

INF 1514

Introdução à Análise de Dados

Material Entrada e saída de arquivos



Entrada de dados

- Acessar e manipular um conjunto de dados é uma tarefa rotineira da análise de dados em geral, e o R, junto com seus pacotes, possui uma série de recursos que facilita a execução dessas tarefas.
- Nesta disciplina veremos como importar, exportar, alterar e manejar dados oriundos de diversas fontes.
- Iremos também analisar mais algumas questões importantes referentes ao uso de data frames e outras estruturas para armazenamentos de dados.
- Vale lembrar que o R tem disponível, já na sua instalação, diversos dados que podem ser usados para aprendizado da linguagem.
- Além disso, a maioria dos pacotes extra do R também apresentam conjuntos de dados para ilustrar algumas das suas funcionalidades.
- Para saber quais conjuntos de dados estão atualmente disponíveis na instalação, use a função **data()**.

Definindo o local dos dados

- O R trabalha com o conceito de *working directory (wd)*, ou seja, um diretório de trabalho (pasta) a partir do qual os scripts irão “ler” e “escrever” os arquivos de dados. Para verificar qual é o diretório de trabalho padrão do ambiente utilize o comando **getwd()**.

```
> getwd()  
[1] "C:/Users/user/Documents"
```

- O comando **setwd()** pode ser utilizado para mudar o diretório de trabalho padrão do R para leitura e escrita de arquivos.


```
> setwd('c:/PUC/INF1514')  
>  
> getwd()  
[1] "c:/PUC/INF1514"
```

```
> setwd('c:\\PUC\\INF1514')  
>  
> getwd()  
[1] "c:/PUC/INF1514"
```

Arquivos CSV

- Arquivos **CSV** (*Comma Separated Values*) são arquivos textos sem formatação alguma (negrito, itálico, etc) que podem ser visualizados por qualquer editor de textos sendo muito utilizados por diversas ferramentas para exportar seus dados ou mesmo para importar novos dados.
- Consiste em uma sequência de linhas onde cada linha representa um registro de dados. Dentro de cada linha, as informações das colunas (campos) são separadas por “,”.

Arquivo CSV com cinco linhas e quatro colunas contendo as colunas sigla, nome, população e PIB (mil R\$) de alguns estados brasileiros.



```
AC,Acre,732793,8477
AL,Alagoas,3120922,24575
AP,Amapá,668689,8266
AM,Amazonas,3480937,59779
BA,Bahia,14021432,154340
```

Arquivos CSV

- Na prática, em função da vírgula apresentar diversas funções, outros padrões, como “;” e “TAB”, também são utilizados para separar as informações das colunas.
- Há muitos pacotes para leitura de arquivos **CSV** como o **readr**, **vroom**, e o **data.table**, sendo que o R também apresenta funções nativas como a **read.csv()** e a **read.csv2()**.
- O comando **help** permite obter maiores detalhes das funções **read.csv()** e **read.csv2()**.

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

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

Importando arquivo CSV

- Antes de importarmos um arquivo **CSV**, é necessário conhecer algumas informações sobre o mesmo, como:
 - qual caractere é utilizado para separar as colunas;
 - se o delimitador das casas decimais é “,” ou “.”.
 - se existe algum caractere especial (aspas simples ou duplas, por exemplo) limitando os dados em cada coluna;
 - se os nomes das colunas estão na primeira linha do arquivo;
 - se o arquivo possui linhas iniciais e finais que precisam ser ignoradas.
- A função **file.head()** do pacote **descr**, que pode ser instalado pelo comando **install.packages("descr")**, permite identificar algumas dessas informações.
- O comando **read_lines()** do pacote **readr**, que pode ser instalado pelo comando **install.packages("readr")**, também pode ser utilizado.

Importando arquivo CSV

- Uso da função **file.head()** do pacote **descr**.

```
> library("descr")
> file.head("PIB2.CSV")
estado;descricao;populacao;pib
"AC";"Acre";"732793";"8477"
"AL";"Alagoas";"3120922";"24575"
"AP";"Amapá";"668689";"8266"
```

- A função nativa **read.csv2()** permite **importar** um arquivo no formato **CSV**.

```
> indicador <- read.csv2("PIB2.CSV", header = TRUE, sep = ";", quote = "'", dec = ",")
```

```
> indicador
```

	estado	descricao	populacao	pib
1	AC	Acre	732793	8477
2	AL	Alagoas	3120922	24575
3	AP	Amapá	668689	8266
4	AM	Amazonas	3480937	59779

O comando simples `read.csv2("PIB2.CSV")` também funcionaria neste caso, devido aos valores padrão dos parâmetros da função `read.csv2()`.

Importando arquivo CSV

- A função nativa **read.csv2()** retorna os dados em um objeto do tipo **data.frame** (detalhado nos próximos materiais) que, de forma simplificada, consiste em um tipo de dado que mantém os dados organizados em linhas e colunas, como uma tabela, podendo ter colunas com tipos diferentes.

```
> class(indicador)
[1] "data.frame"
```

- A função nativa **summary()** pode ser usada para conferir se a importação foi realizada com sucesso.

```
> summary(indicador)
```

estado	descricao	populacao	pib
Length:27	Length:27	Min. : 451227	Min. : 6341
Class :character	Class :character	1st Qu.: 2506152	1st Qu.: 24254
Mode :character	Mode :character	Median : 3512672	Median : 59779
		Mean : 7064174	Mean : 139633
		3rd Qu.: 8622044	3rd Qu.: 151194
		Max. :41252160	Max. :1247596

Importando arquivo CSV

- As funções nativas **head()** e o **tail()** são úteis para visualizar os primeiros e os últimos registros dos dados importados, respectivamente.
- É sempre muito importante observar os dados após a importação, pois, essa prática ajuda a identificar erros básicos.

```
> head(indicador)
```

	estado	descricao	populacao	pib
1	AC	Acre	732793	8477
2	AL	Alagoas	3120922	24575
3	AP	Amapá	668689	8266
4	AM	Amazonas	3480937	59779
5	BA	Bahia	14021432	154340
6	CE	Ceara	8448055	77865

```
> tail(indicador)
```

	estado	descricao	populacao	pib
22	RO	Rondônia	1560501	23561
23	RR	Roraima	451227	6341
24	SC	Santa Catarina	6249682	152482
25	SP	São Paulo	41252160	1247596
26	SE	Sergipe	2068031	23932
27	TO	Tocantins	1383453	17240

Importando arquivo CSV

- Para verificar a estrutura do objeto, ou seja, seus campos e tipos de dados, também pode ser utilizada a função **glimpse** do pacote **dplyr**.

```
> glimpse(indicador)
```

```
Rows: 27
```

```
Columns: 4
```

```
$ estado    <chr> "AC", "AL", "AP", "AM", "BA", ...
```

```
$ descricao <chr> "Acre", "Alagoas", "Amapá", "...
```

```
$ populacao <int> 732793, 3120922, 668689, 3480...
```

```
$ pib       <int> 8477, 24575, 8266, 59779, 154...
```

Importando arquivo CSV

- O pacote **readr** possui a função **read_csv2()** que permite carregar arquivos **CSV** em uma estrutura chamada **tibble**, que é um tipo especial de **data frame** que apresenta mais recursos para organização e manipulação de dados.
- Se os **dados a serem importados** estiverem em uma planilha do Microsoft Excel (**XLS**), algumas alternativas são:
 - **abrir** o arquivo no Excel e **salvar** a planilha como arquivo do tipo **CSV**.
 - usar a função **read_xlsx()** do pacote **readxl**, integrante do pacote **Tidyverse**, que será visto em materiais posteriores.

Arquivos com colunas de largura fixa

- São arquivos texto (**TXT**), como os arquivos **CSV**, mas sem delimitadores de coluna.
- Para sua importação, é preciso saber quantos caracteres cada coluna ocupa.
- Arquivos desse tipo não são muito comuns atualmente, mas muitas bases antigas e importantes estão disponíveis neste formato.
- Normalmente estes arquivos são acompanhados de um dicionário (tabela ou mesmo um texto) indicando a posição inicial e final de cada coluna, seus rótulos e os valores das colunas categóricas.

ACAcre	732793	8477
ALAlagoas	3120922	24575
APAmapá	668689	8266
AMAmazonas	3480937	59779
BABahia	14021432	154340

Importando arquivo TXT

- Uso da função **file.head()** do pacote **descr**.

```
> library("descr")
> file.head("PIB3.TXT")
ACAcre          732793      8477
ALAlagoas       3120922    24575
APAmapá         668689     8266
AMAmazonas      3480937    59779
BABahia         14021432   154340
```

- A função nativa **read.fwf()** permite **importar** um arquivo no formato **TXT**.

```
> indicador <- read.fwf("PIB3.TXT", col.names = c("estado", "descricao",
"populacao", "pib"), widths = c(2, 19, 8, 7), colClasses = c("character",
"character", rep("numeric", 2)))
> indicador
```

	estado	descricao	populacao	pib
1	AC Acre		732793	8477
2	AL Alagoas		3120922	24575
3	AP Amapá		668689	8266

Importando arquivo TXT

- A função interna **read.fwf()** retorna os dados em um objeto do tipo **data.frame** e, desta forma, as funções **summary()**, **head()**, **tail()**, **glimpse()** também se aplicam.
- O pacote **readr** possui a função **read_fwf()** que permite carregar arquivos **TXT** em estruturas **tibble**.

Arquivos JSON

- **JSON** (JavaScript Object Notation - Notação de Objetos JavaScript) é uma formatação leve de troca de dados. Para seres humanos, é fácil de ler e escrever e, para computadores, é fácil de interpretar e gerar.
- **JSON** lembra o padrão **XML** (é texto simples, é autodescritivo, é hierárquico - valores dentro de valores) mas é diferente da **XML** (é mais curto e simples, é mais rápido para processar, não possui palavras reservadas, etc).
- As principais regras de sintaxe **JSON** são:
 - dados **JSON** estão definidos aos pares no formato: **nome** : **valor**.
 - os dados são separados por vírgulas (,).
 - as chaves {} contêm objetos.
 - os colchetes [] expressam **matrizes/vetores**.

Importando arquivo JSON

```
{
  "estados":
    [
      { "sigla":"RJ" , "descricao":"Rio de Janeiro" },
      { "sigla":"SP" , "descricao":"São Paulo" },
      { "sigla":"ES" , "descricao":"Espírito Santo" }
    ]
}
```

```
> library(RJSONIO)
> lista <- fromJSON("estados.json")
> lista
$estados
$estados[[1]]
  sigla descricao
  "RJ" "Rio de Janeiro"
$estados[[2]]
  sigla descricao
  "SP" "São Paulo"
$estados[[3]]
  sigla descricao
  "ES" "Espírito Santo"
```

```
> lista$estados
[[1]]
  sigla descricao
  "RJ" "Rio de Janeiro"
[[2]]
  sigla descricao
  "SP" "São Paulo"
[[3]]
  sigla descricao
  "ES" "Espírito Santo"
> lista$estados[[1]]
  sigla descricao
  "RJ" "Rio de Janeiro"
> lista$estados[[1]][['sigla']]
[1] "RJ"
```


Importando arquivo JSON

```
{
  "investidor":
    [
      "Paulo Silva", "Cris Pinheiro", "Thomas Filho"
    ],
  "acao":
    [
      { "sigla": "PTR04", "quantidade": 77 },
      { "sigla": "IBU05", "quantidade": 25 }
    ],
  "carteira": "GOLD"
}
```

```
> library(RJSONIO)
> carteira <- fromJSON("carteira.json")
> carteira$investidor
[1] "Paulo Silva" "Cris Pinheiro" "Thomas Filho"
```

```
> carteira$acao
[[1]] [[1]]$sigla
[1] "PTR04"

[[1]]$quantidade
[1] 77

[[2]]
[[2]]$sigla
[1] "IBU05"

[[2]]$quantidade
[1] 25
> carteira$acao[[1]]$sigla
[1] "PTR04"
>
> carteira$carteira
[1] "GOLD"
```

Importando arquivo JSON

- Dependendo da estruturação do arquivo **JSON** a função **fromJSON** pode retornar objetos que sejam de tipos de dados distintos do R, como vetores e listas e, desta forma, as funções **summary()**, **head()**, **tail()**, **glimpse()** também podem se utilizadas uma vez que são funções genéricas.