Abrindo dados no R

Leonardo Sangali Barone

March 27, 2017

Abrindo dados no R

Neste tutorial vamos cobrir uma série de métodos disponíveis para abrirmos arquivos de texto e excel no R. Vamos dar atenção aos argumentos das funções de forma a solucionar dificuldades de abertura de dados com diferentes características ou em sistemas operacionais variados.

Pacotes no R

Antes de avançarmos à tarefa principal, vamos aprender um pouco mais sobre pacotes. Já foi destacado diversas vezes que uma das vantagens do R é a existência de uma comunidade produtiva e que desenvolve continuamente novas funcionalidades, tudo em código aberto.

Para instalarmos um novo pacote de R que esteja disponível no CRAN – "The Comprehensive R Archive Network" – utilizamos a função *install.packages*. Veja o exemplo com o pacote *beepr*:

```
install.packages("beepr")
```

Note que o nome do pacote deve estar em parêntese. Além disso, é possível que você tenha sido perguntad@ sobre de qual servidor do CRAN você quer baixar o pacote. A escolha em nada muda o resultado, exceto pelo tempo de duração.

Uma vez que um pacote foi instalado, ele está disponível em seu computador, mas não ainda para uso. Apenas depois de executarmos a função *library* é que teremos o pacote em nossa "biblioteca" de funções.

```
library(beepr)
```

Você pode dispensar as aspas ao usar a função *library*, pois é opcional. A função *require* é semelhante a *library* e a ignoraremos.

Abrindo dados com da funções do pacote utils

Quando você inicia uma nova sessão de R, alguns pacotes já estão automaticamente carregados. *utils* é um deles, e ele contém as funções mais conhecidas de abertura de dados em arquivos de texto.

A principal função é *read.table*. Use a função *args* para explorar seus argumentos:

```
args(read.table)
```

```
## function (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)
## NULL
```

É imprescindível que a função *read.table* receba como primeiro argumento um arquivo de dados. Note que o caminho para o arquivo deve estar completo (ex: "C:\\User\\Documents\\file.txt") ou ele deve estar no seu **working directory** (wd). Mas como eu descubro meu wd?

Caminhos no R

```
getwd()
```

E como eu altero meu wd?

```
setwd("C:\\User\\Documents")
```

Simples e muito útil para evitar escrever "labirintos de pastas" ao importar dados.

Um detalhe fundamental para quem usa Windows: os caminhos devem ser escritos com duas barras no lugar de uma, como no exemplo acima. É uma chatice e a melhor solução é mudar definitivamente para linux.

Vamos supor que você queira abrir diversos arquivos ("file1.txt" e "file2.txt", por exemplo) que estão em uma pasta diferente do seu wd, por exemplo "C:\\User\\Downloads\\". Mudar o wd pode não ser conviente, mas escrever o caminho todo é menos ainda. Uma solução é criar usar *file.path* para cadar arquivo armazenando a pasta e o caminho dos arquivos em algum objeto.

```
pasta <- "C:\\User\\Downloads\\"
path_file1 <- file.path(pasta, "file1.txt")
path_file2 <- file.path(pasta, "file2.txt")
```

O código acima pode parecer pouco inteligente neste momento, mas tente pensar a combinação dele com loops para abrir diversos arquivos.

Use as funções que acabamos de ver para gerenciar caminhos de pastas no R. Vale a pena.

read.table

Voltando à função *read.table*, vamos examinar os argumentos seguintes a *file* usando um exemplo de dados retirado do Portal da Transparência. Extraí dos pagamentos do programa uma amostra de tamanho 10 e salvei em diversos arquivos com caracteristicas distintas.

Para facilitar nossa vida, vamos usar como argumento "file" o endereço dos dados no repositório do curso. O primeiro arquivo está no enderço que guardaremos em "file1".

```
file1 <- "https://raw.githubusercontent.com/leobarone/FLS6397/master/data/bf_amostra_hv.csv"
```

Esse arquivo contém cabeçalho, ou seja, a primeira linha traz o nome das colunas. Por esta razão, informaremos "header = T". Ignore por hora o argumento "sep".

```
dados <- read.table(file1, header = T, sep = ",")
head(dados)
```

```
##   uf codmunic             munic        nis valor
## 1 SP     7107         SAO PAULO 2147483647   124
## 2 PI     1165        PIRACURUCA 2147483647   124
## 3 CE     1423          IRAUCUBA 2147483647   188
## 4 MA      739             BREJO 2147483647   250
## 5 BA     3717 MARCIONILIO SOUZA 2147483647   130
## 6 TO     9385            GURUPI 2147483647   366
```

O que aconteceria se escolhessemos "header = F"?

```
dados <- read.table(file1, header = F, sep = ",")
head(dados)
```

```
##   V1       V2                V3         V4    V5
## 1 uf codmunic             munic        nis valor
## 2 SP     7107         SAO PAULO 2147483647   124
## 3 PI     1165        PIRACURUCA 2147483647   124
## 4 CE     1423          IRAUCUBA 2147483647   188
## 5 MA      739             BREJO 2147483647   250
## 6 BA     3717 MARCIONILIO SOUZA 2147483647   130
```

Em primeiro lugar, o nome das variáveis é inserido automaticamente e na sequência V1, V2, ..., Vn, onde n é o número de colunas. Além disso, os nomes das variável é lido como se fosse uma observação. Disso resulta que todas as variáveis serão lidas com um texto na primeira coluna, resultando, em "character" ou "factor" a depender das características dos dados.

```
str(dados)
```

```
## 'data.frame':    11 obs. of  5 variables:
##  $ V1: Factor w/ 10 levels "AM","BA","CE",..: 10 8 7 3 4 2 9 6 5 5 ...
##  $ V2: Factor w/ 11 levels "1165","1423",..: 11 8 1 2 9 4 10 7 5 6 ...
##  $ V3: Factor w/ 11 levels "BREJO","COIMBRA",..: 8 10 9 5 1 7 3 4 2 11 ...
##  $ V4: Factor w/ 2 levels "2147483647","nis": 2 1 1 1 1 1 1 1 1 1 ...
##  $ V5: Factor w/ 10 levels "124","130","179",..: 10 1 1 4 6 2 9 8 5 7 ...
```

Vamos observar agora uma versão dos dados que não contém a primeira linha como nome das variáveis. Os dados estão no url armazenado em file2:

```
file2 <- "https://raw.githubusercontent.com/leobarone/FLS6397/master/data/bf_amostra_nv.csv"
```

Como já havíamos visto, quando não há cabeçalho na primeira linha, os nomes são inseridos automaticamente:

```
dados <- read.table(file2, header = F, sep = ",")
head(dados)
```

```
##   V1   V2                V3         V4  V5
## 1 SP 7107         SAO PAULO 2147483647 124
## 2 PI 1165        PIRACURUCA 2147483647 124
## 3 CE 1423          IRAUCUBA 2147483647 188
## 4 MA  739             BREJO 2147483647 250
## 5 BA 3717 MARCIONILIO SOUZA 2147483647 130
## 6 TO 9385            GURUPI 2147483647 366
```

E se cometermos o erro de indicar que há cabeçalho quando não há?

```
dados <- read.table(file2, header = T, sep = ",")
head(dados)
```

```
##   SP X7107         SAO.PAULO X2147483647 X124
## 1 PI  1165        PIRACURUCA  2147483647  124
## 2 CE  1423          IRAUCUBA  2147483647  188
## 3 MA   739             BREJO  2147483647  250
## 4 BA  3717 MARCIONILIO SOUZA  2147483647  130
## 5 TO  9385            GURUPI  2147483647  366
## 6 PA   621   IPIXUNA DO PARA  2147483647  294
```

A primeira linha de dados se torna o nome das variáveis (inclusive os números antecedidos por um "X").