# Introdução ao R

Raimundo Marciano de Freitas Neto

2021-02-16

# Contents

# Chapter 1

# Prerequisites

This is a *sample* book written in **Markdown**. You can use anything that Pandoc's Markdown supports, e.g., a math equation $a^2 + b^2 = c^2$.

The **bookdown** package can be installed from CRAN or Github:

```
install.packages("bookdown")
# or the development version
# devtools::install_github("rstudio/bookdown")
```

Remember each Rmd file contains one and only one chapter, and a chapter is defined by the first-level heading `#`.

To compile this example to PDF, you need XeLaTeX. You are recommended to install TinyTeX (which includes XeLaTeX): https://yihui.org/tinytex/.

# Chapter 2

# O que é o R?

R é uma linguagem de programação muito utilizada para o desenvolvimento de estatísticas e cálculos matemáticos. Entretanto, enquanto linguagem de programação, ela possui muitos usos, como a raspagem de sites, a automatização de download de arquivos, e a produção de dashboards e relatórios, sendo útil para a coleta de dados e para o reporte das análises, além de ser gratuita. Mas o que significa ser uma linguagem de programação? Basicamente, você irá dar ordens ao computador por meio de comandos. A vantagem de aprender a estruturar os comandos (ao invés de usar um programa estatístico já consolidado, como o Stata) é que você pode customizar a vontade suas análises. Os programas estatísticos trazem muitas opções pré-prontas. Por exemplo, no Stata é comum que você possa clicar em uma janela e solicitar um quadro com as estatísticas descritivas (média, moda, mediana) de um banco de dados. Entretanto, qualquer funcionalidade fora do padrão oferecido por aqueles pacotes precisará ser gerada pelo próprio usuário (e quando você usa algum recurso como o ssc install é porque alguém gastou um tempinho fazendo isso).

Aliás, o Python (outra linguagem de programação) possui funcionalidades bem similares ao R, embora o modo de escrever os programas seja bem diferente. Imagine que o Português e o Espanhol têm o mesmo propósito - a comunicação - e que mesmo tendo uma mesma raiz (e inclusive diversos vocábulos bem parecidos), possuem algumas regras estruturais bem diferentes, o que inclui, por exemplo, as regras de acentuação e o som produzidos pelas letras: qual a diferença de cajá (PT-BR) para caja (ES); qual a diferença de pastel (PT-BR) para pastel (ES)?

É preferível que o analista iniciante dedique-se a uma das duas (R ou Python). Posteriormente, aprender a outra (ou mais alguma que esteja despontando no mercado, como a Julia) será uma tarefa bem mais simples. Isso se deve ao fato de que diversos elementos da programação se repetem entre as diferentes linguagens, especialmente os conceitos. Um IF tem o mesmo propósito em R ou

em Python ou em Java ou em C++, embora a forma como você deve explicar ao computador o que fazer com esse IF será bem diferente em cada linguagem.

You can label chapter and section titles using `{#label}` after them, e.g., we can reference Chapter 2. If you do not manually label them, there will be automatic labels anyway, e.g., Chapter 7.

Figures and tables with captions will be placed in `figure` and `table` environments, respectively.

```
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```

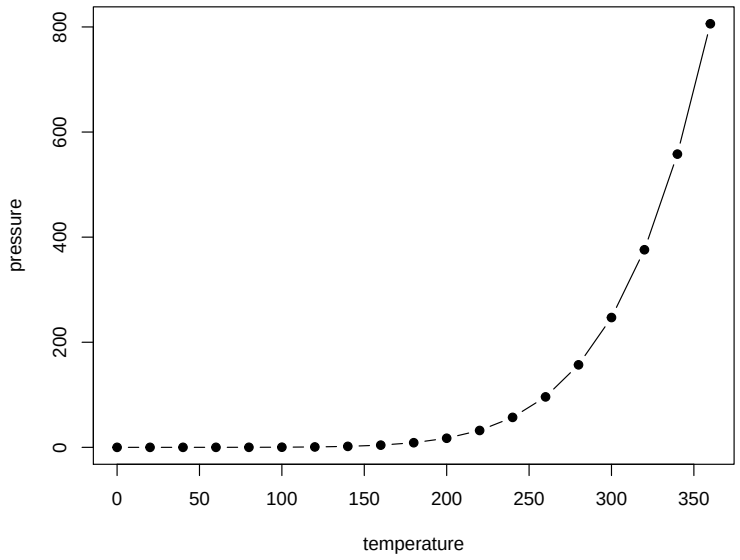


Figure 2.1: Here is a nice figure!

Reference a figure by its code chunk label with the `fig:` prefix, e.g., see Figure 2.1. Similarly, you can reference tables generated from `knitr::kable()`, e.g., see Table 2.1.

```
knitr::kable(
  head(iris, 20), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

You can write citations, too. For example, we are using the **bookdown** package (Xie, 2020) in this sample book, which was built on top of R Markdown and **knitr** (Xie, 2015).

Table 2.1: Here is a nice table!

| Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| 5.4 | 3.9 | 1.7 | 0.4 | setosa |
| 4.6 | 3.4 | 1.4 | 0.3 | setosa |
| 5.0 | 3.4 | 1.5 | 0.2 | setosa |
| 4.4 | 2.9 | 1.4 | 0.2 | setosa |
| 4.9 | 3.1 | 1.5 | 0.1 | setosa |
| 5.4 | 3.7 | 1.5 | 0.2 | setosa |
| 4.8 | 3.4 | 1.6 | 0.2 | setosa |
| 4.8 | 3.0 | 1.4 | 0.1 | setosa |
| 4.3 | 3.0 | 1.1 | 0.1 | setosa |
| 5.8 | 4.0 | 1.2 | 0.2 | setosa |
| 5.7 | 4.4 | 1.5 | 0.4 | setosa |
| 5.4 | 3.9 | 1.3 | 0.4 | setosa |
| 5.1 | 3.5 | 1.4 | 0.3 | setosa |
| 5.7 | 3.8 | 1.7 | 0.3 | setosa |
| 5.1 | 3.8 | 1.5 | 0.3 | setosa |

## 2.1 Scripts

# Chapter 3

# Variáveis

Os dados são inseridos no R como variáveis. Eles podem ser importados (trazidos de um arquivo para o R) ou digitados no próprio console. Quando você deseja armazenar um valor na memória, você deve atribuí-lo, usando o sinal = ou <-.

```
x = 1
print(x)
```

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

```
y <- 1+1
y
```

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

```
texto = "Esse é um texto. Observe as aspas duplas"
assim = 'Também podem ser usadas aspas simples'
sugestao = 'Use aspas simples sempre que "possível" para delimitar a variável. Os
textos em português costumam usadas aspas duplas.'
print(sugestao)
```

```
## [1] "Use aspas simples sempre que \"possível\" para delimitar a variável. Os \ntextos em portu
```

```
writeLines(sugestao)
```

```
## Use aspas simples sempre que "possível" para delimitar a variável. Os
## textos em português costumam usadas aspas duplas.
```

O R é uma linguagem que automaticamente reconhece o tipo do dado que foi inserido. Por uma questão de comparação, se você estivesse usando uma linguagem *tipada*, como o C++, você precisaria declarar explicitamente o tipo da variável que está sendo criada. Assim, seria um erro você criar uma variável usando

```
dois = 2
```

O C++ não entende o que você está querendo dizer com isso. Você precisaria informar que existe uma variável do tipo *integer* (simplesmente *int*) chamada *dois* e cujo valor é o número inteiro 2. Outra particularidade do R é que as linhas de código não precisam ser finalizadas com um ponto-e-vírgula (;), como acontece no C++.

```
int dois = 2;
```

Além disso, as variáveis em R são mutáveis, inclusive quanto ao tipo. Isso quer dizer que uma vez definidas (ou atribuídas), podem ter seus valores e tipo modificados. Assim, uma variável que antes tinha um número, pode passar a ter um texto ou um booleano.

```
# se a variavel receber um número (integer)
dois <- 2
# e depois receber um texto
dois <- "dois"
# seu conteúdo e seu tipo estarão de acordo com a última atribuição
print(dois)
```

```
## [1] "dois"
```

Por fim, para o R a indentação não é relevante. Há linguagens de programação em que a posição relevante das linhas de código é essencial para determinar se elas fazem parte de um bloco ou não. No R, os blocos são definidos com o uso de chaves ({ }). Ainda iremos explorar com detalhes o uso das condicionais, mas podemos ilustrar o seguinte caso: se o resultado for maior que zero, então, sim, tivemos lucro; caso contrário, não tivemos.

```
resultado <- 3000

#se o resultado for maior que zero
if (resultado > 0) {
  # informe: "Tivemos Lucro!"
  print("Tivemos Lucro!")
  # caso contrário
```

```
} else {
  # informe: "Deu ruim :( "
  print("Deu ruim :(")
}
```

```
## [1] "Tivemos Lucro!"
```

No R, esse alinhamento é apenas um facilitador de leitura do código. Se o código estiver organizado, as pessoas terão mais facilidade em entendê-lo. O código a seguir tem exatamente a mesma funcionalidade.

```
resultado <- 3000
if (resultado > 0) { print("Tivemos Lucro!") } else {print("Deu ruim :(")}
```

```
## [1] "Tivemos Lucro!"
```

Embora seja verdade que o espaço ocupado está menor, quando o código começa a adquirir um alto grau de complexidade, torna-se muito desejável que ele esteja melhor organizado e, preferencialmente, comentado.

No Python, onde a identação faz diferença, o código precisaria seguir uma estrutura baseada em espaços, sendo desnecessário o uso das chaves.

```
#se o resultado for maior que zero
if (resultado > 0):
  # informe: "Tivemos Lucro!"
  print("Tivemos Lucro!")
else:
  # informe: "Deu ruim :( "
  print("Deu ruim :(")
```

No Python, portanto, o que importa para definir que o print("Tivemos Lucro!") está associado ao if(resultado $> 0$) é o fato de não haver nenhum espaço entre a margem da página e o if; e o fato de haver quatro espaços entre a margem da página e o print. Isso faz com que esse print esteja subordinado ao if; assim como, o segundo print (que também está mais recuado) é subordinado ao else (que não tem recuo).

## 3.1 Tipos de Variáveis

### 3.1.1 Numéricas

Basicamente existem os inteiros (integers) e os ponto-flutuantes (float). O integer não admite casas decimais, sendo usado para eventos contáveis, como a

quantidade de vezes que algo ocorreu. O float admite casas decimais, sendo indicado para representar valores monetários.

Veja, por exemplo, o caso da idade. Se a idade for calculada como a diferença entre anos (2021 - 1988 = 33), o resultado será um número inteiro, que pode ser representado por um integer. Contudo, se uma idade como 12,5 anos for admitida, então será necessário trabalhar com float. Se você está lendo isso em fevereiro, pense na representação correta da idade de alguém que nasceu em dezembro.

Raramente, isso será uma preocupação. Como dito, o R faz a análise automática e, por padrão, categoriza os números dentro do tipo numeric, que aceita tanto integers como floats. Eventualmente, algum erro pode acontecer, como haver algum canto na planilha que está sendo importada em que foi digitado 2x20 ao invés de 2020. Quando o R tentasse ler esse valor (2x20), ele reconheceria um caractere que não é um número e automaticamente tentaria entender isso como sendo um texto.

### 3.1.2 Textuais

## 3.2 Float

## 3.3 String

## 3.4 Boolean

## 3.5 Date

# Chapter 4

# Estruturas de Dados

## 4.1 Vetores

## 4.2 Data frames

## 4.3 Matrizes

## 4.4 Listas

## 4.5 Factors

# Chapter 5

# Trabalhando com variáveis no R Base

## 5.1 Manuseio de variáveis

## 5.2 Funções numéricas básicas

## 5.3 Funções textuais básicas

# Chapter 6

# Criando sua própria função

## 6.1 Comentários

# Chapter 7

# Methods

We describe our methods in this chapter.

# Chapter 8

# Applications

Some *significant* applications are demonstrated in this chapter.

## 8.1 Example one

## 8.2 Example two

# Chapter 9

# Final Words

We have finished a nice book.

# Bibliography

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2020). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.21.