, mínimos quadrados ordinários) outliers podem influenciar fortemente nos parâmetros estimados, enquanto, nos modelo de regressão LAD (*Least Absolute Deviation*, mínimos desvios absolutos) ou de regressão mediana há maior resistência aos valores extremos, sendo preferido quando o erro não é bem modelado por uma distribuição normal. Formalmente,

na regressão linear clássica (OLS), buscamos

$\hat{\beta}_{OLS} = \arg\min_\beta \sum_{i=1}^n (Y_i - X_i^\top \beta)^2 \quad \Rightarrow$ minimiza L_2

enquanto, na regressão LAD (mínimos desvios absolutos) buscamos

$\hat{\beta}_{LAD} = \arg\min_\beta \sum_{i=1}^n |Y_i - X_i^\top \beta| \quad \Rightarrow$ minimiza L_1,

assim, o resultado é que no segundo caso temos um modelo menos sensível a observações atípicas.

No R, podemos obter o desvio médio da seguinte forma:

```
ap2 %>%
  summarise(
    DM_bar_x = sum(abs(w_age_t - mean(w_age_t))) / length(w_age_t), # pela média
    DM_md = sum(abs(w_age_t - median(w_age_t))) / length(w_age_t) # pela mediana
  ) %>%
  kable(col.names = c("$DM_{\\bar{X}}$", "$DM_{Md}$"))
```

$DM_{\bar{X}}$	DM_{Md}
5.41844	5.400449

3.6.3 Variância

A variância é uma medida de dispersão de uma variável aleatória, descrevendo quanto os valores de X tendem a se afastar da média $\mu = E[X]$. Na verdade, pela forma como é calculada, ela quantifica a expectativa do quadrado dos desvios em relação à média.

Formalmente, uma variância populacional é calculada como

$\sigma^2 = Var(X) = E[(X-\mu)^2] = E[X^2] - (E[X])^2$,

em que σ^2 é variância populacional ($Var(X)$), $\mu = E[X]$ é a média populacional e $E[X^2] - (E[X])^2$ é uma consequência da expansão dos termos na equação, ou seja,

$E[(X-\mu)^2] = E[X^2 - 2\mu X + \mu^2] = E[X^2] - 2\mu E[X] + \mu^2 = E[X^2] - \mu^2$.

Já para uma amostra $X_1, X_2, \dots, X_n$, define-se a variância amostral como:

$s^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2$,

em que $\bar{X} = \frac{1}{n} \sum X_i$ é a média amostral e o divisor $n-1$ é chamado de Correção de Bessel, sendo responsável por eliminar o viés da estimativa de σ^2 a partir da amostra.

Essa correção decorre do fato de que ao calcularmos uma variância populacional, estamos calculando

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

entretanto, na maioria dos casos não saberemos o tamanho da população (N) nem a média populacional (μ), e usamos a média amostral e o tamanho da amostra para estimar a variância (s^2). Se considerarmos uma distribuição amostral $X_1, X_2, \dots, X_n \overset{iid}{\sim} Normal(\mu, \sigma^2)$, ou seja, todas as possíveis amostras que podem ser realizadas daquela população com uma amostra de tamanho n, definimos a variância das médias amostrais como:

$Var(\bar{X}) = Var\left(\frac{1}{n} \sum_{i=1}^n X_i\right) = \frac{\sigma^2}{n}$,

e podemos desenvolver a soma dos desvios amostrais como:

$\sum_{i=1}^{n}(X_i - \bar{X})^2 = \sum_{i=1}^{n}(X_i - \mu)^2 - n(\bar{X} - \mu)^2$

em que $\sum_{i=1}^{n}(X_i - \mu)^2$ é a soma dos desvios em relação à verdadeira média (a populacional) e $n(\bar{X} - \mu)^2$ é a parte da variabilidade da soma dos desvios amostrais em decorrência do deslocamento da média amostral em relação à μ. Avaliando a esperança dos desvios quadráticos da distribuição amostral teremos:

$E[\sum(X_i - \bar{X})^2] = n\sigma^2 - nVar(\bar{X}) = n\sigma^2 - n\frac{\sigma^2}{n} = (n-1)\sigma^2,$

então, dividindo por $n-1$ em ambos os lados, obtemos

$E[\frac{1}{n-1}\sum(X_i - \bar{X})^2] = \sigma^2$

Entre as propriedades da variância, temos que:

1. A média é o valor que minimiza a esperança do erro quadrático - $\sigma^2 = \min_c E[(X-c)^2]$;
2. a variância não é afetada por deslocamentos da variável (adição de constantes), mas é multiplicada pelo quadrado de uma constante multiplicativa - $Var(aX+b) = a^2Var(X)$;
3. $Var(X+Y) = Var(X) + Var(Y) + 2Cov(X,Y)$ ou $Var(X+Y) = Var(X) + Var(Y)$ se X e Y são independentes; e
4. a variância é uma medida sensível a outliers (usa a média que é afetada por outliers e ainda eleva ao quadrado o desvio das observações em relação à média).

Uma outra propriedade da variância que é base para a ANOVA refere-se a decomposição da variância de uma variável (**Lei da Variância Total**). Ela nos informa que dado uma variável X que depende de outra variável Y, podemos decompor a variância total de X como:

$Var(X) = E[Var(X|Y)] + Var(E[X|Y])$

em que o primeiro termo mede a variabilidade dentro de cada nível de Y, enquanto o segundo termo mede a variabilidade entre as médias de cada nível de Y. Assim, para amostras finitas, temos como análogo a decomposição da soma dos quadrados, representada por:

$$\sum_{i=1}^{n}(X_i - \mu)^2 = \underbrace{\sum_{j=1}^{g}\sum_{i\in G_j}(X_i - \bar{X}_j)^2}_{\text{Variabilidade dentro dos grupos}} + \underbrace{\sum_{j=1}^{g} n_j(\bar{X}_j - \mu)^2}_{\text{Variabilidade entre as médias dos grupos}}$$

em que μ é a média da variável X, $\bar{X}_j$ é a média do grupo j de Y e n_j é o tamanho do grupo j. Ou seja, isso representa a $\text{SQ}_{Total} = \text{SQ}_{Entre} + \text{SQ}_{Dentro}$ da ANOVA, o qual dividimos pelos graus de liberdade para estimar $Var_{Total} = Var_{Entre} + Var_{Dentro}$. Então, basicamente, estamos modelando a dispersão total observada na população ($Var(X)$) como a soma da heterogeneidade local ($E[Var(X|Y)]$) e da diferença sistemática entre grupos ($Var(E[X|Y])$).

Por fim, quando consideramos uma distribuição amostral $X_1, X_2, \ldots, X_n \overset{iid}{\sim} \mathcal{N}(\mu, \sigma^2)$, verifica-se que a média amostral e variância amostral apresentam as seguintes distribuições:

$\bar{X} \sim Normal\left(\mu, \frac{\sigma^2}{n}\right)$

$\frac{(n-1)s^2}{\sigma^2} \sim \chi^2_{n-1}$

A segunda distribuição (da variância) pode ser obtida da seguinte forma:

1. primeiro padronizamos a nossa distribuição ($X_i \sim Normal(\mu, \sigma^2)$), de forma que obtemos

$Z_i = \frac{X_i - \mu}{\sigma} \sim Normal(0,1)$

2. em seguida realizamos a soma dos quadrados dessa variável padronizada:

$\sum_{i=1}^{n} Z_i^2 = \sum_{i=1}^{n}\left(\frac{X_i - \mu}{\sigma}\right)^2$

assim, ela passa a ter uma distribuição qui-quadrado com n graus de liberdade:

$\sum_{i=1}^{n} Z_i^2 \sim \chi_n^2$

3. como não sabemos μ, esse parâmetro é estimado por $\bar{X}$ e precisamos corrigir a distribuição pelo fato de $\bar{X}$ depender dos X_i, então:

$\frac{1}{\sigma^2}\sum_{i=1}^{n}(X_i - \bar{X})^2 \sim \chi_{n-1}^2$

4. como $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$, podemos isolar $\sum_{i=1}^{n}(X_i - \bar{X})^2$ e substituir na relação anterior, obtendo

$\frac{(n-1)s^2}{\sigma^2} \sim \chi_{n-1}^2$

Ambas distruições (sob o prossuposto de normalidade da variável) serão fundamentais mais a frente, pois são a base para o cálculo dos intervalos de confiança e da estatística t de Student, que são base para testes de hipótese paramétricos da estatística clássica.

A obtenção da variância no R é bem simples:

```
ap2 %>%
  summarise(
    variancia = var(w_age_t),
    variancia_na_mao = sum((w_age_t - mean(w_age_t))^2) / (length(w_age_t) - 1),
  ) %>%
  kable(col.names = c("$s^2$ com `var()`", "$s^2$ calculado pela definição mat."))
```

s^2 com `var()`	s^2 calculado pela definição mat.
43.02484	43.02484

3.6.4 Desvio-padrão

Como a variância é uma medida da média dos desvios quadrados, sua interpretação pode ser difícil, então tomamos a raiz quadrada da variância (desvio padrão) para quantificar o quanto os valores de uma variável aleatória (ou amostra) se afastam, em média, da sua média.

Assim, seja X uma variável aleatória com média $\mu = E[X]$, o desvio-padrão populacional é definido como:

$\sigma = \sqrt{E[(X-\mu)^2]} = \sqrt{\mathrm{Var}(X)}$

então, para uma amostra, o desvio-padrão amostral é

$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2, \quad \text{e} \quad s = \sqrt{s^2}$

onde $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ é a média amostral. Apesar do denominador $n-1$ tornar a variância amostral um estimador não viesado da variância populacional, isso não ocorre com o desvio-padrão, uma vez que a raiz quadrada é uma operação não linear, por isso $E[s] < \sigma$. Por essa razão, principalmente em amostras pequenas, alguns manuais estatísticos se valem da relação da distribuição das variâncias amostrais ($\frac{(n-1)s^2}{\sigma^2} \sim \chi_{n-1}^2$) para obter um desvio-padrão corrigido, definido como:

$s_{\text{corr}} = \frac{s}{c_4(n)}$,

a qual deriva da relação $E[s] = c_4(n)\sigma$,

onde

$c_4(n) = \sqrt{\frac{2}{n-1}} \cdot \frac{\Gamma(n/2)}{\Gamma((n-1)/2)}$, e $\Gamma(\cdot)$ é a função Gama. De qualquer forma, esse $c_4(n)$ costuma estar tabelado em manuais estatísticos, e essa correção é particularmente útil para amostras com tamanho menor que 30, pois conforme o n aumenta $s \approx s_{\text{corr}}$.

Pelo Teorema Central do Limite, quando n é grande, temos a seguinte distribuição assintótica para o desvio-padrão amostral s:

$\sqrt{n}(s-\sigma) \xrightarrow{d} Normal\left(0, \frac{\mu_4-\sigma^4}{4\sigma^2}\right)$, em que $\mu_4 = E[(X-\mu)^4]$ é o quarto momento central de X (relacionado à curtose).

Especificamente para uma variável X normal ($X \sim Normal(\mu, \sigma^2)$) verifica-se que

$\sqrt{n}(s-\sigma) \xrightarrow{d} Normal\left(0, \frac{\sigma^2}{2}\right)$, o que demonstra que a precisão da estimativa de s em relação ao σ aumenta com $\sqrt{n}$, e que o erro padrão de s é aproximadamente $\sigma/\sqrt{2n}$.

Assim como a variância, o desvio-padrão é altamente sensível a outliers, pois utiliza o quadrado dos desvios. Por isso, em algumas aplicações práticas, estimadores robustos são preferidos, como a MAD ou sua versão escalada para estimar σ ($\hat{\sigma}_{MAD} = 1.4826 \cdot MAD$) quando é plausível pressupor que a variável em estudo é normal. Rousseeuw e Croux (1993) também propõem um estimador robusto alternativo para σ, o Q_n, o qual é definido como

$Q_n = 2.2219 \cdot \{|X_i - X_j| : i < j\}_{(k)}$,

em que $|X_i - X_j|$ representa a diferença entre cada par de observação da variável ordenada e $\{|X_i - X_j| : i < j\}_{(k)}$ significa que desse conjunto de distâncias calculadas, usaremos aquela que está na posição k, sendo

$k = \binom{h}{2} = \frac{h!}{2!(h-2)!}, \quad h = \lfloor n/2 \rfloor + 1.$

Ao multiplicar esse valor pela constante 2.2219, garante-se que, quando $X_i \sim Normal\left(\mu, \sigma^2\right)$, o estimador Q_n é aproximadamente não viesado, ou seja, $E[Q_n] \approx \sigma$. Além disso, os autores que propõem esse estimador demonstraram que ele é adequado também para distribuições assimétricas, possui um ponto de ruptura de 50% (resiste a 50% de contaminação por outliers) e possui uma eficiência em relação ao desvio-padrão na distribuição normal de 82% (precisa de aproximadamente $1/0,82 \approx 1,22$ vezes mais observações para atingir a mesma precisão do s como estimador de σ para dados de uma distribuição normal e sem outliers).

no R, podemos calcular tais estimadores da seguinte forma:

```
Qn_manual <- function(x){
  n <- length(x)
  h <- floor(n / 2) + 1
  k <- choose(h, 2)

  diffs <- abs(outer(x, x, "-")) # todas as distâncias |xi - xj|
  diffs <- diffs[lower.tri(diffs)] # pega somente i < j
  Qn <- 2.2219 * sort(diffs)[k] # aplica constante e seleciona k-ésimo menor
  return(Qn)
}
c4 <- function(n) {
  return(sqrt(2 / (n - 1)) * exp(lgamma(n / 2) - lgamma((n - 1) / 2)))
}

mad_hat <- function(x) {
  return(1.4826 * mad(x, constant = 1))
}

set.seed(123)
pop <- tibble(
  peso = rnorm(1000000, mean=25, sd=3)
```

```
)

pop %>%
  summarise(
    populacao = as.character(n()),
    media = sum(peso) / length(peso),
    dp = sqrt(sum((peso - media)^2) / length(peso)),
  ) %>%
  kable(col.names = c("Populção", "$\\mu$", "$\\sigma$"))
```

Populção	μ	σ
1000000	24.99844	2.99978

```
n_grande <- pop %>%
  slice_sample(n = 385) %>%
  summarise(
    amostra = n(),
    dp_s = sd(peso),
    dp_mad = mad_hat(peso),
    dp_corr = sd(peso) / c4(length(pop$peso)),
    qn = Qn_manual(peso)
  )

n_pequeno <- pop %>%
  slice_sample(n = 25) %>%
  summarise(
    amostra = n(),
    dp_s = sd(peso),
    dp_mad = mad_hat(peso),
    dp_corr = sd(peso) / c4(length(pop$peso)),
    qn = Qn_manual(peso)
  )


kable(bind_rows(n_grande, n_pequeno), col.names = c("$n$", "$s$ com `sd()`", "$\\hat{\\sigma}_{MAD}$", "$s
```

n	s com `sd()`	$\hat{\sigma}_{MAD}$	s_{corr}	Q_n
385	2.863386	3.066885	2.863387	2.879682
25	2.190601	2.393578	2.190602	2.385912

3.6.5 Erro-padrão

O erro padrão mede a variabilidade de um estimador amostral (por exemplo, a média amostral) em relação ao parâmetro populacional verdadeiro. Formalmente, se $\hat{\theta}$ é um estimador de um parâmetro θ, o erro padrão é definido como o desvio-padrão da distribuição amostral do estimador:

$$SE(\hat{\theta}) = \sqrt{Var(\hat{\theta})}$$

Portanto, o erro padrão não mede erro de medição individual, mas sim a incerteza associada à estimativa de um parâmetro com base em uma amostra finita.

Então para a média amostral, se temos $X_1, X_2, \ldots, X_n$ amostras independentes e identicamente distribuídas (i.i.d.), de uma população normal com média μ e variância σ^2, sabe-se que a distribuição amostral das médias amostrais ($\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$) terá:

$E[\bar{X}] = \mu, \quad Var(\bar{X}) = \frac{\sigma^2}{n}$.

Então, o erro padrão da média é:

$SE(\bar{X}) = \sqrt{Var(\bar{X})} = \frac{\sigma}{\sqrt{n}}$.

Como na prática o σ é desconhecido, usamos a variância amostral s^2:

$\widehat{SE}(\bar{X}) = \frac{s}{\sqrt{n}}$.

Como podemos perceber pela definição do erro-padrão, ele diminui conforme n aumenta, uma vez que isso implica em mais informação e menos incerteza sobre a nossa população. Assim, um SE pequeno indica estimativas mais precisas (menos dispersas entre diferentes amostras), enquanto, um SE grande indica maior variabilidade da estimativa, ou seja, a média amostral pode flutuar consideravelmente entre amostras. Um ponto importante que deve ser considerado sobre o SE é que ele pressupõe amostras i.i.d., o que basicamente significa amostragem aleatória e independência entre observações. Se houver autocorrelação (como nas séries temporais) ou amostras dependentes (como em desenhos amostrais de medidas repetidas), o erro-padrão costuma ser subestimado e precisa ser corrigido.

Na prática o SE será a base para calcularmos os intervalos de confiança e testes de hipótese relativos à média quando abordarmos os testes de hipóteses. Supondo normalidade (ou amostras grandes, pelo Teorema Central do Limite), o intervalo de confiança será calculado como:

$IC_{(1-\alpha)}(\mu) = \bar{X} \pm z_{1-\alpha/2} \times SE(\bar{X})$, ou,

$IC_{(1-\alpha)}(\mu) = \bar{X} \pm t_{1-\alpha/2,\, n-1} \times SE(\bar{X})$,

quando a amostra é pequena (menor que 30) uma vez que a distribuição t de Student atua como um fator de correção adicional para a incerteza atribuída ao estimar σ por s. Assim, o erro padrão atua como o "raio de incerteza" multiplicado pelo quantil da distribuição normal (ou t-Student), resultando na margem de erro do intervalo de confiança.

Além disso, o erro padrão também será aplicado à regressão linear na estimativa de erro padrão dos coeficientes do modelo, testes t de significância de cada coeficiente e respectivos intervalos de confiança. No modelo linear clássico, em que temos $Y = X\beta + \varepsilon$, com $\varepsilon \sim N(0, \sigma^2 I)$, ou seja, erros com média zero e homocedásticos, o estimador dos coeficientes é $\hat{\beta} = (X'X)^{-1}X'Y$. A variância amostral dos coeficientes é $Var(\hat{\beta}) = \sigma^2(X'X)^{-1}$. Então, o erro padrão de cada coeficiente $\hat{\beta}_j$ é a raiz quadrada do respectivo elemento da diagonal dessa matriz:

$SE(\hat{\beta}_j) = \sqrt{\hat{\sigma}^2[(X'X)^{-1}]_{jj}}$.

Veja como podemos obter o SE no R e como ele se comporta conforme o n aumenta:

```
# erro padrão estimado de uma amostra
sd(ap2$w_age_t) / sqrt(length(ap2$w_age_t))
```

```
[1] 0.1965246
```

```
# Demonstração da variação de SE conforme o n
set.seed(123)
```

```r
# parâmetros populacionais
mu <- 10
sigma <- 2

# tamanhos amostrais (nossos n)
n_seq <- seq(5, 200, by = 5)

# repetições para cada n (para obter uma distribuição amostral)
B <- 10000

se_emp <- numeric(length(n_seq)) # empiríco (dp das medias amostrais)
se_teo <- numeric(length(n_seq)) # teorico (sigma / raiz de n)

for (i in seq_along(n_seq)) {
  n <- n_seq[i]

  # gerar B amostras e calcular média de cada uma
  means <- replicate(B, mean(rnorm(n, mean = mu, sd = sigma)))

  # erro padrão empírico
  se_emp[i] <- sd(means)

  # erro padrão teórico
  se_teo[i] <- sigma / sqrt(n)
}
```

```r
tibble(
  n = n_seq,
  se_emp = se_emp,
  se_teo = se_teo,
) %>%
  ggplot(aes(x = n, y = se_teo)) +
  geom_line(color = "black", linewidth = 3, alpha = .3) +
  geom_point(aes(y = se_emp), color = "blue", size = 2) +
  geom_line(color = "blue", linetype = "dashed") +
  labs(
    y = "Erro Padrão da Média",
    x = "Tamanho da Amostra (n)",
    title = "Erro Padrão Empírico vs Teórico da Média"
  ) + tema
```

Figura 3.63: Variação do SE conforme o tamanho da amostra

3.6.6 Coeficiente de variação

O coeficiente de variação (CV) calcula a variação dos dados em relação à média. Quanto menor for o seu valor, em tese mais homogêneos serão os dados, ou seja, menor será a dispersão em torno da média. Pode ser calculado como:

$CV = \frac{s}{X} \times 100 \quad (\%)$

Em geral, adota-se a regra de que se $CV \leq 30\%$, então, os dados são considerados razoavelmente homogêneos, lembrando que esse valor é uma convenção e não um critério estatístico formal. Mas, como demonstrado no treinamento anterior, essa medida pode ser fortemente influenciada por transformações das variáveis, então deve-se tomar muito cuidado e não considerar que a variável possui características de uma distribuição normal baseado nessa métrica. Além disso, o CV só é válido para médias positivas e diferentes de zero, sendo um valor não confiável quando temos uma distribuição assimétrica ou com valores negativos.

3.7 Medidas de forma

Medidas de forma nos informam sobre a concentração dos dados em relação à média, ou seja, a forma da distribuição, e são conceituadas como assimetria (*skewness*) e curtose (*kurtosis*).

Para uma variável aleatória X com média $\mu = E[X]$ e desvio padrão $\sigma = \sqrt{Var(X)}$, o um momento central de ordem k é definido como:

$\mu_k = E[(X - \mu)^k]$, e

o momento padronizado é

$\gamma_k = \frac{E[(X-\mu)^k]}{\sigma^k}$.

Para esses momentos, define-se γ_1 como grau de assimetria e γ_2 como grau de curtose.

3.7.1 Assimetria

A assimetria populacional é definida como:

$\gamma_1 = \frac{\mu_3}{\sigma^3} = \frac{E[(X-\mu)^3]}{\sigma^3}$.

γ_1 é uma medida adimensional e expressa o comportamento da cauda da distribuição, de forma que:

- $\gamma_1 > 0$ indica uma assimetria positiva, ou seja, uma cauda direita alongada,
- $\gamma_1 < 0$ indica assimetria negativa, ou seja, uma cauda esquerda alongada, e
- $\gamma_1 = 0$ representa uma distribuição perfeitamente simétrica.

O termo $(X - \mu)^3$ da definição da assimetria amplifica diferenças ao redor da média, assim, valores maiores que μ contribuem positivamente para o valor final, enquanto valores menores que μ contribuem negativamente. Já o uso da terceira potência elimina a simetria que a variância teria, tornando o sinal *informativo* da direção para o qual ocorre a assimetria.

A magnitude de $|\gamma_1|$ é usada para avaliar a intensidade da assimetria:

- leve se $|\gamma_1| < 0,5$
- moderada se $0,5 \leq |\gamma_1| \leq 1$
- forte se $|\gamma_1| > 1$

Na prática estimamos a assimetria como

$g_1' = \frac{m_3}{m_2^{3/2}}$,

onde

$m_k = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^k$.

Ou utilizamos o estimador corregido (Fisher), não viesado assintoticamente, o qual é definido como:

$g_1 = \frac{n}{(n-1)(n-2)}\sum_{i=1}^{n}\left(\frac{X_i-\bar{X}}{s}\right)^3$.

Além dessa avaliação pela magnitude da assimetria, existe um teste de hipótese (teste de assimetria) para identificar se existem evidências de assimetria da distribuição. Nele consideramos que a hipótese nula (H_0) é que a distribuição populacional é simétrica ao redor de um ponto (normalmente a média), de modo que nossa hipótese alternativa é que a distribuição não é simétrica ao redor desse ponto.

Se uma variável X é simétrica em torno de um ponto c, então:

$f_X(c + x) = f_X(c - x), \quad \forall x \in \mathbb{R}$,

onde $f_X(x)$ é a função densidade (ou de probabilidade, se discreta). Ou seja, o lado direito à c da distruição é um espelho do lado esquerodo, de modo que $P(X > c + d) = P(X < c - d)$. Esse é exatamente o comportamento da distribuição normal para o $c = \mu$. Então, para realizar o teste considerando uma distribuição simétrica em torno da média (distribuição normal), calculamos o g_1 corrigido de Fisher e sob a condição de H_0 esperamos:

$E[g_1] = 0, \quad Var(g_1) \approx \frac{6}{n}$,

portanto, a estatística padronizada de g_1 é:

$Z = \frac{g_1}{\sqrt{6/n}}$.

Para $n \geq 50$, $Z \sim Normal(0,1)$, então se $|Z| > z_{\alpha/2}$ (por exemplo, $z_{0.025} = 1.96$ para 5%), rejeitamos a hipótese nula e consideramos que há evidências de assimetria da distribuição populacional.

No R a medida de assimetria pode ser obtida pela função `skewness()` do pacote `moments`:

```
instala_se_nao_existe("moments")
```

```
NULL
```

```
require(moments)

# assimetria estimada de uma amostra
skewness(ap2$w_age_t)
```

```
[1] 0.2779448
```

```
# Simulação de distribuições com diferentes formas
set.seed(123)
n <- 100000

# Assimetria negativa: distribuição Beta(5, 2)
assim_neg <- rbeta(n, 5, 2)

# Assimetria praticamente nula: Normal(0,1)
assim_neutra <- rnorm(n, 0, 1)

# Assimetria positiva: Qui-quadrado(3)
assim_pos <- rchisq(n, df = 3)

dados_assim <- tibble(
  valor = c(
    assim_neg,
    assim_neutra,
    assim_pos
  ),
  tipo = factor(
    rep(c("Negativa", "Simétrica", "Positiva"), each = n),
    levels = c("Negativa", "Simétrica", "Positiva")
  ),
  dist = factor(
    rep(c("Beta(5, 2)", "Normal(0,1)", "Qui-quadrado(3)"), each = n),
    levels = c("Beta(5, 2)", "Normal(0,1)", "Qui-quadrado(3)")
  )
)

res_assim <- dados_assim %>%
  group_by(tipo) %>%
  summarise(across(valor, function(x) round(skewness(x), 2)))

res_assim
```

```
# A tibble: 3 x 2
  tipo      valor
```

```
  <fct>     <dbl>
1 Negativa  -0.6
2 Simétrica  0
3 Positiva   1.63
```

```r
dados_assim %>%
  ggplot(aes(x = valor, fill = tipo)) +
  geom_density(alpha = 0.5) +
  facet_wrap(~dist, scales = "free", ncol = 1) +
  labs(
    title = "Comparação de Assimetria",
    subtitle = "Distribuições com diferentes graus e direções de skewness",
    x = "Valor",
    y = "Densidade"
  ) +
  tema +
  theme(legend.position = "none")
```

Figura 3.64: Comparação dos diferentes tipos de assimetria

3.7.2 Curtose

A curtose mede quanto peso está nas caudas e no centro, o que graficamente representa o grau de achatamento de uma distribuição de frequências, o que indica a concentração das probabilidades ao redor da média. A curtose populacional é definida como:

$\gamma_2 = \frac{\mu_4}{\sigma^4} - 3 = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] - 3.$

A subtração de 3 torna $\gamma_2 = 0$ para uma distribuição Normal (padrão de referência). Assim, para:

- $\gamma_2 > 0$ temos uma distribuição leptocúrtica, de caudas pesadas, com grande concentração central e caudas longas,
- $\gamma_2 = 0$ temos uma distribuição mesocúrtica de forma semelhante à normal, e, para
- $\gamma_2 < 0$ temos uma distribuição platicúrtica, de caudas leves, uma distribuição mais achatada e com menor concentração central.

Em relação aos estimadores de curtose, temos a versão não corrigida definida como:

$g_2' = \frac{m_4}{m_2^2} - 3$,

e a versão corrigida de Fisher:

$g_2 = \frac{n(n+1)}{(n-1)(n-2)(n-3)} \sum \left(\frac{X_i - \bar{X}}{s}\right)^4 - \frac{3(n-1)^2}{(n-2)(n-3)}$.

Assim como na assimetria, temos um teste de hipótese para a curtose normal. Nela definimos a hipótese nula (H_0) de que $\gamma_2 = 0$, de modo que, sob a condição de uma distribuição normal, temos uma estatística padronizada na forma:

$Z = \frac{g_2}{\sqrt{24/n}} \sim N(0,1)$.

Assim, se $|Z| > z_{\alpha/2}$, rejeitamos a hipótese nula e consideramos que há evidências de que a curtose da distribuição em estudo não é de uma normal ($\gamma_2 \neq 0$).

No R a medida de assimetria pode ser obtida pela função `kurtosis()` do pacote `moments`:

```
# curtose estimada de uma amostra
kurtosis(ap2$w_age_t)
```

```
[1] 2.449314
```

```
# Simulação de distribuições com diferentes formas
set.seed(123)
n <- 100000

# Curtose platicúrtica: Uniforme(0,1)
curt_plat <- runif(n, 0, 1)

# Curtose mesocúrtica: Normal(0,1)
curt_meso <- rnorm(n, 0, 1)

# Curtose leptocúrtica: t-Student com poucos graus de liberdade (caudas pesadas)
curt_lepto <- rt(n, df = 3)

dados_curt <- tibble(
  valor = c(
    curt_plat,
    curt_meso,
    curt_lepto
  ),
  tipo = factor(
    rep(c("Platicúrtica", "Mesocúrtica", "Leptocúrtica"), each = n),
    levels = c("Platicúrtica", "Mesocúrtica", "Leptocúrtica")
  ),
  dist = factor(
    rep(c("Uniforme(0,1)", "Normal(0,1)", "t(3)"), each = n),
```

```r
    levels = c("Uniforme(0,1)", "Normal(0,1)", "t(3)")
  )
)

res_curt <- dados_curt %>%
  group_by(tipo) %>%
  summarise(across(valor, function(x) kurtosis(x) - 3))

res_curt
```

```
# A tibble: 3 x 2
  tipo            valor
  <fct>           <dbl>
1 Platicúrtica  -1.20
2 Mesocúrtica    0.0161
3 Leptocúrtica 105.
```

```r
dados_curt %>%
  ggplot(aes(x = valor, fill = tipo)) +
  geom_density(alpha = 0.5) +
  facet_wrap(~dist, scales = "free", ncol = 1) +
  labs(
    title = "Comparação de Curtose",
    subtitle = "Distribuições com caudas leves (platicúrtica), normais e pesadas (leptocúrtica)",
    x = "Valor",
    y = "Densidade"
  ) +
  tema +
  theme(legend.position = "none")
```

Figura 3.65: Comparação dos diferentes tipos de assimetria

3.7.3 Teste combinado de normalidade para assimetria e curtose

Além dos testes em separado para assimetria e curtose, existe um teste que utiliza das duas métricas denominado teste de Jarque-Bera (JB). Nesse teste, definimos as seguintes hipóteses:

H_0: A amostra provém de uma distribuição Normal ($N(\mu, \sigma^2)$), H_1: A amostra não é Normal (assimétrica ou com curtose diferente da normal).

A ideia é que se a distribuição subjacente da amostra é normal os coeficientes de assimetria e curtose serão iguais a zero. Assim, o teste Jarque–Bera mede justamente o quanto as estimativas amostrais desses coeficientes se afastam dos valores teóricos esperados sob a normalidade. Assim, calculados os estimadores g_1 (assimetria amostral) e g_2 (curtose amostral), definimos a estatística JB como:

$JB = n\left(\frac{g_1^2}{6} + \frac{g_2^2}{24}\right)$,

em que n é o tamanho amostral. Sob a hipótese nula e para amostras grandes (teorema central do limite),

$JB \sim \chi^2_2$, ou seja, essa medida segue uma distribuição qui-quadrado com 2 graus de liberdade (devido aos dois parâmetros analisados, simetria e curtose). Se $JB > \chi^2_{(1-\alpha,2)}$ rejeitamos H_0, ou seja, há evidências de que a distribuição não é normal.

No R, o pacote `tseries` de análise de séries temporais implementa esse teste com a função `jarque.bera.test()` e retorna um p-valor do teste (nesse caso rejeitamos H_0 se $p < 0,05$):

```
instala_se_nao_existe("tseries")
```

```
NULL
```

```
set.seed(123)
x1 <- rnorm(200, 0, 1)
```

```r
x2 <- rexp(200, 1)

test_norm <- tseries::jarque.bera.test(x1)
test_exp <- tseries::jarque.bera.test(x2)

tibble(
  dist = c("Normal(0,1)", "Exponencial(1)"),
  jb = c(
    test_norm$statistic,
    test_exp$statistic
  ),
  p = c(
    test_norm$p.value,
    test_exp$p.value
  )
) %>%
  kable(col.names = c("Distribuição", "JB", "p-valor"))
```

Distribuição	JB	p-valor
Normal(0,1)	4.334818	0.1144739
Exponencial(1)	89.138360	0.0000000

3.7.4 Desigualdade Fundamental de Pearson

A Desigualdade Fundamental de Pearson (ou relação entre momentos de terceira e quarta ordem) é uma propriedade matemática que impõe limites entre a assimetria e a curtose de qualquer distribuição cujos momentos sejam finitos. Ela garante que assimetria e curtose são vinculados por restrições derivadas da positividade da variância e da forma da distribuição.

Seja X uma variável aleatória, seus momentos padronizados serão definidos como $\beta_1 = \frac{\mu_3^2}{\mu_2^3}$ e $\beta_2 = \frac{\mu_4}{\mu_2^2}$. Pearson demonstrou que, para qualquer distribuição real com momentos até a quarta ordem finitos, $\beta_2 \geq \beta_1 + 1$.

Em termos de assimetria e curtose, temos:

$\gamma_2 = \beta_2 - 3, \qquad \gamma_1 = \frac{\mu_3}{\sigma^3}$, então,

substituindo na desigualdade anterior, obtemos $\gamma_2 \geq \gamma_1^2 - 2$.

Essa desigualdade define uma região viável em relação ao γ_1 e γ_2 para o qual a combinação desses valores corresponde a uma distribuição real, de modo que, por exemplo, nenhuma distribuição real pode ter $\gamma_2 < -2$. Isso implica que, para uma distribuição real, quanto mais assimetria ($|\gamma_1|$ grande) maior deve ser a curtose (caudas mais pesadas ou pico mais proeminente), portanto, não existem distribuições muito assimétricas e que sejam também fortemente achatadas.

```r
set.seed(123)

dists <- tibble(
  nome = c("Normal", "Uniforme", "Exponencial", "Chi-quadrado(3)"),
  dados = list(rnorm(1e5), runif(1e5), rexp(1e5), rchisq(1e5, df=3))
) %>%
```

```r
  mutate(
    skew = map_dbl(dados, skewness),
    kurt = map_dbl(dados, kurtosis) - 3
  )

# Curva do limite teórico
gamma1_grid <- seq(-4, 4, by = 0.1)
limite <- tibble(
  gamma1 = gamma1_grid,
  gamma2 = gamma1_grid^2 - 2
)

ggplot() +
  geom_line(
    data = limite,
    aes(x = gamma1, y = gamma2),
    color = "red", size = 1.1, linetype = "dashed"
  ) +
  geom_point(
    data = dists,
    aes(x = skew, y = kurt, color = nome),
    size = 4
  ) +
  labs(
    title = "Desigualdade Fundamental de Pearson",
    subtitle = expression(gamma[2] >= gamma[1]^2 - 2),
    x = expression("Assimetria ( " ~ gamma[1] ~ " )"),
    y = expression("Excesso de Curtose ( " ~ gamma[2] ~ " )")
  ) +
  tema
```

Figura 3.66: Posição da assimetria e curtose de distribuições reais em relação à desigualdade de Pearson

Hyndman, Rob J., e Yanan Fan. 1996. "Sample Quantiles in Statistical Packages". *The American Statistician* 50 (4): 361–65. https://doi.org/10.2307/2684934.

Rousseeuw, Peter J., e Christophe Croux. 1993. "Alternatives to the Median Absolute Deviation". *Journal of the American Statistical Association* 88 (424): 1273–83. https://doi.org/10.2307/2291267.