



Importação e exportação de dados

Giovanna Segantini

giovanna.ufrn@gmail.com

Introdução

A primeira etapa de uma pesquisa é a importação de dados.

Objetivos da aula:

- ▶ Aprender a importar e exportar dados contidos em arquivos locais
- ▶ Aprender a importar dados de forma dinâmica, utilizando pacotes especializados e uma conexão da internet.

Formatos de armazenamento

- ▶ Dados delimitados em texto (csv)
- ▶ Microsoft Excel (xls,xlsx);
- ▶ Arquivos nativos do R (RData e rds)
- ▶ Formato fst (fst)
- ▶ SQLite (SQLITE)
- ▶ Texto não estruturado (txt).

Em geral o formato mais flexível e utilizado é .csv

Mostrando o diretório de trabalho

É com base nesse diretório o R procura os arquivos para importar dados.

É nesse mesmo diretório que o R salva arquivos.

```
my.dir <-getwd()  
print(my.dir)
```

```
## [1] "C:/Users/gato/Desktop/Github/analise-de-dados"
```

O resultado apresenta a pasta a onde os arquivos serão salvos e resgatados. ** O Rcloud não permite a mudança no diretório.**

Importando arquivos CSV

Os arquivos CSV são bastante utilizados para disponibilização de dados. É um formato bem antigo e que utiliza vírgulas para separação dos valores. No entanto, implementações mais sofisticadas utilizam aspas duplas ("") entorno do conteúdo e podem ter outro tipo de separador, como o ponto e vírgula (;).

Como sugestão para evitar problemas, antes de prosseguir para a importação de dados em um arquivo .csv, deve-se abrir o arquivo em um editor de texto qualquer e verificar:

- ▶ A existência de texto antes dos dados e a necessidade de ignorar algumas linhas iniciais;
- ▶ A existência ou não dos nomes das colunas;
- ▶ O símbolo separador de colunas;
- ▶ O símbolo de decimal, o qual deve ser o ponto (.).

Um pacote bastante popular para a importação de arquivos CSV é o readr. Caso não o tenha instalado, faça a instalação e carregue o mesmo utilizando:

```
install.packages("readr")  
  
library(readr)  
  
cia_2016 <- read_delim("empresas_economatica_2016.csv",  
  ";", escape_double = FALSE, trim_ws = TRUE)  
  
View(cia_2016)
```

Agora, vamos checar a classe das nossas colunas. Para isso, utilizamos a função *glimpse* do pacote *dplyr*:

```
#install.packages("dplyr")
```

```
library("dplyr")
```

```
# print column classes  
dplyr::glimpse(cia_2016)
```

```
#ou
```

```
str(cia_2016)
```

NOTA: O uso da mensagem com as classes das colunas é particularmente útil quando o arquivo importado tem várias colunas e a definição manual de cada classe exige muita digitação.

Importando arquivos Excel (xls e xlsx)

Os principais pacotes para importar esses tipos de arquivo são: XLConnect (Mirai Solutions GmbH 2016), xlsx (Dragulescu 2014) e readxl (Wickham 2016a).


```
#install.packages("readxl")  
  
library(readxl)  
  
#Ajustando a importação do banco de dados  
my.df <- read_excel("cotacoes.xlsx", skip=1)  
View(my.df)
```

Em alguns casos, o arquivo onde estão as informações possui em suas primeiras linhas instruções para utilização. Nestes casos, você pode usar o argumento skip para indicar quantas linhas devem ser puladas.

Importando dados em outros formatos

Pode-se importar dados diretamente dos programas estatísticos.

```
#install.packages("haven")  
  
library(haven)  
Segmentao_Mercado <- read_sav("Segmentao_Mercado.sav")  
View(Segmentao_Mercado)
```

Importando dados via Pacotes

- ▶ BatchGetSymbols: Yahoo Finance e Google Finance.
- ▶ Quandl: dados da Quandl
- ▶ BETS: Fundação Getúlio Vargas
- ▶ rbcbr: acesso direto ao API do BCB
- ▶ GetDFPDData: DF distribuídas pela B3 e pela CVM
- ▶ GetHFData: dados de alta frequência na B3.
- ▶ GetTDDData: preços e retorno do Tesouro Direto

Carregando os pacotes

```
my.pkgs <- c('BatchGetSymbols', 'Quandl', 'BETS',  
'rbcB', 'GetDFPData',  
'GetHFData', 'GetTDDData', 'dplyr')  
  
install.packages(my.pkgs)
```

BatchGetSymbols

Esse pacote faz a comunicação do R com os dados financeiros disponíveis no Yahoo Finance e Google Finance.

```
library(BatchGetSymbols)
library(dplyr)

help(package = BatchGetSymbols )

# set tickers
my.tickers <- c("PETR4.SA", "CIEL3.SA",
  "GGBR4.SA", "GOAU4.SA")
```

```
# set dates and other inputs
first.date <- Sys.Date()-30
last.date <- Sys.Date()

thresh.bad.data <- 0.95 # sets percent threshold for bad

bench.ticker <- "^BVSP" # set benchmark as ibovespa

cache.folder <- "BGS_Cache" # set folder for cache

l.out <- BatchGetSymbols(tickers = my.tickers,
  first.date = first.date,
  last.date = last.date,
  bench.ticker = bench.ticker,
  thresh.bad.data = thresh.bad.data,
  cache.folder = cache.folder)
```

A saída de `BatchGetSymbols` é um objeto do tipo lista, semelhante ao dataframe porém mais flexível. O acesso a cada elemento de uma lista pode ser feito pelo operador `$`.

```
print(l.out$df.control)
```

Objeto *df.control* mostra que todos tickers foram válidos, com um total de 22 observações para cada ativo. Note que as datas batem 100% com o Ibovespa.

Quanto aos dados financeiros, esses estão contidos no elemento *df.tickers* de *l.out*:

```
print(l.out$df.tickers)
```

Outra função útil do pacote é `BatchGetSymbols::GetIbovStocks`, a qual importa a composição atual do índice Ibovespa diretamente do site da B3.

```
# set tickers  
df.ibov <- GetIbovStocks()  
my.tickers <- paste0(df.ibov$tickers, '.SA')
```

Note que utilizamos a função *paste0* para adicionar o texto 'SA' para cada ticker em `df.ibov$tickers`

```
# set dates and other inputs  
first.date <- Sys.Date()-30  
last.date <- Sys.Date()  
thresh.bad.data <- 0.95      # sets percent threshold for bad  
bench.ticker <- '^BVSP'     # set benchmark as ibovespa  
cache.folder <- 'BGS_Cache' # set folder for cache
```



```
l.out <- BatchGetSymbols(tickers = my.tickers,  
                        first.date = first.date,  
                        last.date = last.date,  
                        bench.ticker = bench.ticker,  
                        thresh.bad.data = thresh.bad.data,  
                        cache.folder = cache.folder)
```

GetDFPData

Pacote GetDFPData (Perlin 2017) fornece uma interface aberta para todas as demonstrações financeiras distribuídas pela B3 e pela CVM nos sistemas DFP, FRE e FCA.

Usuários podem selecionar as empresas disponíveis, o intervalo de datas e baixar os dados como uma planilha do Excel ou um arquivo compactado com vários arquivos csv.

Leia mais sobre o pacote <http://bibliotecadigital.fgv.br/ojs/index.php/rbfin/article/view/78654/77107>

```
library(GetDFPDData)
```

```
df.info <- gdfpd.get.info.companies(type.data =  
                                     'companies')
```

```
##
```

```
## Reading info file from github
```

```
## Found 49063 lines for 525 companies [Actives = 524 In
```

```
## Last file update: 2020-04-17
```

```
## Caching RDATA into tempdir()
```

```
dplyr::glimpse(df.info)
```

```
## Rows: 525
```

```
## Columns: 16
```

```
## $ name.company      <chr> "521 PARTICIPAÇÕES S.A. - P
```

```
## $ id.company        <int> 16330, 16284, 24732, 21725,
```

```
## $ cnpj              <dbl> 1.547749e+12, 1.851771e+12,
```

```
## $ date.registration <date> 1997-07-11, 1997-05-30, 20
```

```
## $ date.constitution <date> 1996-07-30, 1997-04-02, 20
```

Essa tabela disponibiliza os identificadores numéricos das empresas, setores de atividades, atual segmento de governança, tickers negociados na bolsa e situação atual (ativa ou não).

A função *gdfpd.search.company* permite que o usuário procure o nome oficial de uma instituição.

```
gdfpd.search.company("ambev")
```

```
##
```

```
## Reading info file from github
```

```
## Found 49063 lines for 525 companies [Actives = 524 In
```

```
## Last file update: 2020-04-17
```

```
## Caching RDATA into tempdir()
```

```
##
```

```
## Found 1 companies:
```

```
## AMBEV S.A. | situation = ATIVO | first date = 2012-12-31
```

```
## [1] "AMBEV S.A."
```

Usamos a função principal do pacote, `gdfpd.GetDFPDData`, para baixar informações.

```
library(GetDFPDData)
```

```
library(dplyr)
```

```
# set options
```

```
name.companies <- 'AMBEV S.A.'
```

```
first.date <- '2017-12-31'
```

```
last.date <- '2018-12-31'
```

```
# download data
```

```
df.reports <- gdfpd.GetDFPDData(name.companies = name.companies,  
                                first.date = first.date,  
                                last.date = last.date)
```

O objeto resultante de *gdfpd.GetDFPDData* é um tibble, um tipo de objeto tabular que permite colunas do tipo lista. Vamos dar uma olhada no conteúdo de *df.reports*:

```
dplyr::glimpse(df.reports)
```

Para uma melhor visualização dos demonstrativos financeiros, o pacote também inclui função *gdfpd.convert.to.wide* para converter os dados para formato wide, com as datas preenchendo colunas. Veja a seguir:

```
# convert assets to wide
fr.assets.wide <- gdfpd.convert.to.wide(fr.assets)

# display it
print(fr.assets.wide)
```

Exportação dos dados usando o pacote *GetDFPDData*

A exportação dos dados também é fácil, basta selecionar um formato ('csv' ou 'xlsx') e usar *gdfpd.export.DFP.data*: