Programação Estatística

Introdução ao R - Manipulação de dados
Tratamento de dados

Rachid Muleia, PhD in Statistics

Universidade Eduardo Mondlane
Faculdade de Ciências
Departamento de Matemática e Informática

2023-03-23

Análise exploratória de dados

Objectivos

No final desta aula, deverá saber:

- Identificar os diferentes tipos de variáveis.
- Sumarizar os dados usando estatísticas descritivas.
- Verificar a presença de valores omissos

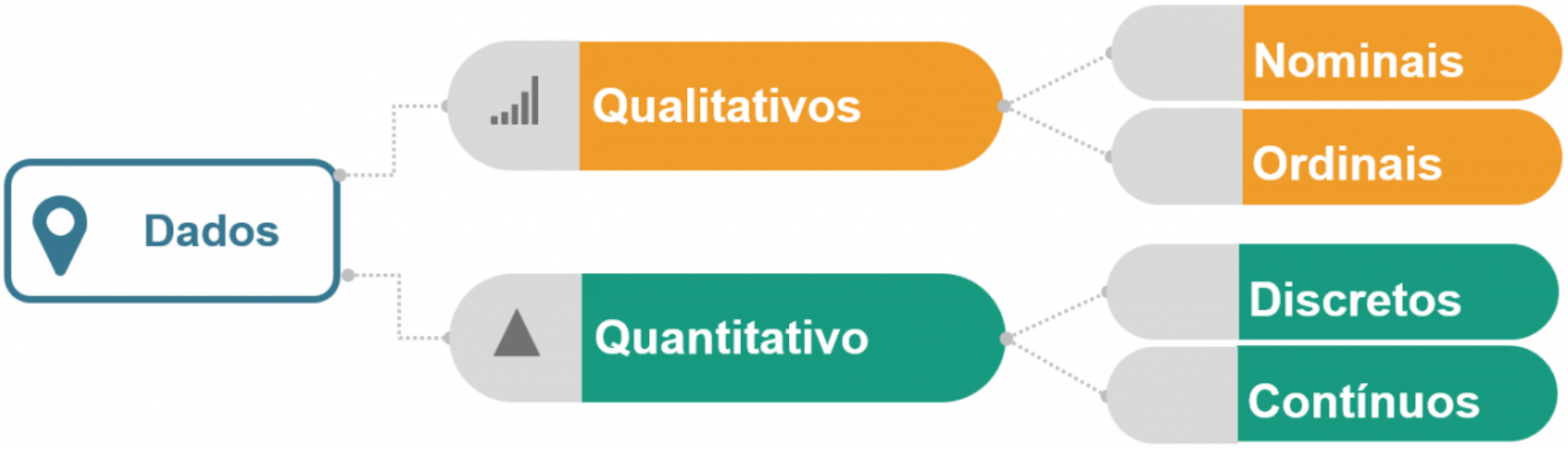
Análise exploratória de dados (AED)

Na análise estatística, a AED é um dos primeiros passos antes de avançar com análises mais complexas.

Porquê?

- Identificar anomalias nos dados.
- Familiarizarmo-nos mais com os dados.
- Descobrir padrões e tendências.
- Formular hipóteses.

Tipo de dados/variáveis



- Quantitativo: altura, peso, temperatura, idade,...
- Qualitativo: cor dos olhos, nome, classificação numa competição,...

Medidas sumárias/ descritivas

Para variáveis quantitativas/númericas, pode-se resumir os dados usando as seguintes funções:

- Medidas de tendência central- Média, Mediana e Moda.
- Medidas de Posição: Quartis, Decis e Percentis,. . .
- Medidas de Dispersão: Variância, Desvio-Padrão, Coeficiente de variação:

Medidas sumárias/ descritivas

O R possui várias funcionalidades para sumarizar os dados.

- `summary()`: retorna várias medidas descritivas.
- `mean()` e `median()`: retorna a média e a mediana, respectivamente.
- `quantile()`: retorna quartis, decis ou percentis.
- `var()` e `sd()`: retorna a variância e o desvio-padrão, respectivamente.

Medidas sumárias/ descritivas

Vamos usar a base de dados `iris` para ilustrar o cálculo de medidas sumárias no R.

```
data('iris') # chama uma base interna do R.
str(iris) # mostra a estrutura da base de dados.
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
 Median :5.800   Median :3.000   Median :4.350   Median :1.300
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
       Species
 setosa    :50
 versicolor:50
 virginica :50
```

Medidas sumárias/ descritivas

Para além de usarmos as função `summary()`, podemos usar funções específicas, `mean()`, `median()`, `sd()`, ...

```
mean(iris$Sepal.Length)
[1] 5.843333
median(iris$Sepal.Length)
[1] 5.8
sd(iris$Sepal.Length)
[1] 0.8280661
# coeficiente de  variação
CV=sd(iris$Sepal.Length)/mean(iris$Sepal.Length)
CV
[1] 0.1417113
```

Medidas sumárias/ descritivas

Podemos usar igualmente a função `summarize()` da livraria `dplyr`

```
library(dplyr)
summarize(iris, media=mean(Sepal.Length),
          mediana=median(Sepal.Length),
          desvi_padrao=sd(Sepal.Length),
          cv=sd(Sepal.Length)/mean(Sepal.Length))
     media mediana desvi_padrao        cv
1 5.843333     5.8    0.8280661 0.1417113
# alternativamente
iris %>% summarize( media=mean(Sepal.Length),
          mediana=median(Sepal.Length),
          desvi_padrao=sd(Sepal.Length),
          cv=sd(Sepal.Length)/mean(Sepal.Length))
     media mediana desvi_padrao        cv
1 5.843333     5.8    0.8280661 0.1417113
```

Medidas sumárias/ descritivas

A base de dados tem flores por espécies. Por exemplo, podemos sumarizar os dados por espécie.

```
iris %>% group_by(Species) %>% summarize( media=mean(Sepal.Length),
          mediana=median(Sepal.Length),
          desvi_padrao=sd(Sepal.Length),
          cv=sd(Sepal.Length)/mean(Sepal.Length))
# A tibble: 3 x 5
  Species    media mediana desvi_padrao     cv
  <fct>      <dbl>   <dbl>        <dbl>  <dbl>
1 setosa      5.01     5          0.352 0.0704
2 versicolor  5.94     5.9        0.516 0.0870
3 virginica   6.59     6.5        0.636 0.0965
```

Dados omissos/missing cases

A manipulação dos dados omissos será feita sobre a base de dados `iris` com dados omissos introduzidos propositadamente.

```
iris_miss <- read.csv('iris_missing.csv',header = TRUE)
```

- Os dados omissos, no R, são denotados por `NA`.
- Pode-se usar a função `is.na()` para verificar a presença de dados omissos.
- `na.omit()` retorna um objecto/base de dados sem dados omissos.
- `complete.cases()` retorna um vector lógico indicando quais casos estão completos.

Dados omissos/missing cases

- Vamos examinar quantos dados omissos temos na base de dados.
- A contagem pode se fazer usando a função `colSums()`.

```
colSums(is.na(iris_miss))
           X Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
           0           15           15           15           15           15
```

- Podemos remover os dados omissos, apagando todas as linhas com `NA`.

```
iris_miss <- read.csv('iris_missing.csv',header = TRUE)
iris_no_miss <- na.omit(iris_miss)
colSums(is.na(iris_no_miss))
           X Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
           0            0            0            0            0            0
```

Dados omissos/missing cases

Por vezes há interesse de ver quais linhas/observações apresentam valores omissos.

```
iris_miss[!complete.cases(iris_miss),]
      X Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1     1           NA         3.5          1.4         0.2     setosa
2     2          4.9         3.0          1.4         0.2       <NA>
6     6           NA         3.9          1.7         0.4     setosa
7     7          4.6         3.4           NA          NA     setosa
11   11          5.4         3.7           NA         0.2     setosa
13   13          4.8          NA          1.4         0.1     setosa
17   17          5.4         3.9          1.3         0.4       <NA>
20   20           NA         3.8           NA         0.3     setosa
22   22           NA         3.7          1.5         0.4     setosa
25   25           NA         3.4          1.9         0.2     setosa
28   28           NA         3.5          1.5         0.2     setosa
29   29          5.2          NA          1.4         0.2     setosa
32   32          5.4          NA          1.5          NA     setosa
37   37          5.5          NA           NA         0.2     setosa
38   38           NA         3.6          1.4         0.1     setosa
40   40          5.1         3.4          1.5          NA     setosa
42   42          4.5         2.3           NA         0.3     setosa
44   44          5.0         3.5          1.6         0.6       <NA>
59   59           NA         2.9           NA         1.3 versicolor
60   60          5.2         2.7           NA         1.4 versicolor
61   61          5.0         2.0           NA         1.0 versicolor
62   62           NA         3.0          4.2         1.5 versicolor
65   65          5.6         2.9          3.6         1.3       <NA>
70   70          5.6         2.5          3.9          NA versicolor
74   74          6.1          NA          4.7         1.2       <NA>
```

Dados omissos/missing cases

A instrução `!complete.case()` nos da todas as linhas com valores omissos para todas as variáveis da base de dados.

Como fazer, se o interesse for verificar a presença de valores omissos em uma única variável? Acho que a função `is.na()` pode ajudar.

```
is.na(iris_miss$Sepal.Length) # retorna um vector de TRUE ou FALSE  indicando se um dado valor é ou não NA
  [1]  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
 [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE
 [25]  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [37] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
 [61] FALSE  TRUE 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  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE
 [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
[109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[121]  TRUE  TRUE 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  TRUE FALSE FALSE FALSE
```

Dados omissos/missing cases

Para saber a posição vamos usar a função `which()`.

```
pos <- which(is.na(iris_miss$Sepal.Length))
iris_miss[pos, ]
```

	X	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	1	NA	3.5	1.4	0.2	setosa
6	6	NA	3.9	1.7	0.4	setosa
20	20	NA	3.8	NA	0.3	setosa
22	22	NA	3.7	1.5	0.4	setosa
25	25	NA	3.4	1.9	0.2	setosa
28	28	NA	3.5	1.5	0.2	setosa
38	38	NA	3.6	1.4	0.1	setosa
59	59	NA	2.9	NA	1.3	versicolor
62	62	NA	3.0	4.2	1.5	versicolor
90	90	NA	NA	4.0	1.3	versicolor
95	95	NA	2.7	4.2	1.3	<NA>
105	105	NA	3.0	5.8	2.2	virginica
121	121	NA	3.2	5.7	NA	virginica
122	122	NA	2.8	4.9	2.0	virginica
147	147	NA	NA	5.0	1.9	virginica

Os valores omissos poder ser representados por diversos caracteres: 9999, *, ? , " “, . . .

Como informar o R que estes caracteres representam valores omissos?

- Durante a leitura dos dados.

```
iris_spec <- read.csv('iris_missing_spec_car.csv',header=TRUE, sep=',' , na.strings = "9999")
colSums(is.na(iris_spec))
Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
          15           15           15           15           15
```

- Após a leitura,

```
iris_spec1 <- read.csv('iris_missing_spec_car.csv',header=TRUE, sep=',')
iris_spec1[iris_spec1 == 9999] <- NA
colSums(is.na(iris_spec1))
Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
          15           15           15           15           15
```

Dados omissos/missing cases vs estatísticas descritivas

```
summary(iris_spec)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.10
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.30
 Median :5.800   Median :3.000   Median :4.400   Median :1.30
 Mean   :5.867   Mean   :3.072   Mean   :3.764   Mean   :1.16
 3rd Qu.:6.400   3rd Qu.:3.350   3rd Qu.:5.100   3rd Qu.:1.80
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.50
 NA's   :15      NA's   :15      NA's   :15      NA's   :15
   Species
 Length:150
 Class :character
 Mode  :character



mean(iris_spec$Sepal.Length,na.rm = TRUE)
[1] 5.866667
```

Tabelas de frequência

Para variáveis de natureza qualitativa (não-numéricas) não se pode calcular as estatísticas sumárias. Mas pode-se sumarizar através de contagens ou percentagens.

- Para construir uma tabela de frequência vamos usar a função `table()`.

```
table(iris$Species)

    setosa versicolor  virginica
        50         50         50
```

Para ter dados em percentagem vamos a usar a função `prop.table()`

```
prop.table(table(iris$Species))

    setosa versicolor  virginica
 0.3333333  0.3333333  0.3333333
```

Tabelas de frequência

A livraria dplyr também permite fazer tabelas de frequência e ter os resultados num formato 'agradável'.

```
library(dplyr) # não se esqueça de chamar esta livraria
iris%>% group_by(Species)%>%
  summarize (N = n())%>% mutate(Perc = N/sum(N))
# A tibble: 3 x 3
  Species         N  Perc
  <fct>       <int> <dbl>
1 setosa         50 0.333
2 versicolor     50 0.333
3 virginica      50 0.333
```

Análise bivariada

- No R, podemos também construir tabelas cruzadas
- Verificar relações entre duas variáveis , qualitativas/ quantitativas
- Para construir tabelas cruzadas temos várias opções:
  - `table()`.
  - `crosstab()` e `CrossTable()` da livraria `descr`.
  - `CrossTable()` da livraria `gmodels`.

Análise bivariada- Tabelas cruzadas

```
library(haven)
dados_pesca <- read_spss('BaseDados.sav')
# tipo de pesca vs metodo de conservação
table(dados_pesca$PTP, dados_pesca$MC)
```

```
     1   2   3   4   5   6
  1 53  11  41 384 120   6
  2 28  12  70  21 119  12
  3 37  11  24   6  19  16
  4  7   7   1   9   5   3
  5 76   6  39  66  30  46
  9 145  3  26  50  20  31
```

Análise bivariada- Tabelas cruzadas

```
library(descr)
# tipo de pesca vs metodo de conservação
crosstab(dados_pesca$PTP,dados_pesca$MC, plot=FALSE)
   Cell Contents
|-------------------------|
|                   Count |
|-------------------------|

=============================================================================
                                Metodo para conservar e/ou processar pescado
prncpl tp d psc q a s fml prt       1      2      3      4      5      6   Total
-----------------------------------------------------------------------------
1                                  53     11     41    384    120      6     615
-----------------------------------------------------------------------------
2                                  28     12     70     21    119     12     262
-----------------------------------------------------------------------------
3                                  37     11     24      6     19     16     113
-----------------------------------------------------------------------------
4                                   7      7      1      9      5      3      32
-----------------------------------------------------------------------------
5                                  76      6     39     66     30     46     263
-----------------------------------------------------------------------------
9                                 145      3     26     50     20     31     275
-----------------------------------------------------------------------------
Total                             346     50    201    536    313    114    1560
==============================================================================
```

Análise bivariada- Tabelas cruzadas

Como verificar associação entre duas várias

- `chisq.test()`.
- `crosstab` e especificar o argumento `chisq=TRUE`.

```
library(descr)
# tipo de pesca vs metodo de conservação
chisq.test(dados_pesca$PTP,dados_pesca$MC)
Warning in chisq.test(dados_pesca$PTP, dados_pesca$MC): Chi-squared
approximation may be incorrect

    Pearson's Chi-squared test

data:  dados_pesca$PTP and dados_pesca$MC
X-squared = 776.21, df = 25, p-value < 2.2e-16
```

Análise bivariada- Variáveis quantitativas

Para variáveis quantitativas a análise é feita por via:

- Diagrama de dispersão.
- Análise das correlações, calculado usando a função `cor()`.
- Análise de regressão, feita a usando a função `lm()`.

Análise bivariada- Variáveis quantitativas

calculo do coeficiente de correlação de pearson

```
dados_carros <- mtcars[, c(1,3,4,5,6,7)]
# calcular o coeficiente de correlação de pearson para duas
cor(dados_carros$mpg,dados_carros$disp)
[1] -0.8475514
# matriz das correlações
cor(dados_carros)
            mpg       disp         hp        drat         wt        qsec
mpg   1.0000000 -0.8475514 -0.7761684  0.68117191 -0.8676594  0.41868403
disp -0.8475514  1.0000000  0.7909486 -0.71021393  0.8879799 -0.43369788
hp   -0.7761684  0.7909486  1.0000000 -0.44875912  0.6587479 -0.70822339
drat  0.6811719 -0.7102139 -0.4487591  1.00000000 -0.7124406  0.09120476
wt   -0.8676594  0.8879799  0.6587479 -0.71244065  1.0000000 -0.17471588
qsec  0.4186840 -0.4336979 -0.7082234  0.09120476 -0.1747159  1.00000000
```

Análise bivariada- Variáveis quantitativas

Regressão linear

```
# regressão linear simples
lm(mpg~disp,data = dados_carros)

Call:
lm(formula = mpg ~ disp, data = dados_carros)

Coefficients:
(Intercept)         disp
   29.59985     -0.04122
#ou
reg<- lm(mpg~disp,data = dados_carros)
summary(reg)

Call:
lm(formula = mpg ~ disp, data = dados_carros)

Residuals:
    Min      1Q  Median      3Q     Max
-4.8922 -2.2022 -0.9631  1.6272  7.2305

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 29.599855   1.229720  24.070  < 2e-16 ***
disp        -0.041215   0.004712  -8.747 9.38e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.251 on 30 degrees of freedom
Multiple R-squared:  0.7183,    Adjusted R-squared:  0.709
F-statistic: 76.51 on 1 and 30 DF,  p-value: 9.38e-10
```

Funções/procedimentos

Funções

- Um dos aspectos bastante atractivo do R é a facilidade de poder escrever funções próprias;
- Funções podem receber um ou mais argumentos e devolvem um ou "mais valores".

```
Fun1 = function(x){
  return(x*x)
}
```

A função `Fun1` recebe um valor `x` e devolve o quadrado desse valor.

Os elementos chaves na definição de uma função são:

- A palavra chave `function`;
- Os parênteses `()` para acomodar os argumentos da função;
- Sequência de operações dentro de chavetas `{}`;
- o comando `return`;

```
Fun1(5) ; Fun1(10)
```

```
## [1] 25
```

```
## [1] 100
```

```
z=c(1,2,3,4,5); Fun1(z)
```

```
## [1]  1  4  9 16 25
```

A função a seguir recebe um vector, e devolve a média e o desvio padrão

```
media_var = function(values){
  return(c(mean(values),sd(values)))
}

w = rep(c(1,3,4), each =5)
media_var(w)
```

```
## [1] 2.666667 1.290994
```

Escopo da variável

Quando se invoca uma função, o R primeiro procura por variáveis locais, e depois por variáveis globais. Por exemplo, a função `Fun4` usa uma variável global.

```
blob = 5
Fun4 = function(){
    return(blob)
 }
Fun4()
```

```
## [1] 5
```

Geralmente, as variáveis/objectos devem ser definidas localmente.

Exercício

Escreva uma função para calcular as raízes de uma equação quadrática.