Módulo 2 - Lendo e gerando dados

Marcus Suassuna Santos

18/08/2020

Leitura de dados no R

No módulo anterior, vimos que é possível criar tabelas diretamente no console do R, por exemplo, usando o código.

```
DF <- data.frame(x = c(1:4), y = rnorm(4))
DF
```

```
##   x           y
## 1 1 -0.76623541
## 2 2  1.08123622
## 3 3 -0.07845436
## 4 4 -0.20062306
```

```
class(DF)
```

```
## [1] "data.frame"
```

Leitura de dados no R

Essa forma de criar tabelas, ainda que possível, é pouco prática e limitada, pois requer que os dados sejam inseridos manualmente pelo console. Além disso, é muito comum que se utilize planilhas Excel, arquivos .csv ou .txt para compartilhamento de dados.

ASsim, o R possui várias ferramentas que permitem fazer essa operação de leitura de dados de forma mais prática e eficiente.

Aqui, vamos ver duas formas: usando funções básicas do R e o pacote `readr`.

Leitura de dados no R

A função mais básica para leitura de dados no R é a função `read.table()`. Atentar ao usa da função `head()` que apresenta as seis primeiras linhas de um `data.frame`. A função `tail()` funciona de forma semelhante, porém mostra as 6 últimas linhas.

```
PV <- read.table("dados/portoVelho.csv", header = TRUE, sep = ";")
head(PV)
```

```
##         Data Vazao
## 1 1966-09-01 26021
## 2 1967-09-01 31095
## 3 1968-09-01 25731
## 4 1969-09-01 28658
## 5 1970-09-01 32650
## 6 1971-09-01 35369
```

Leitura de dados no R

A função mais básica para leitura de dados no R é a função `read.table()`. Atentar ao usa da função `head()` que apresenta as seis primeiras linhas de um `data.frame`. A função `tail()` funciona de forma semelhante, porém mostra as 6 últimas linhas.

```
PV <- read.table("dados/portoVelho.csv", header = TRUE, sep = ";")
tail(PV)
```

```
##          Data Vazao
## 49 2014-09-01 43684
## 50 2015-09-01 34717
## 51 2016-09-01 33450
## 52 2017-09-01 40324
## 53 2018-09-01 44068
## 54 2019-09-01 38221
```

Leitura de dados no R

As funções `read.csv()`, `read.csv2()`, `read.delim()` e `read.delim2()` executam a mesma tarefa da função `read.table()`, porém com alguns parâmetros pré-definidos.

Description

Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.

Usage

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
           row.names, col.names, as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = default.stringsAsFactors(),
           fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)

read.csv(file, header = TRUE, sep = ",", quote = "\"",
         dec = ".", fill = TRUE, comment.char = "", ...)

read.csv2(file, header = TRUE, sep = ";", quote = "\"",
          dec = ",", fill = TRUE, comment.char = "", ...)

read.delim(file, header = TRUE, sep = "\t", quote = "\"",
           dec = ".", fill = TRUE, comment.char = "", ...)

read.delim2(file, header = TRUE, sep = "\t", quote = "\"",
            dec = ",", fill = TRUE, comment.char = "", ...)
```

Leitura de dados no R

Assim, se fosse utilizada a função `read.csv2()`, nenhum outro argumento da função necessitaria ser explicitado, pois todos os argumentos desta função são adequados para a leitura do arquivo `portoVelho.csv`, localizado na pasta `dados`.

```
PV <- read.csv2("dados/portoVelho.csv")
head(PV)
```

```
##         Data Vazao
## 1 1966-09-01 26021
## 2 1967-09-01 31095
## 3 1968-09-01 25731
## 4 1969-09-01 28658
## 5 1970-09-01 32650
## 6 1971-09-01 35369
```

Leitura de dados no R

A vantagem de saber usar a função `read.table()` é o fato de se poder ajustar a leitura de dados para qualquer formato de arquivo.

```
PV <- read.csv2("dados/portoVelho.csv")
head(PV)
```

```
##         Data Vazao
## 1 1966-09-01 26021
## 2 1967-09-01 31095
## 3 1968-09-01 25731
## 4 1969-09-01 28658
## 5 1970-09-01 32650
## 6 1971-09-01 35369
```

A vantagem de usar a função `read.table()` é o fato de se poder ajustar a leitura de dados para qualquer formato de arquivo.

Leitura de dados no R

Além do `header`, `sep` e `dec` Um número bastante grande de argumentos é cabível na função `read.table()`. Alguns importantes:

- `na.strings`: como o R deve ler dados faltantes (alguns bancos de dados usam valores como -999.999 para dados faltantes, por exemplo, e é importante indicar isso, eventualmente);
- `colClasses`: vetor de classes que devem ser atribuídas a cada coluna. Essa predefinição faz com que o R não tenha que adivinhar qual a classe da coluna e torna os códigos mais rápidos, o que é importante na leitura de grandes bancos de dados;
- `skip`: número de linhas que não devem ser lidas até que se leia os dados.

Leitura de dados no R

- `nrows`: máximo número de linhas a serem lidas;
- `encoding`: Latin-1 ou UTF-8 - auxilia o R a lidar com algarismos latinos;
- **Sugestão:** consultar com frequência a ajuda da função usando `?read.text`

Leitura de dados no R - exemplo

```
PV <- read.csv2("dados/portoVelho.csv", nrows = 5)
classes <- sapply(PV, class)
classes
```

```
##        Data       Vazao
## "character"   "integer"
```

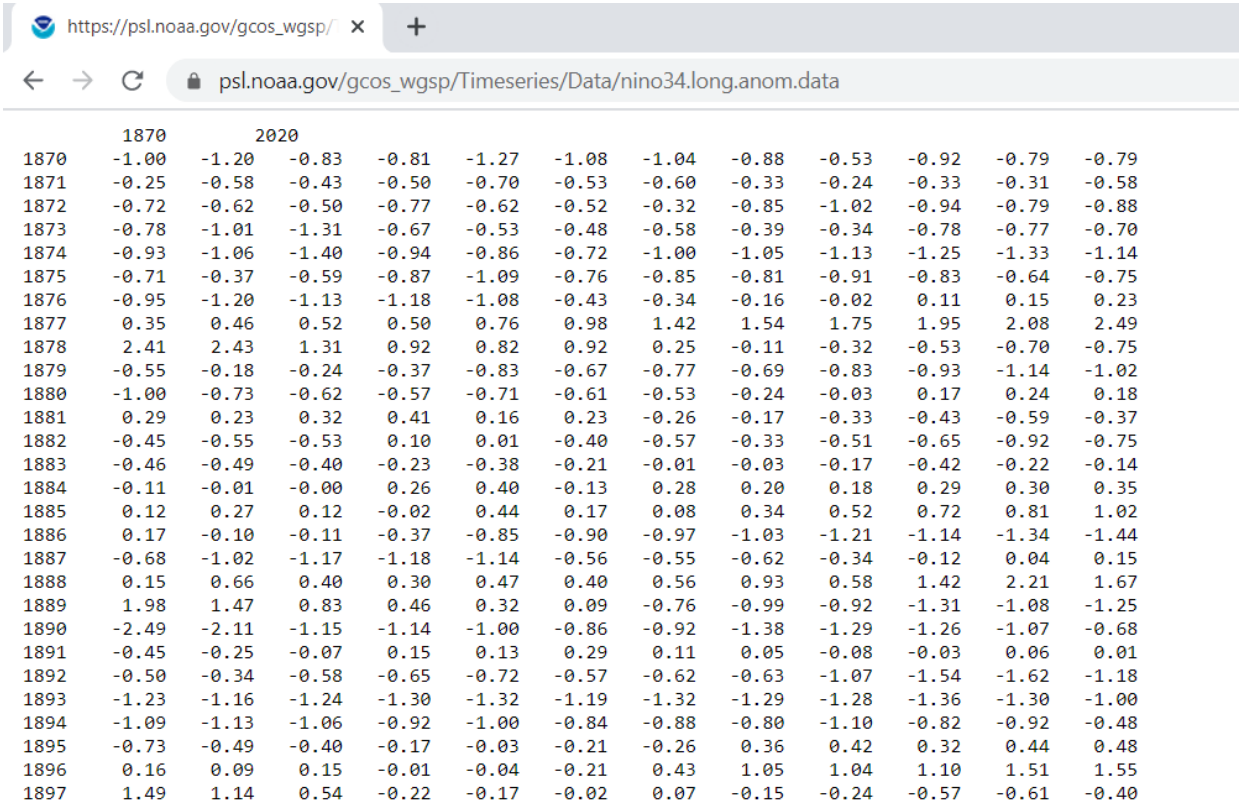
```
PV <- read.csv2("dados/portoVelho.csv", colClasses = classes)
head(PV)
```

```
##         Data Vazao
## 1 1966-09-01 26021
## 2 1967-09-01 31095
## 3 1968-09-01 25731
## 4 1969-09-01 28658
## 5 1970-09-01 32650
## 6 1971-09-01 35369
```

Lendo direto de uma URL

As funções `read.table()` podem fazer leitura de dados diretamente de uma URL.

Por exemplo, os dados da anomalia de temperatura oceânia na região do NINO 3.4:



	1870						2020					
1870	-1.00	-1.20	-0.83	-0.81	-1.27	-1.08	-1.04	-0.88	-0.53	-0.92	-0.79	-0.79
1871	-0.25	-0.58	-0.43	-0.50	-0.70	-0.53	-0.60	-0.33	-0.24	-0.33	-0.31	-0.58
1872	-0.72	-0.62	-0.50	-0.77	-0.62	-0.52	-0.32	-0.85	-1.02	-0.94	-0.79	-0.88
1873	-0.78	-1.01	-1.31	-0.67	-0.53	-0.48	-0.58	-0.39	-0.34	-0.78	-0.77	-0.70
1874	-0.93	-1.06	-1.40	-0.94	-0.86	-0.72	-1.00	-1.05	-1.13	-1.25	-1.33	-1.14
1875	-0.71	-0.37	-0.59	-0.87	-1.09	-0.76	-0.85	-0.81	-0.91	-0.83	-0.64	-0.75
1876	-0.95	-1.20	-1.13	-1.18	-1.08	-0.43	-0.34	-0.16	-0.02	0.11	0.15	0.23
1877	0.35	0.46	0.52	0.50	0.76	0.98	1.42	1.54	1.75	1.95	2.08	2.49
1878	2.41	2.43	1.31	0.92	0.82	0.92	0.25	-0.11	-0.32	-0.53	-0.70	-0.75
1879	-0.55	-0.18	-0.24	-0.37	-0.83	-0.67	-0.77	-0.69	-0.83	-0.93	-1.14	-1.02
1880	-1.00	-0.73	-0.62	-0.57	-0.71	-0.61	-0.53	-0.24	-0.03	0.17	0.24	0.18
1881	0.29	0.23	0.32	0.41	0.16	0.23	-0.26	-0.17	-0.33	-0.43	-0.59	-0.37
1882	-0.45	-0.55	-0.53	0.10	0.01	-0.40	-0.57	-0.33	-0.51	-0.65	-0.92	-0.75
1883	-0.46	-0.49	-0.40	-0.23	-0.38	-0.21	-0.01	-0.03	-0.17	-0.42	-0.22	-0.14
1884	-0.11	-0.01	-0.00	0.26	0.40	-0.13	0.28	0.20	0.18	0.29	0.30	0.35
1885	0.12	0.27	0.12	-0.02	0.44	0.17	0.08	0.34	0.52	0.72	0.81	1.02
1886	0.17	-0.10	-0.11	-0.37	-0.85	-0.90	-0.97	-1.03	-1.21	-1.14	-1.34	-1.44
1887	-0.68	-1.02	-1.17	-1.18	-1.14	-0.56	-0.55	-0.62	-0.34	-0.12	0.04	0.15
1888	0.15	0.66	0.40	0.30	0.47	0.40	0.56	0.93	0.58	1.42	2.21	1.67
1889	1.98	1.47	0.83	0.46	0.32	0.09	-0.76	-0.99	-0.92	-1.31	-1.08	-1.25
1890	-2.49	-2.11	-1.15	-1.14	-1.00	-0.86	-0.92	-1.38	-1.29	-1.26	-1.07	-0.68
1891	-0.45	-0.25	-0.07	0.15	0.13	0.29	0.11	0.05	-0.08	-0.03	0.06	0.01
1892	-0.50	-0.34	-0.58	-0.65	-0.72	-0.57	-0.62	-0.63	-1.07	-1.54	-1.62	-1.18
1893	-1.23	-1.16	-1.24	-1.30	-1.32	-1.19	-1.32	-1.29	-1.28	-1.36	-1.30	-1.00
1894	-1.09	-1.13	-1.06	-0.92	-1.00	-0.84	-0.88	-0.80	-1.10	-0.82	-0.92	-0.48
1895	-0.73	-0.49	-0.40	-0.17	-0.03	-0.21	-0.26	0.36	0.42	0.32	0.44	0.48
1896	0.16	0.09	0.15	-0.01	-0.04	-0.21	0.43	1.05	1.04	1.10	1.51	1.55
1897	1.49	1.14	0.54	-0.22	-0.17	-0.02	0.07	-0.15	-0.24	-0.57	-0.61	-0.40

Lendo direto de uma URL

O R é capaz de ler o documento `nino34.long.anom.data` diretamente no seu domínio `psl.noaa.gov`, utilizando a URL `https://psl.noaa.gov/gcos_wgsp/Timeseries/Data/nino34.long.anom.data`

```
NINO <- read.table("https://psl.noaa.gov/gcos_wgsp/Timeseries/Data/nino34.long.anom.data",
                   skip = 1,
                   sep = "\t")
head(NINO)
```

```
##
## 1  1870    -1.00   -1.20   -0.83   -0.81   -1.27   -1.08   -1.04   -0.88   -0.53   -0.92   -0.79
## 2  1871    -0.25   -0.58   -0.43   -0.50   -0.70   -0.53   -0.60   -0.33   -0.24   -0.33   -0.31
## 3  1872    -0.72   -0.62   -0.50   -0.77   -0.62   -0.52   -0.32   -0.85   -1.02   -0.94   -0.79
## 4  1873    -0.78   -1.01   -1.31   -0.67   -0.53   -0.48   -0.58   -0.39   -0.34   -0.78   -0.77
## 5  1874    -0.93   -1.06   -1.40   -0.94   -0.86   -0.72   -1.00   -1.05   -1.13   -1.25   -1.33
## 6  1875    -0.71   -0.37   -0.59   -0.87   -1.09   -0.76   -0.85   -0.81   -0.91   -0.83   -0.64
```

Leitura de dados no R

Outro procedimento útil com grandes bancos de dados, é estimar o tamanho do arquivos. Por exemplo, uma tabela com 1.000.000 de linhas e 10 colunas, na maior parte dos computadores atuais, irá armazenar cada dado utilizando 64 bits de memória, ou seja, 8 bytes. Assim, grosseiramente, esse arquivo irá conter 1.000.000 x 10 x 8 bytes = 8 x 10^7 bytes = 8 x 10^7 / 2^20 MB = 76 MB.

Esse valor deve ser comparado à memória RAM, contudo, deve-se ter em mente que outros programas podem estar usando a memória RAM do computados, além de outros objetos do R. Assim, esse conhecimento prévio é importante para evitar surpresas com arquivos muito grandes.

Leitura de dados no R

Função `readLines()`: útil para leitura de textos sem uma estrutura predefinida.

```
texto <- readLines("dados/texto.txt")
texto
```

```
## [1] "Uma linha de texto"         "Uma segunda linha de texto"
## [3] "Ãšltima linha de texto"
```

```
texto <- readLines("dados/texto.txt", encoding = "UTF-8");
texto
```

```
## [1] "Uma linha de texto"         "Uma segunda linha de texto"
## [3] "Última linha de texto"
```

Leitura de dados no R

Útil em mineração de texto, que não é o foco deste curso. Mas segue apenas um exemplo:

```
sort(table(unlist(strsplit(texto, " "))))
```

```
##
## segunda  Última     Uma      de   linha   texto
##       1       1       2       3       3       3
```

```
grep("Última", texto)
```

```
## [1] 3
```

```
grep("Última", texto, value = TRUE)
```

```
## [1] "Última linha de texto"
```

Leitura de dados no R

Observações

A leitura de planilhas Excel diretamente do R engloba mais aspectos do que a leitura de arquivos .txt e .csv.

Pessoalmente, eu prefiro exportar planilhas do formato .xls ou .xlsx para .csv ou .txt e depois ler no R.

Mas se o usuário preferir fazer a leitura diretamente de dados .xlsx, pode utilizar pacotes desenvolvidos para essa finalidade, por exemplo, os pacotes `readxl` ou `xlsx`.

Nesses pacotes, outras opções são necessárias, por exemplo, nome ou número da planilha.

Exportando dados do R para tabelas

Para cada função `read.table()`, existe uma função `write` corespondente.

Por exemplo, despois de uma série de cálculo e processamentos, o usuário pretende salvar suas tabelas em um arquivo .csv:

```
PV <- read.csv2("dados/portoVelho.csv", nrows = 5)
PV$padronizado <- scale(PV$Vazao)
PV
```

```
##         Data Vazao padronizado
## 1 1966-09-01 26021 -0.92086011
## 2 1967-09-01 31095  0.74193142
## 3 1968-09-01 25731 -1.01589550
## 4 1969-09-01 28658 -0.05669352
## 5 1970-09-01 32650  1.25151771
```

Exportando dados do R para tabelas

Para cada função `read.table()`, existe uma função `write` corespondente.

Por exemplo, despois de uma série de cálculo e processamentos, o usuário pretende salvar suas tabelas em um arquivo .csv:

```
write.table(PV, "dados/portoVelho_2.csv",
            row.names = FALSE, sep = ";", dec = ",",
            na = "")
```

Usando o pacote `readr`

O pacote `readr` foi desenvolvido por Hadley Wickham, anos após a linguagem R e também, após o desenvolvimento das funções `read.table()`, fazendo com a leitura fosse enormemente otimizada e algumas surpresas frequentes que resultam das funções `read.table()`.

Em geral, para cada função `read.table()`, `read.csv()`, etc., existe uma função equivalente no pacote `readr`, por exemplo, `read_table()` e `read_csv()`.

Um dos recursos interessantes do pacote `readr` é uma barra de progresso da leitura de um determinado arquivo.

Usando o pacote **readr**

Exemplo:

```
library(readr)
PV <- read_csv2("dados/portoVelho.csv")
head(PV)
```

```
## # A tibble: 6 x 2
##   Data       Vazao
##   <date>     <dbl>
## 1 1966-09-01 26021
## 2 1967-09-01 31095
## 3 1968-09-01 25731
## 4 1969-09-01 28658
## 5 1970-09-01 32650
## 6 1971-09-01 35369
```

Usando o pacote `readr`

O resultado da leitura usando `readr` é um tipo especial de data frame, chamado `tibble`. Falaremos mais deste tipo de objeto quando aprofundarmos no manuseio de dados e tabelas.

```
## # A tibble: 6 x 2
##   Data       Vazao
##   <date>     <dbl>
## 1 1966-09-01 26021
## 2 1967-09-01 31095
## 3 1968-09-01 25731
## 4 1969-09-01 28658
## 5 1970-09-01 32650
## 6 1971-09-01 35369
```

Usando o pacote **readr**

Em geral a definição da classe de um objeto usando `readr` é eficiente, porém, também sujeita a falhas. Para especificar o tipo de objeto de cada coluna, o argumento `col_types` pode ser utilizado. Por exemplo:

```
PV <- read_csv2("dados/portoVelho.csv",
                col_types = "Dd")
head(PV)
```

```
## # A tibble: 6 x 2
##   Data       Vazao
##   <date>     <dbl>
## 1 1966-09-01 26021
## 2 1967-09-01 31095
## 3 1968-09-01 25731
## 4 1969-09-01 28658
## 5 1970-09-01 32650
## 6 1971-09-01 35369
```

Usando o pacote `readr`

Em geral a definição da classe de um objeto usando `readr` é eficiente, porém, também sujeita a falhas. Para especificar o tipo de objeto de cada coluna, o argumento `col_types` pode ser utilizado. Por exemplo:

```
PV <- read_csv2("dados/portoVelho.csv",
                col_types = "cd")
head(PV)
```

```
## # A tibble: 6 x 2
##   Data       Vazao
##   <chr>      <dbl>
## 1 1966-09-01 26021
## 2 1967-09-01 31095
## 3 1968-09-01 25731
## 4 1969-09-01 28658
## 5 1970-09-01 32650
## 6 1971-09-01 35369
```

Usando o pacote `readr`

Principal vantagem: velocidade.

Barra de progresso, o que é interessante na leitura de grandes bancos de dados.

Também lê URL diretamente, assim como as funções `read.table()`.