Leitura de Dados

Manejo, Visualização e Compartilhamento de Dados

Nicholas A. C. Marino
github.com/nacmarino/compartilhaR

Estrutura da Aula

1. Revisando a leitura de dados no R
2. Usando o pacote `readr` para leitura de dados
3. Importando tabelas do Excel
4. Exportando tabelas
5. Exportando dados no formato de listas

Revisando a leitura de dados no R

- É bastante comum utilizarmos as funções `read.csv` ou `read.table`, ambas disponíveis na `base` do R, para realizar a leitura de um conjunto de dados armazenados no seu computador com extensão `.csv` e `txt`, respectivamente.
- Os dados importados através destas funções estarão disponíveis no ambiente de trabalho da sua sessão, e serão um objeto pertencente à classe `data.frame`.
    - Um `data.frame` contém duas dimensões, as linhas e as colunas. As colunas contém os valores das diferentes variáveis, que caracterizam cada unidade observacional registradas nas linhas.
    - Cada coluna armazena um único tipo de valor - somente números, somente caracteres, somente argumentos lógicos.

Exercício 1

1. Carregue o conjunto de dados `dados dos projetos.csv` no R e atribua a um objeto.
2. Determine a classe à qual pertença o objeto criado.
3. Determine a classe de objeto à qual pertence cada coluna do dados que você carregou.
4. Qual o tamanho das dimensões do conjunto de dados que você carregou?

Usando o pacote `readr` para a leitura de dados

- De maneira alternativa às funções disponíveis na `base` do R, também podemos importar e exportar um conjunto de dados no R utilizando as funções do pacote `readr`.
  - `read_<extensão>`, para importar;
  - `write_<extensão>`, para exportar.
- A sintaxe das funções deste pacote é similar àquela da `base`. No entanto, as funções do pacote `readr` atribuem **três** classes de objeto à tabela importanta: `tbl_df`, `tbl` e `data.frame`.

```
library(readr)
class(read_csv(file = "dados dos projetos.csv"))
```

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

tibble

- Toda a tabela de dados importada com o `readr` é um `tibble`:
  - Possui a mesma estrutura e dimensões de um `data.frame`;
  - Objetivo é facilitar a visualização dos dados na tabela;
  - A classe de objeto dos elementos de cada coluna é apresentada junto da própria tabela;
  - Previne a automatização da atribuição de certas classes de objeto a algumas colunas (*e.g.*, toda coluna com caracteres é um fator, o que torna difícil sua manipulação).

```
dados <- read_csv(file = "dados dos projetos.csv")
dados
```

tibble

```
## # A tibble: 119 x 26
##     Processo Chamada id_coordenador sexo  inicio termino instituicao uf
##        <dbl> <chr>             <int> <chr> <chr>  <chr>   <chr>       <chr>
##  1   4.86e10 Univer…               1 M     06/11… 05/11/… FURG        RS
##  2   4.01e10 BJT                   2 F     12/09… 11/09/… UFSCAR      SP
##  3   4.82e10 Univer…               3 M     08/11… 30/11/… UFPR-PRPG   PR
##  4   4.72e10 Univer…               4 F     14/12… 31/12/… UFRPE       PE
##  5   4.76e10 Univer…               5 M     30/10… 31/10/… UFPB        PB
##  6   4.82e10 Univer…               6 M     07/05… 05/05/… INPA        AM
##  7   4.80e10 Univer…               7 M     19/11… 18/11/… INPA        AM
##  8   4.46e10 Incuba…               8 M     02/08… 31/12/… PUC GOIAS   GO
##  9   4.58e10 PPBio …               9 M     12/12… 11/12/… UFRJ        RJ
## 10   4.72e10 Univer…              10 M     28/11… 30/11/… UERJ        RJ
## # ... with 109 more rows, and 18 more variables: cidade <chr>,
## #   regiao <chr>, duracao <int>, brasileiro <chr>, produtividade <chr>,
## #   doutorado <int>, doutorado_exterior <chr>, sanduiche <chr>,
## #   posdoc <chr>, indiceH <int>, bolsa_contratado <dbl>,
## #   bolsa_gasto <int>, capital_contratado <dbl>, capital_gasto <dbl>,
## #   custeio_contratado <dbl>, custeio_gasto <dbl>, total_fornecido <dbl>,
## #   total_gasto <dbl>
```

Importando tabelas do Excel {#anchor3}

- Na maior parte das vezes, no entanto, armazenamos nossos dados nos formtatos-padrão do Microsoft Excel, com as extensões `.xls` ou `.xlsx`.
- Nestes casos, podemos utilizar as funções do pacote `readxl` para importar tabelas de dados com estas extensões.
  - `read_xls`, para extensão `.xls`
  - `read_xlsx`, para entensão `.xlsx`
  - `read_excel`, como uma função genérica tanto ambas as extensões

```
library(readxl)
args(read_excel)
```

```
## function (path, sheet = NULL, range = NULL, col_names = TRUE,
##     col_types = NULL, na = "", trim_ws = TRUE, skip = 0, n_max = Inf,
##     guess_max = min(1000, n_max))
## NULL
```

Exercício 2

1. Importe os arquivos `autores.xls` e `revistas.xlsx` e armazene cada um deles em um objeto distinto.
2. A importação ocorreu sem problemas?
3. Qual o tamanho das dimensões de cada um destes objetos?
4. Selecione apenas a segunda coluna da tabela `revistas.xlsx`.
5. Selecione apenas as linhas 10 a 40 da tabela `revistas.xlsx`.
6. Exclua apenas a primeira coluna da tabela `revistas.xlsx`.

Exportando tabelas

- O processamento e análise de dados podem gerar um enorme volume de informações, que muitas vezes serão empregados em outro contexto ou plataforma que não aquela do ambiente de programação - por exemplo:
  - Criação de subconjuntos dos dados;
  - Resultados de análise estatísticas;
  - Predições de modelos para confecção de figuras.
- Portanto, uma importante tarefa também é exportar as tabelas de dados geradas dentro do ambiente de programação.
- Podemos organizar os dados que queremos exportar dentro de um `data.frame` (ou `tibble`), e utilizar as funções `write_csv`, `write_tsv` ou, ainda `write_delim` para salvar esta nova tabela de dados no diretório que você preferir.

Exportando tabelas

- No exemplo abaixo, estamos criando um `data.frame` com duas colunas, `x` e `y` e exportando o objeto que representa este `data.frame` utilizando a função `write_csv`.
- As funções `rep` e `seq` são utilizadas, respectivamente, para repetir e criar uma sequência de valores.

```
exemplo <- data.frame(x = rep(x = 1:3, each = 3), y = seq(from = 2, to = 18, by = 2))
write_csv(x = exemplo, path = "exemplo_exportacao.csv")
```

Exercício 3

1. Do objeto onde você armazenou a tabela de dados `revistas.xlsx`, retenha apenas as observacoes que correspondam às revistas brasileiras, e armazene o resultado dessa operação em um novo objeto.
2. Exporte este objeto, para o diretório e utilizando a extensão de sua escolha.

Exportando dados no formato de listas

- A vantagem de uma tabela de dados (`data.frame` ou `tibble`) é a facilidade com a qual podemos adicionar, remover e manipular seus elementos.
- Apesar disso, nem sempre é útil ou possível que armazenemos um conjunto de dados neste formato. Alguns exemplos disso são:
  - Quando precisamos trabalhar com múltiplas tabelas de dados similares, mas que representam entidades diferentes (e.g., amostragens realizadas em anos diferentes, dados de espécies diferentes,…);
  - Quando precisamos trabalhar com múltiplas tabelas de dados diferentes, mas todas dentro do mesmo contexto (e.g., *rasters* diferentes para a construção de um mesmo mapa);
  - Quando queremos manter juntos todos os dados e resultados utilizados em uma análise.

Exportando dados no formato de listas

- Nestas situações podemos armazenar todas estas fontes de dados em uma lista, utilizando a função `list`.

```
list(projetos = dados,
     exemplo_da_aula = exemplo,
     vetor = 1:100)
```

- As listas são úteis pois podem acomodar objetos de classe diferentes, objetos com dimensões distintas e, também, múltiplos `data.frame` onde não haja correspondência entre a ordem das colunas.

Exercício 4

1. Com os dois `data.frame` abaixo, tente juntá-los em uma único objeto utilizando a função `rbind.data.frame` e `list`.

```
set.seed(33)
tab1 <- data.frame(x = rnorm(n = 10, mean = 4, sd = 2),
                   y = runif(n = 10, min = 0, max = 10))
set.seed(83)
tab2 <- data.frame(z = 1:5, k = rpois(n = 5, lambda = 20))
```

Exportando dados no formato de listas

- Podemos exportar uma lista contendo múltiplos elementos através da função `write_rds` (do pacote `readr`) ou `write.rds` (da `base` do R).

```
write_rds(x = list(tab1, tab2), path = "lista salva.rds")
```

Exercício 5

1. Crie uma lista contendo os dados presentes em `dados dos projetos.csv`, `autores.xls`, `publicacoes.xls` e `revistas.xlsx`, nomeando cada elemento desta lista com o nome do arquivo onde está cada dado.
2. Salve esta lista em algum diretório no seu computador.
3. Importe o arquivo contendo esta lista novamente para o R, e atribua ela a um objeto.
4. Selecione o terceiro elemento da lista que você importou.

Resumindo

- Podemos utilizar as funções dos pacotes `readr` e `readxl` para importar um conjunto de dados para o R.
- Toda a função utilizada para importar um conjunto de dados começa com `read`, enquanto toda função utilizada para exportação dos mesmos começa com `write`.
- O ideal é que armazenemos as tabelas de dados primários (ou brutos) em um formato não-proprietário, como aquele com extensão `.csv`.
- Podemos (e devemos) abusar da flexibilidade das listas ao armazenar e exportar os resultados de processamentos e análises de dados.