

Comandos básicos do R: Guia de bolso

Lucas C. Germano

2022-05-27

Contents

Sobre este livro	5
1 Leitura de arquivos de texto	7
1.1 Diretório de trabalho	7
1.2 Leitura de arquivos	8
1.3 Escrita de arquivos	22
2 Cross-references	29
2.1 Chapters and sub-chapters	29
2.2 Captioned figures and tables	29
3 Parts	33
4 Footnotes and citations	35
4.1 Footnotes	35
4.2 Citations	35
5 Blocks	37
5.1 Equations	37
5.2 Theorems and proofs	37
5.3 Callout blocks	37
6 Sharing your book	39
6.1 Publishing	39
6.2 404 pages	39
6.3 Metadata for sharing	39

Sobre este livro

Sejam bem-vindos!

O objetivo deste livro é disponibilizar para consulta anotações de códigos R de forma prática e rápida. Não há explicações aprofundadas nem se pretende esgotar as possibilidades do conteúdo apresentado, assim, esta documentação deve ser utilizada somente como um guia rápido, pois não passa de um conjunto de rascunhos apreendidos no dia-a-dia da manipulação de dados e na apresentação de resultados. O conteúdo poderá ser baixado nos formatos **.pdf** ou **epub**, mas a proposta é que o conteúdo seja dinâmico, com atualizações semanais. A estrutura de construção está disponível no GitHub.

Críticas, sugestões ou contribuições de código e conteúdo podem ser enviadas para lucascgermano@gmail.com. Ficarei muito feliz, qualquer que seja o motivo do contato.

Chapter 1

Leitura de arquivos de texto

1.1 Diretório de trabalho

Abaixo são transcritos alguns comandos e métodos para se definir e conhecer o diretório de trabalho, bem como manipular criação e exclusão de pastas e arquivos.

Comando	Definição
<code>base::setwd()</code>	Define diretório de trabalho.
<code>base::getwd()</code>	Identifica diretório ativo.
<code>base::dir()</code>	Retorna todo o conteúdo do diretório ativo.
Ctrl + Shift + h	Abre janela de navegação para definir diretório.
<code>base::file.choose()</code>	Abre janela de navegação e ao selecionar o arquivo, ele retorna o caminho (diretório). Pode-se usar também dentro do comando, como em <code>read.csv2(file = file.choose())</code> .
No RStudio: Ir em Session, Setting Working Directory Inserir aspas ' ' + Tab entre elas	Equivalente a Ctrl + Shift + h
<code>base::dir.create()</code>	Navegação que pode servir para explorar caminhos.
<code>base::unlink()</code>	Cria uma pasta de trabalho.
	Deleta uma pasta, ex. <code>unlink("some_directory", recursive = TRUE)</code> .
<code>base::file.create()</code>	Cria um arquivo no diretório ex. <code>file.create("text_file.txt")</code> (docx, csv, etc).

Comando	Definição
<code>base::file.copy()</code>	Copia um arquivo. Ex. <code>file.copy(from = "source_file.txt", to = "destination_folder")</code> .
<code>base::file.remove()</code>	Deleta um arquivo, ex. <code>file.remove("csv_file.csv")</code> . Pode-se usar também <code>unlink('csv_file.csv')</code> .
<code>base::list.files()</code> <code>here::here()</code>	Lista os arquivos presentes no diretório. Cria um caminho relativo para um arquivo no diretório de trabalho, preferencialmente em um projeto, o que facilita ser reproduzido em diversas máquinas, ex. <code>here('arquivos', 'dados.csv')</code> . É similar ao <code>base::file.path()</code> , cuja sintaxe é a mesma.

Exemplo - `list.files()`

```
list.files(path = 'dados/',      # Caminho do arquivo
           pattern = '.ods',     # Formato especificado
           full.names = FALSE,   # Somente nome
           recursive = TRUE,     # Pesquisa em subpastas
           ignore.case = FALSE) # Ignora tamanhos das letras
```

```
## [1] "planilha_ods.ods"
```

1.2 Leitura de arquivos

1.2.1 `utils::read.csv2()`

`read.csv` = Arquivos separados por vírgula.

`read.csv2` = Arquivos separados por ponto e vírgula.

Os argumentos das funções são os mesmos, por isso o exemplo será dado somente para `.csv2` (mais usado)

```
dados <- read.csv2(file = 'dados/dados.csv')
head(dados, 5)      # Exibir as 5 primeiras linhas dos dados.
```

```
##   X      data code_mn      muni  faixa casos obitos masc fem  ano mes semana
## 1 1 2020-01-01 353070 Mogi Guaçu 30 a 39      1      0    0  1 2020  1      1
## 2 2 2020-01-20 353070 Mogi Guaçu 50 a 59      1      0    1  0 2020  1      3
```



```
## 3 3 2020-01-29 352380      Itobi 30 a 39      1      0      1      0 2020      1      5
## 4 4 2020-01-30 353050      Mococa 70 a 79      1      0      0      1 2020      1      5
## 5 5 2020-02-02 353080 Mogi Mirim 40 a 49      1      0      0      1 2020      2      5
##      pop
## 1 150713
## 2 150713
## 3      7830
## 4      68788
## 5      92715
```

Argumentos principais

Os argumentos são os mesmos da função `read.table()`.

Argumento	Definição
<code>file</code>	Nome do arquivo que será lido, contendo o caminho do diretório.
<code>header</code>	Logical. Indica se o arquivo contém os nomes das colunas na primeira linha.
<code>sep</code>	Tipo de separador de campo. Default é = “;”.
<code>dec</code>	Tipo de separador de decimal. Default é = “.”.
<code>nrows</code>	Integer. Número máximo de linhas a serem lidas.
<code>skip</code>	Integer. Número de linhas que serão puladas antes de iniciar a leitura dos dados.
<code>fill</code>	Logical. Se TRUE, caso as linhas tenham comprimento desigual, serão adicionados campos em branco.
<code>blank.lines.skip</code>	Logical. Se TRUE linhas vazias serão ignoradas.
<code>stringsAsFactors</code>	Logical. Se TRUE os vetores <code>character</code> serão convertidos para <code>factors</code> . Se houver distorção dos caracteres, utilizar FALSE para sem conversão.
<code>fileEncoding</code>	Character string. Define o encoding que será usado. Ex. <code>fileEncoding</code> = “UTF-8” ou “Latin-1” ou “ISO-8859-1”.
<code>skipNull</code>	Logical. Se TRUE os nulos (NA) devem ser ignorados.

Argumento	Definição
colClasses	character. Um vetor de classes referentes as colunas. Valores possíveis são NA (default, quando type.convert é usado), “NULL” (quando a coluna é pulada), um vetor atômico de classes (logical, integer, numeric, complex, character, raw), or “factor”, “Date” or “POSIXct”.

1.2.2 readr::read_csv2()

Exemplo 1

```
dados <- readr::read_csv2(file = 'dados/dados.csv', # Caminho e arquivo
                          col_select = c(2,4:7),    # Seleção de colunas
                          guess_max = 1000,         # Máximo de linhas utilizadas para
                          skip_empty_rows = TRUE)    # Pular linhas vazias

head(dados, 5)
```

```
## # A tibble: 5 x 5
##   data      muni      faixa casos obitos
##   <date>    <chr>    <chr>   <dbl>  <dbl>
## 1 2020-01-01 Mogi Guaçu 30 a 39     1      0
## 2 2020-01-20 Mogi Guaçu 50 a 59     1      0
## 3 2020-01-29 Itobi    30 a 39     1      0
## 4 2020-01-30 Mococa   70 a 79     1      0
## 5 2020-02-02 Mogi Mirim 40 a 49     1      0
```

Exemplo 2

```
dados <- readr::read_csv2(
  file = 'dados/dados.csv', # Caminho e arquivo
  guess_max = 1000,         # Linhas utilizadas para classes
  skip_empty_rows = TRUE,   # Pular linhas vazias
  skip = 1,                 # Pular primeira linha
  col_names = c('a','b','c','d','e'), # Definir nomes das colunas
  col_select = c('a','b','c','d','e')) # Selecionar colunas

head(dados, 5)
```

```
## # A tibble: 5 x 5
##       a b           c d           e
```

```
##      <dbl> <date>          <dbl> <chr>      <chr>
## 1      1 2020-01-01 353070 Mogi Guaçu 30 a 39
## 2      2 2020-01-20 353070 Mogi Guaçu 50 a 59
## 3      3 2020-01-29 352380 Itobi      30 a 39
## 4      4 2020-01-30 353050 Mococa     70 a 79
## 5      5 2020-02-02 353080 Mogi Mirim 40 a 49
```

Argumentos principais

Argumento	Definição
file	Nome do arquivo que será lido, contendo o caminho do diretório (admite http). Arquivos terminados em .gz, .bz2, .xz, ou .zip serão automaticamente descomprimidos.
col_names	TRUE ou FALSE ou um vetor tipo caracter com nomes das colunas. Se TRUE, a primeira linha será usada para nomear as colunas. Se FALSE, nomes das colunas serão gerados automaticamente (X1, X2, X3 etc). Se col_names for um vetor com nomes, os valores serão usados como nomes das colunas, mas a primeira linha será considerada no banco (nomes errados), assim, pode-se usar o argumento renomeando as colunas, mas fazendo a leitura sem considerar a primeira linha, com [-1,] ou skip = 1. Colunas sem nome (NA) receberão nomes fictícios.

Argumento	Definição
<code>col_types</code>	Se for NULL, todas as classes de coluna serão imputadas a partir do máximo de linhas lidas (<code>guess_max</code>) intercaladas por todo o arquivo. Se a imputação falhar, você precisará aumentar o <code>guess_max</code> ou fornecer os tipos corretos você mesmo. As especificações de coluna criadas por <code>list()</code> ou <code>cols()</code> devem conter uma especificação de coluna para cada coluna. Se você quiser ler apenas um subconjunto das colunas, use <code>cols_only()</code> . Para compactar um vetor com as classes, usar as letras <code>c</code> = character, <code>i</code> = integer, <code>n</code> = number, <code>d</code> = double, <code>l</code> = logical, <code>f</code> = factor, <code>D</code> = date, <code>T</code> = date time, <code>t</code> = time, <code>?</code> = guess. Por padrão, a definição de classe é automática.
<code>col_select</code>	Colunas a serem incluídas nos resultados, equivale a <code>dplyr::select()</code> para se referir às colunas pelo nome. Use <code>c()</code> ou <code>list()</code> para usar mais de uma expressão de seleção. Embora esse uso seja menos comum, <code>col_select</code> também aceita um índice de coluna numérica.
<code>locale</code>	A localidade controla os padrões que variam de lugar para lugar. A localidade padrão é centrada nos EUA (como R), mas você pode usar <code>locale()</code> para criar sua própria localidade que controla coisas como o fuso horário padrão, codificação, marca decimal, marca grande e nomes de dia e mês.
<code>na</code>	Vetor de caracteres de strings para interpretar como valores ausentes. Defina esta opção como <code>character()</code> para indicar que não há valores ausentes.

Argumento	Definição
trim_ws	Os espaços em branco à esquerda e à direita (espaços e tabulações ASCII) devem ser cortados de cada campo antes de analisá-lo?
skip	Número de linhas para pular antes de ler os dados.
n_max	Número máximo de linhas a ler.
guess_max	Número máximo de linhas a serem usadas para adivinhar os tipos de coluna.
show_col_types	Se FALSE, não mostre os tipos de coluna adivinhados. Se TRUE sempre mostra os tipos de coluna, mesmo que sejam fornecidos. Se NULL (o padrão) mostrar apenas os tipos de coluna se eles não forem fornecidos explicitamente pelo argumento col_types.
skip_empty_rows	As linhas em branco devem ser ignoradas completamente? ou seja, se esta opção for TRUE, as linhas em branco não serão representadas. Se for FALSE, eles serão representados por valores NA em todas as colunas.

1.2.3 data.table::fread()

Tem a vantagem de realizar a leitura de arquivos grandes de forma rápida. Além disso, tem boa capacidade de identificar automaticamente o separador, encoding e tipos de classes. O resultado padrão é um objeto `data.table`, mas pode-se mudar para `data.frame`.

Exemplo 1

[illegible]

```
##      data.in.sin  municipio num_casos
## 1:  2020-01-01 Mogi Guaçu         1
## 2:  2020-01-20 Mogi Guaçu         1
## 3:  2020-01-29      Itobi          1
## 4:  2020-01-30      Mococa         1
## 5:  2020-02-02 Mogi Mirim         1
```

Argumentos principais

Argumento	Definição
file	Nome do arquivo no diretório de trabalho, caminho para o arquivo ou um URL começando http:, etc. Arquivos compactados ‘.gz’ e ‘.bz2’ são suportados se o pacote R.utils estiver instalado.
sep	O separador entre colunas.
nrows	Número máximo de linhas a serem lidas.
header	Logical. Primeira linha é o nome das colunas.
na.strings	Para ler NA, como NA, defina na.strings=“NA”. Para ler „ como string em branco “”, defina na.strings=NULL.
stringsAsFactors	Converter todas as colunas de caracteres em fatores?
skip	skip >0 ignora as primeiras linhas. skip=“string” procura por “string” no arquivo (por exemplo, uma substring da linha de nomes de coluna) e começa nessa linha (inspirada em read.xls no pacote gdata).
select	Um vetor de nomes de colunas ou números para manter e eliminar as demais. Pode especificar também tipos da mesma forma que colClasses; ou seja, um vetor de pares colname=type, ou uma lista de pares type=col(s). Em todas as formas de seleção, a ordem em que as colunas são especificadas determina a ordem das colunas no resultado.

Argumento	Definição
drop	Vetor de nomes de colunas ou números a serem descartados, mantenha o resto.
colClasses	Pode receber um vetor ou lista nomeado especificando tipos para um subconjunto das colunas por nome. O padrão NULL significa que os tipos são inferidos automaticamente. Ex1 - <code>colClasses = c("Date", "character", "integer")</code> , neste caso as classes vão compor as classes das colunas na ordem posta. Ex2 - <code>colClasses = c("data" = "Date", "idade" = "integer")</code> , nesse caso estou indicando as classes somente de algumas variáveis. Funciona também no <code>read.csv2</code> .
dec	Separador de decimal como em <code>read.csv2</code> .
col.names	Inserir um vetor de nomes para as colunas se quiser substituir os originais. Se houver alguma coluna original sem título (NA), ela será renomeada automaticamente com "V"+ o numero que corresponde no banco (V1,V2,V3).
encoding	Default is "unknown". Outras possíveis opções são "UTF-8" e "Latin-1". Porém, não é usado para recodificar, em vez disso, permite o manuseio de strings codificadas em sua codificação nativa.
strip.white	O padrão é TRUE. Retira espaços em branco à esquerda e à direita de campos não citados. Se FALSE, apenas os espaços à direita do cabeçalho serão removidos.
fill	Logical, o padrão é FALSE. Se TRUE, caso as linhas tenham comprimento desigual, os campos em branco serão preenchidos implicitamente.

Argumento	Definição
blank.lines.skip	Logical, o padrão é FALSE. Se TRUE, as linhas em branco serão ignoradas.
showProgress	TRUE exibe o progresso no console se o ETA for maior que 3 segundos.
data.table	TRUE retorna um data.table (default). FALSE retorna um data.frame. O default para este argumento pode ser modificado com opções(datatable.fread.datatable=FALSE).
nThread	Número de threads a serem usados. Experimente para ver o que funciona melhor para seus dados em seu hardware.
KeepLeadingZeros	Se for TRUE, dados numéricos com zeros à esquerda são lidos como caracterer, caso contrário, os zeros à esquerda serão removidos e convertidos em numéricos.

1.2.4 readODS::read_ods()

Leitura de arquivos no formato .ods do Libre Office, em que le uma planilha individual e retorna um data.frame.

Exemplo 1

```
dados <- readODS::read_ods(path = 'dados/planilha_ods.ods', # Caminho do arquivo
                           col_names = FALSE,             # Primeira linha contém n
                           sheet = 1,                     # Seleção da planilha
                           range = "A7:B14")              # Intervalo para leitura
head(dados)
```

```
##      A      B
## 1 113 381
## 2  29 112
## 3  23  25
## 4  29 152
## 5  87  NA
## 6  40  27
```

Argumentos principais

Argumento	Definição
path	Caminho do arquivo ods.
sheet	Planilha que será lida. Default e 1. Pode ser o nome da planilha (ex. "semana1") ou um número correspondente a planilha.
col_names	Indica se a primeira linha contem os nomes das colunas.
skip	Número de linhas a pular antes de iniciar a leitura dos dados.
formula_as_formula	Exibir fórmulas como fórmulas "SUM(A1:A3)" ou como valores "3" ou "8".
range	Seleção de retângulo usando intervalo de células semelhante ao Excel, como intervalo = "D12:F15" ou intervalo = "R1C12:R6C15". O processamento de intervalo de células é tratado pelo pacote cellranger.
row_names	Indica se o arquivo contém os nomes das linhas na primeira coluna.
strings_as_factors	Logical. Se variáveis tipo character serão convertidas a fatores.

1.2.5 readxl::read_excel()

Leitura de arquivos extensão .xls e .xlsx.

Exemplo 1

```
dados <- readxl::read_excel(path = "dados/planilha_xlsx.xlsx",
                             sheet = 1,
                             col_names = c('vel', 'dist'),
                             col_types = c("numeric", "numeric"),
                             range = "A3:B19")
head(dados, 5)
```

```
## # A tibble: 5 x 2
##   vel  dist
##   <dbl> <dbl>
## 1    72   360
## 2    68   410
## 3    NA   255
## 4    76   239
```

5 88 209

Argumentos principais

Argumento	Definição
path	Caminho para o arquivo xls/xlsx.
sheet	Planilha a ser lida. Aceita o nome da planilha ou o número correspondente. Default é a primeira planilha.
reange	Intervalo de células para leitura, ex. “B3:D87” ou “Orçamento!B2:G14”.
col_names	Se TRUE a primeira linha será usada para nomear as colunas. FALSE o número das colunas será uma sequência automática de X1 a Xn, ou um vetor de nomes para cada coluna.
col_types	Se NULL os tipos de classes serão adivinhados, senão inserir um vetor indicando as classes “blank”, “numeric”, “date” or “text”.
na	Valores ausentes. Por default o readxl converte células em branco para valores ausentes. Pode-se inserir um valor padrão caso se deseje cobrir os valores ausentes.
skip	Número de linhas para pular antes de iniciar a leitura dos dados.
n_max	Número máximo de linhas a serem lidas.
guess_max	Máximo de linhas utilizados para adivinhar classes das colunas.

1.2.6 foreign::read.dbf()

A função lê arquivos .dbf como dataframe, convertendo por default campos character em factor. Tem apenas dois argumentos, o file (caminho) e o as.is (se FALSE não converte os campos em factor). Por não ser muito usado, o desenvolvedor já alerta que nem todos os arquivos poderão ser lidos normalmente.

Exemplo

```
dados <- foreign::read.dbf(file = 'dados/planilha_dbf.dbf')
head(dados, 5)
```

```
##  peso altura
## 1  222    160
## 2  132    164
## 3  137    169
## 4   63    209
## 5  223    166
```

1.2.7 rio::import()

O rio é um pacote que vem com a iniciativa de simplificar o procedimento de importação e exportação de arquivos de dados. Na importação, é capaz de ler uma vasta lista de extensões de arquivos, sem a necessidade de especificar o formato. Ver documentação.

Exemplo

```
dados <- rio::import(file = 'dados/planilha_dbf.dbf') # Caminho e nome.
head(dados, 3)
```

```
##  peso altura
## 1  222    160
## 2  132    164
## 3  137    169
```

Argumentos principais

Argumento	Definição
file	Caminho e nome do arquivo. Pode ter extensão .zip ou .tar.
format	Formato do arquivo. A definição é opcional, mas pode ser “,”, “;”, “ ”
setclass	Classe do objeto (opcional). Default é um “data.frame”. Os valores permitidos incluem “tbl_df”, “tbl” ou “tibble” (se estiver usando dplyr) ou “data.table” (se estiver usando data.table). Outros valores são ignorados, de modo que um data.frame é retornado.

Argumento	Definição
<code>which</code>	Controla a importação de arquivos multi-objeto; como regra, <code>import</code> apenas retorna um único quadro de dados (use <code>import_list</code> para importar vários quadros de dados de um arquivo multi-objeto). Se o arquivo for um diretório compactado, que pode ser uma cadeia de caracteres especificando um nome de arquivo ou um número inteiro especificando qual arquivo (na ordem de classificação de localidade) extrair do diretório compactado. Para planilhas do Excel, isso pode ser usado para especificar um nome ou número de planilha. Para arquivos <code>.Rdata</code> , pode ser um nome de objeto. Para arquivos HTML, identifica qual tabela extrair (da ordem do documento). Ignorado de outra forma. Um valor de cadeia de caracteres será usado como uma expressão regular, de modo que o arquivo extraído seja a primeira correspondência da expressão regular com os nomes de arquivo no arquivo.

1.2.8 Arquivos da web

Pode-se usar o endereço do apresentado no navegador ou contido nas propriedades (clique com botão direito). O endereço deverá ser inserido entre aspas nos argumentos `file` ou `path` da maioria das funções de leitura, como no exemplo abaixo:

```
read.csv2(file = 'https://raw.githubusercontent.com/seade-R/dados-covid-sp/master/data/dados_covid_sp.csv')
```

Ou atribuir o link a um objeto e usá-lo na função.

```
link <- 'https://raw.githubusercontent.com/seade-R/dados-covid-sp/master/data/dados_covid_sp.csv'
```

É possível também baixar o arquivo (inclusive imagens) por meio da seguinte função:

```
download.file(url = 'https://raw.githubusercontent.com/seade-R/dados-covid-sp/master/data/dados_covid_sp.csv', destfile = 'dados/baixado_web.csv')
```

1.2.9 Encoding

Se houver distorção de caracteres especiais, considerar como possibilidades para resolver o problema utilizar o argumento correspondente a `stringsAsFactors = F`. Esse comando faz com que os caracteres permaneçam como caracteres, ao invés de serem convertidos para factor, e `encoding = "UTF-8"` ou `encoding = "ISO-8859-1"` para reconhecer os caracteres especiais. O argumento `fileEncoding = "UTF-8"` também pode ser necessário.

Descobrir o encoding

Verifica somente de um vetor

```
stringi::stri_enc_detect(str = cars$speed[1])
```

```
## [[1]]
##   Encoding Language Confidence
## 1    UTF-8                0.15
```

Converter encoding

```
base::iconv(x = cars$speed[1:3], # Dataframe ou vetor
            from = "UTF-8",      # Encoding anterior
            to = "ISO-8859-1")   # Novo encoding
```

```
## [1] "4" "4" "7"
```

Pode-se também utilizar a função `base::enc2utf8` para transformar uma codificação em UTF-8, porém, deve ser sempre aplicado a um vetor (ou coluna do banco) de dados do tipo character, se for preciso, transformar antes com a função `base::as.character`

```
dados <- base::as.character(iris$Species)
dados <- base::enc2utf8(dados)
```

Encoding via Import Dataset

É possível controlar o encoding pelos argumentos da função escolhida para leitura do arquivo, ou então pela leitura realizada pela interface gráfica do RStudio. Entrar no menu “File”, “Import Dataset”, “From text (base)...”, após isso será aberta uma janela, onde o campo encoding

permite selecionar uma codificação entre centenas. Veja figura abaixo:

Import Dataset

Name: dados

Input File: "data";"code_mn";"muni";"faixa";"casos";"obitos";"nasc";"fem";"ano";
 "1";2020-01-01;353070;"Mogi Guaçu";"30 a 39";1;0;0;1;2020;1;1;150713
 "2";2020-01-20;353070;"Mogi Guaçu";"50 a 59";1;0;1;0;2020;1;3;150713
 "3";2020-01-29;352380;"Itobi";"30 a 39";1;0;1;0;2020;1;5;7830
 "4";2020-01-30;353050;"Mococa";"70 a 79";1;0;0;1;2020;1;5;68788
 "5";2020-02-02;353080;"Mogi Mirim";"40 a 49";1;0;0;1;2020;2;5;92715
 "6";2020-02-05;353080;"Mogi Mirim";"40 a 49";1;0;0;1;2020;2;6;92715
 "7";2020-02-06;353070;"Mogi Guaçu";"30 a 39";1;0;0;1;2020;2;6;150713
 "8";2020-02-13;353070;"Mogi Guaçu";"0 a 4";1;0;1;0;2020;2;7;150713
 "9";2020-02-22;353070;"Mogi Guaçu";"60 a 69";1;0;0;1;2020;2;8;150713
 "10";2020-03-01;351080;"Casa Branca";"20 a 29";1;0;0;1;2020;3;9;30237
 "11";2020-03-02;350040;"Águas da Prata";"50 a 59";1;0;1;0;2020;3;9;8137
 "12";2020-03-02;351390;"Divinolândia";"20 a 29";1;0;0;1;2020;3;9;11208
 "13";2020-03-04;354910;"São João da Boa Vista";"60 a 69";1;0;1;0;2020;3
 "14";2020-03-05;352260;"Itapira";"50 a 59";1;0;1;0;2020;3;10;74299
 "15";2020-03-06;350030;"Aguai";"50 a 59";1;0;0;1;2020;3;10;35954
 "16";2020-03-07;354970;"São José do Rio Pardo";"20 a 29";1;0;1;0;2020;3

Encoding: Automatic

Heading: Yes

Row names: Automatic

Separator: Semicolon

Decimal: Period

Quote: Double (")

Comment: None

na.strings: NA

☐ Strings as factors

Data Frame

x	data	code_mn	muni	faixa	casos	o
1	2020-01-01	353070	Mogi Guaçu	30 a 39	1	0
2	2020-01-20	353070	Mogi Guaçu	50 a 59	1	0
3	2020-01-29	352380	Itobi	30 a 39	1	0
4	2020-01-30	353050	Mococa	70 a 79	1	0
5	2020-02-02	353080	Mogi Mirim	40 a 49	1	0
6	2020-02-05	353080	Mogi Mirim	40 a 49	1	0
7	2020-02-06	353070	Mogi Guaçu	30 a 39	1	0
8	2020-02-13	353070	Mogi Guaçu	0 a 4	1	0
9	2020-02-22	353070	Mogi Guaçu	60 a 69	1	0
10	2020-03-01	351080	Casa Branca	20 a 29	1	0
11	2020-03-02	350040	Águas da Prata	50 a 59	1	0
12	2020-03-02	351390	Divinolândia	20 a 29	1	0
13	2020-03-04	354910	São João da Boa Vista	60 a 69	1	0
14	2020-03-05	352260	Itapira	50 a 59	1	0
15	2020-03-06	350030	Aguai	50 a 59	1	0
16	2020-03-07	354970	São José do Rio Pardo	20 a 29	1	0

Import Cancel

1.3 Escrita de arquivos

1.3.1 utils::write.csv2()

É possível salvar um arquivo de dados que foi trabalhado no R em diferentes formatos, no caso, separado por ponto e vírgula. Um ponto negativo é que essa função, ao salvar o arquivo, cria uma coluna com nomes das linhas (em números).

Exemplo

```
write.csv2(x = iris, # Dados ativos
           file = 'dados/iris.csv', # Caminho e nome do arquivo
           fileEncoding = "UTF-8") # Encoding

read.csv2('dados/iris.csv', nrows = 4)
```

```
## X Sepal.Length Sepal.Width Petal.Length Petal.Width Species
```

```
## 1 1      5.1      3.5      1.4      0.2  setosa
## 2 2      4.9      3.0      1.4      0.2  setosa
## 3 3      4.7      3.2      1.3      0.2  setosa
## 4 4      4.6      3.1      1.5      0.2  setosa
```

Argumentos principais

Argumento	Definição
x	Objeto a ser escrito, preferencialmente uma matriz ou data.frame.
file	Nome do arquivo criado (pode conter o caminho) utilizando aspas ” “.
append	Logical. Se TRUE os dados serão adicionados à última linha de um arquivo já existente, que deve ter o nome descrito em file, se FALSE qualquer arquivo com o nome descrito será sobrescrito.
na	String usada para valores ausentes nos dados.
dec	String para definir divisor de decimal, ex. dec = “.”.
col.names	Logical. Indica se os nomes das colunas de x devem ser escritos junto com x, ou um vetor de caracteres dos nomes das colunas a serem escritos.
row.names	Logical. Cria coluna com nomes para linhas.
fileEncoding	String. Declara a codificação a ser usada para que possam ser recodificados à medida que são gravados.

1.3.2 readr::write_csv2()

É semelhante à função anterior, mas executa a tarefa duas vezes mais rápido, com a vantagem de não criar uma coluna com nomes das linhas.

Exemplo

```
readr::write_csv2(x = iris, file = 'dados/iris.csv')
read.csv2(file = 'dados/iris.csv', nrow = 4)
```

```
## Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1          3.5          1.4          0.2 setosa
## 2          4.9          3.0          1.4          0.2 setosa
## 3          4.7          3.2          1.3          0.2 setosa
## 4          4.6          3.1          1.5          0.2 setosa
```

Argumentos principais Argumento|Definição ———|———— x
 A data frame or tibble to write to disk.

file

File or connection to write to.

delim

Delimiter used to separate values. Defaults to " " for write_delim(), ",", for write_excel_csv() and ";" for write_excel_csv2(). Must be a single character.

na

String used for missing values. Defaults to NA. Missing values will never be quoted; strings with the same value as na will always be quoted.

append

If FALSE, will overwrite existing file. If TRUE, will append to existing file. In both cases, if the file does not exist a new file is created.

col_names

If FALSE, column names will not be included at the top of the file. If TRUE, column names will be included. If not specified, col_names will take the opposite value given to append.

quote

How to handle fields which contain characters that need to be quoted.

needed - Only quote fields which need them.

all - Quote all fields.

none - Never quote fields.

escape

The type of escape to use when quotes are in the data.

double - quotes are escaped by doubling them.

backslash - quotes are escaped by a preceding backslash.

none - quotes are not escaped.

eol

num_threads Number of threads to use when reading and materializing vectors. If your data contains newlines within fields the parser will automatically be forced to use a single thread only.

progress

Display a progress bar? By default it will only display in an interactive session and not while knitting a document. The display is updated every 50,000 values and will only display if estimated reading time is 5 seconds or more. The automatic progress bar can be disabled by setting option `readr.show_progress` to `FALSE`.

1.3.3 writexl::write_xlsx()

Salvar arquivo de dados como Excel, formato .xlsx.

Exemplo 1

```
writexl::write_xlsx(x = iris,
                    path = 'dados/iris.xlsx',
                    col_names = TRUE,
                    format_headers = TRUE)
readxl::read_excel('dados/iris.xlsx', n_max = 4)
```

```
## # A tibble: 4 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##   <dbl>         <dbl>         <dbl>         <dbl> <chr>
## 1         5.1         3.5         1.4         0.2 setosa
## 2         4.9         3         1.4         0.2 setosa
## 3         4.7         3.2         1.3         0.2 setosa
## 4         4.6         3.1         1.5         0.2 setosa
```

Exemplo 2

```
writexl::write_xlsx(path = "dados/conjuntodadosnativos.xlsx",
                    x = list(sheet1=iris, sheet2=cars, sheet3=mtcars))
readxl::read_excel(path = 'dados/conjuntodadosnativos.xlsx',
                    sheet = 3,
                    n_max = 4)
```

```
## # A tibble: 4 x 11
##   mpg   cyl  disp    hp  drat    wt  qsec    vs  am  gear  carb
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  21     6   160   110   3.9   2.62  16.5     0     1     4     4
## 2  21     6   160   110   3.9   2.88  17.0     0     1     4     4
## 3 22.8     4   108    93   3.85   2.32  18.6     1     1     4     1
## 4 21.4     6   258   110   3.08   3.22  19.4     1     0     3     1
```

Argumentos principais

Argumento	Definição
x	Data frame ou lista de data frames que serão salvos em planilhas (sheets).
path	Nome do arquivo criado.
col_names	Se TRUE, primeira linha traz os nomes das colunas.
format_headers	Inserir nomes das colunas.

1.3.4 data.table::fwrite()

Salvar arquivos de texto (csv, csv2, tab, etc), semelhante ao write.table() do pacote utils, só que mais rápido.

Exemplo

```
dados <- data.table::fread(file = 'dados/dados.csv', nrows = 20)
data.table::fwrite(x = dados,                                # Objeto a ser escrito
                   file = 'dados/dados20max.csv',           # Nome e caminho (arquivo ja existe)
                   append = TRUE,                           # Salva na ultima linha do arquivo j
                   sep = ';',                               # Separados de colunas
                   showProgress = TRUE)                     # Mostrar progresso
```

Argumentos principais

Argumento	Definição
x	Objeto a salvar. Deve estar como data.frame ou data.table.
file	Nome do arquivo.
append	Se TRUE , o arquivo é salvo em acrescimo à última linha de um arquivo existente, sem incluir os nomes das colunas.
sep	Separador de colunas. Default é “,”.
na	Um string a ser usada para valores ausentes. O padrão é uma string em branco “”.
dec	Separador de decimal, default é “.”.
row.names	Nome das linhas. Usar somente se for data.frame, porque é incompatível com data.table..

Argumento	Definição
col.names	Primeira linha como nomes das colunas.
logical01	Os valores lógicos devem ser escritos como 1 e 0 em vez de “TRUE” e “FALSE”?
showProgress	Exibir um medidor de progresso no console. Ignorado quando file == “.”.
compress	Se compress = “auto” e se o arquivo termina em .gz, o formato de saída é gzipado csv. Se compress = “none”, o formato de saída é sempre csv. Se compress = “gzip”, o formato é csv compactado com gzip. A saída para o console nunca é compactada com gzip mesmo se compress = “gzip”. Por padrão, compress = “auto”.

1.3.5 rio::export()

Semelhante a outros comandos de escrita de arquivos, o `rio::export()` permite gravar um data frame nos formatos habituais de texto. Para exportar uma lista de arquivos, usar o `rio::export_list()`. Ver documentação.

Exemplo

```
rio::export(x = iris,                # Objeto que será exportado.
            file = 'dados/iris.xlsx') # Caminho, nome e extensão.
```

Argumentos principais

Argumento	Definição
x	Matriz ou data frame a ser escrita. Exceções são que x pode ser uma lista de dados se o formato de arquivo de saída for uma pasta de Excel .xlsx. Para exportar uma lista de quadros de dados para vários arquivos, use <code>export_list</code> em vez disso.
file	Nome do arquivo. Deve especificar file e/ou format.

Argumento	Definição
format	Sequência de caracteres opcional contendo o formato de arquivo, que pode ser usado para substituir o formato inferido a partir de file ou, em vez de especificar file, um arquivo com o nome do símbolo de x e a extensão de arquivo especificada será criado. Os atalhos incluem: “,” ou “;” ou ”

Chapter 2

Cross-references

Cross-references make it easier for your readers to find and link to elements in your book.

2.1 Chapters and sub-chapters

There are two steps to cross-reference any heading:

1. Label the heading: `# Hello world {#nice-label}`.
 - Leave the label off if you like the automated heading generated based on your heading title: for example, `# Hello world = # Hello world {#hello-world}`.
 - To label an un-numbered heading, use: `# Hello world {-#nice-label}` or `{# Hello world .unnumbered}`.
2. Next, reference the labeled heading anywhere in the text using `\@ref(nice-label)`; for example, please see Chapter 2.
 - If you prefer text as the link instead of a numbered reference use: any text you want can go here.

2.2 Captioned figures and tables

Figures and tables *with captions* can also be cross-referenced from elsewhere in your book using `\@ref(fig:chunk-label)` and `\@ref(tab:chunk-label)`, respectively.

See Figure 2.1.

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

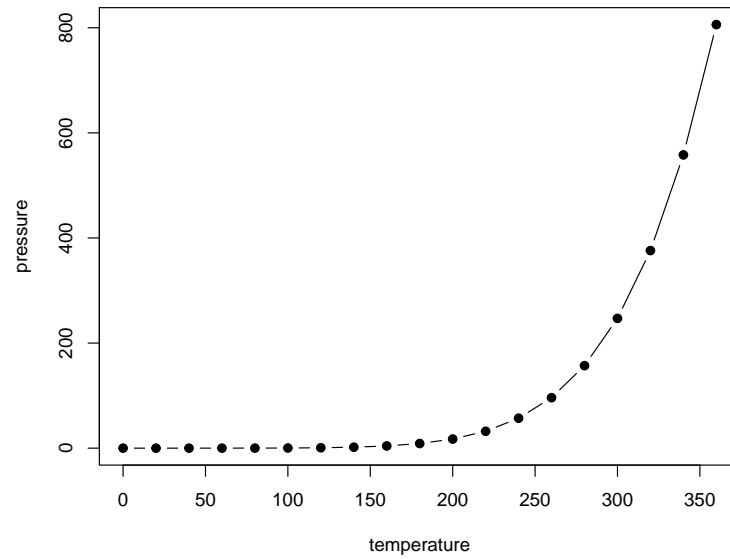


Figure 2.1: Here is a nice figure!

Don't miss Table 2.1.

```
knitr::kable(  
  head(pressure, 10), caption = 'Here is a nice table!',  
  booktabs = TRUE  
)
```

Table 2.1: Here is a nice table!

temperature	pressure
0	0.0002
20	0.0012
40	0.0060
60	0.0300
80	0.0900
100	0.2700
120	0.7500
140	1.8500
160	4.2000
180	8.8000

Chapter 3

Parts

You can add parts to organize one or more book chapters together. Parts can be inserted at the top of an .Rmd file, before the first-level chapter heading in that same file.

Add a numbered part: `# (PART) Act one {-}` (followed by `# A chapter`)

Add an unnumbered part: `# (PART*) Act one {-}` (followed by `# A chapter`)

Add an appendix as a special kind of un-numbered part: `# (APPENDIX) Other stuff {-}` (followed by `# A chapter`). Chapters in an appendix are prepended with letters instead of numbers.

Chapter 4

Footnotes and citations

4.1 Footnotes

Footnotes are put inside the square brackets after a caret `^[]`. Like this one ¹.

4.2 Citations

Reference items in your bibliography file(s) using `@key`.

For example, we are using the **bookdown** package [Xie, 2022] (check out the last code chunk in `index.Rmd` to see how this citation key was added) in this sample book, which was built on top of R Markdown and **knitr** [Xie, 2015] (this citation was added manually in an external file `book.bib`). Note that the `.bib` files need to be listed in the `index.Rmd` with the YAML `bibliography` key.

The RStudio Visual Markdown Editor can also make it easier to insert citations: <https://rstudio.github.io/visual-markdown-editing/#/citations>

¹This is a footnote.

Chapter 5

Blocks

5.1 Equations

Here is an equation.

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k} \quad (5.1)$$

You may refer to using `\@ref{eq:binom}`, like see Equation (5.1).

5.2 Theorems and proofs

Labeled theorems can be referenced in text using `\@ref{thm:tri}`, for example, check out this smart theorem 5.1.

Theorem 5.1. *For a right triangle, if c denotes the length of the hypotenuse and a and b denote the lengths of the **other** two sides, we have*

$$a^2 + b^2 = c^2$$

Read more here <https://bookdown.org/yihui/bookdown/markdown-extensions-by-bookdown.html>.

5.3 Callout blocks

The R Markdown Cookbook provides more help on how to use custom blocks to design your own callouts: <https://bookdown.org/yihui/rmarkdown-cookbook/custom-blocks.html>

Chapter 6

Sharing your book

6.1 Publishing

HTML books can be published online, see: <https://bookdown.org/yihui/bookdown/publishing.html>

6.2 404 pages

By default, users will be directed to a 404 page if they try to access a webpage that cannot be found. If you'd like to customize your 404 page instead of using the default, you may add either a `_404.Rmd` or `_404.md` file to your project root and use code and/or Markdown syntax.

6.3 Metadata for sharing

Bookdown HTML books will provide HTML metadata for social sharing on platforms like Twitter, Facebook, and LinkedIn, using information you provide in the `index.Rmd` YAML. To setup, set the `url` for your book and the path to your `cover-image` file. Your book's `title` and `description` are also used.

This `gitbook` uses the same social sharing data across all chapters in your book—all links shared will look the same.

Specify your book's source repository on GitHub using the `edit` key under the configuration options in the `_output.yml` file, which allows users to suggest an edit by linking to a chapter's source file.

Read more about the features of this output format here:

<https://pkgs.rstudio.com/bookdown/reference/gitbook.html>

Or use:

```
?bookdown::gitbook
```


Bibliography

Yihui Xie. *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition, 2015. URL <http://yihui.org/knitr/>. ISBN 978-1498716963.

Yihui Xie. *bookdown: Authoring Books and Technical Documents with R Markdown*, 2022. URL <https://CRAN.R-project.org/package=bookdown>. R package version 0.26.