Programação Estatística

Introdução ao R -Importação de dados

Rachid Muleia, PhD in Statistics

Universidade Eduardo Mondlane
Faculdade de Ciências
Departamento de Matemática e Informática

2023-03-16

1 Importação & Exportação de dados

Importação & Exportação de dados

Objectivos da sessão

No final desta sessão deverá:

- Conhecer o formato ideal de uma `data frame`.
- Saber importar uma `data frame` para o R.
- Saber inspeccionar uma `data.frame`.
- Identificar anomalias na `data frame`.
- Exportar objectos em R para outro ambiente.

Passos para importação e exportação de `data frames`

1. Importar seus dados.
2. Inspeccionar, limpar e preparar os dados.
3. Fazer as análises.
4. Exportar os seus resultados
5. Limpar o ambiente R e fechar a sessão.

Como devem se apresentar os seus dados?

- As colunas devem representar variáveis.
- As linhas devem representar observações.
- Use a primeira linha para os nomes das variáveis.
- Todas as células em branco devem ser preenchidas com **NA**
- Armazene os dados no formato `.csv` e `.txt`, pois podem ser facilmente importados para o R.

NOTA IMPORTANTE: Todos os valores da mesma variável DEVEM estar na mesma coluna!

Exemplo: Tratamento para redução do peso.

Tratamento	Controlo	Placebo
65	70	80
70	80	86
55	90	78
68	100	98

Formato ideal de uma data frame.

Resposta	Método
65	Tratamento
70	Tratamento
55	Tratamento
68	Tratamento
70	Controlo
80	Controlo
90	Controlo
100	Controlo
80	Placebo
86	Placebo
78	Placebo
98	Placebo

Importação dados

Importe dados usando as funções `read.table()` ou `read.csv`

```
> meus_dados <- read.csv(file = 'MeusDados,txt')
> meus_dados <- read.csv(file = 'MeusDados.csv')
>
> # criam uma data frame meus_dados
```

Importação dados

Importe dados usando as funções `read.table()` ou `read.csv`

```
> meus_dados <- read.csv(file = 'MeusDados.csv')
> Error in file(file, "rt") : cannot open the connection
> In addition: Warning message:
> In file(file, "rt") :
>   cannot open file 'MeusDados.csv': No such file or directory
```

NOTA:

- Especificar o directório de trabalho usando a função `setwd()`.
- `setwd()` permite que o R saiba em que pasta está a base de dados.

Importação de dados - Argumentos

Veja a documentação da função `?read.table` e `?read.csv`

```
read.table(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 = FALSE,
           fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)
```

Importação de dados - Argumentos importantes

Algumas dicas para reduzir possíveis erros na importação de dados.

- `header = TRUE` - informa ao R que a primeira linha da base de dados representa as variáveis.
- `sep = ','` informa ao R que os campos são separados por vírgulas.
- `strip.white = TRUE` remove o espaço em branco antes ou após caracteres que foram inseridos por engano durante a digitação de dados (EX: “pequeno” vs "pequeno ").
- `na.strings = ''` permite substituir células vazias por `NA`.

Importação de dados

Primeiro especifique o directório de trabalho

```
setwd('C:/Users/Rachid/Dropbox/Analise de Dados DMI/Slides Aulas')
```

Agora vamos importar a base de dados `snail_feeding.csv` usando a função `read.csv()`

```
> snail_data <- read.csv(file = "Snail_feeding.csv",
+ header = TRUE, strip.white = TRUE,
+ na.strings = " ")
```

Inspeccionando os dados

Após a importação dos dados, podemos usar as funções `str()`, `head()` e `tail()` para inspeccionar os dados.

```
> str(snail_data)
'data.frame':   769 obs. of  10 variables:
 $ Snail.ID: int  1 1 1 1 1 1 1 1 1 1 ...
 $ Sex     : chr  "male" "male" "males" "male" ...
 $ Size    : chr  "small" "small" "small" "small" ...
 $ Feeding : logi  FALSE FALSE FALSE FALSE FALSE TRUE ...
 $ Distance: num  0.17 0.87 0.22 0.13 0.36 0.84 0.69 0.6 0.85 0.59 ...
 $ Depth   : num  1.66 1.26 1.43 1.46 1.21 1.56 1.62 162 1.96 1.93 ...
 $ Temp    : int  21 21 18 19 21 21 20 20 19 19 ...
 $ X       : logi  NA NA NA NA NA NA ...
 $ X.1     : logi  NA NA NA NA NA NA ...
 $ X.2     : logi  NA NA NA NA NA NA ...
```

Inspecção de dados

- No slide anterior o execução do comando `str` mostra-nos que temos 10 variáveis, o que não é verdade.
- As colunas indesejadas podem ser excluídas.

```
> snail_data <- snail_data[ , 1:7]
> str(snail_data)
'data.frame':   769 obs. of  7 variables:
 $ Snail.ID: int  1 1 1 1 1 1 1 1 1 1 ...
 $ Sex     : chr  "male" "male" "males" "male" ...
 $ Size    : chr  "small" "small" "small" "small" ...
 $ Feeding : logi  FALSE FALSE FALSE FALSE FALSE TRUE ...
 $ Distance: num  0.17 0.87 0.22 0.13 0.36 0.84 0.69 0.6 0.85 0.59 ...
 $ Depth   : num  1.66 1.26 1.43 1.46 1.21 1.56 1.62 162 1.96 1.93 ...
 $ Temp    : int  21 21 18 19 21 21 20 20 19 19 ...
```

Inspecção de dados

A execução do comando `str` mostra-nos que a variável `Sex` tem 4 categorias. Esquisito !!!

```
> unique(snail_data$Sex)
[1] "male"   "males"  "Male"   "female"
```

Pode se também usar as seguintes funções para fazer uma breve inspecção nos dados:

- `summary()` : estatísticas sumárias para cada variável
- `head()` : retorna as 6 primeiras linhas da base de dados
- `tail()` : retorna as seis últimas linhas da base de dados
- `names()`: para aceder aos nomes das variáveis

Inspecção de dados

Para transformar “males” ou “Male” em “male correto, pode usar o o operador `[ ]` acompanhado da função `which()`:

```
snail_data$Sex[which(snail_data$Sex == "males")]<- "male"

snail_data$Sex[which(snail_data$Sex == "Male")]<- "male"
```

Inspecção de dados

Para transformar “males” ou “Male” em “male correto, pode usar o o operador `[ ]` acompanhado da função `which()`:

```
snail_data$Sex[which(snail_data$Sex =="males")]<- "male"

snail_data$Sex[which(snail_data$Sex == "Male")]<- "male"
```

ou

```
snail_data$Sex[which(snail_data$Sex == "males" |
snail_data$Sex == "Male")] <- "male"
```

Inspecção de dados

Funções que podem ser úteis para preparar seus dados: `sort()` e `order()`

Ordenar os dados em ordem decrescente em função da variável `Temp`.

```
snail_data[order(snail_data$Depth, snail_data$Temp,decreasing=TRUE), ]
    Snail.ID    Sex  Size Feeding Distance  Depth Temp
8          1   male small    TRUE  0.60000 162.00   20
762       16 female large   FALSE  0.92000   2.00   21
412        9 female small    TRUE  0.48000   2.00   19
37         1   male small   FALSE  0.67000   2.00   18
155        4   male small   FALSE  0.38000   2.00   18
434       10 female small   FALSE  0.49000   2.00   18
644       14 female large   FALSE  0.79000   1.99   21
692       15 female large   FALSE  0.87000   1.99   21
217        5   male large   FALSE  0.86000   1.99   20
568       12 female small   FALSE  0.48000   1.99   20
675       15 female large   FALSE  0.08000   1.99   20
397        9 female small   FALSE  0.48000   1.99   19
116        3   male small   FALSE  0.24000   1.99   18
320        7   male large   FALSE  0.50000   1.99   18
411        9 female small    TRUE  0.45000   1.99   18
299        7   male large    TRUE  0.46000   1.98   21
749       16 female large    TRUE  0.98000   1.98   21
290        7   male large   FALSE  0.44000   1.98   20
345        8   male large    TRUE  0.37000   1.98   20
752       16 female large   FALSE  0.75000   1.98   18
```

Inspecção de dados

Indentificação e remoção de duplicados usando a função `duplicated()`

```
duplicated(snail_data)
  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [13] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[145] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[157] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[169] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[181] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[193] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[205] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[217] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[229] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[241] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[253] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[265] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[277] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[289] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[301] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

Inspecção de dados

Para remover linhas duplicadas, pode usar o operador `[ ]` acompanhado a função `duplicated()`:

```
snail_data<- snail_data[!duplicated(snail_data), ]
```

OU

```
snail_data<- unique(snail_data)
```

Importação de dados armazenados na web

O R também pode ler dados armazenados usando uma `url`.

```
> data_web = read.table("http://www.sthda.com/upload/decathlon.txt",
+                       header=TRUE, sep='\t',strip.white = TRUE)
> head(data_web)
     name X100m Long.jump Shot.put High.jump X400m X110m.hurdle Discus
1  SEBRLE 11.04      7.58    14.83      2.07 49.81        14.69  43.75
2    CLAY 10.76      7.40    14.26      1.86 49.37        14.05  50.72
3  KARPOV 11.02      7.30    14.77      2.04 48.37        14.09  48.95
4 BERNARD 11.02      7.23    14.25      1.92 48.93        14.99  40.87
5  YURKOV 11.34      7.09    15.19      2.10 50.42        15.31  46.26
6 WARNERS 11.11      7.60    14.31      1.98 48.68        14.23  41.10
  Pole.vault Javeline X1500m Rank Points Competition
1       5.02    63.19  291.7    1   8217           1
2       4.92    60.15  301.5    2   8122           1
3       4.92    50.31  300.2    3   8099           1
4       5.32    62.77  280.1    4   8067           1
5       4.72    63.44  276.4    5   8036           1
6       4.92    51.77  278.1    6   8030           1
```

Importação de dados armazenados na web

- Ler dados a partir da web não é uma boa prática.
- O aconselhável é baixar os dados e ler localmente.
- O R possui uma função que permite baixar os dados, a função `download.file()`

```
download.file(url = "http://www.sthda.com/upload/decathlon.txt",destfile='decathlon')
```

- A função download pode baixar ficheiros com outras extensões.

Importação de dados no formato SPSS, SAS e STATA

- O R oferece recursos para ler ficheiros SPSS, SAS e STATA.
- Os recursos estão disponíveis na livraria `haven`. Como aceder??
- Precisamos instalar a livraria `haven` usando a função `install.packages()`.

Importação de dados no formato SPSS, SAS e STATA

- `read_sas()`: lê ficheiros `.sas7bdat` e `.sas7bcat` do SAS.
- `read_sav` ou `read_spss`: lê ficheiros `.sav` do SPSS.
- `read_dta`: lê ficheiros `.dta` do Stata.

```
> install.packages(haven)
> library(haven)
> dados_spss <- read_spss(file ='MZAR72FL.SAV')
```

Importação de dados no formato SPSS, SAS e STATA

Após ler os dados no formato spss, podemos aceder os nomes das variáveis.

```
> #install.packages(haven)
> library(haven)
> dados_spss <- read_spss(file ='MZAR72FL.SAV')
> attr(dados_spss$HIV03, 'label') # descrição da variável
[1] "Blood test result"
> attr(dados_spss$HIV03, 'labels') # labels dos valores assumido pela variável
                          HIV negative                           HIV  positive
                                     0                                      1
                         HIV2 positive                  HIV1 & HIV2 positive
                                     2                                      3
                   ERROR : V-, W+, M+                    ERROR : V-, W+, M-
                                     4                                      5
                   ERROR : V-, W-, M+                         Indeterminate
                                     6                                      7
Not enough samples to complete protocol                          Inconclusive
                                     8                                      9
```

Exportação de Dados

- O R permite exportar/gravar dados em vários formatos.
- Para exportação de dados usa-se a função `write.table()` ou `write.csv()`.

```
write.table(snail_data,  # Objecto que pretendemos exportar
            file='snail_inspecionado.txt', # nome do ficheiro
            sep=",", # separador de campos
           row.names=FALSE) # exlcuir os nomes das linhas.
```

- Para exportar dados no formato SPSS, SAS ou STATA usamos as funções, `write_sav()`, `write_sas()` e `write_dta()`, respectivamente.