Ciência de Dados Aplicada à Saúde Materno-Infantil

Observatório Obstétrico Brasileiro

10/10/2023

Índice

Prefácio

Sumário

1 Introdução

1.1 Bases de dados

Os dados considerados nas aplicações deste livro são provenientes do Sistema de Informação da Vigilância Epidemiológica da Gripe (SIVEP-Gripe), sistema oficial para o registro dos casos e óbitos por Síndrome Respiratória Aguda Grave (SRAG) disponibilizado pelo Ministério da Saúde. Os dados correspondem a registros de gestantes e puérperas de 10 a 55 anos hospitalizadas com SRAG por COVID-19 confirmada por teste de PCR. O conjunto de dados é utilizado para ilustrar e demonstrar diversos aspectos dos conceitos abordados no texto, e pode ser baixado em https://github.com/observatorioobstetrico/dados_livro_cd_saude.

1.1.1 Dados de COVID-19 em gestantes e puérperas

Essa base consiste em 11.523 registros de gestantes e puérperas diagnosticadas com COVID-19 no período de março de 2020 a dezembro de 2021. Alguns estudos conduzidos pelo OOBr usaram esses dados, dentre os quais podem ser citados: Características demográficas e epidemiológicas sobre mulheres grávidas e puérperas que morreram de Síndrome Respiratória Aguda Grave no Brasil, Mortalidade materna associada à COVID-19 no Brasil em 2020 e 2021: comparação com mulheres não grávidas e homens e Desfechos da COVID-19 em puérperas, gestantes e mulheres não gestantes e nem puérperas hospitalizadas.

O dicionários das variáveis a ser considerado neste livro está na Tabela **??**.

Tabela 1.1: Dicionário das variáveis da base de dados de COVID-19 em gestantes e puérperas.

Variável	Descrição	Valores
DT_NOTIFIC	Data de preenchimento da ficha de notificação	Dia/Mês/Ano
DT_SIN_PRI	Data de primeiros sintomas do caso	Dia/Mês/Ano
DT_NASC	Data de nascimento da gestante ou puérpera	Dia/Mês/Ano
DT_INTERNA	Data em que gestante ou puérpera foi hospitalizada	Dia/Mês/Ano
SEM_PRI	Semana epidemiológica do início dos sintomas	1 a 52

Variável	Descrição	Valores
CS_RACA	Raça da gestante ou puérpera	1- branca; 2- preta; 3- amarela; 4- parda; 5-indígena; 9- ignorado
CS_ESCOL_N	Nível de escolaridade da gestante ou puérpera	0- sem escolaridade (analfabeto); 1- fundamental 1° ciclo (1ª a 5ª série); 2- fundamental 2 (6ª a 9ª série); 3- medio (1° ao 3° ano); 4- superior; 5- não se aplica; 9- ignorado
idade	Idade, em anos, da gestante ou puérpera	10 a 55
CS_GESTANT	Momento gestacional ou puerpério	1- 1° trimestre; 2- 2° trimestre; 3- 3° trimestre; 4- idade gestacional ignorada; 5- não; 9- ignorado
PUERPERA	Se paciente é puérpera ou parturiente (mulher que pariu recentemente - até 45 dias do parto)	1- sim; 2- não; 9- ignorado
SG_UF	Sigla do estado de residência da gestante ou puérpera	Sigla padronizada pelo IBGE
ID_MN_RESI	Nome do município de residência da gestante ou puérpera	Nomes padronizados pelo IBGE
CO_MUN_RES	Código do município de residência da gestante ou puérpera	Código definido pelo IBGE
CS_ZONA	Tipo de zona de residência da gestante ou puérpera	1- urbana; 2- rural; 3- periurbana; 9- ignorado
FEBRE	Se gestante ou puérpera manifestou sintoma de febre	1- sim; 2- não; 9- ignorado
TOSSE	Se gestante ou puérpera manifestou sintoma de tosse	1- sim; 2- não; 9- ignorado
GARGANTA	Se gestante ou puérpera manifestou sintoma de dor de garganta	1- sim; 2- não; 9- ignorado
DISPNEIA	Se gestante ou puérpera manifestou sintoma de dispneia	1- sim; 2- não; 9- ignorado
DESC_RESP	Se gestante ou puérpera manifestou sintoma de desconforto respiratório	1- sim; 2- não; 9- ignorado
SATURACAO	Se gestante ou puérpera manifestou sintoma de saturação	1- sim; 2- não; 9- ignorado

Variável	Descrição	Valores
DIARREIA	Se gestante ou puérpera manifestou sintoma de diarreia	1- sim; 2- não; 9- ignorado
VOMITO	Se gestante ou puérpera manifestou sintoma de vômito	1- sim; 2- não; 9- ignorado
DOR_ABD	Se gestante ou puérpera manifestou sintoma de dor abdominal	1- sim; 2- não; 9- ignorado
FADIGA	Se gestante ou puérpera manifestou sintoma de fadiga	1- sim; 2- não; 9- ignorado
PERD_OLFT	Se gestante ou puérpera manifestou sintoma de perda de olfato	1- sim; 2- não; 9- ignorado
PERD_PALA	Se gestante ou puérpera manifestou sintoma de perda de paladar	1- sim; 2- não; 9- ignorado
ASMA	Se gestante ou puérpera tem asma	1- sim; 2- não; 9- ignorado
DIABETES	Se gestante ou puérpera tem diabetes *mellitus*	1- sim; 2- não; 9- ignorado
NEUROLOGIC	Se gestante ou puérpera tem doença neurológica	1- sim; 2- não; 9- ignorado
PNEUMOPATI	Se gestante ou puérpera tem outra pneumopatia crônica	1- sim; 2- não; 9- ignorado
IMUNODEPRE	Se gestante ou puérpera tem imunodeficiência ou imunodepressão (diminuição da função do sistema imunológico)	1- sim; 2- não; 9- ignorado
RENAL	Se gestante ou puérpera tem doença renal crônica	1- sim; 2- não; 9- ignorado
OBESIDADE	Se gestante ou puérpera tem obesidade	1- sim; 2- não; 9- ignorado
CARDIOPATI	Se gestante ou puérpera tem doença cardiovascular crônica	1- sim; 2- não; 9- ignorado
HEMATOLOGI	Se gestante ou puérpera tem doença hematológica crônica	1- sim; 2- não; 9- ignorado
HEPATICA	Se gestante ou puérpera tem doença hepática crônica	1- sim; 2- não; 9- ignorado
VACINA_COV	Se gestante ou puérpera recebeu vacina COVID-19	1- sim; 2- não; 9- ignorado
DOSE_1_COV	Data em que gestante ou puérpera recebeu a 1ª dose da vacina COVID-19	Dia/Mês/Ano
DOSE_2_COV	Data em que gestante ou puérpera recebeu a 2ª dose da vacina COVID-19	Dia/Mês/Ano

Variável	Descrição	Valores
FAB_COV_1	Fabricante da vacina que a gestante ou puérpera recebeu na 1ª dose	
FAB_COV_2	Fabricante da vacina que a gestante ou puérpera recebeu na 2ª dose	
SUPORT_VEN	Se gestante ou puérpera precisou de ventilação mecânica; se sim, se foi invasiva ou não	1- sim, invasivo; 2- sim, não invasivo; 3- não; 9- ignorado
UTI	Se gestante ou puérpera foi internada na UTI	1- sim; 2- não; 9- ignorado
DT_ENTUTI	Data de entrada da gestante ou puérpera na UTI	Dia/Mês/Ano
DT_SAIDUTI	Data de saída da gestante ou puérpera na UTI	Dia/Mês/Ano
EVOLUCAO	Evolução do caso da gestante ou puérpera	1- cura; 2- óbito; 3- óbito por outras causas; 9- ignorado

2 Manipulação de dados

Neste capítulo falaremos alguns princípios básicos sobre manipulação de dados. Iremos trabalhar em um cenário mais próximo da realidade possível, ou seja, iremos trabalhar em cima de uma base de dados real. O objetivo é manipular a base e torná-la pronta para ser usada nos capítulos seguintes. Será mostrado desde como importar a base até como criar novas variáveis que poderão ser utilizadas em análises. Não será possível cobrir todo o ramo de manipulação em um só capítulo, mas iremos trabalhar com o máximo de ferramentas possíveis. Pacotes ou funções que não forem utilizadas aqui, mas que são interessantes serão mencionados ao longo do capítulo junto a links que contenham explicações de como utilizá-las. Vale ressaltar que estamos em um cenário mais básico e introdutório. Vamos começar.

2.0.1 Importação de dados

Um dos caminhos mais simples para importar dados no R é utilizando a função `read.table()`. Está função é simples pois ja vem instalada com o R, faz parte do pacote base `utils`, e importa arquivos nos formatos cvs e txt.

A utilização do pacote é bem simples, não preciso carregá-lo na memória usando `library()`.

```
dados1 <- read.table(file = "dados.csv", sep = ";")
dados2 <- read.table(file = "caminho-para-o-arquivo/dados.csv", sep = ";")
```

Observe que na função temos os argumentos `file` e `sep`. O `file` indica o nome do arquivo que será importado e `sep` indica qual o símbolo separador de colunas, que neste caso é a virgula. Note também que usamos dois exemplos, o primeiro considera que o seu arquivo está no diretório de trabalho (quando criamos o projeto e colocamos os arquivos de dados na pasta criada pelo projeto), não sendo necessário especificar o caminho até do arquivo. O outro exemplo mostra como especificar o local do seu arquivo. A função possui mais argumentos que você pode explorar usando o help, mas no geral, esses dois são os mais utilizados.

2.0.1.1 Extensão .txt ou .csv

Caso esteja trabalhando com arquivos do tipo cvs ou txt o pacote `readr` irá servir muito bem. As funções deste pacote são bem rapidas e algumas delas são focadas em tranformar arquivos simples em `data.frame`. Aglumas funções do pacote são

- `read_cvs()`: para arquivos delimitados por vírgulas.
- `read_cvs2()`: para arquivos delimitados por ponto e vírgula.
- `read_tsv()`: para arquivos delimitados por tabulações.
- `read_delim()`: para aquivos com qualquer delimitador.
- `read_fwf()`: para arquivos compactos que devem ter a largura de cada coluna especificada.
- `read_table()`: para arquivos de texto tabulas com colunas separas por espaço.

Caso esta seja a primeira vez que você ira utilizar este pacote, será necessário instalá-lo em seu computar. Você pode fazer isso utilizando a função `install.packages("readr")` e é claro, antes de usar qualquer pacote que não faça parte do R base, você deve carregá-lo. Como exemplo, consideramos um arquivo chamado dados1 que queremos importar para o R.

```
library(readr)
dados_csv <- read_csv(file = "caminho-para-o-arquivo/dados1.csv")
dados_txt <- read_delim(file = "caminho-para-o-arquivo/dados1.txt", delim = " ")
```

Apesar dos argumentos deste pacote serem semelhantes aos da função `read.table()`, devemos nos atentar a algumas diferenças. Aqui é o argumento `delim` que indica qual o separador das colunas no arquivo texto.

Vale ressaltar que para cada função `read_`, existe umas respectiva função `write_` para exportar o arquivo no formato de interesse. Como exemplo, queremos salvar a base de dados mtcars na pasta do meu computador com o nome cars:

```
write_csv(x = mtcars, path = "cars.csv")
write_delim(x = mtcars, delim = " ", path = "cars.txt")
```

2.0.1.2 Arquivos em Excel

Arquivos em formato xlsx são muito utilizados, porém o R não possui uma função nativa para importar este tipo de arquivo. Existem diversos pacotes para importar dados neste e formato e os principais são `redxl`, `xlsx`, `XLConnect` e `tydixl`. Apesar destes pacotes terem objetivos semelhantes, cada um tem suas peculiaridades, então aconselhamos estudar cada um desses pacotes e assim decidir qual melhor atende às suas necessidades. Aqui vamos mostrar apenas o pacote `readxl`, pois é um dos mais facéis e diretos de se utilizar. Este pacote serve para importar e ler planilhas do Excel nos formatos xlsx ou xls. A seguir estão listadas algumas funções para importação e leitura de dados:

- `read_excel()`: esta função detecta automaticamente a extensão do arquivo, e importa arquivos do tipo xsl e xlsx.
- `read_xsl()`: importa arquivos no formato xsl.
- `read_xlsx()`: importa arquivos no formato xlsx.

Novamente, é necessário à instalação e carregamento do pacote caso não o tenha em seu computador. Para exemplicar consideramos um arquivo chamado dados2 que queremos importar para o R.

```
library(readxl)
dados_excel1 <- read_excel(path = "dados2.xls")
dados_excelx1 <- read_excel(path = "dados2.xlsx")
```

Por meio da função `read_excel` conseguimos importar tanto um arquivo no formato xls quanto no formato xlsx.

Podemos também exportar um arquivo em excel (.xls e .xlsx) ao considerar a função `write_xlsx` do pacote `writexl`. Suponha que temos o interesse em salvar a base de dados `dados` em excel na pasta do computador (exportar) com o nome de `dados_correto`:

```
library(writexl)
write_xlsx(dados, "dados_correto.xlsx")
```

2.0.2 Análise de consistência e tratamento de dados

O tratamento dos dados toma muitas vezes a maior parte do tempo de uma análise estatística.

A análise de consistência consiste em realizar uma primeira análise dos dados com o intuito de encontrar inconsistências. São exemplos de inconsistências:

- boas práticas para nome das variáveis.
- como erros de digitação;
- indivíduos imputados mais de uma vez na planilha de dados de maneira errada;
- identificar casos missings e avaliar se a observação está ausente de maneira correta ou não;
- identificar as categorias de variáveis qualitativas.

A partir daqui iremos trabalhar com a nossa base de dados de COVID-19 em gestantes e puérperas.

Importando os dados

Como já aprendemos a importar os dados, vamos direto ao ponto. Nos dados estão no forma **rds** que não foi mencionado anteriormente, mas o pacote `readr` tem uma função para importar este tipo de arquivo.

```
dados <- readr::read_rds("dados/dados_covid[SUJO].rds")
knitr::kable(head(dados))
```

15/[illegible]	[illegible]	16	2	NA	1	1	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	2	2
18/[illegible]	CURITIBA	23	2	2	2	2	2	1	2	2	2	2	1	2	NA	NA	NA	NA	2	2	2	2	2	2	2	2	2	2	NA	NA	NA	NA	NA	2	3
30/[illegible]	SAO CAETANO DO SUL	24	9	1	1	2	2	2	2	2	2	2	2	2	NA	NA	NA	NA	2	2	2	2	2	2	2	2	2	2	NA	NA	NA	NA	NA	2	3
11/[illegible]	[illegible]	33	5	1	1	1	2	2	1	2	1	2	2	2	NA	NA	NA	NA	2	2	2	2	2	2	2	2	2	2	NA	NA	NA	NA	NA	2	2
01/[illegible]	SANTA MARIA	23	5	1	2	2	2	2	2	2	2	2	1	2	NA	1	NA	NA	2	2	2	2	2	2	2	2	2	2	NA	NA	NA	NA	NA	2	3
09/[illegible]	PORTO VELHO	35	3	2	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	NA	NA	NA	NA	NA	2	3

2.0.2.1 Tratamento da base de dados

Inicialmente, vamos verificar os nomes das variáveis na base de dados por meio da função `names`. Note que os nomes estão, de certa forma, padronizados. Todos maíusculos (com exceção de "idade"), separados por "_". Este ainda não é o cenário ideal para trabalharmos, mas poderia ser pior, contendo maiúsculas, espaços e acentos. Utilizar os dados com essas características não impossibilita as futuras análises, mas pode atrapalhar quando precisamos selecionar algumas dessas variáveis.

```
names(dados)
```

```
 [1] "DT_NOTIFIC" "DT_SIN_PRI" "DT_NASC"    "DT_INTERNA" "SEM_PRI"
 [6] "SG_UF"      "ID_MN_RESI" "CO_MUN_RES" "CS_ZONA"    "CS_RACA"
[11] "CS_ESCOL_N" "idade"      "CS_GESTANT" "PUERPERA"   "FEBRE"
[16] "TOSSE"      "GARGANTA"   "DISPNEIA"   "DESC_RESP"  "SATURACAO"
[21] "DIARREIA"   "VOMITO"     "FADIGA"     "PERD_OLFT"  "PERD_PALA"
```

```
[26] "DOR_ABD"    "CARDIOPATI" "HEMATOLOGI" "HEPATICA"   "ASMA"
[31] "DIABETES"   "NEUROLOGIC" "PNEUMOPATI" "IMUNODEPRE" "RENAL"
[36] "OBESIDADE"  "VACINA_COV" "DOSE_1_COV" "DOSE_2_COV" "FAB_COV_1"
[41] "FAB_COV_2"  "DT_ENTUTI"  "DT_SAIDUTI" "UTI"        "SUPORT_VEN"
[46] "EVOLUCAO"
```

uma boa prática consiste em padronizar os nomes das variáveis, até para facilitar a lembrança deles. Para isso, utilizaremos o pacote `janitor` para a arrumação da base de dados. Usamos a função `clean_names()` para primeiro ajuste dos nomes das variáveis.

```
dados <- janitor::clean_names(dados)
names(dados)
```

```
 [1] "dt_notific" "dt_sin_pri" "dt_nasc"    "dt_interna" "sem_pri"
 [6] "sg_uf"      "id_mn_resi" "co_mun_res" "cs_zona"    "cs_raca"
[11] "cs_escol_n" "idade"      "cs_gestant" "puerpera"   "febre"
[16] "tosse"      "garganta"   "dispneia"   "desc_resp"  "saturacao"
[21] "diarreia"   "vomito"     "fadiga"     "perd_olft"  "perd_pala"
[26] "dor_abd"    "cardiopati" "hematologi" "hepatica"   "asma"
[31] "diabetes"   "neurologic" "pneumopati" "imunodepre" "renal"
[36] "obesidade"  "vacina_cov" "dose_1_cov" "dose_2_cov" "fab_cov_1"
[41] "fab_cov_2"  "dt_entuti"  "dt_saiduti" "uti"        "suport_ven"
[46] "evolucao"
```

Veja que ele deixou todos os nomes minúsculos. Neste caso não foi feito, mas a função substitui o espaço por "_" e tira acentos. Isso ajuda a evitar problemas futuros em algumas análises que não lidam muito bem com acentos e espaços nos nomes das variáveis.

Outro problema comum é a presença de linhas e colunas vazias. Na base de dados em questão, não há linhas nem colunas em branco, como pode ser visto na saída abaixo.

```
janitor::remove_empty(dados,"rows")
janitor::remove_empty(dados,"cols")
```

2.0.2.2

Identificando casos duplicados

Outra boa prática é identificar casos duplicados, isto é, identificar se há casos erroneamente repetidos. O ideal é utilizar variável chave do seu banco de dados, ou seja, aquela em que cada observação é única. Por exemplo, em uma base de dados de funcionários de uma empresa,

uma variável chave poderia ser o CPF. Uma variável chave também pode ser a combinação de variáveis, gerando assim observações únicas. Para identificar casos duplicados, usamos a função `get_dupes` do pacote `janitor`. Em nosso banco de dados não tempos uma varíavel chave, então não vamos especificá-la na função, assim a função irá procurar observações repetidas considerando todas as variáveis, ou seja, linhas repetidas.

```
janitor::get_dupes(dados)
```

```
No variable names specified - using all columns.
```

```
No duplicate combinations found of: dt_notific, dt_sin_pri, dt_nasc, dt_interna, sem_pri, sg
```

```
 [1] dt_notific dt_sin_pri dt_nasc    dt_interna sem_pri    sg_uf
 [7] id_mn_resi co_mun_res cs_zona    cs_raca    cs_escol_n idade
[13] cs_gestant puerpera   febre      tosse      garganta   dispneia
[19] desc_resp  saturacao  diarreia   vomito     fadiga     perd_olft
[25] perd_pala  dor_abd    cardiopati hematologi hepatica   asma
[31] diabetes   neurologic pneumopati imunodepre renal      obesidade
[37] vacina_cov dose_1_cov dose_2_cov fab_cov_1  fab_cov_2  dt_entuti
[43] dt_saiduti uti        suport_ven evolucao   dupe_count
<0 rows> (or 0-length row.names)
```

Em nosso caso, não temos casos duplicados. Caso tivesse, seria necessário remover as linhas duplicadas. Isto pode ser feito com o uso da função `distinct` do pacote `dplyr`.

2.0.3 Identificar problemas nas variáveis da base de dados

Outra etapa importante na análise de consistência é identificar o tipo de variável e ver se o R está interpretando corretamente o tipo de cada variável.

Temos na nossa base de dados variáveis de data, além de variáveis qualitativas e quantitativas (veja o dicionário das variáveis na em: **refenciar parte**). Assim, precisamos entender se o R realmente entendeu todas as variáveis da maneira correta. Uma maneira de identificar isso e também de ver algumas descritivas das variáveis que nos auxiliam a ver possíveis inconsistências na base de dados é a a função `glimpse` do pacote `dplyr`. A função `skim` do pacote `skimr` também pode ajudar nisso.

```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
glimpse(dados)
```

```
Rows: 11,523
Columns: 46
$ dt_notific <chr> "15/05/2020", "18/05/2020", "30/04/2020", "11/05/2020", "01~
$ dt_sin_pri <chr> "06/05/2020", "10/05/2020", "20/04/2020", "04/05/2020", "12~
$ dt_nasc    <chr> "03/06/2003", "07/07/1996", "26/03/1996", "02/06/1986", "11~
$ dt_interna <chr> "15/05/2020", "15/05/2020", "24/04/2020", "09/05/2020", "30~
$ sem_pri    <int> 19, 20, 17, 19, 24, 24, 26, 27, 28, 24, 14, 29, 28, 10, 36,~
$ sg_uf      <chr> "CE", "PR", "SP", "PA", "DF", "RO", "PI", "RS", "PE", "MA",~
$ id_mn_resi <chr> "MORRINHOS", "CURITIBA", "SAO CAETANO DO SUL", "MARABA", "S~
$ co_mun_res <int> 230890, 410690, 354880, 150420, 530150, 110020, 221100, 431~
$ cs_zona    <int> NA, 1, 1, 1, 1, 1, 1, 1, 1, 1, NA, 1, 1, 1, NA, 1, 1, NA, 2~
$ cs_raca    <int> 4, 1, 9, 4, 9, 4, 9, 1, 4, 9, 4, 4, 4, 4, 4, 9, 4, 9, 4, 4,~
$ cs_escol_n <int> NA, 2, 9, 4, NA, 2, 4, 2, NA, NA, NA, 9, 9, 3, 3, NA, 3, NA~
$ idade      <dbl> 16, 23, 24, 33, 23, 35, 31, 17, 22, 29, 28, 22, 27, 25, 26,~
$ cs_gestant <int> 2, 2, 9, 5, 5, 3, 1, 5, 3, 3, 4, 3, 5, 3, 3, 3, 3, 9, 3, 3,~
$ puerpera   <int> NA, 2, 1, 1, 1, 2, NA, 1, NA, 1, NA, NA, 1, 2, 1, NA, NA, 1~
$ febre      <int> 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, NA, 1, 1, 1, 1, 1, 1, 1, 1~
$ tosse      <int> 1, 2, 2, 1, 2, 1, 1, 1, NA, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1~
$ garganta   <int> NA, 2, 2, 2, 2, 2, 2, NA, 1, NA, 2, NA, 2, 1, 1, 2, 2, NA, ~
$ dispneia   <int> NA, 1, 2, 2, 2, 2, 2, 1, NA, NA, 1, NA, 2, 1, 2, 1, 1, NA, ~
$ desc_resp  <int> NA, 2, 2, 1, 2, 2, 1, 1, NA, 1, 1, NA, 2, 1, 2, 2, 1, NA, 1~
$ saturacao  <int> NA, 2, 2, 2, 2, 2, 2, 1, NA, NA, 2, NA, 2, 2, 2, 1, 1, NA, ~
$ diarreia   <int> NA, 1, 2, 2, 1, 2, 2, 1, NA, NA, 2, NA, 2, 1, 2, 2, 2, NA, ~
$ vomito     <int> NA, 2, 2, 2, 2, 2, 2, 2, NA, NA, 2, NA, 2, 1, 2, 2, 2, NA, ~
$ fadiga     <int> NA, NA, NA, NA, NA, 2, NA, NA, NA, NA, 2, NA, NA, 2, 2, 2, ~
$ perd_olft  <int> NA, NA, NA, NA, 1, 2, NA, NA, NA, NA, 2, NA, NA, 1, 2, 1, 2~
$ perd_pala  <int> NA, NA, NA, NA, NA, 2, NA, NA, NA, NA, 2, NA, NA, 1, 2, 2, ~
$ dor_abd    <int> NA, NA, NA, NA, NA, 2, NA, NA, NA, NA, 2, NA, NA, 2, 2, 2, ~
```

```
$ cardiopati <int> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 1, NA, NA, ~
$ hematologi <int> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ hepatica   <int> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ asma       <int> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ diabetes   <int> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ neurologic <int> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 1, NA, NA, ~
$ pneumopati <int> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ imunodepre <int> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ renal      <int> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ obesidade  <int> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ vacina_cov <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ dose_1_cov <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ dose_2_cov <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ fab_cov_1  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ fab_cov_2  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ dt_entuti  <chr> "", "", "", "", "", "", "", "30/06/2020", "", "12/07/2020",~
$ dt_saiduti <chr> "", "", "", "", "", "", "", "29/07/2020", "", "", "", "", "~
$ uti        <int> 2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2, 2, 2, 2, 2, 2, 1, 1, NA, 2~
$ suport_ven <int> 2, 3, 3, 2, 3, 3, 9, 1, 3, 3, 3, 3, 3, 2, 3, 2, 1, 2, 3, 2,~
$ evolucao   <int> 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, NA, 1, 1, 1, 1, 1, 1~
```

No R, as variáveis qualititativas são nomeadas "factor", as variáveis quantitativas são nomeadas "numeric" e as variáveis de data são "date". Note que na importação dos dados o R não entendeu corretamente os tipos de variáveis. Mas vamos corrigir isso no que segue.

Começando pela data, vamos rodar o seguinte código:

```
dados$dt_notific  <- as.Date(dados$dt_notific, format = "%d/%m/%Y")
dados$dt_sin_pri  <- as.Date(dados$dt_sin_pri, format = "%d/%m/%Y")
dados$dt_nasc  <- as.Date(dados$dt_nasc, format = "%d/%m/%Y")
dados$dt_interna  <- as.Date(dados$dt_interna, format = "%d/%m/%Y")
dados$dt_entuti  <- as.Date(dados$dt_entuti, format = "%d/%m/%Y")
dados$dt_saiduti  <- as.Date(dados$dt_saiduti, format = "%d/%m/%Y")
```

A função `as.Date` informa para o R que a variável indicada é de data. O argumento `format` indica o formato que está a data, nesse caso, "dia/mês/ano". Aqui é possível verificar todos os formatos de datas da função. Vamos ver como ficou:

```
glimpse(dados)
```

```
Rows: 11,523
Columns: 46
```

```
$ dt_notific <date> 2020-05-15, 2020-05-18, 2020-04-30, 2020-05-11, 2020-07-01~
$ dt_sin_pri <date> 2020-05-06, 2020-05-10, 2020-04-20, 2020-05-04, 2020-06-12~
$ dt_nasc    <date> 2003-06-03, 1996-07-07, 1996-03-26, 1986-06-02, 1996-12-11~
$ dt_interna <date> 2020-05-15, 2020-05-15, 2020-04-24, 2020-05-09, 2020-06-30~
$ sem_pri    <int> 19, 20, 17, 19, 24, 24, 26, 27, 28, 24, 14, 29, 28, 10, 36,~
$ sg_uf      <chr> "CE", "PR", "SP", "PA", "DF", "RO", "PI", "RS", "PE", "MA",~
$ id_mn_resi <chr> "MORRINHOS", "CURITIBA", "SAO CAETANO DO SUL", "MARABA", "S~
$ co_mun_res <int> 230890, 410690, 354880, 150420, 530150, 110020, 221100, 431~
$ cs_zona    <int> NA, 1, 1, 1, 1, 1, 1, 1, 1, 1, NA, 1, 1, 1, NA, 1, 1, NA, 2~
$ cs_raca    <int> 4, 1, 9, 4, 9, 4, 9, 1, 4, 9, 4, 4, 4, 4, 4, 9, 4, 9, 4, 4,~
$ cs_escol_n <int> NA, 2, 9, 4, NA, 2, 4, 2, NA, NA, NA, 9, 9, 3, 3, NA, 3, NA~
$ idade      <dbl> 16, 23, 24, 33, 23, 35, 31, 17, 22, 29, 28, 22, 27, 25, 26,~
$ cs_gestant <int> 2, 2, 9, 5, 5, 3, 1, 5, 3, 3, 4, 3, 5, 3, 3, 3, 3, 9, 3, 3,~
$ puerpera   <int> NA, 2, 1, 1, 1, 2, NA, 1, NA, 1, NA, NA, 1, 2, 1, NA, NA, 1~
$ febre      <int> 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, NA, 1, 1, 1, 1, 1, 1, 1, 1~
$ tosse      <int> 1, 2, 2, 1, 2, 1, 1, 1, NA, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1~
$ garganta   <int> NA, 2, 2, 2, 2, 2, 2, NA, 1, NA, 2, NA, 2, 1, 1, 2, 2, NA, ~
$ dispneia   <int> NA, 1, 2, 2, 2, 2, 2, 1, NA, NA, 1, NA, 2, 1, 2, 1, 1, NA, ~
$ desc_resp  <int> NA, 2, 2, 1, 2, 2, 1, 1, NA, 1, 1, NA, 2, 1, 2, 2, 1, NA, 1~
$ saturacao  <int> NA, 2, 2, 2, 2, 2, 2, 1, NA, NA, 2, NA, 2, 2, 2, 1, 1, NA, ~
$ diarreia   <int> NA, 1, 2, 2, 1, 2, 2, 1, NA, NA, 2, NA, 2, 1, 2, 2, 2, NA, ~
$ vomito     <int> NA, 2, 2, 2, 2, 2, 2, 2, NA, NA, 2, NA, 2, 1, 2, 2, 2, NA, ~
$ fadiga     <int> NA, NA, NA, NA, NA, 2, NA, NA, NA, NA, 2, NA, NA, 2, 2, 2, ~
$ perd_olft  <int> NA, NA, NA, NA, 1, 2, NA, NA, NA, NA, 2, NA, NA, 1, 2, 1, 2~
$ perd_pala  <int> NA, NA, NA, NA, NA, 2, NA, NA, NA, NA, 2, NA, NA, 1, 2, 2, ~
$ dor_abd    <int> NA, NA, NA, NA, NA, 2, NA, NA, NA, NA, 2, NA, NA, 2, 2, 2, ~
$ cardiopati <int> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 1, NA, NA, ~
$ hematologi <int> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ hepatica   <int> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ asma       <int> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ diabetes   <int> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ neurologic <int> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 1, NA, NA, ~
$ pneumopati <int> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ imunodepre <int> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ renal      <int> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ obesidade  <int> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ vacina_cov <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ dose_1_cov <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ dose_2_cov <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ fab_cov_1  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ fab_cov_2  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ dt_entuti  <date> NA, NA, NA, NA, NA, NA, NA, 2020-06-30, NA, 2020-07-12, NA~
$ dt_saiduti <date> NA, NA, NA, NA, NA, NA, NA, 2020-07-29, NA, NA, NA, NA, NA~
```

```
$ uti        <int> 2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2, 2, 2, 2, 2, 2, 1, 1, NA, 2~
$ suport_ven <int> 2, 3, 3, 2, 3, 3, 9, 1, 3, 3, 3, 3, 3, 2, 3, 2, 1, 2, 3, 2,~
$ evolucao   <int> 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, NA, 1, 1, 1, 1, 1, 1~
```

Agora vamos lidar com as variáveis qualitativas. Veja que "cs_zona" foi identificada como *int*. Isso acontece porque ela foi tabulada como número, como posteriormente variáveis deste tipo serão recodificadas de acordo com o dicionário, precisamos tratá-la como fator. Já as demais variáveis qualitativas estão identificadas como *numeric*, *dbl* ou *chacacter* pois na tabulação suas categorias estão codificadas com números ou textos. Para então dizer ao R o verdadeiro tipo dessas variáveis, vamos utilizar os seguintes comandos:

```
dados$cs_raca <- as.factor(dados$cs_raca)
dados$cs_escol_n <- as.factor(dados$cs_escol_n)
dados$cs_gestant <- as.factor(dados$cs_gestant)
dados$puerpera <- as.factor(dados$puerpera)
dados$cs_zona <- as.factor(dados$cs_zona)
dados$febre <- as.factor(dados$febre)
dados$tosse <- as.factor(dados$tosse)
dados$suport_ven <- as.factor(dados$suport_ven)
dados$uti <- as.factor(dados$uti)
dados$evolucao <- as.factor(dados$evolucao)
```

Uma forma um pouco mais eficiente de fazer isso é selecionar as variáveis por meio de um vetor, por exemplo, quero que as variáveis da coluna 10 até a coluna 20 sejam fatores. Podemos fazer isso com a ajuda a função `lapply`. Essa função, em resumo, nos possibilita aplicar uma função em uma lista de elementos e retorna uma lista de mesmo tamanho em que o resultado é a aplicação desta função a cada elemento da lista. Neste caso, aplicamos a função `as.factor` nas colunas selecionadas (lista de elementos). Veja como é feito.

```
dados[,c(17:37)] <- lapply(dados[,c(17:37)], as.factor)
glimpse(dados)
```

```
Rows: 11,523
Columns: 46
$ dt_notific <date> 2020-05-15, 2020-05-18, 2020-04-30, 2020-05-11, 2020-07-01~
$ dt_sin_pri <date> 2020-05-06, 2020-05-10, 2020-04-20, 2020-05-04, 2020-06-12~
$ dt_nasc    <date> 2003-06-03, 1996-07-07, 1996-03-26, 1986-06-02, 1996-12-11~
$ dt_interna <date> 2020-05-15, 2020-05-15, 2020-04-24, 2020-05-09, 2020-06-30~
$ sem_pri    <int> 19, 20, 17, 19, 24, 24, 26, 27, 28, 24, 14, 29, 28, 10, 36,~
$ sg_uf      <chr> "CE", "PR", "SP", "PA", "DF", "RO", "PI", "RS", "PE", "MA",~
$ id_mn_resi <chr> "MORRINHOS", "CURITIBA", "SAO CAETANO DO SUL", "MARABA", "S~
```

```
$ co_mun_res <int> 230890, 410690, 354880, 150420, 530150, 110020, 221100, 431~
$ cs_zona    <fct> NA, 1, 1, 1, 1, 1, 1, 1, 1, 1, NA, 1, 1, 1, NA, 1, 1, NA, 2~
$ cs_raca    <fct> 4, 1, 9, 4, 9, 4, 9, 1, 4, 9, 4, 4, 4, 4, 4, 9, 4, 9, 4, 4,~
$ cs_escol_n <fct> NA, 2, 9, 4, NA, 2, 4, 2, NA, NA, NA, 9, 9, 3, 3, NA, 3, NA~
$ idade      <dbl> 16, 23, 24, 33, 23, 35, 31, 17, 22, 29, 28, 22, 27, 25, 26,~
$ cs_gestant <fct> 2, 2, 9, 5, 5, 3, 1, 5, 3, 3, 4, 3, 5, 3, 3, 3, 3, 9, 3, 3,~
$ puerpera   <fct> NA, 2, 1, 1, 1, 2, NA, 1, NA, 1, NA, NA, 1, 2, 1, NA, NA, 1~
$ febre      <fct> 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, NA, 1, 1, 1, 1, 1, 1, 1, 1~
$ tosse      <fct> 1, 2, 2, 1, 2, 1, 1, 1, NA, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1~
$ garganta   <fct> NA, 2, 2, 2, 2, 2, 2, NA, 1, NA, 2, NA, 2, 1, 1, 2, 2, NA, ~
$ dispneia   <fct> NA, 1, 2, 2, 2, 2, 2, 1, NA, NA, 1, NA, 2, 1, 2, 1, 1, NA, ~
$ desc_resp  <fct> NA, 2, 2, 1, 2, 2, 1, 1, NA, 1, 1, NA, 2, 1, 2, 2, 1, NA, 1~
$ saturacao  <fct> NA, 2, 2, 2, 2, 2, 2, 1, NA, NA, 2, NA, 2, 2, 2, 1, 1, NA, ~
$ diarreia   <fct> NA, 1, 2, 2, 1, 2, 2, 1, NA, NA, 2, NA, 2, 1, 2, 2, 2, NA, ~
$ vomito     <fct> NA, 2, 2, 2, 2, 2, 2, 2, NA, NA, 2, NA, 2, 1, 2, 2, 2, NA, ~
$ fadiga     <fct> NA, NA, NA, NA, NA, 2, NA, NA, NA, NA, 2, NA, NA, 2, 2, 2, ~
$ perd_olft  <fct> NA, NA, NA, NA, 1, 2, NA, NA, NA, NA, 2, NA, NA, 1, 2, 1, 2~
$ perd_pala  <fct> NA, NA, NA, NA, NA, 2, NA, NA, NA, NA, 2, NA, NA, 1, 2, 2, ~
$ dor_abd    <fct> NA, NA, NA, NA, NA, 2, NA, NA, NA, NA, 2, NA, NA, 2, 2, 2, ~
$ cardiopati <fct> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 1, NA, NA, ~
$ hematologi <fct> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ hepatica   <fct> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ asma       <fct> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ diabetes   <fct> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ neurologic <fct> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 1, NA, NA, ~
$ pneumopati <fct> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ imunodepre <fct> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ renal      <fct> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ obesidade  <fct> NA, 2, 2, 2, 2, 2, NA, 2, NA, NA, NA, NA, 2, 2, 2, NA, NA, ~
$ vacina_cov <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ dose_1_cov <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ dose_2_cov <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ fab_cov_1  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ fab_cov_2  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ dt_entuti  <date> NA, NA, NA, NA, NA, NA, NA, 2020-06-30, NA, 2020-07-12, NA~
$ dt_saiduti <date> NA, NA, NA, NA, NA, NA, NA, 2020-07-29, NA, NA, NA, NA, NA~
$ uti        <fct> 2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2, 2, 2, 2, 2, 2, 1, 1, NA, 2~
$ suport_ven <fct> 2, 3, 3, 2, 3, 3, 9, 1, 3, 3, 3, 3, 3, 2, 3, 2, 1, 2, 3, 2,~
$ evolucao   <fct> 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, NA, 1, 1, 1, 1, 1, 1~
```

Ótimo! Corrigimos as inconsistências das variáveis qualitativas. Mas outra questão surge: como faço para usar um rótulo nos números codificados nas categorias das variáveis

qualitativas? Para o grupo, por exemplo, ao invés de aparecer 1 quero que apareça "sim". Para isso, vamos utilizar o pacote `forcats` que lida com variáveis qualitativas (categóricas). Para renomear as categorias das variáveis, vamos usar a função `fct_recode` desse pacote:

```
dados$cs_raca <- forcats::fct_recode(dados$cs_raca,
                                     branca = "1",
                                     preta = "2",
                                     amarela = "3",
                                     parda = "4",
                                     indigena = "5",
                                     ignorado = "9")

dados$cs_escol_n <- forcats::fct_recode(dados$cs_escol_n,
                                       "sem escola"  = "0",
                                       fund1 = "1",
                                       fund2 = "2",
                                       medio = "3",
                                       superior = "4",
                                       ignorado = "9")

dados$cs_gestant <- forcats::fct_recode(dados$cs_gestant,
                                       "1tri" = "1",
                                       "2tri" = "2",
                                       "3tri" = "3",
                                       IG_ig = "4",
                                       nao = "5",
                                       ignorado = "9")

dados$puerpera <- forcats::fct_recode(dados$puerpera,
                                       sim = "1",
                                       nao = "2",
                                       ignorado = "9")

dados$cs_zona <- forcats::fct_recode(dados$cs_zona,
                                     urbana = "1",
                                     rural = "2",
                                     periurbana = "3",
                                     ignorado = "9")

dados$febre <- forcats::fct_recode(dados$febre,
                                     sim = "1",
                                     nao = "2",
```

```
                                    ignorado = "9")

dados$suport_ven <-forcats::fct_recode(dados$suport_ven,
                                       "sim, invasivo" = "1",
                                       "sim, nao invasivo" = "2",
                                       nao = "3",
                                       ignorado = "9")

dados$uti <-forcats::fct_recode(dados$uti,
                                       sim = "1",
                                       nao = "2",
                                       ignorado = "9")

dados$evolucao <-forcats::fct_recode(dados$evolucao,
                                       cura = "1",
                                       obito = "2",
                                       "obito por outras causas" = "3",
                                       ignorado = "9")
```

Este tramanto foi feito para todas as variáveis qualitativas da base, mas por conta do tamanho do código, omitimos algumas da saída.

Finalmente chegamos nas variáveis quantitativas. Uma forma de identificar problemas em variáveis quantitativas é avaliar os valores mínimo e máximo de cada variável e ver se tem algum valor impossível para a mesma. Em nosso caso podemos verificar a variável idade. Seria meio estranho encontrar alguém com valores extremamente altos ou negativos, concorda?! A função `summary` pode ser uma opção boa aqui, ela nos formece algumas medidas descritivas como, media, mínimo, máximo, entre outros.

```
summary(dados$idade)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  10.00   25.00   30.00   30.25   35.00   55.00       9
```

Aparentemente nossa variável esta dentro do esperado, sem valores inesperados.

2.0.3.1 Transformação de dados

É possível modificar ou criar novas variáveis na base de dados por meio da função `mutate` do pacote `dplyr`, você pode veificar melhor essa função clicando aqui. Também podemos criar categorias com base em alguma condição por meio da função `case_when` também do pacote

`dplyr`, veja melhor aqui. Para ficar mais claro, vamos a um exemplo combinando as duas funções. Vamos criar a variável "faixa_et", onde as observações serão as faixas etárias. São essas: "<20", "20-34" e ">=". Veja como faz:

```
dados <- dados |>
  mutate(faixa_et = case_when(
    idade < 20 ~ "<20",
    idade >= 20 & idade < 34 ~ "20-34",
    idade >= 34 ~ ">=34"
  ))

table(dados$faixa_et)#table nos mostra as observações da quela variável e a sua frequência
```

```
<20  >=34 20-34
714  3862  6938
```

Aqui fizemos a utilização da função "pipe" `|>` que agora está no pacote base do R, mas que antes era necessário carregá-la por meio de pacotes. Essa função é de extrema importância, facilita a programção no R de uma forma inimaginável. É válido dedicar um pouco de seu tempo para entender melhor essa função. Separamos alguns links que pode te ajudar a entender melhor e você pode acessá-los clickando aqui, aqui ou aqui. Como foi mencionado acima, a função foi adicionada ao R base há pouco tempo, então esses links se referem ao pipe "antigo", mas fique tranquilo, a função é a mesma. Para resumir, o pipe pega a saída de uma função e a passa para outra função como um argumento. Isso nos permite vincular uma sequência de etapas de análise.

2.0.3.2 Manipulação de datas

Algo interessante também é trabalhar com a varíavel de datas. Podemos calulcar a diferença entre duas datas no R de forma bem simples por meio da função `difftime` do pacote base do R. Para exemplificar vamos criar a variável "dias_uti" que vai ser ser quantos dias a pessoa ficou internada na uti. Vamos fazer isso calculando a diferença entre a data de saída e a data de entrada na uti e queremos o resultado em dias.

```
dados$dias_uti <- difftime(dados$dt_saiduti, dados$dt_entuti, units = "days")
glimpse(dados)
```

```
Rows: 11,523
Columns: 48
```

```
$ dt_notific <date> 2020-05-15, 2020-05-18, 2020-04-30, 2020-05-11, 2020-07-01~
$ dt_sin_pri <date> 2020-05-06, 2020-05-10, 2020-04-20, 2020-05-04, 2020-06-12~
$ dt_nasc    <date> 2003-06-03, 1996-07-07, 1996-03-26, 1986-06-02, 1996-12-11~
$ dt_interna <date> 2020-05-15, 2020-05-15, 2020-04-24, 2020-05-09, 2020-06-30~
$ sem_pri    <int> 19, 20, 17, 19, 24, 24, 26, 27, 28, 24, 14, 29, 28, 10, 36,~
$ sg_uf      <chr> "CE", "PR", "SP", "PA", "DF", "RO", "PI", "RS", "PE", "MA",~
$ id_mn_resi <chr> "MORRINHOS", "CURITIBA", "SAO CAETANO DO SUL", "MARABA", "S~
$ co_mun_res <int> 230890, 410690, 354880, 150420, 530150, 110020, 221100, 431~
$ cs_zona    <fct> NA, urbana, urbana, urbana, urbana, urbana, urbana, urbana,~
$ cs_raca    <fct> parda, branca, ignorado, parda, ignorado, parda, ignorado, ~
$ cs_escol_n <fct> NA, fund2, ignorado, superior, NA, fund2, superior, fund2, ~
$ idade      <dbl> 16, 23, 24, 33, 23, 35, 31, 17, 22, 29, 28, 22, 27, 25, 26,~
$ cs_gestant <fct> 2tri, 2tri, ignorado, nao, nao, 3tri, 1tri, nao, 3tri, 3tri~
$ puerpera   <fct> NA, nao, sim, sim, sim, nao, NA, sim, NA, sim, NA, NA, sim,~
$ febre      <fct> sim, nao, sim, sim, nao, sim, sim, sim, sim, sim, sim, NA, ~
$ tosse      <fct> sim, nao, nao, sim, nao, sim, sim, sim, NA, sim, sim, sim, ~
$ garganta   <fct> NA, nao, nao, nao, nao, nao, nao, NA, sim, NA, nao, NA, nao~
$ dispneia   <fct> NA, sim, nao, nao, nao, nao, nao, sim, NA, NA, sim, NA, nao~
$ desc_resp  <fct> NA, nao, nao, sim, nao, nao, sim, sim, NA, sim, sim, NA, na~
$ saturacao  <fct> NA, nao, nao, nao, nao, nao, nao, sim, NA, NA, nao, NA, nao~
$ diarreia   <fct> NA, sim, nao, nao, sim, nao, nao, sim, NA, NA, nao, NA, nao~
$ vomito     <fct> NA, nao, nao, nao, nao, nao, nao, nao, NA, NA, nao, NA, nao~
$ fadiga     <fct> NA, NA, NA, NA, NA, nao, NA, NA, NA, NA, nao, NA, NA, nao, ~
$ perd_olft  <fct> NA, NA, NA, NA, sim, nao, NA, NA, NA, NA, nao, NA, NA, sim,~
$ perd_pala  <fct> NA, NA, NA, NA, NA, nao, NA, NA, NA, NA, nao, NA, NA, sim, ~
$ dor_abd    <fct> NA, NA, NA, NA, NA, nao, NA, NA, NA, NA, nao, NA, NA, nao, ~
$ cardiopati <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ hematologi <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ hepatica   <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ asma       <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ diabetes   <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ neurologic <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ pneumopati <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ imunodepre <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ renal      <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ obesidade  <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ vacina_cov <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ dose_1_cov <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ dose_2_cov <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ fab_cov_1  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ fab_cov_2  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ dt_entuti  <date> NA, NA, NA, NA, NA, NA, NA, 2020-06-30, NA, 2020-07-12, NA~
$ dt_saiduti <date> NA, NA, NA, NA, NA, NA, NA, 2020-07-29, NA, NA, NA, NA, NA~
```

```
$ uti        <fct> nao, nao, nao, nao, nao, nao, nao, sim, nao, sim, nao, nao,~
$ suport_ven <fct> "sim, nao invasivo", "nao", "nao", "sim, nao invasivo", "na~
$ evolucao   <fct> cura, cura, cura, cura, cura, cura, cura, obito, cura, cura~
$ faixa_et   <chr> "<20", "20-34", "20-34", "20-34", "20-34", ">=34", "20-34",~
$ dias_uti   <drtn> NA days, NA days, NA days, NA days, NA days, NA days, NA d~
```

Note que não utilizamos a função `mutate` para criar está nova variável, utilizamos apenas o `$` para representar a variável e atribuímos a função. Assim, o R já entende como uma variável.

2.0.3.3 Manipulação de dados

Já temos a nossa base de dados devidamente tratada para prosseguir com a análise descritiva, mas quando falamos de manipulação de dados, um leque de possibilidades aparece. Em diversos cenários precisamos filtrar observações, reordená-las, selecionar variáveis específicas, entre outras coisas. Não poderíamos deixar de mencionar o poderoso `tidyverse`. O `tidyverse` é um pacote que contém um coleção de outros pacotes que são utilizados para manipulação, exploração e visualização de dados e que compartilham uma filosofia de design bem parecida, por isso de forma combinada permitem que você consiga fazer inúmeros trabalhos. Os pacotes que fazem parte desse universo são: `dplyr`, `tidyr`, `ggplot2`, `forcats`, `purrr`, `stringr`, `tibble` e `readr`. Anteriormente já trabalhamos com alguns destes pacotes, mas agora é válido aprofundarmos um pouco mais em alguns deles. Aqui você irá acessar o site do `tidyverse` onde podera navegar por cada pacote e aprender mais sobre suas utilidades e aqui você irá acessar um post escrito pelo **Laboratório de Data Scinence - UFES** (daslab) que contem diversos exemplos práticos de uso de todos os pacotes do universo `tidyverse`. Neste capítulo iremos trabalhar com algumas funções específicas.

2.0.3.3.1 Pacote dplyr

O `dplyr` é extremamente útil e nos ajuda a resolver os desafios mais comuns de manipulação de dados.

Suas principais funções são:

- filter() - filtra linhas;
- group_by() - agrupa pela(s) variável(is) no argumento. Função muito útil quando usada a funçaõ summurise.
- summarise() - reduz vários valores a um único resumo.
- select() - seleciona colunas;
- arrange() - ordena a base;
- mutate() - cria/modifica colunas.

Já utilizamos algumas funções do pacote, vamos falar sobre outras. Como já avançamos um pouco sobre a utilização de funções, vamos combinar algumas funções, o que geralmente é feito no dia a dia.

```
#criando um novo banco de dados selecionando 3 variáveis
dados_tratamento <- dados |>
  select(sg_uf, cs_zona, idade)

glimpse(dados_tratamento)
```

```
Rows: 11,523
Columns: 3
$ sg_uf   <chr> "CE", "PR", "SP", "PA", "DF", "RO", "PI", "RS", "PE", "MA", "M~
$ cs_zona <fct> NA, urbana, urbana, urbana, urbana, urbana, urbana, urbana, ur~
$ idade   <dbl> 16, 23, 24, 33, 23, 35, 31, 17, 22, 29, 28, 22, 27, 25, 26, 20~
```

Aqui criamos a base "dados_tratamento" onde apenas selecionamos algumas colunas da base de dados inicial com a função `select`.

```
dados_tratamento2 <- dados_tratamento |>
  filter(cs_zona == "urbana") |>
  group_by(sg_uf) |>
  summarise(media = mean(idade, na.rm = TRUE)) |>
  arrange(desc(media))

knitr::kable(head(dados_tratamento2))
```

sg_uf	media
AP	36.66667
BA	31.04152
RR	31.00000
SP	30.94237
RJ	30.93570
MG	30.93257

Vamos entender o código acima. Primeiro acessamos a base "dados_tratamento" e com função `filter` selecionamos apenas as observações "urbana". Após isso utilizamos a função `group_by` para agrupar nossas observações pela variável "sg_uf" e por últimos, combinamos com a função `summarise` para criar a variável "media" que será a media da variável idade. Note que nesta função utilizamos o argumento `na.rm - TRUE`. Este argumento serve para indicar para

a função se ela deve ou não remover valores `NA's` do cálculo, o default é `FALSE`. Como não é possível calcular a média de valores ausentes e temos variáveis ausentes, foi necessário utilizar este argumento. Caso contrário, Estados com valores faltantes ficariam com `NA`. Por último, utilizamos a função `arrange` para ordernar os dados em em ordem descrente pela variavel media. Uma dica para tentar entender melhor o funcionamento das funções é tentar refazer o código utilizando uma função de cada vez e ir vendo como fica. Então, em poucas linhas de códigos conseguimos criar uma base com a idade média dos Estados considerando apenas zonas urbanas, legal né?

2.0.3.3.2 Pacote stringr

Um desafio muito grande na manipulação de dados é extrair informações de caracteres. Em resumo caracteres são letras, símbolos, sinais, números que representem algo escrito, etc.. Essa sequência de caracteres formam o que chamamos de `string`. Diversas vezes encontramos variáveis com categorias não padronizadas, como, por exemplo, uma variável contendo "São Paulo", "sao paulo" e "sp". Apesar de representarem o mesmo estado, elas são diferentes. Nesse sentido, uma parte muito importante no tratamento de dados é "lapidar" esse conjunto de caracteres para que seja possível usá-los nas análises. Essa é a introdução do post do **daslab** onde é passado de uma maneira muito prática como trabalhar com strings utilizando o pacote `stringr`, la você vai aprender também sobre expressões regulares, que com certeza serão úteis em vários momentos da sua carreira. Link do post. Como as variáveis de texto do nosso banco de dados já estão bem padronizadas não será necessário realizar nenhum tratamento, mas por ser um pacote de extrema importância e que não havia sido mencionado ainda, deixamos ele aqui para que você possa se aprofundar mais. Como em nossa base dados as variáveis de texto estão padrozinados, não será necessário realizar nenhum tratamento.

2.0.3.4 Manipulando o formato da base de dados

Em certos casos é necessário mudar o formato das bases de dados, fazer com que colunas se tornem linhas vice-versa. Vamos utilizar a base "dados_tratamento". Veja que ela está no formato *long*, em que as avaliações do mesmo indivíduo (variável de identificação de indivíduo é "registro") estão nas linhas. Queremos que as zonas fiquem nas colunas, com as três colunas (vamos tirar valores ignorados): urbana, rural e periurbana, ou seja, queremos o formato *wide*. Um pacote do R que pode nos auxiliar a transformar formato *long* em *wide* e vice-versa é o `tidyr`. A função que usaremos é `spread`, como segue:

```
library(tidyr)

dados_formato <- dados_tratamento |>
  filter(!is.na(cs_zona) & cs_zona != "ignorado") |>
  mutate(id = row_number())
```

```r
knitr::kable(head(dados_formato))
```

sg_uf	cs_zona	idade	id
PR	urbana	23	1
SP	urbana	24	2
PA	urbana	33	3
DF	urbana	23	4
RO	urbana	35	5
PI	urbana	31	6

Fizemos pequenas alterações na base de dados. Primeiro realizamos um filtro para retirarmos valores faltantes da variável "cs_zona", pois essa passará a

```r
dados_formato2 <- dados_formato |>
  pivot_wider(names_from = cs_zona, values_from = idade)

knitr::kable(head(dados_formato2))
```

sg_uf	id	urbana	rural	periurbana
PR	1	23	NA	NA
SP	2	24	NA	NA
PA	3	33	NA	NA
DF	4	23	NA	NA
RO	5	35	NA	NA
PI	6	31	NA	NA

```r
dados_formato3 <- dados_formato2 |>
  pivot_longer(cols = c("urbana",   "rural",    "periurbana"), names_to = "cs_zona", value
```

```r
knitr::kable(head(dados_formato3))
```

sg_uf	id	cs_zona	idade
PR	1	urbana	23
PR	1	rural	NA
PR	1	periurbana	NA
SP	2	urbana	24
SP	2	rural	NA

sg_uf	id	cs_zona	idade
SP	2	periurbana	NA

2.0.3.5 Combinando bases de dados

Quando estamos trabalhando com dados, nem sempre uma única base irá conter todas as informações que precisamos, na verdade, isso é mais comum do que se possa imaginar. Assim, saber juntar duas bases de dados é indispensável. Vamos começar então falando sobre chave primária. Em resumo, chave primária se refere a um ou mais campos, onde combinados (no caso de mais de uma chave primária), não se repete na mesma tabela. Em outras palavras, uma chave primária no meu banco dados seria uma variável onde as observações não se repetem ou a combinação de variáveis que tornam as observações únicas. Para exemplicar vamos pegar nossa base de dados e separar em duas, para que posteriormente possamos juntalas. Como em nossa base de dados não temos naturalmente nenhuma chave primária, vamos utilizar a função `mutate(id = row_number())` para criarmos um identificar único para este exemplo. Após isso, vamos dividir a nossa base de dados em duas, mantendo em comum entre elas apenas a nossa chave primária, neste caso, a variável "id"

```
dados <- dados |>
  mutate(id = row_number()) |>
  select(id, everything())#selecionar variavel id e todas as outras

dados1 <- dados[, c(1:24)]

glimpse(dados1)
```

```
Rows: 11,523
Columns: 24
$ id         <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ~
$ dt_notific <date> 2020-05-15, 2020-05-18, 2020-04-30, 2020-05-11, 2020-07-01~
$ dt_sin_pri <date> 2020-05-06, 2020-05-10, 2020-04-20, 2020-05-04, 2020-06-12~
$ dt_nasc    <date> 2003-06-03, 1996-07-07, 1996-03-26, 1986-06-02, 1996-12-11~
$ dt_interna <date> 2020-05-15, 2020-05-15, 2020-04-24, 2020-05-09, 2020-06-30~
$ sem_pri    <int> 19, 20, 17, 19, 24, 24, 26, 27, 28, 24, 14, 29, 28, 10, 36,~
$ sg_uf      <chr> "CE", "PR", "SP", "PA", "DF", "RO", "PI", "RS", "PE", "MA",~
$ id_mn_resi <chr> "MORRINHOS", "CURITIBA", "SAO CAETANO DO SUL", "MARABA", "S~
$ co_mun_res <int> 230890, 410690, 354880, 150420, 530150, 110020, 221100, 431~
$ cs_zona    <fct> NA, urbana, urbana, urbana, urbana, urbana, urbana, urbana,~
$ cs_raca    <fct> parda, branca, ignorado, parda, ignorado, parda, ignorado, ~
$ cs_escol_n <fct> NA, fund2, ignorado, superior, NA, fund2, superior, fund2, ~
```

```
$ idade      <dbl> 16, 23, 24, 33, 23, 35, 31, 17, 22, 29, 28, 22, 27, 25, 26,~
$ cs_gestant <fct> 2tri, 2tri, ignorado, nao, nao, 3tri, 1tri, nao, 3tri, 3tri~
$ puerpera   <fct> NA, nao, sim, sim, sim, nao, NA, sim, NA, sim, NA, NA, sim,~
$ febre      <fct> sim, nao, sim, sim, nao, sim, sim, sim, sim, sim, sim, NA, ~
$ tosse      <fct> sim, nao, nao, sim, nao, sim, sim, sim, NA, sim, sim, sim, ~
$ garganta   <fct> NA, nao, nao, nao, nao, nao, nao, NA, sim, NA, nao, NA, nao~
$ dispneia   <fct> NA, sim, nao, nao, nao, nao, nao, sim, NA, NA, sim, NA, nao~
$ desc_resp  <fct> NA, nao, nao, sim, nao, nao, sim, sim, NA, sim, sim, NA, na~
$ saturacao  <fct> NA, nao, nao, nao, nao, nao, nao, sim, NA, NA, nao, NA, nao~
$ diarreia   <fct> NA, sim, nao, nao, sim, nao, nao, sim, NA, NA, nao, NA, nao~
$ vomito     <fct> NA, nao, nao, nao, nao, nao, nao, nao, NA, NA, nao, NA, nao~
$ fadiga     <fct> NA, NA, NA, NA, NA, nao, NA, NA, NA, NA, nao, NA, NA, nao, ~
```

```r
dados2 <- dados[, c(1, 25:49)]

glimpse(dados2)
```

```
Rows: 11,523
Columns: 26
$ id         <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ~
$ perd_olft  <fct> NA, NA, NA, NA, sim, nao, NA, NA, NA, NA, nao, NA, NA, sim,~
$ perd_pala  <fct> NA, NA, NA, NA, NA, nao, NA, NA, NA, NA, nao, NA, NA, sim, ~
$ dor_abd    <fct> NA, NA, NA, NA, NA, nao, NA, NA, NA, NA, nao, NA, NA, nao, ~
$ cardiopati <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ hematologi <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ hepatica   <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ asma       <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ diabetes   <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ neurologic <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ pneumopati <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ imunodepre <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ renal      <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ obesidade  <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ vacina_cov <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ dose_1_cov <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ dose_2_cov <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ fab_cov_1  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ fab_cov_2  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ dt_entuti  <date> NA, NA, NA, NA, NA, NA, NA, 2020-06-30, NA, 2020-07-12, NA~
$ dt_saiduti <date> NA, NA, NA, NA, NA, NA, NA, 2020-07-29, NA, NA, NA, NA, NA~
$ uti        <fct> nao, nao, nao, nao, nao, nao, nao, sim, nao, sim, nao, nao,~
```

```
$ suport_ven <fct> "sim, nao invasivo", "nao", "nao", "sim, nao invasivo", "na~
$ evolucao   <fct> cura, cura, cura, cura, cura, cura, cura, obito, cura, cura~
$ faixa_et   <chr> "<20", "20-34", "20-34", "20-34", "20-34", ">=34", "20-34",~
$ dias_uti   <drtn> NA days, NA days, NA days, NA days, NA days, NA days, NA d~
```

Em "dados1" selecionamos as colunas de 1 até a 24, onde a coluna 1 é a variável "id". Em dados 2 selecionamos a coluna depois e as colunas de 25 até a 49. Agora temos dois banco de dados e precisamos juntá-los.

Há algumas funções de combinação de duas bases de dados no pacote `dplyr`. Elas recebem três argumentos: a primeira base a ser declarada `(x=)`, a segunda base a ser declarada `(y=)` e a variável de identificação informada no argumento `by=`. Aqui estão as funções mais úteis:

- `left_join()` - retorna todas as linhas da base de dados no argumento `x` e todas as colunas das duas bases de dados. Linhas da base de dados de `x` sem correspondentes em `y` receberão `NA` na base de dados combinada.
- `right_join()` - retorna todas as linhas da base de dados no argumento `y` e todas as colunas das duas bases de dados. Linhas da base de dados de `y` sem correspondentes em `x` receberão `NA` na base de dados combinada.
- `full_join()` - retorna todas as linhas e todas as colunas de `x` e de `y`. As linhas sem correspondência entre as bases receberão `NA` na base de dados combinada.
- `inner_join()` - filtra a base de dados no argumento `x` apenas onde tem valores correspondentes na base de dados no argumento `y` e todas as colunas das duas bases de dados.
- `semi_join()` - filtra a base de dados no argumento `x` apenas onde tem valores correspondentes na base de dados no argumento `y`, mantendo apenas as colunas da base de dados de `x`.
- `anti_join()` - filtra a base de dados no argumento `x` para incluir apenas valores que **não possuem** correspondências na base de dados no argumento `y`.

Assim sendo, no nosso exemplo, tanto as funções `left_join()`, `right_join()`, `full_join()` e `inner_join()` retornarão a mesma combinação, pois "dados1" e "dados2" possuem exatamente os mesmos indivíduos, ou seja, não há nenhuma linha que esteja em uma das bases de dados e que não está na outra. Este cenário foi um pouco mais simples, mas pense que no dia a dia você irá encontrar bases onde você precisará encontrar chaves primarias entre elas. Além disso, varios problemas podem vir acompanhados, por exemplo, imagine que para juntar duas bases você utilizará uma chave formada pela combinação de duas variáveis: UF e Município. Em uma base a sua UF está no formato de sigla e na outra está sendo representada pelo código da UF atribuido pelo IBGE. Já na variável de Município, Em uma base os dados estão todos padronizados, maiúsculos e sem acentuação, já na outra base está no formato "padrão" com a

primeira letra maiúscula e acentuação. Veja que será necessário um bom tratamento de dados para pode juntar essas bases. Voltando para o nosso exemplo, vamos a prática.

```
dados_todos <- full_join(dados1, dados2, by=c("id"))

glimpse(dados_todos)
```

```
Rows: 11,523
Columns: 49
$ id          <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ~
$ dt_notific <date> 2020-05-15, 2020-05-18, 2020-04-30, 2020-05-11, 2020-07-01~
$ dt_sin_pri <date> 2020-05-06, 2020-05-10, 2020-04-20, 2020-05-04, 2020-06-12~
$ dt_nasc    <date> 2003-06-03, 1996-07-07, 1996-03-26, 1986-06-02, 1996-12-11~
$ dt_interna <date> 2020-05-15, 2020-05-15, 2020-04-24, 2020-05-09, 2020-06-30~
$ sem_pri    <int> 19, 20, 17, 19, 24, 24, 26, 27, 28, 24, 14, 29, 28, 10, 36,~
$ sg_uf      <chr> "CE", "PR", "SP", "PA", "DF", "RO", "PI", "RS", "PE", "MA",~
$ id_mn_resi <chr> "MORRINHOS", "CURITIBA", "SAO CAETANO DO SUL", "MARABA", "S~
$ co_mun_res <int> 230890, 410690, 354880, 150420, 530150, 110020, 221100, 431~
$ cs_zona    <fct> NA, urbana, urbana, urbana, urbana, urbana, urbana, urbana,~
$ cs_raca    <fct> parda, branca, ignorado, parda, ignorado, parda, ignorado, ~
$ cs_escol_n <fct> NA, fund2, ignorado, superior, NA, fund2, superior, fund2, ~
$ idade      <dbl> 16, 23, 24, 33, 23, 35, 31, 17, 22, 29, 28, 22, 27, 25, 26,~
$ cs_gestant <fct> 2tri, 2tri, ignorado, nao, nao, 3tri, 1tri, nao, 3tri, 3tri~
$ puerpera   <fct> NA, nao, sim, sim, sim, nao, NA, sim, NA, sim, NA, NA, sim,~
$ febre      <fct> sim, nao, sim, sim, nao, sim, sim, sim, sim, sim, sim, NA, ~
$ tosse      <fct> sim, nao, nao, sim, nao, sim, sim, sim, NA, sim, sim, sim, ~
$ garganta   <fct> NA, nao, nao, nao, nao, nao, nao, NA, sim, NA, nao, NA, nao~
$ dispneia   <fct> NA, sim, nao, nao, nao, nao, nao, sim, NA, NA, sim, NA, nao~
$ desc_resp  <fct> NA, nao, nao, sim, nao, nao, sim, sim, NA, sim, sim, NA, na~
$ saturacao  <fct> NA, nao, nao, nao, nao, nao, nao, sim, NA, NA, nao, NA, nao~
$ diarreia   <fct> NA, sim, nao, nao, sim, nao, nao, sim, NA, NA, nao, NA, nao~
$ vomito     <fct> NA, nao, nao, nao, nao, nao, nao, nao, NA, NA, nao, NA, nao~
$ fadiga     <fct> NA, NA, NA, NA, NA, nao, NA, NA, NA, NA, nao, NA, NA, nao, ~
$ perd_olft  <fct> NA, NA, NA, NA, sim, nao, NA, NA, NA, NA, nao, NA, NA, sim,~
$ perd_pala  <fct> NA, NA, NA, NA, NA, nao, NA, NA, NA, NA, nao, NA, NA, sim, ~
$ dor_abd    <fct> NA, NA, NA, NA, NA, nao, NA, NA, NA, NA, nao, NA, NA, nao, ~
$ cardiopati <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ hematologi <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ hepatica   <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ asma       <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ diabetes   <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ neurologic <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
```

```
$ pneumopati <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ imunodepre <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ renal      <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ obesidade  <fct> NA, nao, nao, nao, nao, nao, NA, nao, NA, NA, NA, NA, nao, ~
$ vacina_cov <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ dose_1_cov <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ dose_2_cov <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ fab_cov_1  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ fab_cov_2  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ dt_entuti  <date> NA, NA, NA, NA, NA, NA, NA, 2020-06-30, NA, 2020-07-12, NA~
$ dt_saiduti <date> NA, NA, NA, NA, NA, NA, NA, 2020-07-29, NA, NA, NA, NA, NA~
$ uti        <fct> nao, nao, nao, nao, nao, nao, nao, sim, nao, sim, nao, nao,~
$ suport_ven <fct> "sim, nao invasivo", "nao", "nao", "sim, nao invasivo", "na~
$ evolucao   <fct> cura, cura, cura, cura, cura, cura, cura, obito, cura, cura~
$ faixa_et   <chr> "<20", "20-34", "20-34", "20-34", "20-34", ">=34", "20-34",~
$ dias_uti   <drtn> NA days, NA days, NA days, NA days, NA days, NA days, NA d~
```

Pronto, temos nossa base completa e aprendemos um pouco sobre manipular dados. O pacote `tidyverse` será um grande aliado seu no R de forma geral. Como mencionado anteriormente, não cobrimos tudo o que é necessário saber para trabalhar com manipulação de dados, é necessário entender a demanda e pesquisar soluções. Saber traduzir o seu problema para que consiga pesquisar com mais facilidade é uma habilidade muito importante. Recomendamos que treine, trabalhe com diferentes tipos de dados, pesquise pacotes, funções, etc. Com o tempo fará com tranquilidade coisas que hoje considera difícil. Lembre-se, a pratica leva a perfeição.

3 Tabulação de dados

Na sessão 5 obtivemos nosso primeiro contato com os vários tipos e estrutura de dados no qual basearemos nossas análises, integer, Character, Complex etc. Além das possíveis formas de disposição desses dados nos atuais softwares de programação, tomando como exemplos os *DataFrames* apresentados para a linguagem R.

3.1 Variáveis

Os objetos apresentados, ou variáveis, podem ser denotados como o armazenamento de informações sobre a característica de interesse a respeito de cada unidade amostral, variáveis socioeconômicas como raça, renda e escolaridade são um ótimo exemplo. As variáveis podem ser divididas em dois tipos:

- Variáveis Qualitativas: cujos valores podem ser separados por categorias não numéricas. Sendo chamadas de variáveis qualitativas ordinais quando há presença de uma ordenação entre as categorias (Ex.: Escolaridade), e variáveis qualitativas nominais caso contrário (Ex.: Raça, Sexo)
- Variáveis Quantitativas: onde os valores são expressos em números resultantes de uma contagem ou mensuração. Podendo ser quantitativas discretas, quando resultam de um conjunto finito ou enumerável de possíveis valores (Ex.: Número de vitórias ou de filhos), ou ainda variáveis quantitativas continuas quando assumem valores em uma escala continua (Ex.: Peso, Altura).

Observe as 10 unidades amostrais para as variáveis da base de dados COVID-19 para melhor compreensão, onde a idade representa variável quantitativa discreta, a raça represeta qualitativa nominal e a escolaridade é relativa a qualitativa ordinal.

	idade_anos	raca	escol
5	39	parda	superior
6	34	branca	superior
8	29	branca	medio
11	28	branca	medio
13	37	parda	fund2
16	27	branca	medio

	idade_anos	raca	escol
17	44	branca	medio
23	31	branca	medio
24	33	amarela	medio
25	25	parda	medio

Podemos olhar uma variável por outra perspectiva, assumindo um outro tipo de classificação. Isso pode soar um pouco estranho a princípio, mas olher o exemplo a seguir para melhor compreensão, considere a variável idade, podemos transformar em faixas de idade para classificação em criança, jovem, adulto e idoso. Observe:

```
#criacao da variavel classificacao
classificacao <- idade_anos |>
  lapply(function(x) ifelse(x < 12, 'crianca',
                            ifelse(x < 25, 'jovem',
                                   ifelse( x < 60 ,'adulto','idoso'))))
#tabela concatenando idade e classificacao
classificacao |>
  unlist() |>
  cbind(idade_anos) |>
  head(10) |> knitr::kable()
```

	idade_anos
jovem	24
adulto	31
adulto	27
jovem	20
adulto	39
adulto	34
adulto	34
adulto	29
adulto	44
adulto	27

Agora, temos uma variável categórica ordinal.

3.2 Como tabular

É perceptível, até mesmo quando trabalhamos com *DataFrames* e matrizes, a forma proposta de visualização e armazenamento dessas variáveis. Por colunas onde cada coluna representa uma das características (no nosso exemplo, idade, raça e escolaridade).

Fazemos isso de forma a facilitar nossa análise, sendo cada linha um indíviduo e, cada uma das observações dentro dessa linha, suas características.

Assim como discutido, podemos obter nossas bases de dados de diversas fontes, como planilhas excel, arquivos .csv, bases SQL, ou até mesmo criá-las no nosso próprio *R script* com a função `data.frame()` como já apresentado. Por ser mais intuitivo e mais utilizado no dia a dia, vamos tomar o excel para exemplificar todo o processo. Você irá notar que o processo é realizado de forma bem simples.

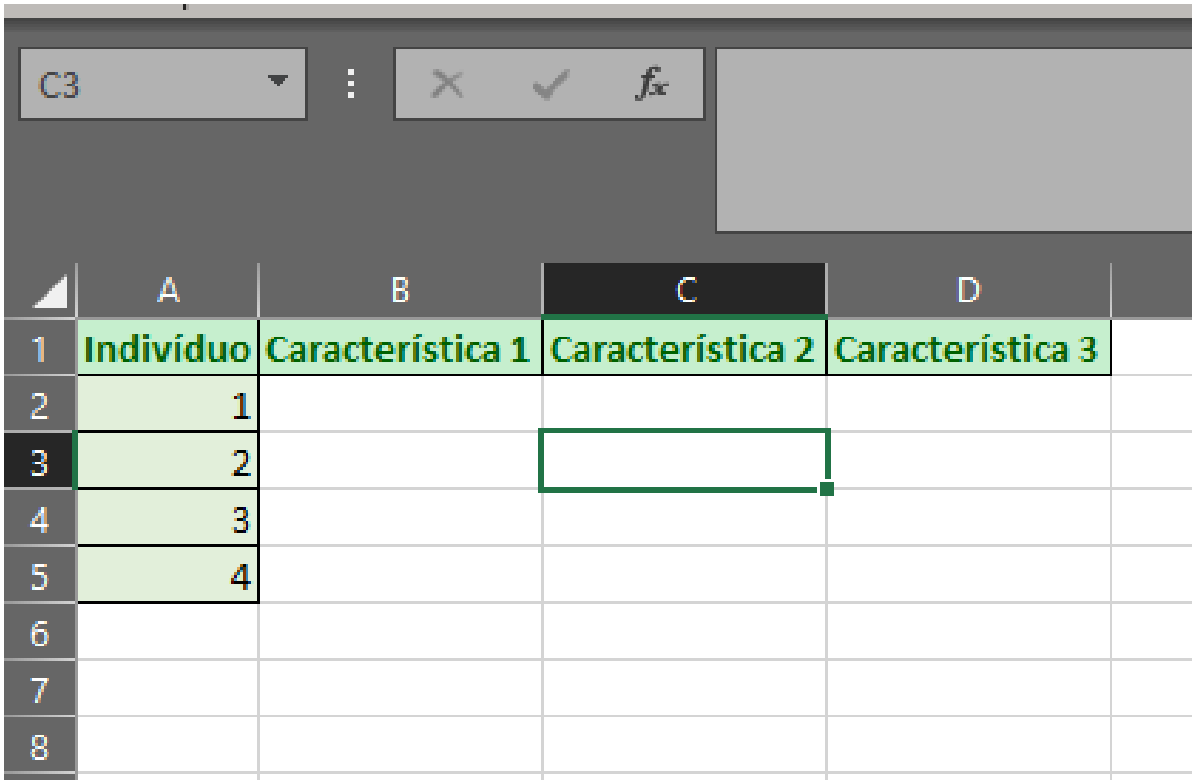


Figura 3.1: Tabulação das variáveis no excel

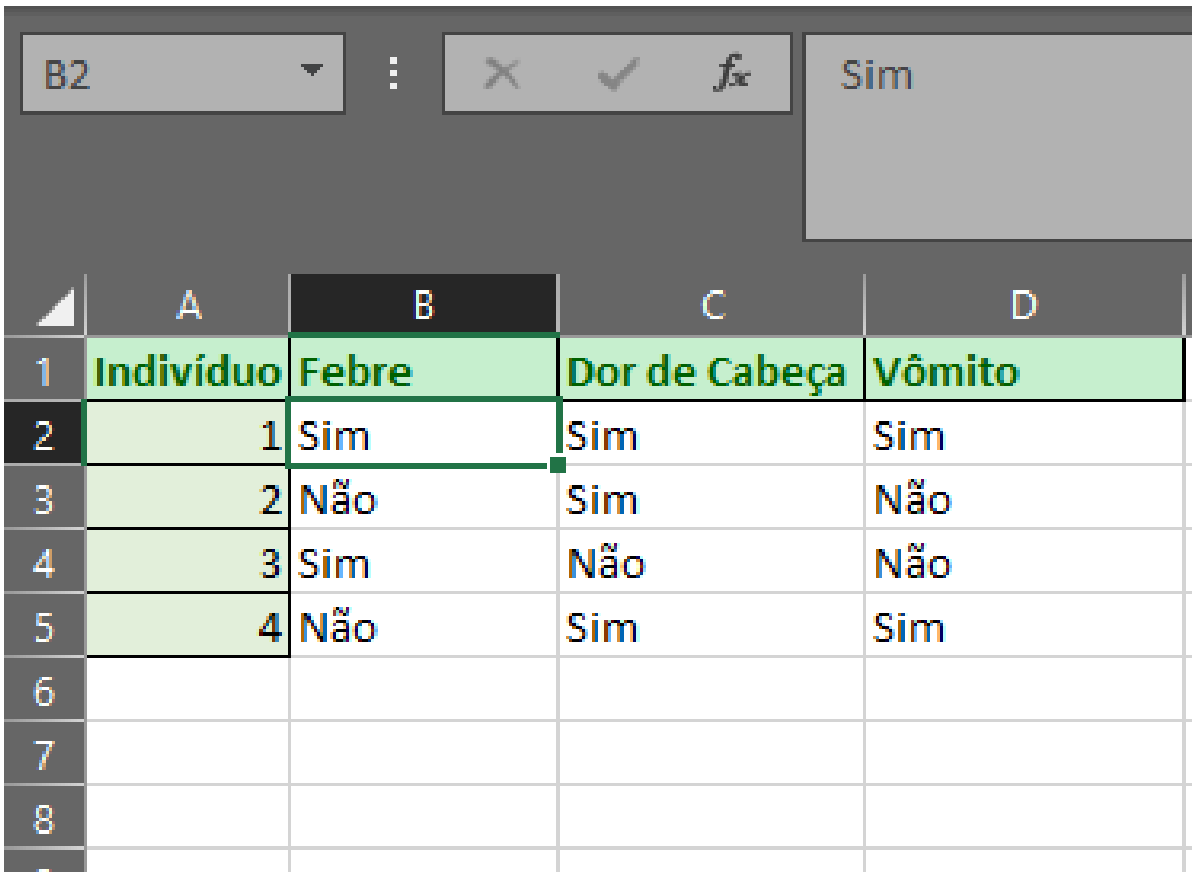
Cada uma das células receberá um valor x referente a alguma característica indicada pela coluna e um indivíduo representado pela linha, em nosso caso temos 3 características para cada uma das 4 observações.

3.3 Alguns problemas no meio do caminho

É valido ressaltar que é possível se deparar com alguns problemas que talvez possam vir a ser solucionados da maneira errada.

A forma como tabulamos nossos dados pode vir a ser um facilitar ou empecilho em nossas análises, um belo exemplo é a forma citada anteriormente de classificação dos dados ou transformação para que sejam salvos em alguma outra categoria, como faixa etária ou idade.

Outro problema é quando trabalhamos com dados que possas vir a ter mais de uma resposta. Por exemplo: Quais sintomas estava sentindo? O melhor a se fazer nesse caso é criar uma coluna para cada um dos possíveis sintomas.



	A	B	C	D
1	Indivíduo	Febre	Dor de Cabeça	Vômito
2	1	Sim	Sim	Sim
3	2	Não	Sim	Não
4	3	Sim	Não	Não
5	4	Não	Sim	Sim
6				
7				
8				

Figura 3.2: Mais de uma opção de escolha na variável

Uma outra forma seria:

Devemos lembrar sempre de anexar um `ID` ou forma de identificação única para cada uma das observações. É possível criar uma ou trabalhar com alguma já existente, um exemplo de uma já existente é o próprio CPF ou RG quando trabalhamos com pessoas.

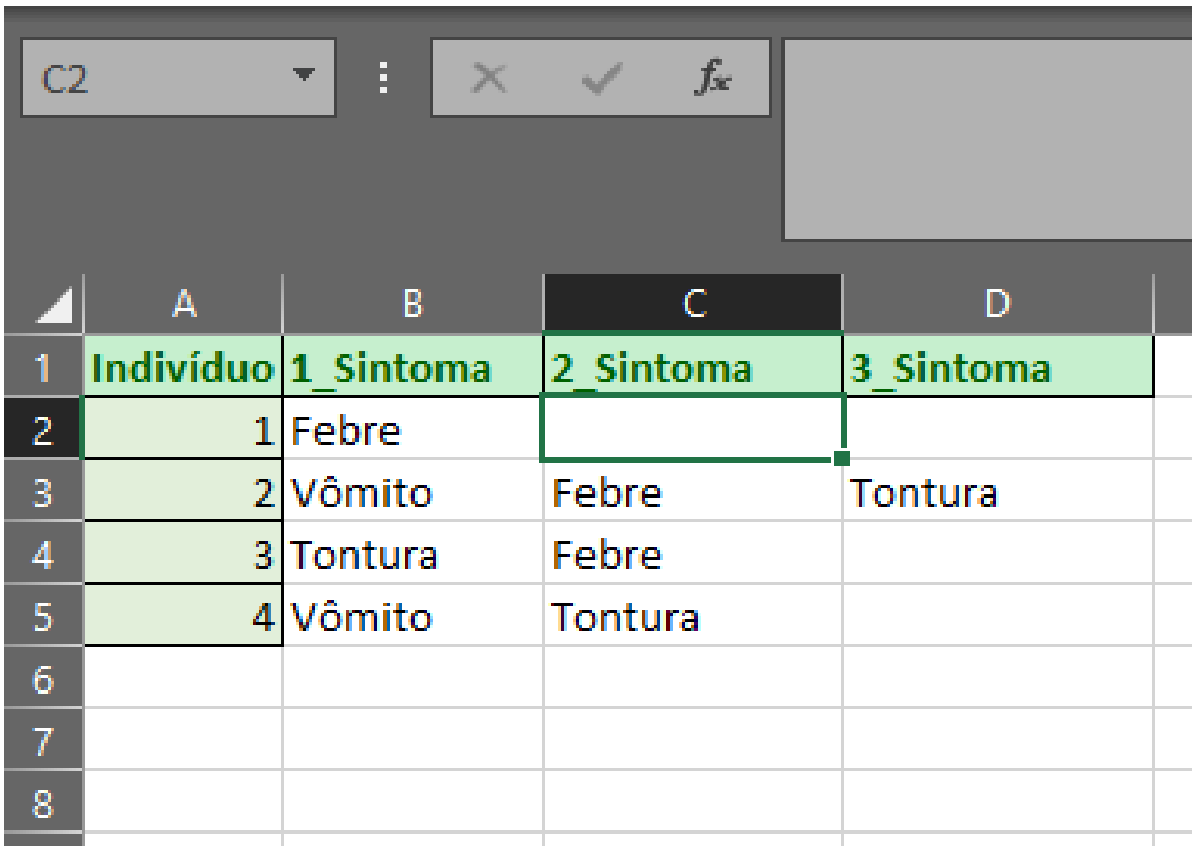


Figura 3.3: Mais de uma opção de escolha na variável, outra forma

Vale ressaltar outras boas práticas ao realizar a tabulação:

- Se trabalhando com Excel ou Softwares parecidos, deixe a planilha apenas com a tabela de dados, evite armazenar na mesma planilha várias informações avulssas que não façam parte da sua tabela;
- No R conseguimos especificar qual planilha de um arquivo `.xlsx` queremos transferir, porém pode vir a ser um pouco confuso as vezes, então é sugerido deixar todas as suas informações em uma única tabela em uma única planilha;
- Padronização é extremamente importante, salve todos os dados para cada coluna em apenas um determinado formato (Ex.: Coluna Idade - *Integer*, Coluna Raça - *Character*), lembrando sempre de manter um padrão de medida (cm, L), variáveis do tipo categórico tambem precisam de padronização (Evite coisas como: Não, nao, n, N, não);
- Cuidado ao classificar dados faltantes, uma prática errada é preencher esses dados com 0, isso pode vir a atrapalhar toda sua análise
- Foi citado CPF como forma de identificação, mas pode haver casos em que teremos mais de uma linha contendo um mesmo indivíduo dependendo do nosso tipo de dados. Ou seja, esteja atento para que não haja duplicidade de variável identificadora ou `ID`.

4 Análise exploratória dos dados

Após o tratamento de uma base de dados, em geral, temos o interesse em examinar e estudar as características de cada uma das variáveis presentes no banco, de forma a identificar comportamentos e relações entre as variáveis. Morettin e Singer (2020) apresentam um conjunto de questões que normalmente se tenta responder ao se realizar uma análise dos dados, sendo elas:

i. qual a frequência com que cada valor aparece no conjunto de dados ou seja, qual a distribuição de frequências dos dados?

ii. quais são alguns valores típicos do conjunto de dados, como mínimo e máximo?

iii. qual seria um valor para representar a posição (ou localização) central do conjunto de dados?

iv. qual seria uma medida da variabilidade ou dispersão dos dados?

v. existem valores atípicos ou discrepantes (*outliers*) no conjunto de dados?

vi. os dados podem ser considerados simétricos?

Para responder a essas questões, fazemos uso de um conjunto de técnicas numéricas e gráficas que nos permitem detectar padrões, resumir informação e apresentar visivelmente características das variáveis de um conjunto de dados. A essa metodologia denominamos análise descritiva ou análise exploratória de dados e será objeto de estudo desse capítulo. Para ilustrar as técnicas apresentadas, vamos realizar a análise descritiva de algumas das variáveis presentes no banco de dados de COVID-19 em gestantes e puérperas já devidamente tratadas no capítulo anterior.

A decisão por qual técnica empregar para a análise descritiva de uma variável, passa por qualificar corretamente a natureza da variável em estudo em qualitativa nominal, qualitativa ordinal, quantitativa discreta ou quantitativa contínua. Maiores detalhes sobre essa teoria podem ser vistos na Seção XX.

Uma das principais ferramentas de resumo de informações de variáveis qualitativas é a organização tabular que fornece a distribuição de frequências de cada variável, conforme apresentado na Seção XX.

Em se tratando de variáveis quantitativas, utilizar uma tabela de frequências para resumir informações da variável, especialmente nos casos de variáveis contínuas, pode não ser a melhor

estratégia, visto que é comum obter frequências muito pequenas (em geral, 1) para os diferentes valores da variável, não atingindo o propósito de resumir a informação. Nesse sentido, vamos apresentar aqui medidas-resumo que podem ser utilizadas para variáveis quantitativas.

4.1 Medidas-resumo

Uma medida-resumo é uma construção matemático/estatística que tenta capturar em um único número um comportamento presente nos dados. Quatro grandes grupos de medidas podem ser considerados para resumir variáveis quantitativas, são elas: posição, dispersão, assimetria e curtose. Vejamos agora que medidas são essas e como interpretá-las.

4.1.1 Medidas de posição

As medidas de posição, como o nome diz, indicam posições de interesse de valores da variável. Por exemplo, se idade é a variável de interesse investigada em um grupo de pessoas, e se quer trazer um informação resumida dela no grupo, apresentar a menor e a maior idade encontrada, valores típicos da idade no grupo, são exemplos de medidas de posição.

De maneira geral, vamos explorar aqui as seguintes medidas posição: valor mínimo, valor máximo, percentis e medidas de tendência central, tais como moda, média e mediana. Os valores mínimo e máximo de uma variável quantitativa estão relacionados, respectivamente, com o menor e o maior valor observado da variável analisada.

Medidas que buscam descrever um valor típico que a variável apresenta são chamados de medidas de tendência ou posição central. Mas o que seria uma valor típico? Como podemos definir isso? A resposta não é única e, por isso, existem diferentes medidas de tendência central. Por exemplo, se o valor típico considerado for aquele que mais se repete no conjunto de dados para variável, o que temos é a moda. Se o valor típico for aquele que ocupa uma posição central no conjunto de dados, de tal forma que 50% dos dados observados estão abaixo desse valor e os demais 50% estão acima, o que temos é a mediana. Agora, se o valor típico considerado for pensado como um ponto de equilíbrio das observações da variável, então temos a média.

Por definição, a medida estatística moda corresponde aos(s) valor(es) mais frequente(s) do conjunto de dados observados para uma variável. Conjunto de dados que não apresentam valores repetidos são considerados amodais. Um conjunto de dados é bimodal se tiver duas modas, indicando que não apenas um único valor, mas dois valores do conjunto de dados apresentam frequências igualmente mais altas que os demais valores. Usando de mesmo raciocínio, havendo três ou mais valores modais em um conjunto de dados, dizemos que o conjunto de dados é trimodal ou multimodal, respectivamente. Vale citar que a moda também pode ser obtida para variáveis qualitativas.

A média é a medida obtida ao somar todos os valores da variável e dividí-la pela quantidade de dados observados. Matematicamente, considere x_1, x_2, ..., x_n as observações de uma variável X, assim a média é definida como:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}. \tag{4.1}$$

Para entender essa medida como ponto de equilíbrio, vamos representar cada valor observado como pesos de mesma massa e distribuí-los sobre uma reta de massa desprezível nas posições referentes aos valores da variável em questão. Nosso objetivo agora é encontrar um ponto de apoio nessa reta de tal forma que ela e os pesos corretamente posicionados nela fiquem perfeitamente equilibrados, similar a uma balança. A média é o único local em que se pode localizar o ponto de apoio na reta de forma a obter um perfeito equilíbrio da reta e dos pesos. Para ilustrar essa ideia, apresentamos a seguir uma representação gráfica considerando o subconjunto da variável idade {22, 28, 29, 34, 34, 35, 36, 36, 37, 39}, cuja média é 33.

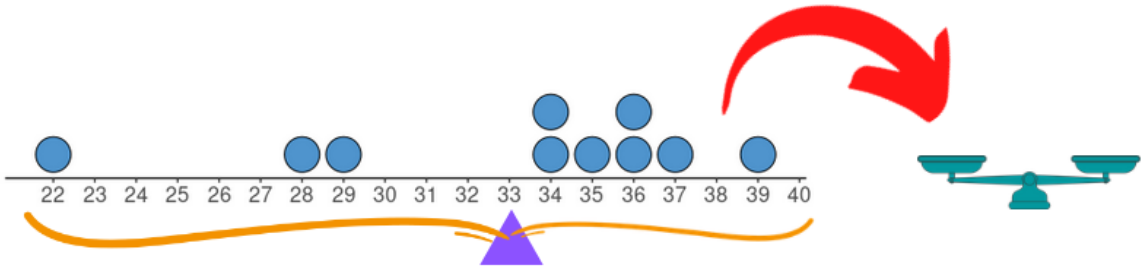


Figura 4.1: (a) Ponto de equilíbrio na média

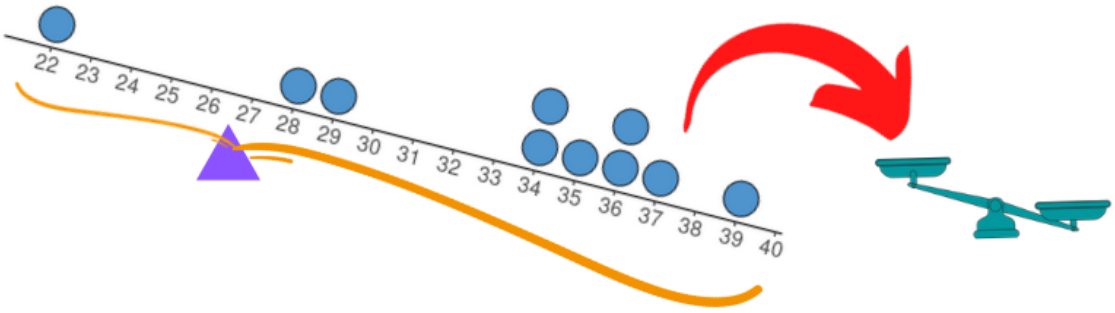


Figura 4.2: (b) Ponto de equilíbrio fora da média

Figura 4.3: Apresentando a média como ponto de equilíbrio

No início dessa seção, apresentamos a noção intuitiva do que representa o valor mediano, mas não como obtê-lo formalmente. A construção dessa medida passa por organizar os dados de maneira crescente e calcular a posição central dos dados via $\frac{n+1}{2}$, em que n representa o

tamanho do conjunto de dados relacionada a variável de interesse. O valor mediano é o valor na amostra ordenada que ocupa a posição $\frac{n+1}{2}$.

Quando n é ímpar, a expressão $\frac{n+1}{2}$ vai sempre gerar um valor inteiro, facilitando a obtenção da mediana. Por exemplo, o subconjunto a seguir é formado por nove valores retirados da variável idade, sendo eles: 21, 28, 24, 22, 31, 26, 22, 38, 16.

Ordenando esse subconjunto, obtemos 16, 21, 22, 22, 24, 26, 28, 31, 38.

Como $n = 9$, a posição em que se encontra a mediana será $\frac{9+1}{2} = 5$. Assim, a mediana será 24, pois é o valor que está na quinta posição do subconjunto ordenado.

Se n é par, a expressão $\frac{n+1}{2}$ gerará um valor não inteiro que apresenta apenas uma única casa decimal após a vírgula igual a 5. Por exemplo, se a variável idade apresenta apenas 8 valores então $n = 8$ e a posição em que a mediana está localizada é dada por $\frac{n+1}{2} = \frac{8+1}{2} = 4,5$. Como inferir um valor para a mediana quando a posição que ela ocupa é decimal? Note que a posição $4,5$ está exatamente no meio das posições 4 e 5, então o valor mediano será definido como a média entre os valores que ocupam as posições 4 e 5.

O subconjunto abaixo também consiste de valores retirados da variável idade, porém note que nesse exemplo há 8 valores, ou seja, $n = 8$.

$$27, 16, 31, 43, 26, 42, 17, 40.$$

Ao ordenarmos, temos:

$$16, 17, 26, 27, 31, 40, 42, 43.$$

A mediana será o valor que está na posição $\frac{8+1}{2} = 4,5$. Logo, visto que a mediana está entre os valores que ocupam a quarta e quinta posição, corresponde à média entre esses valores, sendo $\frac{27+31}{2} = 29$.

Com ideia correlata a mediana, podemos apresentar medidas de posição não centrais, as quais denominamos quantis ou percentis. O percentil 20, por exemplo, é o valor da variável em que 20% das observações no conjunto de dados apresentam valores menores ou iguais a ele. Por consequência, as restantes 80% das observações possuem valores acima do percentil 20. De maneira geral, podemos definir o percentil de ordem p como o valor da variável em que $100p\%$ $(0 < p < 1)$ das observações estão à sua esquerda, ou seja, são menores ou iguais que ele.

Alguns percentis destacam-se por serem muito utilizados na análise de dados, não só numericamente como graficamente. Esses percentis são conhecido como quartis e basicamente dividem o conjunto de dados em 4 partes de mesmo tamanho. O primeiro quartil (Q_1) é o percentil 25, o segundo quartil (Q_2) é o percentil 50 e o terceiro quartil (Q_3) é o percentil 75. Vale notar que o segundo quartil é a mediana. De posse desses valores, como veremos mais a

frente nesse capítulo, iremos construir o gráfico do tipo *boxplot*, bastante utilizado na análise de dados da saúde.

Ainda com respeito aos percentis, outro termo comum na literatura é o decil que refere-se a divisão em 10 partes de mesmo tamanho do conjunto de dados associado a variável analisada. O primeiro decil, por exemplo, é o percentil 10 e o sexto decil é o percentil 60.

Todas as medidas de posição aqui apresentadas tem em comum terem a mesma unidade de medida dos valores da variável observada, o que traz bastante interpretabilidade.

4.1.2 Medidas de dispersão

Por mais que as medidas de posição apresentadas sejam muito úteis na análise dados, elas por si só não se bastam como medidas resumo das observações de uma variável em um conjunto de dados. É possível construir diferentes conjuntos de dados para uma mesma variável que apresentam os mesmo valores de medida central (média, mediana e moda), mas tem comportamentos absolutamente diferentes. Por exemplo, veja a figura a seguir.

(a) Dados: 2 ,3, 5 , 5, 7, 8.

(b) Dados: 5,5,5,5.

Figura 4.4: Exemplos de conjuntos de dados com mesma média, moda e mediana.

Os dois conjuntos de dados apresentam os mesmos valores de média, mediana e moda. O que diferencia os dois conjuntos? O quão diferentes ou parecidos são as observações entre si em cada conjunto da variável. Na Figura Figura **??**, notamos que os quatro valores observados são iguais entre si e que, portanto, as observações nesse conjunto não variam, diferente do que ocorre para os dados que geraram o gráfico da Figura Figura **??**. Medidas de dispersão ou variabilidade são as medidas responsáveis por quantificar o quão diferente são os dados entre si. De forma bastante intuitiva temos que se os dados observados da variável não variam, então a medida de dispersão dela é zero e, caso haja diferenças entre os valores observados, então essa medida vai ser um valor positivo. Quanto maior a medida de variabilidade, mais diferente são os dados observados da variável entre si.

Não existe uma única medida de dispersão na literatura, aqui vamos considerar as seguintes medidas: amplitude, intervalo interquartil, variância, desvio padrão e coeficiente de variação.

De fácil obtenção e interpretabilidade, a amplitude é a diferença entre o valor máximo e o valor mínimo da variável analisada no conjunto de dados e nos dá uma ideia do intervalo de variação dos dados. Uma desvantagem é que essa medida é absolutamente influenciada pela presença de valores discrepantes ou *outliers*. O intervalo interquartil é uma medida mais robusta do que a amplitude intervalar e é calculada como a diferença entre o terceiro e o primeiro quartil, ou seja, é a amplitude entre os 50% dos dados centrais.

Por mais informativas que sejam as medidas de amplitude e intervalo interquartil, queremos uma medida de dispersão que não considere apenas dois valores da amostra (mínimo e máximo ou primeiro e terceiro quartis) e sim todos os dados. Uma medida bastante intuitiva seria considerar a soma dos desvios de cada uma das observações em torno da média. Mas aí temos um problema: a soma dos desvios da média é sempre zero! Isso acontece porque sempre há desvios positivos e negativos que quando somados se anulam. Uma solução para essa questão é considerar alguma função que considere apenas o valor do desvio e não o seu sinal. Uma função candidata é a função quadrática (lembre que, por exemplo, $(-2)^2 = 4$). Nessa construção surge a variância: soma dos desvios quadrados dividida pelo total de observações (n), ou seja, a média dos desvios quadrados. Assim, a variância quantifica o quanto os dados estão dispersos da média, em média.

Matematicamente, considere x_1, x_2, ..., x_n as observações de uma variável X e $\bar{x}$ a média observada dessa variável. A variância seria calculada como:

$$\text{Var(X)} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}. \tag{4.2}$$

Por mais intuitiva que seja essa construção, programas como o R e similares utilizam em sua análise uma versão modificada do cálculo da variância acima apresentado, em que a soma dos desvios quadrados é dividida por $n-1$, não por n. Justificativas para isso se devem a propriedades inferenciais. A maioria dos conjuntos de dados considerados nos estudos referem-se a análise de amostras de uma população e não a análise de todos os elementos de uma população. Ao mesmo tempo, um dos principais objetivos da análise estatística é fazer análises para a população e não apenas para a amostra considerada no estudo. Basicamente, se temos interesse de conhecer o valor médio de uma variável na população (μ), na impossibilidade de analisar todos os elementos dela e obter a medida, o fazemos de forma aproximada investigando o valor médio dessa variável na amostra ($\bar{x}$). Esse mesmo raciocínio ocorre para a variância, na impossibilidade de obter a variância da variável para todos os elementos da população (σ^2), analisamos essa medida via amostra, o caso é que é possível mostrar que para amostras de tamanho pequeno, a variância apresentada em (Equação **??**) não aproxima-se bem do valor de σ^2. Matematicamente, é possível mostrar que tal dificuldade é contornada fazendo uso do divisor igual a $n-1$ em (Equação **??**). Na literatura esse cálculo muitas vezes é denominado como variância amostral e representado pelo símbolo S^2 de tal forma que

$$S^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}. \tag{4.3}$$

Vale ressaltar também que a medida que se considera tamanhos de amostra maiores, calcular a variância com divisor n ou $n-1$ torna-se indiferente.

Como a unidade de medida da variância é o quadrado da unidade de medida da variável correspondente, convém definir outra medida de dispersão que mantenha a unidade de medida original. Uma medida com essa propriedade é a raiz quadrada da variância, conhecida por desvio padrão.

Caso o interesse seja calcular e comparar a dispersão entre variáveis com unidades dimensionais de natureza diferente, por exemplo, comprimento (em metros) e massa (em kg), não convém utilizar as medidas de dispersão apresentadas anteriormente pois todas as medidas apresentadas carregam consigo a unidade de medida considerada para a variável. Nesse caso, podemos fazer uso do coeficiente de variação (CV) para cada uma das variáveis analisadas, já que o CV é uma medida de dispersão relativa adimensional, calculada via razão entre o desvio-padrão e a média observada para a variável e quanto maior o seu valor, maior a dispersão dos dados em termos relativos a média.

4.1.3 Medidas de assimetria e curtose

Além das medidas de posição e variabilidade, existe um conjunto de medidas dedicadas a explorar a forma da distribuição de frequências dos dados. Especificamente aqui estudaremos algumas: coeficientes de assimetria e de curtose e variações destas.

Como boa parte dos estudos na área de saúde é realizado através de amostras da variável de interesse na população, vamos precisar definir os momentos amostrais centrais que serão ferramenta fundamental para a construção dos coeficientes de assimetria, curtose e seus derivados. Por definição, o momento amostral centrado (na média) de ordem r é dado por

$$m_r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^r}{n},\ r = 1, 2, \cdots.$$

A versão populacional do momento centrado de ordem r é expressa por $\mu_r = \frac{\sum_{i=1}^{n}(x_i-\mu)^r}{N}$, em que $r = 1, 2, \cdots$ e μ e N referem-se, respectivamente, a média da variável de interesse e a quantidade de elementos investigados na população.

Dessa forma, o coeficiente de assimetria amostral é dado por $\frac{m_3}{m_2^{3/2}}$. Populações cuja a distribuição da variável é simétrica apresentam coeficiente de assimetria igual a zero. Distribuições assimétricas à direita apresentam valores positivos de coeficiente de assimetria para a variável analisada populacionalmente, assim como distribuições assimétricas à esquerda apresentam coeficiente de assimetria negativo.

Analogamente, $\frac{\mu_3}{\mu_2^{3/2}}$ é o coeficiente de correlação populacional.

Populações cuja a distribuição da variável é simétrica apresentam coeficiente de assimetria igual a zero. Distribuições assimétricas à direita apresentam valores positivos de coeficiente de assimetria para a variável analisada populacionalmente, assim como distribuições assimétricas à esquerda apresentam coeficiente de assimetria negativo.

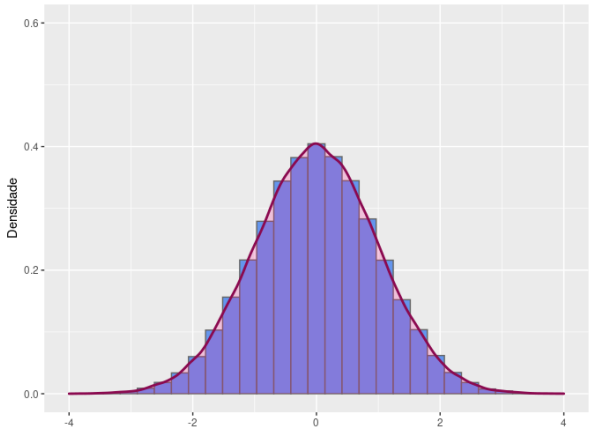


Figura 4.5: (a) Simétrico

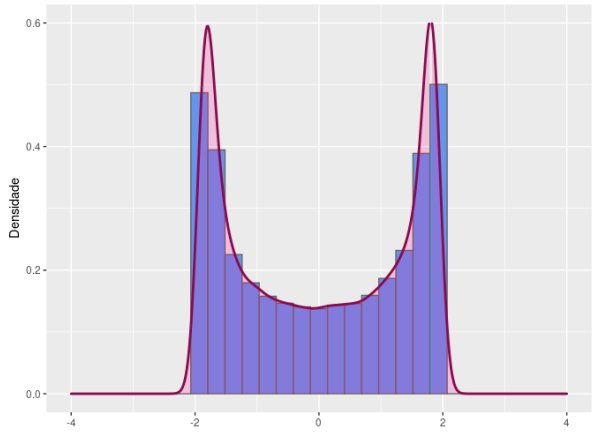


Figura 4.6: (b) Simétrico

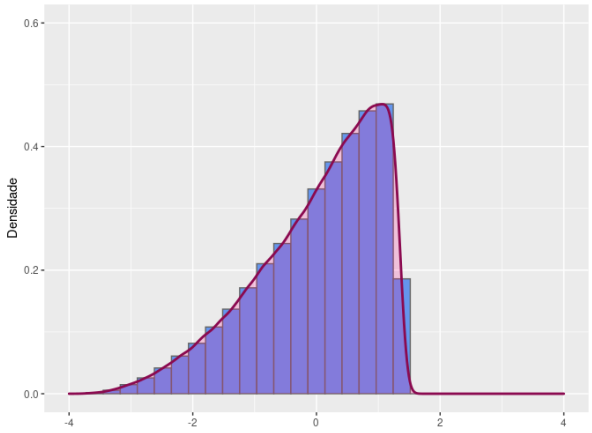


Figura 4.7: (c) Assimétrico à esquerda

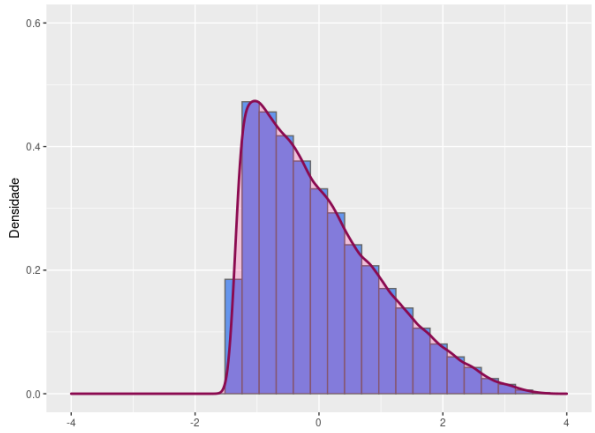


Figura 4.8: (d) Assimétrico à direita

Figura 4.9: Histogramas e funções de densidade

Coeficientes de assimetria amostrais diferentes de zero devem ser interpretados com cautela, uma vez que por se tratar de uma amostra não significa que necessariamente o comportamento da variável na população seja assimétrico. Testes estatísticos devem ser realizados para avaliar a hipotese de simetria da variável na população.

Ainda com respeito a forma da distribuição da variável, podemos avaliar o comportamento em suas caudas através do coeficiente de curtose amostral que se define por $\frac{m_4}{m_2^2}$. Distribuições de variáveis com valor de curtose igual a 3 são denominadas mesocúrticas. Tomada muitas vezes como referência, a distribuição normal apresenta coeficiente de curtose igual a três. Distribuições com coeficiente de curtose menores que 3 são denominadas platicúrticas e apresentam caudas mais leves ("finas") do que a da distribuição normal. Distribuições com coeficiente de curtose maiores que 3 são denominadas leptocúrticas e apresentam caudas mais pesadas ("grossas") do que a da distribuição normal. A distribuição t-Student é um exemplo de distribuição leptocúrtica.

Na literatura é muito comum ser apresentado uma variante do coeficiente de curtose denominada excesso de curtose. Esse excesso é avaliado em relação a curtose do modelo normal por isso seu valor é calculado fazendo o coeficiente de curtose subtraído de 3. Dessa forma, o excesso de curtose em distribuições mesocúrticas é igual a zero, em distribuições leptocúrticas é maior que zero e em distribuições platicúrticas é menor que zero.

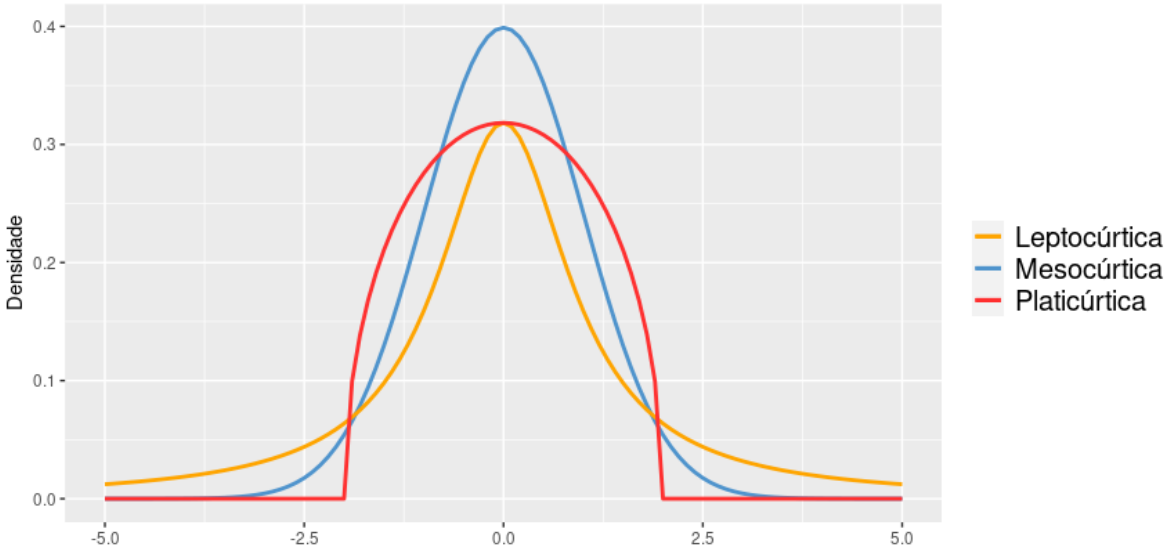


Figura 4.10: Exemplo de funções de densidade com diferentes medidas de curtose.

Ressalva similar feita ao coeficiente de assimetria deve ser considerado para o coeficiente de curtose ou de excesso de curtose. Um coeficiente de curtose amostral diferente de três ou, de forma equivalente, com excesso de curtose amostral diferente de zero, não implica necessariamente que a distribuição da variável na população possui caudas mais leves ou mais pesadas do que a da distribuição normal. Para que se possa fazer tal afirmação é necessária a realização de testes estatísticos inferenciais. Os coeficientes amostrais de assimetria e curtose tão somente nos dão uma medida da forma da distribuição de frequências e *insights* do comportamento da variável na população, que devem ser verificados via análise inferencial

estatística.

No R, para obter essas medidas resumo vamos utilizar a função `descr` também do pacote `summarytools`. No comando abaixo pedimos ao R as medidas descritivas da variável quantitativa "idade".

```
descr(dados$idade)
```

```
Descriptive Statistics
dados$idade
N: 11523

                      idade
----------------- ----------
             Mean      30.25
          Std.Dev       7.04
              Min      10.00
               Q1      25.00
           Median      30.00
               Q3      35.00
              Max      55.00
              MAD       7.41
              IQR      10.00
               CV       0.23
         Skewness       0.17
      SE.Skewness       0.02
         Kurtosis      -0.10
          N.Valid   11514.00
        Pct.Valid      99.92
```

Note que os nomes de algumas das estatísticas apresentadas pela função `descr` estão em inglês. $Mean$ refere-se ao valor médio da variável analisada, $Std.Dev$ corresponde ao desvio-padrão, IQR é o símbolo para o intervalo interquartil, MAD é o desvio-médio absoluto, $Skewness$ é o coeficiente de assimetria, $SE.Skewness$ é o erro-padrão do coeficiente de assimetria, $Kurtosis$ é o coeficiente de excesso de curtose e $N.Valid$ e $Pct.Valid$ correspondem, respectivamente, ao número de observações válidas e seu percentual no conjunto de dados considerado para a variável.

Especificamente para a variável idade, podemos notar que das 11523 observações, apenas 11514 (99.92 %) foram consideradas válidas. Isso acontece porque nesse conjunto de dados, 9 pessoas não declararam a idade, ficando com a casela vazia (NA). Todas as medidas-resumo foram calculadas considerando apenas as observações válidas. Sendo assim, algumas análises que podem ser realizadas para a variável idade através da função `descr` são que a idade média