Importando dados para o R

Live 03 - youtube.com/analisemacro

Vítor Wilher

Cientista de Dados | Mestre em Economia

Plano de Voo

Introdução

A função `read_csv()`

Strings

Analisando um arquivo

Arquivos txt

Arquivos xlsx

A função `read_excel`

Arquivos zipados

Introdução

Em qualquer projeto de *análise de dados*, é comum que os dados estejam dispostos em arquivos com outras extensões. Assim, a pergunta imediata é: como inseri-los no R?[1] Para isso, vamos mostrar nessa `Live` um pouco do que vemos no nosso `Curso de R para Análise de Dados`, sobre importação de dados para o R. Para começar, vamos carregar os pacotes `tidyverse`, como abaixo:

```
library(tidyverse)
```

[1] Essa seção está baseada em Grolemund and Wickham [2017].

Introdução

Vamos começar pelo pacote `readr`, que lida com arquivos do tipo `csv`. As funções mais básicas desse pacote são:

- `read_csv()`lê arquivos com separação com vírgulas `,`.
- `read_csv2()` lê arquivos separados com `;`.
- `read_delim()` lê arquivos com qualquer tipo de separação.
- `read_fwf()` lê arquivos com distância fixa.
- `read_table()` faz isso com arquivos separados por espaços em braco - muito comum para bases disponibilizadas em extensão `.txt`.
- `read_log()` lê arquivos em estilo Apache de logs

A função `read_csv()`

Abaixo, vemos um exemplo de utilizaçaõ da função `read_csv()`.

```
heights <- read_csv("heights.csv")
```

A função `read_csv()`

Se você colocar o objeto `heights` no Console, verá que o output dele é o de um `tibble`...

```
heights
```

```
## # A tibble: 1,192 x 6
##     earn height sex       ed   age race
##    <dbl>  <dbl> <chr>  <dbl> <dbl> <chr>
##  1 50000   74.4 male      16    45 white
##  2 60000   65.5 female    16    58 white
##  3 30000   63.6 female    16    29 white
##  4 50000   63.1 female    16    91 other
##  5 51000   63.4 female    17    39 white
##  6  9000   64.4 female    15    26 white
##  7 29000   61.7 female    12    49 white
##  8 32000   72.7 male      17    46 white
##  9  2000   72.0 male      15    21 hispanic
## 10 27000   72.2 male      12    26 white
## # ... with 1,182 more rows
```

Usando argumentos da função `read_csv()`

Lembre-se que você sempre poderá utilizar `?read_csv` para consultar os argumentos da função - ou de qualquer função, diga-se, do R! Vamos usar abaixo alguns dos mais comuns.

```
sidra = read_csv2('sidra.csv', skip=5,
                  col_names = c('mes', 'vendas', 'vendas_sa',
                                'receita', 'receita_sa'),
                  col_types =
                     cols(
                       mes = col_date(format='%d/%m/%Y'),
                       vendas = col_double(),
                       vendas_sa = col_double(),
                       receita = col_double(),
                       receita_sa = col_double()),
                  locale=locale(decimal_mark = ','))
```

Strings

Vamos agora ver como analisar vetores que contenham caracteres ou *strings*. Para isso, podemos utilizar a função `parse_character`. O problema com *strings* é que existem várias formas de representá-los. Para entender melhor, vamos começar com o exemplo abaixo.

```
charToRaw('Seção')
```

```
## [1] 53 65 e7 e3 6f
```

Strings

Cada número hexadecimal representa um *byte* de informação: 53 é o S. A relação entre o número hexadecimal e o caractere é chamado de `encoding`. O problema é quando o idioma não é o inglês, como no nosso caso. O pacote `readr` irá sempre assumir como *default* que o arquivo está com *encoding* **UTF-8** e acabará lendo o vetor que contenha caracteres especiais de forma errada. Para resolver isso, podemos fazer como no código abaixo.

```
parse_character('Seção', locale = locale(encoding = 'Latin1'))
```

```
## [1] "Seção"
```

Strings

```
subitem = read_csv2('subitem.csv')
subitem
```

```
## # A tibble: 383 x 3
##    subitem                                    `varia\xe7\xe3o`   peso
##    <chr>                                                 <dbl>  <dbl>
##  1 "1101002.Arroz"                                       -0.41 0.581
##  2 "1101051.Feij\xe3o - mulatinho"                       17.0  0.0262
##  3 "1101052.Feij\xe3o - preto"                           12.6  0.0824
##  4 "1101053.Feij\xe3o - macassar (fradinho)"              9.45 0.0301
##  5 "1101073.Feij\xe3o - carioca (rajado)"                12.9  0.280
##  6 "1101075.Feij\xe3o - branco"                          29.2  0.002
##  7 "1102001.Farinha de arroz"                            -4.73 0.0131
##  8 "1102006.Macarr\xe3o"                                  1.13 0.283
##  9 "1102008.Fub\xe1 de milho"                             1.02 0.028
## 10 "1102009.Amido de milho"                               0.06 0.0043
## # ... with 373 more rows
```

Strings

```
subitem = read_csv2('subitem.csv',
                    locale = locale(encoding='Latin1'))

subitem
```

```
## # A tibble: 383 x 3
##    subitem                                variação   peso
##    <chr>                                     <dbl>  <dbl>
##  1 1101002.Arroz                             -0.41 0.581
##  2 1101051.Feijão - mulatinho                17.0  0.0262
##  3 1101052.Feijão - preto                    12.6  0.0824
##  4 1101053.Feijão - macassar (fradinho)       9.45 0.0301
##  5 1101073.Feijão - carioca (rajado)         12.9  0.280
##  6 1101075.Feijão - branco                   29.2  0.002
##  7 1102001.Farinha de arroz                  -4.73 0.0131
##  8 1102006.Macarrão                           1.13 0.283
##  9 1102008.Fubá de milho                      1.02 0.028
## 10 1102009.Amido de milho                     0.06 0.0043
## # ... with 373 more rows
```

Analisando um arquivo

`readr` usa uma heurística para descobrir o tipo de cada coluna: lê as primeiras 1000 linhas e usa algumas heurísticas (moderadamente conservadoras) para descobrir o tipo de cada coluna. Você pode emular esse processo com um vetor de caracteres usando `guess_parser()`, que retorna a melhor estimativa de `readr` e `parse_guess()`, que usa essa suposição para analisar a coluna:

Analisando um arquivo

```
guess_parser("2010-10-01")
```

```
## [1] "date"
```

```
guess_parser("15:01")
```

```
## [1] "time"
```

```
guess_parser(c("TRUE", "FALSE"))
```

```
## [1] "logical"
```

```
guess_parser(c("1", "5", "9"))
```

```
## [1] "double"
```

```
guess_parser(c("12,352,561"))
```

```
## [1] "number"
```

```
str(parse_guess("2010-10-10"))
```

```
##  Date[1:1], format: "2010-10-10"
```

Analisando um arquivo

A heurística tenta cada um dos seguintes tipos, parando quando encontra uma correspondência:

- lógico: contém apenas “F”, “T”, “FALSE” ou “TRUE”.
- integer: contém apenas caracteres numéricos (e -).
- double: contém apenas duplas válidas (incluindo números como 4.5e-5).
- number: contém duplas válidas com a marca de agrupamento dentro.
- time: corresponde ao time_format padrão.
- date: corresponde ao padrão date_format.
- data e hora: qualquer data ISO8601.

Analisando um arquivo

Esses padrões nem sempre funcionam para arquivos maiores. Existem dois problemas básicos:

- As primeiras mil linhas podem ser um caso especial, e o readr adivinha um tipo que não é suficientemente geral. Por exemplo, você pode ter uma coluna de duplas que contenha apenas inteiros nas primeiras 1000 linhas.
- A coluna pode conter muitos valores ausentes. Se as primeiras 1000 linhas contiverem apenas `NA`s, o readr irá adivinhar que é um vetor de caracteres, enquanto você provavelmente desejará analisá-lo como algo mais específico.

Arquivos txt

```
doc.text <- readLines('comunicado226.txt')
```

Arquivos xlsx

Anteriormente, vimos em detalhes como é possível analisar um arquivo através do pacote `readr`. Fizemos isso porque ela representa a ideia geral de como os pacotes `tidyverse` fazem a importação de dados para o R.

Já aqui, vamos ver um pouco do pacote `readxl`, para importar arquivos do tipo `xls` e `xlsx`.

A função `read_excel`

A função `readxl::read_excel()` irá adivinhar os tipos de coluna, por padrão, ou você pode fornecê-los explicitamente através do argumento `col_types`.

```
library(readxl)
## Importar uma planilha da internet
url = 'https://analisemacro.com.br/download/27398/'
download.file(url, 'resultado.xlsx', mode='wb')
data_stn = read_excel("resultado.xlsx", sheet='1.1',
                      range='B5:JV79', col_types = 'numeric')
```

Arquivos zipados

```
temp <- tempfile()
download.file("http://www.bcb.gov.br/ftp/notaecon/Divggnp.zip",temp)
datazip <- unzip(temp, files='Divggnp.xls')
```

Arquivos zipados

```
dados_covid <- vroom::vroom("https://data.brasil.io/dataset/covid19/cas
```

G. Grolemund and H. Wickham. *R for Data Science*. O'Reilly Media, 2017.