Programação em R

Julho, 2016

O processo de análise de dados

Todo processo de análise de dados envolve, em geral, três grandes etapas: a entrada de dados no programa; o processamento e análise desses dados; a saída de dados do programa. Nesta aula apresentaremos a primeira e a última etapa: alguns modos de entrada e saída de dados no R.

Arquivos, pastas e os formatos próprios do R: RData e rds

Diretório de trabalho

Toda vez que você abre uma sessão do R, ele realizará operações de leitura e gravação de dados no diretório de trabalho.

```
# qual é o diretório de trabalho atual?
# no seu computador o resultado vai ser diferente
getwd() # get working directory
[1] "C:/"
```

A qualquer momento você pode alterar o diretório de trabalho com `setwd()`.

```
setwd("D:/Curso de R")
getwd()
[1] "D:/Curso de R"
```

Diretório de trabalho

Note que o separador é o contrário do que se usa no windows, pois a barra \ é um comando especial. Uma opção, caso você queira usar o padrão do windows, é usar duas barras.

```
setwd("C:\\diretorio1\\diretorio2")
```

Existe, ainda, uma função de conveniência que cria o caminho do arquivo para você a `file.path()`

```
file.path("C:",
          "diretorio1",
          "diretorio2",
          "arquivo.csv")
## [1] "C:/diretorio1/diretorio2/arquivo.csv"
```

Manipulando arquivos e pastas

O R possui uma série de funções para manipulação de arquivos e pastas. Vejamos algumas:

```
# listar os arquivos e diretórios que estão em uma pasta
list.files()
## [1] "Dados"     "io.pdf"    "io.Rmd"    "Rprof.out"

# listar os diretórios e subdiretórios que estão
# em uma pasta
list.dirs()
## [1] "."                   "./Dados"             "./Dados/Agosto"
## [4] "./Dados/outra pasta" "./Dados/Setembro"
```

Manipulando arquivos e pastas

```
# verificar se um arquivo existe
file.exists("io.pdf")
## [1] TRUE

file.exists("arquivoquenaoexiste")
## [1] FALSE
```

Manipulando arquivos e pastas

```
# criar arquivo
file.create("arquivoquenaoexiste")
## [1] TRUE

file.exists("arquivoquenaoexiste")
## [1] TRUE
```

Manipulando arquivos e pastas

```
# remover arquivo
file.remove("arquivoquenaoexiste")
## [1] TRUE

file.exists("arquivoquenaoexiste")
## [1] FALSE
```

Manipulando arquivos e pastas

```
# criar pastas
dir.create("Nova Pasta"); list.dirs()
## [1] "."                    "./Dados"              "./Dados/Agosto"
## [4] "./Dados/outra pasta" "./Dados/Setembro"     "./Nova Pasta"

file.remove("Nova Pasta"); list.dirs()
## [1] TRUE
## [1] "."                    "./Dados"              "./Dados/Agosto"
## [4] "./Dados/outra pasta" "./Dados/Setembro"
```

Manipulando arquivos e pastas

É possível também renomear e mover arquivos com `file.rename()`, ou copiar arquivos `file.copy()` entre outras funções. No limite, você pode usar diretamente os comandos do `shell` (DOS) do Windows.

```
# Cria diretório via DOS
shell("md teste")

# Remove diretório via DOS
shell("rmdir teste")
```

RData, rda e rds

Como vimos em exemplos durante as aulas, o R tem dois formatos próprios:

- `RData` ou `rda`: salva um ou vários objetos da área de trabalho. Carrega com o mesmo nome que foi salvo.
- `rds`: salva apenas um objeto. É possível carregar com nome diferente.

RData, rda e rds

Você pode salvar objetos no formato `RData` ou `rda` com a função `save()`. Para carregar, use a função `load()`. É possível salvar mais de um objeto ao mesmo tempo.

```
mtcars <- mtcars
# salva em rdata
save(mtcars, file = "mtcars.RData")
rm(mtcars)
ls()
## character(0)

# carrega novamente
load(file = "mtcars.RData")
ls()
## [1] "mtcars"
```

RData, rda e rds

A segunda opção para salvar e ler objetos do R são os objetos do tipo `rds`. Para tanto você irá utilizar as funções `saveRDS()` e `readRDS()`. Uma das principais diferenças com relação a `save()` e `load()` é que, enquanto estas salvam e carregam os objetos com o seu nome original, as funções RDS permitem a você carregar o objeto com um nome diferente.

```
saveRDS(mtcars, file = "mtcars.rds")
rm(mtcars)

# carrega em um objeto com nome diferente
dados <- readRDS("mtcars.rds")
ls()
## [1] "dados"
```

Planilhas e web: csv, xlsx, XML e JSON

CSV

Uma das formas mais convenientes de importar e exportar dados pelo R e por meio de arquivos `.csv`. O csv é um formato entendido por virtualmente quase todo software. Existe uma série de funções (derivadas da `read.table`) que fazem este trabalho, veja mais em `?read.table`.
Argumentos que necessitam atenção especial:

- `dec`: determina o símbolo utilizado para decimal. No Brasil, utilizamos "," mas em muitos outros lugares o padrão é ".".
- `sep`: como os dados estão separados? Por tab (\t), por vírgula (,), por ponto e vírgula (;), por espaço (" ")?
- `fileEncoding`: o padrão do R, em geral, é trabalhar com caracteres ASCII. No Brasil, são comuns os padrões latin1 ou UTF-8. Ler o arquivo com o padrão errado pode resultar em caracteres "estranhos".

CSV

Para ilustrar como salvar um csv, vamos utilizar a função `write.csv2()`. Ela e igual à `write.csv()` mas já tem como padrão `sep = ";"` e `dec = ","`, que são comuns no Brasil.

```
write.csv2(mtcars, "mtcars.csv")
```

csv

Para ilustrar como ler um csv, vamos utilizar a função `read.csv()`, colocando os parâmetros corretos para leitura:

```
carros <- read.csv("mtcars.csv", dec = ",", sep = ";")
```

Neste caso poderíamos ter lido também com `read.csv2()` que já teria os parâmetros desejados.

xlsx

Outra forma de interação bastante comum é com dados em planilha Excel. Há vários pacotes que permitem ler e salvar arquivos em Excel. Esse pacotes, em geral, usam bibliotecas externas escritas em outras linguagens, por exemplo:

Pacote	Acessa o Excel por
readxl	**C e C++ (somente leitura)**
xlsx	**Java**
XLconnect	Java
openxlsx	C++
RODBC	ODBC
gdata	perl

É importante saber esse detalhe porque você pode encontrar alguma incompatibilidade dependendo do computador que estiver usando. Para leitura, recomendo o pacote `readxl` e para gravação o `xlsx`.

xlsx

Vejamos um exemplo de como salvar os dados em excel com o pacote `xlsx`. Para salvar o objeto `mtcars` como uma planilha excel, você digitaria o seguinte comando:

```
library(xlsx)
write.xlsx(mtcars, "mtcars.xlsx")
```

xlsx

Vamos agora ler os dados da planilha excel que acabamos de criar tanto com o pacote `xlsx` quanto com o pacote `readxl`:

```
ex1 <- read.xlsx("mtcars.xlsx",sheetIndex = 1)

# com read_excel
library(readxl)
ex2 <- read_excel("mtcars.xlsx", sheet = 1)
```

JSON e XML

Dados provenientes da web em geral vêm em formatos JSON ou XML. Os pacotes que recomendo para leitura desses dados são o `jsonlite` e `xml2`, respectivamente. Vimos um exemplo de leitura de dados JSON quando carregamos nossos dados de buscas do Google.

Formato	Pacote recomendado
JSON	jsonlite
XML	xml2

Pacotes estatísticos e bancos de dados

STATA, SPP e SAS

Para ler dados de pacotes estatísticos, existem dois pacotes principais: o `foreign` e o `haven`. Aqui recomendo utilizar o `haven`. As funções são bem intuitivas e simples de utilizar. Principais funções:

Função do Haven	Pacote estatístico
`read_dta()`	Lê arquivos do STATA
`read_stata()`	Lê arquivos do STATA
`write_stata()`	Salva arquivos STATA
`read_por()`	Lê arquivos do SPSS
`read_sav()`	Lê arquivos do SPSS
`read_spss()`	Lê arquivos do SPSS
`write_sav()`	Salva arquivos do SPSS
`read_sas()`	Lê arquivos do SAS

Bancos de dados

Por fim, há pacotes do R para conversar com bancos de dados externos, entre eles cabe destacar:

Pacote	Conversa com...
RMySQL	MySQL
RDOBC	SQL Server via ODBC
RPostgres	Postgres
RSQLite	SQLite
rmongodb, mongolite	MongoDB
RCassandra	Cassandra

... e vários outros pacotes!