Café com estatística e R

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

Marcelo Teixeira Paiva

2025-10-08

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

Lista de Figuras

Lista de Tabelas

```r
# Pacotes
library(tidyverse)
library(gridExtra)
library(plotly)
library(gt)
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.

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

```
[1] 1270
```

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

```
[1] 297
```

```
# 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.

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

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

```
[22] "Autoloads"          "package:base"
```

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

```
      abind      ARTool     askpass   backports
    "abind"    "ARTool"   "askpass" "backports"
```

```
length(instalados)
```

```
[1] 330
```

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

```
[1] 1
```

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

```
[1] TRUE
```

```
"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.

```
# 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.

```
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
update.packages(ask = FALSE)  # todos, mas exige confirmação

# 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 4.3.3; ; 2025-10-07 18:16:38 UTC; unix

-- File: /home/marcelo/R/x86_64-pc-linux-gnu-library/4.3/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.

A BibTeX entry for LaTeX users is

  @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)
}
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.

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

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

```
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)
```

```
Dimensões: 1114 21
Tipos de variáveis:
$farm_id
[1] "numeric"

$batch_id
[1] "numeric"

$litt_id
[1] "numeric"

$pig_id
[1] "numeric"

$parity
```

```
[1] "numeric"

$vacc_mp
[1] "haven_labelled" "vctrs_vctr"     "double"

$seas_fin
[1] "haven_labelled" "vctrs_vctr"     "double"

$age_t
[1] "numeric"

$w_age_t
[1] "numeric"

$age_t6
[1] "numeric"

$w_age_t6
[1] "numeric"

$dwg_fin
[1] "numeric"

$ap2_t
[1] "haven_labelled" "vctrs_vctr"     "double"

$mp_t
[1] "haven_labelled" "vctrs_vctr"     "double"

$infl_t
[1] "haven_labelled" "vctrs_vctr"     "double"

$prrs_t
[1] "haven_labelled" "vctrs_vctr"     "double"

$ap2_t6
[1] "haven_labelled" "vctrs_vctr"     "double"

$mp_t6
[1] "haven_labelled" "vctrs_vctr"     "double"

$infl_t6
[1] "haven_labelled" "vctrs_vctr"     "double"

$prrs_t6
[1] "haven_labelled" "vctrs_vctr"     "double"

$ap2_sc
[1] "haven_labelled" "vctrs_vctr"     "double"
```

```
Primeiras linhas:
# A tibble: 3 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
# 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>

Resumo estatístico:
    farm_id         batch_id       litt_id          pig_id
 Min.   :1.000   Min.   : 1.0   Min.   :  1.0   Min.   :   1.0
 1st Qu.:2.000   1st Qu.:11.0   1st Qu.:124.0   1st Qu.: 371.2
 Median :3.000   Median :22.0   Median :256.5   Median : 766.5
 Mean   :3.273   Mean   :21.6   Mean   :252.2   Mean   : 752.9
 3rd Qu.:5.000   3rd Qu.:33.0   3rd Qu.:388.0   3rd Qu.:1156.8
 Max.   :6.000   Max.   :41.0   Max.   :491.0   Max.   :1466.0

     parity          vacc_mp          seas_fin         age_t
 Min.   : 1.000   Min.   :0.0000   Min.   :0.000   Min.   :52.00
 1st Qu.: 2.000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:62.00
 Median : 4.000   Median :1.0000   Median :0.000   Median :68.00
 Mean   : 3.769   Mean   :0.6239   Mean   :0.342   Mean   :68.91
 3rd Qu.: 5.000   3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:76.00
 Max.   :11.000   Max.   :1.0000   Max.   :1.000   Max.   :83.00

    w_age_t          age_t6         w_age_t6         dwg_fin
 Min.   :12.10   Min.   :101.0   Min.   : 30.00   Min.   : 231.0
 1st Qu.:22.30   1st Qu.:117.0   1st Qu.: 61.00   1st Qu.: 647.5
 Median :26.80   Median :124.0   Median : 70.00   Median : 758.5
 Mean   :27.43   Mean   :125.6   Mean   : 69.97   Mean   : 757.5
 3rd Qu.:32.10   3rd Qu.:131.0   3rd Qu.: 78.00   3rd Qu.: 872.8
 Max.   :47.20   Max.   :161.0   Max.   :107.00   Max.   :1188.0

     ap2_t            mp_t            infl_t          prrs_t
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
 Median :0.0000   Median :0.0000   Median :0.0000   Median :0.0000
 Mean   :0.1059   Mean   :0.2738   Mean   :0.1795   Mean   :0.2127
 3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.0000
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000

     ap2_t6           mp_t6           infl_t6          prrs_t6
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
 Median :1.0000   Median :0.0000   Median :0.0000   Median :0.0000
 Mean   :0.5566   Mean   :0.3293   Mean   :0.4668   Mean   :0.4102
 3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

```
                NA's   :33
    ap2_sc
 Min.   :0.00
 1st Qu.:0.00
 Median :1.00
 Mean   :0.51
 3rd Qu.:1.00
 Max.   :1.00
 NA's   :118

Valores missing por coluna:
 farm_id batch_id  litt_id   pig_id   parity  vacc_mp seas_fin    age_t
       0        0        0        0        0        0        0        0
 w_age_t   age_t6 w_age_t6  dwg_fin    ap2_t     mp_t   infl_t   prrs_t
       0        0        0        0        0        0        0        0
  ap2_t6    mp_t6  infl_t6  prrs_t6   ap2_sc
       0       33        0        0      118

Structura dos dados:
tibble [1,114 x 21] (S3: tbl_df/tbl/data.frame)
 $ farm_id : num [1:1114] 1 1 1 1 1 1 1 1 1 1 ...
  ..- attr(*, "label")= chr "farm identification"
  ..- attr(*, "format.stata")= chr "%5.0f"
 $ batch_id: num [1:1114] 1 1 1 1 1 1 1 1 1 1 ...
  ..- attr(*, "label")= chr "batch identifiaction number"
  ..- attr(*, "format.stata")= chr "%5.0f"
 $ litt_id : num [1:1114] 1 1 1 2 2 2 3 3 3 4 ...
  ..- attr(*, "label")= chr "litter identification number"
  ..- attr(*, "format.stata")= chr "%5.0f"
 $ pig_id  : num [1:1114] 1 2 3 4 5 6 7 8 9 10 ...
  ..- attr(*, "label")= chr "pig identification"
  ..- attr(*, "format.stata")= chr "%5.0f"
 $ parity  : num [1:1114] 8 8 8 8 8 8 6 6 6 6 ...
  ..- attr(*, "label")= chr "the farrowing no. of the sow"
  ..- attr(*, "format.stata")= chr "%3.0f"
 $ vacc_mp : dbl+lbl [1:1114] 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
   ..@ label       : chr "the batch vaccinated against M.hyop yes=1"
   ..@ format.stata: chr "%8.0f"
   ..@ labels      : Named num [1:2] 0 1
   .. ..- attr(*, "names")= chr [1:2] "not vac." "vac"
 $ seas_fin: dbl+lbl [1:1114] 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
   ..@ label       : chr "prod. season in finishing unit: winther=1"
   ..@ format.stata: chr "%6.0f"
   ..@ labels      : Named num [1:2] 0 1
   .. ..- attr(*, "names")= chr [1:2] "summer" "winter"
 $ age_t   : num [1:1114] 70 70 70 60 60 60 67 67 67 61 ...
  ..- attr(*, "label")= chr "pig-age transfer from weaning to finishing unit"
  ..- attr(*, "format.stata")= chr "%5.0f"
 $ w_age_t : num [1:1114] 33.8 32.9 29.4 19.8 20.4 20.3 21 32.4 30.3 22.5 ...
  ..- attr(*, "label")= chr "weight in kg. at age_t"
  ..- attr(*, "format.stata")= chr "%5.1f"
```

```
$ age_t6  : num [1:1114] 116 116 116 106 106 106 113 113 113 107 ...
 ..- attr(*, "label")= chr "age_tra plus approx. 6 weeks"
 ..- attr(*, "format.stata")= chr "%5.0f"
$ w_age_t6: num [1:1114] 80 80 81 54 64 63 59 79 72 66 ...
 ..- attr(*, "label")= chr "weight in kg. at age_t6"
 ..- attr(*, "format.stata")= chr "%5.1f"
$ dwg_fin : num [1:1114] 1004 1024 1122 743 948 ...
 ..- attr(*, "label")= chr "dwg in g. between age_t and age_t6"
 ..- attr(*, "format.stata")= chr "%5.0f"
$ ap2_t   : dbl+lbl [1:1114] 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
  ..@ label       : chr "serological reac. against A.pleuropneumoniae serotype 2 at age_t"
  ..@ format.stata: chr "%3.0f"
  ..@ labels      : Named num [1:2] 0 1
  .. ..- attr(*, "names")= chr [1:2] "neg" "pos"
$ mp_t    : dbl+lbl [1:1114] 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0,...
  ..@ label       : chr "serological reac. against M.hyopneumoniae at age_t"
  ..@ format.stata: chr "%3.0f"
  ..@ labels      : Named num [1:2] 0 1
  .. ..- attr(*, "names")= chr [1:2] "neg" "pos"
$ infl_t  : dbl+lbl [1:1114] 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
  ..@ label       : chr "serological reac. against Influenza virus at age_t"
  ..@ format.stata: chr "%3.0f"
  ..@ labels      : Named num [1:2] 0 1
  .. ..- attr(*, "names")= chr [1:2] "neg" "pos"
$ prrs_t  : dbl+lbl [1:1114] 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,...
  ..@ label       : chr "serological reac. against PRRS virus at age_t"
  ..@ format.stata: chr "%3.0f"
  ..@ labels      : Named num [1:2] 0 1
  .. ..- attr(*, "names")= chr [1:2] "neg" "pos"
$ ap2_t6  : dbl+lbl [1:1114] 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
  ..@ label       : chr "serological reac. against A.pleuropneumoniae serotype 2 at age_t6"
  ..@ format.stata: chr "%3.0f"
  ..@ labels      : Named num [1:2] 0 1
  .. ..- attr(*, "names")= chr [1:2] "neg" "pos"
$ mp_t6   : dbl+lbl [1:1114] 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0,...
  ..@ label       : chr "serological reac. against M.hyopneumoniae at age_t6"
  ..@ format.stata: chr "%3.0f"
  ..@ labels      : Named num [1:2] 0 1
  .. ..- attr(*, "names")= chr [1:2] "neg" "pos"
$ infl_t6 : dbl+lbl [1:1114] 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,...
  ..@ label       : chr "serological reac. against Influenza virus at age_t6"
  ..@ format.stata: chr "%3.0f"
  ..@ labels      : Named num [1:2] 0 1
  .. ..- attr(*, "names")= chr [1:2] "neg" "pos"
$ prrs_t6 : dbl+lbl [1:1114] 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
  ..@ label       : chr "serological reac. against PRRS virus at age_t6"
  ..@ format.stata: chr "%3.0f"
  ..@ labels      : Named num [1:2] 0 1
  .. ..- attr(*, "names")= chr [1:2] "neg" "pos"
$ ap2_sc  : dbl+lbl [1:1114] 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
  ..@ label       : chr "seroconversion to ap2 during the finishing period"
```

```
  ..@ format.stata: chr "%3.0f"
  ..@ labels      : Named num [1:2] 0 1
  .. ..- attr(*, "names")= chr [1:2] "neg" "pos"
 - attr(*, "notes")= chr [1:2] "5 Aug 2002 16:24 data provided by Dr. Haakan Vigre, Denmark" "1"
```

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.

```r
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()`
SAS	`haven`	`read_sas()`
SPSS	`haven`	`read_sav()`
Stata	`haven`	`read_stata()`, `read_dta()`
Múltiplos	`rio`	`import()`

Chapter 2

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. Entretanto, antes de obter tais estatísticas precisamos aprender como manipular os dados no R.

2.1 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:

```r
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")`.

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

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

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

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

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

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

2.1.1 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.

```r
# Sem pipe, com funções aninhadas
resultado <- summarise(
  group_by(
```

```
    filter(dados_tidy, peso_kg > 10),
    especie
  ),
  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))
```

2.1.2 Manipulação dos dados com o `dplyr`

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

```
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)
)
```

```
# 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
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.

```
# 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.

```
# 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.

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

# Ordenação decrescente por uma variável
df_exemplo %>%
```

```r
  arrange(desc(peso))

# 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)
```

2.1.3 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 <- 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 %>%
  separate(
    info,
```

```
    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 = ",")
```

2.1.4 5.3 Preenchimento de Valores Faltantes

```
# Criar dados com NAs
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() - preenche NAs com valores anteriores/posteriores
dados_na %>%
  fill(temperatura, .direction = "down")  # Preenche para baixo
```

```
# A tibble: 7 x 3
    dia temperatura medicacao
  <int>       <dbl> <chr>
1     1        38.5 A
2     2        38.5 <NA>
3     3        38.7 <NA>
4     4        38.7 B
5     5        38.7 <NA>
6     6        39   <NA>
7     7        38.6 C
```

```
dados_na %>%
  fill(everything(), .direction = "up")   # Preenche para cima
```

```
# A tibble: 7 x 3
    dia temperatura medicacao
  <int>       <dbl> <chr>
1     1        38.5 A
2     2        38.7 B
3     3        38.7 B
```

```
4     4        39   B
5     5        39   C
6     6        39   C
7     7        38.6 C
```

```
# replace_na() - substitui NAs por valor específico
dados_na %>%
  replace_na(list(
    temperatura = 38.5,
    medicacao = "Nenhuma"
  ))
```

```
# A tibble: 7 x 3
    dia temperatura medicacao
  <int>       <dbl> <chr>
1     1        38.5 A
2     2        38.5 Nenhuma
3     3        38.7 Nenhuma
4     4        38.5 B
5     5        38.5 Nenhuma
6     6        39   Nenhuma
7     7        38.6 C
```

```
# drop_na() - remove linhas com NA
dados_na %>%
  drop_na()  # Remove qualquer linha com NA
```

```
# A tibble: 2 x 3
    dia temperatura medicacao
  <int>       <dbl> <chr>
1     1        38.5 A
2     7        38.6 C
```

```
dados_na %>%
  drop_na(temperatura)  # Remove apenas se temperatura é NA
```

```
# A tibble: 4 x 3
    dia temperatura medicacao
  <int>       <dbl> <chr>
1     1        38.5 A
2     3        38.7 <NA>
3     6        39   <NA>
4     7        38.6 C
```

2.1.5 5.4 Aninhamento (Nesting)

```
# Criar dados aninhados
dados_aninhados <- df_exemplo %>%
  group_by(especie) %>%
  nest() %>%
  mutate(
    n_observacoes = map_int(data, nrow),
```

```
    modelo = map(data, ~lm(peso ~ idade, data = .x))
  )

print(dados_aninhados)

# Desaninhar
dados_desaninhados <- dados_aninhados %>%
  select(-modelo) %>%
  unnest(data)
```

2.2 6. JOINS: COMBINANDO TABELAS

2.2.1 6.1 Mutating Joins

```
# Criar tabelas exemplo
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),  # Note: id 6 não existe em 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")
)

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

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

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

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

2.2.2 6.2 Filtering Joins

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

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

2.2.3 6.3 Joins Complexos

```
# Join com múltiplas chaves
proprietarios <- tibble(
  id_prop = 1:3,
  nome_prop = c("João", "Maria", "Pedro"),
  telefone = c("1234", "5678", "9012")
)

animais_prop <- tibble(
  id_animal = 1:5,
  nome_animal = c("Rex", "Mia", "Bob", "Luna", "Max"),
  id_proprietario = c(1, 2, 1, 3, 2)
)

# Join triplo
resultado_completo <- animais_prop %>%
  left_join(proprietarios, by = c("id_proprietario" = "id_prop")) %>%
  left_join(consultas, by = c("id_animal" = "id_animal"))

print(resultado_completo)

# Join com condições diferentes
# Usando joining by different names
exames <- tibble(
  codigo_animal = 1:4,
  tipo_exame = c("sangue", "urina", "raio-x", "sangue"),
  resultado = c("normal", "alterado", "normal", "normal")
)

animais %>%
  left_join(exames, by = c("id" = "codigo_animal"))
```

2.3 7. PURRR: PROGRAMAÇÃO FUNCIONAL

2.3.1 7.1 Família map()

```
# map() retorna lista
lista_modelos <- df_exemplo %>%
  split(.$especie) %>%
  map(~lm(peso ~ idade, data = .x))

# map_dbl() retorna vetor numérico
r_squared <- lista_modelos %>%
  map_dbl(~summary(.x)$r.squared)

print(r_squared)

# map_df() retorna data frame
coeficientes <- lista_modelos %>%
  map_df(~broom::tidy(.x), .id = "especie")

print(coeficientes)

# map2() - itera sobre 2 vetores
x <- list(1:3, 4:6, 7:9)
y <- list(10, 20, 30)

resultado <- map2(x, y, ~.x + .y)
print(resultado)

# pmap() - itera sobre múltiplos vetores
params <- tibble(
  mean = c(10, 20, 30),
  sd = c(1, 2, 3),
  n = c(100, 100, 100)
)

amostras <- pmap(params, rnorm)
```

2.3.2 7.2 Operações Iterativas

```
# walk() - para efeitos colaterais (não retorna resultado)
plots <- df_exemplo %>%
  split(.$especie) %>%
  map(~ggplot(.x, aes(idade, peso)) +
        geom_point() +
        geom_smooth(method = "lm"))

# walk(plots, print)  # Imprimiria cada gráfico

# reduce() - reduz lista a valor único
numeros <- list(1:3, 4:6, 7:9)
```

```r
reduce(numeros, c)  # Concatena todos

# accumulate() - como reduce mas mantém resultados intermediários
accumulate(1:5, `+`)  # Soma cumulativa
```

2.4 8. STRINGR: MANIPULAÇÃO DE TEXTO

```r
# Dados exemplo
diagnosticos <- 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 <- diagnosticos %>%
  mutate(
    # Remover espaços extras
    descricao_limpa = str_trim(descricao),

    # Converter para minúsculas
    descricao_lower = str_to_lower(descricao_limpa),

    # Converter para título
    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+")
  )

print(diagnosticos_limpos)

# Expressões regulares
telefones <- c("(11) 1234-5678", "11-98765.4321", "1112345678", "11 1234 5678")
```

```r
# Padronizar telefones
telefones_limpos <- telefones %>%
  str_remove_all("[^0-9]") %>%  # Remove tudo exceto números
  str_replace("^(\\d{2})(\\d{4,5})(\\d{4})$", "(\\1) \\2-\\3")

print(telefones_limpos)
```

2.5 9. LUBRIDATE: MANIPULAÇÃO DE DATAS

```r
library(lubridate)

# Criar e converter datas
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_ymd = ymd("2023-12-25")
)

# Extrair componentes
consultas_datas <- df_exemplo %>%
  mutate(
    ano = year(data_consulta),
    mes = month(data_consulta, label = TRUE),
    dia = day(data_consulta),
    dia_semana = wday(data_consulta, label = TRUE),
    trimestre = quarter(data_consulta),
    dia_ano = yday(data_consulta)
  )

# Aritmética de datas
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(duracao_dias / 2),

    # Intervalos e períodos
    intervalo = interval(inicio, fim),
    periodo = as.period(intervalo),
    duracao_periodo = as.duration(intervalo)
  )

print(intervencoes)
```

```r
# 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"),
  dia = floor_date(agora, "day"),
  semana = floor_date(agora, "week"),
  mes = floor_date(agora, "month")
)
```

2.6 10. FORCATS: MANIPULAÇÃO DE FATORES

```r
library(forcats)

# Criar dados com fatores
dados_fatores <- df_exemplo %>%
  mutate(
    especie_fator = factor(especie),

    # Reordenar níveis por frequência
    especie_freq = fct_infreq(especie_fator),

    # Reordenar por outra variável
    especie_peso = fct_reorder(especie_fator, peso, mean),

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

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

# Visualizar importância dos fatores
dados_fatores %>%
  count(especie_freq) %>%
  ggplot(aes(x = especie_freq, y = n)) +
  geom_col() +
  labs(title = "Frequência de Espécies (ordenado)")
```

- Estatística descritiva univariada

  - Tabela de distribuição de frequências
    - Tabela de distribuição de frequências para variáveis qualitativas
    - Tabela de distribuição de frequências para dados discretos
    - Tabela de distribuição de frequências para dados contínuos agrupados em classes
  - Representação gráfica dos dados
  - Representação gráfica para variáveis qualitativas
    - Gráfico de barras
    - Gráfico de setores ou pizza
    - Diagrama de Pareto
  - Representação gráfica para variáveis quantitativas
    - Gráfico de linhas
    - Gráfico de pontos ou dispersão
    - Histograma
    - Gráfico de ramo-e-folhas
    - Boxplot ou diagrama de caixa
  - Medidas-resumo
    - Medidas de posição ou localização
      - Medidas de tendência central
      - Medidas separatrizes
      - Identificação de existência de outliers univariados
    - Medidas de dispersão ou variabilidade
      - Amplitude
      - Desvio-médio
      - Variância
      - Desvio-padrão
      - Erro-padrão
      - Coeficiente de variação
  - Medidas de forma
    - Medidas de assimetria
    - Medidas de curtose
- Estatística descritiva bivariada
  - Associação entre duas variáveis qualitativas
    - Tabelas de distribuição conjunta de frequências
  - Medidas de associação
    - Estatística qui-quadrado
    - Outras medidas de associação baseadas no qui-quadrado
    - coeficiente de Spearman
  - Correlação entre duas variáveis quantitativas
    - Tabelas de distribuição conjunta de frequências
    - Representação gráfica por meio de um diagrama de dispersão
  - Medidas de correlação
    - Covariância
    - Coeficiente de correlação de Pearson