Importar_Arquivo_de_Dados

April 15, 2020

- José O. Siqueira (siqueira@usp.br)
- Paulo S. P Silveira (paulo.silveira@fm.usp.br)
- Koichi Sameshima (koichi.sameshima@fm.usp.br)

1 IMPORTANDO ARQUIVOS DE DADOS

1.1 Exemplo 4.1 de Sardanelli & Di Leo (2008)

Example 4.1. Measuring myocardial delayed enhancement in cardiac MR imaging. Let us suppose we want to evaluate the difference in delayed enhancement of the myocardium provided by **two contrast agents (CAs)**. A sample of 50 post-ischemic patients undergo a cardiac MR with inversion recovery turbo- gradient-echo sequence ten minutes after the injection of 0.1 mmol/kg of CA 1. The signal intensity (SI), expressed in arbitrary units (a.u.), is measured in a region of interest placed in the infarcted myocardium. A second sample made up of another 50 post-ischemic patients is studied with the same technique but using 0.1 mmol/kg of CA 2. Data are summarized in Tables 4.2 and 4.3.

1.1.1 Resultados com CA1

Table 4.2. Signal intensity measurements after the administration of CA 1

Individual	SI (a.u.)	Individual	SI (a.u.)	Individual	SI (a.u.)
1	38.74	19	39.39	37	42.25
2	39.26	20	40.30	38	36.40
3	39.13	21	39.65	39	36.50
4	40.56	22	38.48	40	35.62
5	37.18	23	41.99	41	39.52
6	38.61	24	36.27	42	39.65
7	37.40	25	37.05	43	40.30
8	40.17	26	37.57	44	38.48
9	40.56	27	40.82	45	38.74
10	38.22	28	41.08	46	38.60
11	37.96	29	39.13	47	39.00
12	38.87	30	39.78	48	39.13
13	38.30	31	39.91	49	38.74
14	37.18	32	38.61	50	39.13
15	41.34	33	38.87		
16	41.86	34	38.09	m_1	39.0
17	39.26	35	39.13	s_1	1.5
18	38.87	36	39.26	SE_1	0.2

SI = signal intensity; CA = contrast agent; a.u. = arbitrary units.

1.1.2 Resultados com CA2

Table 4.3. Signal intensity measurements after the administration of CA 2

Individual	SI (a.u.)	Individual	SI (a.u.)	Individual	SI (a.u.)
1	50.36	19	51.21	37	54.93
2	51.04	20	52.39	38	47.32
3	50.87	21	51.55	39	47.45
4	52.73	22	50.02	40	46.31
5	48.33	23	54.59	41	51.38
6	50.19	24	47.15	42	51.55
7	48.62	25	48.17	43	52.39
8	52.22	26	48.84	44	50.02
9	52.73	27	53.07	45	50.36
10	49.69	28	53.40	46	50.18
11	49.35	29	50.87	47	50.70
12	50.53	30	51.71	48	50.87
13	49.79	31	51.88	49	50.36
14	48.33	32	50.19	50	50.87
15	53.74	33	50.53		
16	54.42	34	49.52	m_2	50.7
17	51.04	35	50.87	s_2	1.9
18	50.53	36	51.04	SE_2	0.3

SI = signal intensity; CA = contrast agent; a.u. = arbitrary units.

1.2 Arquivos e Diretórios/Pastas no R

Em ambiente R e no Jupyter Notebook, todos os arquivos serão lidos e salvos no diretório ativo, a menos que se especifique o contrário. Logo, a primeira coisa que você precisa se preocupar é como obter e configurar seu **diretório de trabalho**.

```
# Para obter o diretório de trabalho corrente, execute:
getwd ()

#Configure o diretório de trabalho:
setwd ("C:/Documents and Settings/Aula06")
```

1.2.1 Diretório de Trabalho

Os códigos ou comandos executados a seguir assumirão que os arquivos de dados estão localizados num diretório relativo ao **diretório de trabalho R do Jupyter Notebook**. O diretório atual pode ser verificado com a função getwd. Assim no meu caso:

[1]:
```
getwd()              # get current working directory
```

'/Users/koichi/Dropbox/0Aulas/mdr5728_2020/Aula06/R_Tidy_data'

Poderá selecionar diferentes diretórios de trabalho com a função `setwd()`, evitando-se assim a necessidade de especificar o caminho completo para arquivos de dados para leitura e gravação. Neste exemplo criou-se um diretório "Sardanelli" no caminho obtido por getwd(). Vamos verificar isso por meio da função `list.files()`:

[2]:
```
list.files()
```

1. 'anorexia_A.txt' 2. 'anorexia_B.txt' 3. 'anorexigenos_log.xls' 4. 'anorexigenos.xls' 5. 'cheat_sheet_data-import.pdf' 6. 'Data' 7. 'image' 8. 'Importar_Arquivo_de_Dados.ipynb' 9. 'Sardanelli' 10. 'Sardanelli_demo_exemplos_2008.ipynb' 11. 'tidy_data_slides.ipynb' 12. 'tidy_data.ipynb' 13. 'Untitled.ipynb'

Pronto, podemos verificar que há um arquivo ou diretório denominado **Sardanelli**.

[3]:
```
dir.exists("Sardanelli")  # Vamos testar se Sardanelli existe como um diretório ou não.
```

TRUE

[4]:
```
# Verifica-se neste caso que Sardanelli é um diretório. Assim teremos sucesso em␣
↪executar o seguinte comando
setwd("Sardanelli")   # set working directory para o subdiretório ./Sardanelli
```

[5]:
```
getwd()
```

'/Users/koichi/Dropbox/0Aulas/mdr5728_2020/Aula06/R_Tidy_data/Sardanelli'

Como pode ser verificado por meio de **`getwd()`**, o diretório corrente foi configurado para o subdiretório `./Aula06/Tidying_data/Sardanelli`. Note que a barra normal (/) deve ser usada no abiente R como separador de diretórios, mesmo na plataforma Windows, ao invés da barra invertida (). Por exemplo `> setwd("C:/Users/usuario/Documents")`

1.2.2 Listando os arquivos e as pastas no diretório

```
#lista arquivos no diretório de trabalho
list.files()

#lista arquivos in a specific folder
list.files (path = "C:/Folder/Subfolder1/Subfolder2")

#lista arquivos no diretório escolhido, choose.dir() funciona somente no Windows
list.files(path = choose.dir())

#lista de subdiretórios e arquivos dentro dos subdiretórios recursivamente
list.files(recursive = TRUE)

#obtém o nome completo (caminho e nome de arquivos) de cada arquivo
list.files(full.name = TRUE)
```

Se a lista obtida for muito longa, poderá filtrar os resultados usando o argumento padrão da função `list.files()`

```
#lista todos os arquivos que contêm a palavra "anorex"
list.files(pattern = "anorex")

#lista arquivos que termina com a palavra "csv"
list.files (pattern = "csv$")

#lista arquivos que começam com a palavra "anorex" seguida por quaisquer characteres e finalizado com "xlsx"
list.files(pattern = "^anorex(.*)xlsx$")
```

Expressões regulares

> Se ficou curioso o que são os símbolos "^", ".", "*",$" nos comandos acima, bem estes são chamados **metacaracteres em expressões regulares**. Se você quiser saber mais sobre expressões regulares, consultem estas duas fontes: Cheat Sheet de "Expressões Regulares" ou "Expressões Regulares" que um programador R deve saber.

[6]:
```
# Retornando ao diretório inicial
setwd("../")
# Verificando se o arquivo anorexigenos.xls
getwd()
list.files(pattern = "^anorex(.*)xls$")
```

'/Users/koichi/Dropbox/0Aulas/mdr5728_2020/Aula06/R_Tidy_data'

1. 'anorexigenos_log.xls' 2. 'anorexigenos.xls'

1.2.3 Checking if a file or folder exists

Como exercício vamos checar se o arquivo "anorexigenos.xls" existe no diretório de trabalho. A função para verificação é: > `file.exists("filename.extension")`

[7]:
```
file.exists("anorexigenos.xls")
```

TRUE

Podemos ver que o resultado é TRUE, de modo que o arquivo em questão existe!

Agora vamos inicialmente checar se a pasta "Data" existe diretório corrente, se não existir cria-se o diretório:

[8]:
```
ifelse(!dir.exists("Data"), dir.create("Data"), "Diretorio já existe!") # Execute esta␣
↪linha de comando mais uma vez.
```

'Diretorio já existe!'

1.3 Importando dados para Data.Frame

Tutorial genérico para importação dados em R

1.3.1 Arquivos textos

[9]:
```
si_ca1 = read.table("./Sardanelli/table_4_2.txt",header=TRUE)  # read text file
si_ca2 = read.table("./Sardanelli/table_4_3.txt",header=TRUE)  # read second text file
```

[10]:
```
head(si_ca1)
```

A data.frame: 6 × 2

	Individual <int>	SI <dbl>
1	1	38.74
2	2	39.26
3	3	39.13
4	4	40.56
5	5	37.18
6	6	38.61

[11]:
```
head(si_ca2)
```

A data.frame: 6 × 2

	Individual <int>	SI <dbl>
1	1	50.36
2	2	51.04
3	3	50.87
4	4	52.73
5	5	48.33
6	6	50.19

[12]:
```
tail(si_ca1)
mean(si_ca1$SI)
```

A data.frame: 6 × 2

	Individual <int>	SI <dbl>
45	45	38.74
46	46	38.60
47	47	39.00
48	48	39.13
49	49	38.74
50	50	39.13

39.0182

[13]:
```
tail(si_ca2)
mean(si_ca2$SI)
```

A data.frame: 6 × 2

	Individual <int>	SI <dbl>
45	45	50.36
46	46	50.18
47	47	50.70
48	48	50.87
49	49	50.36
50	50	50.87

50.724

1.3.2 Importando arquivo em formato CSV

CSV = comma separated values

[14]:
```
si_ca1_csv = read.csv("./Sardanelli/table_4_2.csv")  # read text file
si_ca2_csv = read.csv("./Sardanelli/table_4_3.csv")  # read second text file
head(si_ca1_csv)
head(si_ca2_csv)
```

A data.frame: 6 × 2

	Individual <int>	SI <dbl>
1	1	38.74
2	2	39.26
3	3	39.13
4	4	40.56
5	5	37.18
6	6	38.61

A data.frame: 6 × 2

	Individual <int>	SI <dbl>
1	1	50.36
2	2	51.04
3	3	50.87
4	4	52.73
5	5	48.33
6	6	50.19

Vamos calcular o sumário da Tabela 4.2:

[15]:
```
summary(si_ca1_csv)
```

```
   Individual          SI       
 Min.   : 1.00   Min.   :35.62  
 1st Qu.:13.25   1st Qu.:38.34  
 Median :25.50   Median :39.06  
 Mean   :25.50   Mean   :39.02  
 3rd Qu.:37.75   3rd Qu.:39.75  
 Max.   :50.00   Max.   :42.25  
```

1.4 Cálculo do Erro Padrão da Média (EPM)

Leia sobre EPM em An R Companion for the Handbook of Biological Statistics - SEM e neste site de John H. McDonald.

Recomendamos: Este site elaborado por Salvatore S. Mangiafico é uma excelente fonte para praticantes da análise estatística em R.

No R, o erro padrão da média (EPM) pode ser calculado com funções padrão no pacote de estatísticas nativas. Similar à função `summary( )`, existe uma função de **descrição no pacote psych**, `describe( )`, que inclui o cálculo do erro padrão da média junto com outras estatísticas descritivas. Esta função é útil para resumir várias variáveis em um quadro de dados. Nesse caso é interessante que os dados estejam no formato WIDE.

[16]:
```
# Instala o pacote/bliblioteca `psych` se ainda não está presente
if (!require("psych")) install.packages("psych",repo="https://vps.fmvz.usp.br/CRAN/
 ↪",dep=TRUE)
```

```
Loading required package: psych
```

[17]:
```
library(psych)
```

1.4.1 Cálculo de EPM manualmente

$$EPM = \frac{\text{desvio-padrão}}{\sqrt{N}}$$

[18]:
```
# Aplicação acima para cálculo do EPM para os contrastes CA1 e CA2.
se_ca1 <- sd(si_ca1$SI)/sqrt(length(si_ca1$SI[!is.na(si_ca1$SI)])) # se = standard␣
 ↪error ~ epm
se_ca2 <- sd(si_ca2$SI)/sqrt(length(si_ca2$SI[!is.na(si_ca2$SI)]))
```

```
cat('sd_ca1 = ', sd(si_ca1$SI), '; sd_ca2 = ', sd(si_ca2$SI), '; ')
cat('se_ca1 = ', se_ca1, '; se_ca2 = ', se_ca2)
```

```
sd_ca1 =  1.460874 ; sd_ca2 =  1.899586 ; se_ca1 =  0.2065988 ; se_ca2 =
0.2686421
```

[19]:
```
str(si_ca1)
```

```
'data.frame':   50 obs. of  2 variables:
 $ Individual: int  1 2 3 4 5 6 7 8 9 10 …
 $ SI        : num  38.7 39.3 39.1 40.6 37.2 …
```

[20]:
```
tail(si_ca1)
```

A data.frame: 6 × 2

	Individual <int>	SI <dbl>
45	45	38.74
46	46	38.60
47	47	39.00
48	48	39.13
49	49	38.74
50	50	39.13

[21]:
```
head(si_ca1)
```

A data.frame: 6 × 2

	Individual <int>	SI <dbl>
1	1	38.74
2	2	39.26
3	3	39.13
4	4	40.56
5	5	37.18
6	6	38.61

[22]:
```
help(read.table)
```

[23]:
```
filename <- "./Sardanelli/table_4_2.txt"
data1_csv <- read.csv(filename, sep = " ", quote = "\"",
         dec = ".", fill = TRUE)
```

[24]:
```
head(data1_csv)
```

A data.frame: 6 × 2

	Individual <int>	SI <dbl>
1	1	38.74
2	2	39.26
3	3	39.13
4	4	40.56
5	5	37.18
6	6	38.61

1.4.2 Acrescentando terceira coluna, a variável tipo de contrast

[25]:
```
si_ca1$contrast <- "CA1"
si_ca2$contrast <- "CA2"

head(si_ca1)
head(si_ca2)
```

A data.frame: 6 × 3

	Individual <int>	SI <dbl>	contrast <chr>
1	1	38.74	CA1
2	2	39.26	CA1
3	3	39.13	CA1
4	4	40.56	CA1
5	5	37.18	CA1
6	6	38.61	CA1

A data.frame: 6 × 3

	Individual <int>	SI <dbl>	contrast <chr>
1	1	50.36	CA2
2	2	51.04	CA2
3	3	50.87	CA2
4	4	52.73	CA2
5	5	48.33	CA2
6	6	50.19	CA2

1.5 Juntando dois dataframes ou arquivos

Dois conjunto de dataframes com nomes de colunas comuns podem ser juntados

[26]:
```
si_total <- rbind(si_ca1, si_ca2)
head(si_total)
tail(si_total)
```

A data.frame: 6 × 3

	Individual <int>	SI <dbl>	contrast <chr>
1	1	38.74	CA1
2	2	39.26	CA1
3	3	39.13	CA1
4	4	40.56	CA1
5	5	37.18	CA1
6	6	38.61	CA1

A data.frame: 6 × 3

	Individual <int>	SI <dbl>	contrast <chr>
95	45	50.36	CA2
96	46	50.18	CA2
97	47	50.70	CA2
98	48	50.87	CA2
99	49	50.36	CA2
100	50	50.87	CA2

1.5.1 LONG => WIDE

[27]:
```
library(tidyr)
si_total_wide <- spread(si_total, contrast, SI)
```

[28]:
```
head(si_total_wide)
```

A data.frame: 6 × 3

	Individual <int>	CA1 <dbl>	CA2 <dbl>
1	1	38.74	50.36
2	2	39.26	51.04
3	3	39.13	50.87
4	4	40.56	52.73
5	5	37.18	48.33
6	6	38.61	50.19

1.6 Gerando sumários dos dados

[29]:
```
library(psych)
```

[30]:
```
summary(si_total_wide)
```

```
   Individual         CA1             CA2       
 Min.   : 1.00   Min.   :35.62   Min.   :46.31  
 1st Qu.:13.25   1st Qu.:38.34   1st Qu.:49.85  
 Median :25.50   Median :39.06   Median :50.78  
 Mean   :25.50   Mean   :39.02   Mean   :50.72  
 3rd Qu.:37.75   3rd Qu.:39.75   3rd Qu.:51.67  
 Max.   :50.00   Max.   :42.25   Max.   :54.93  
```

[31]:
```
# A função describe do pacote psych requer dados no formato WIDE
describe(si_total_wide)
```

A psych: 3 × 13

	vars <int>	n <dbl>	mean <dbl>	sd <dbl>	median <dbl>	trimmed <dbl>	mad <dbl>	min <dbl>	max <dbl>
Individual	1	50	25.5000	14.577380	25.500	25.50000	18.532500	1.00	50.00
CA1	2	50	39.0182	1.460874	39.065	39.01375	1.097124	35.62	42.25
CA2	3	50	50.7240	1.899586	50.785	50.71800	1.423296	46.31	54.93

Note que os valores de EPM, **se** em inglês, estão na última coluna à direita. `describe()` emite um relatório mais completo que a função `summary()`.

1.7 Visualizar os dados por box plots

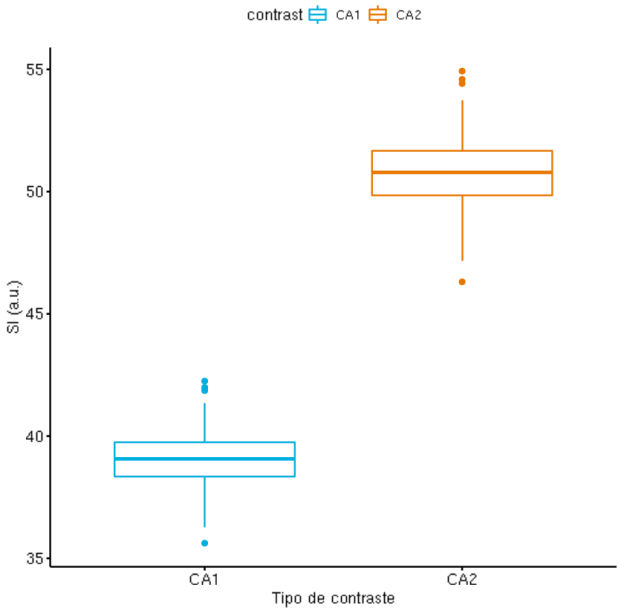
Fonte: Statistical tools for high-throughput data analysis

[39]:
```
if (!require("ggpubr")) install.packages("ggpubr",repo="https://vps.fmvz.usp.br/CRAN/
↪",dep=TRUE)

library(ggpubr)

# Note que aqui é necessário usar o formato LONG de dados
```

```
ggboxplot(si_total, x = "contrast", y = "SI",
          color = "contrast", palette = c("#00AFDD", "#E77700"),
          order = c("CA1", "CA2"),
          ylab = "SI (a.u.)", xlab = "Tipo de contraste")
```



Pela visualização do **box plot** fica evidente que o sinal gerado pelo contraste CA2 é bem maior que o do contraste CA1.

Com isso poderíamos ter feito um teste monocaudal de tal sorte que a hipótese alternativa poderia ser $H_1 : si_{CA2} > si_{CA2}$

Mas insistiremos no teste bicaudal:

1.8 Teste t não pareado bicaudal

Neste caso o teste estatístico é

$H_0 : \mu_{CA1} = \mu_{CA2}$;
$H_1 : \mu_{CA1} \neq \mu_{CA2}$.
$\alpha = 5$

[33]:
```
?t.test # Para visualizar o help de t.test
```

[34]:
```
t.test(si_total_wide$CA2,
       si_total_wide$CA1,
       paired=FALSE,
       alternative="two.sided",  # bicaudal
       mu = 0,                   #
       conf.level=0.95)
```

```
Welch Two Sample t-test
```

```
data:  si_total_wide$CA2 and si_total_wide$CA1
t = 34.541, df = 91.94, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 11.03271 12.37889
sample estimates:
mean of x mean of y
  50.7240   39.0182
```

1.9 Reordenando a posição das colunas

Parece ser lógico ordenar as variáveis de coluna hierarquicamente, por exemplo, nessa ordem: **contrast** => **Individual** => **SI**. Há diversas maneiras de se obter este efeito em R. Vejamos quatro métodos:

[ ]:
```r
### Reordenação método 1
si_total[, c(3, 1, 2)]  # A primeira vírgula sinaliza que todas as linhas devem ser␣
 ↪mantidas,
                        # e 3, 1, 2 refere-se a ordem das colunas.
```

[ ]:
```r
### Reordenação método 2
si_total[, c("contrast", "Individual", "SI")] # Vide observação acima.
```

[ ]:
```r
### Reordenação método 3
subset(si_total, select = c(3, 1, 2))
```

[ ]:
```r
### Reordenação ou pipeline por meio do dplyr método 4
if (!require("dplyr")) install.packages("dplyr", repo = "https://vps.fmvz.usp.br/CRAN/",
    dep = TRUE)
library(dplyr)

si_total %>% select(contrast, Individual, SI) # %>% indica operação de pipeline do dplyr
```