Introdução ao R para Análise de Dados

Minicurso

Paulo Alexandrino

Março 2023

Sumário

1 Introdução **2**

2 Conceitos Fundamentais **3**

2.1 Operadores 3

2.1.1 Operadores matemáticos 3

2.1.2 Operadores lógicos 4

2.2 Variáveis e Tipos de Dados 6

2.3 Funções 6

2.3.1 Aplicando funções em vetores numéricos 7

2.4 Baixando e excutando pacotes externos 8

3 Importação **9**

3.1 Arquivos separados por vírgula (.csv) 9

3.2 Arquivos Excel (.xls e .xlsx) 10

4 Visualização **12**

4.1 Gráfico de linha 12

4.2 Gráfico de dispersão 13

5 Organização e Transformação **15**

6 Modelagem **16**

7 Próximos Passos **17**

Capítulo 1

Introdução

Capítulo 2

Conceitos Fundamentais

2.1 Operadores

2.1.1 Operadores matemáticos

A forma mais elementar de utilizar o R é como uma calculadora que nos permite realizar todo tipo de operação matemática desejada.

```
# Soma (+)
5 + 7
```

```
## [1] 12
```

```
# Subtração (-)
10 - 16
```

```
## [1] -6
```

```
# Multiplicação (*)
2 * 7
```

```
## [1] 14
```

```
# Exponenciação (^)
2^4
```

```
## [1] 16
```

```r
# Divisão (/)
5 / 2
```

```
## [1] 2.5
```

```r
# Divisão Inteira (%/%)
5 %/% 2
```

```
## [1] 2
```

```r
# Módulo - Resto da divisão inteira (%%)
5 %% 2
```

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

2.1.2 Operadores lógicos

Operadores lógicos nos permitem estabelecer relações entre objetos e julgar proposições (orações declarativas) como verdadeiras (`TRUE`) ou falsas (`FALSE`). Serão muito úteis a construção de filtros.

```r
# Menor que (<)
5 < 3
```

```
## [1] FALSE
```

```r
# Menor ou igual que (<=)
2.5 <= 7
```

```
## [1] TRUE
```

```r
# Maior que (>)
10 > 15
```

```
## [1] FALSE
```

```r
# Maior ou igual que (>=)
11 >= 11
```

```
## [1] TRUE
```

```r
# Igual a (==)
2 == 3
```

```
## [1] FALSE
```

```r
# Diferente de (!=)
5 != 4
```

```
## [1] TRUE
```

A negação de uma proposição é seu inverso lógico.

```r
# Negação (!)
2 > 5
```

```
## [1] FALSE
```

```r
!(2 > 5)
```

```
## [1] TRUE
```

Os conectivos de conjunção ($\wedge$, lê-se `E`) e disjunção ($\vee$, lê-se `OU`) são operadores lógicos que permitem criar proposições compostas. Sejam p e q proposições quaisquer:

- A conjunção $p \wedge q$ é verdadeira se p e q são ambas verdadeiras; se ao menos uma delas for falsa, $p \wedge q$ é falsa.

p	q	p $\wedge$ q
V	V	V
V	F	F
F	V	F
F	F	F

```r
# E (&)
(2 > 1) & (3 != 0)
```

```
## [1] TRUE
```

```r
(3 < 2) & (5 < 11)
```

```
## [1] FALSE
```

- A disjunção $p \vee q$ é verdadeira se p e q se ao menos uma das proposições p ou q é verdadeira; se p e q são ambas falsas, então $p \vee q$ é falsa.

p	q	p V q
V	V	V
V	F	V
F	V	V
F	F	F

```r
# OU (|)
(3 < 2) | (5 < 11)
```

```
## [1] TRUE
```

```r
(5 == 6) | (12 < 8)
```

```
## [1] FALSE
```

2.2 Variáveis e Tipos de Dados

2.3 Funções

Funções são ferramentas, porções de código reutilizáveis. Elas nos permitem realizar os mais variados tipos de operações, seja criar ou modificar objetos, realizar comandos no computador, e muito mais.

Funções quase sempre recebem argumentos, de forma a ajustar seu comportamento de acordo com algum objetivo específico. Alguns argumentos são obrigatórios, outros opicionais, ou seja, possuem valores pré-definidos.

Caso queira obter mais informações sobre uma função, você pode acessar sua documentação. A documentação contém tudo, ou a maior parte, do que se precisa saber sore uma função: quais seus argumentos, como eles se comportam, quais os valores pré-definidos. Para acessar a documentação basta digitar um ponto de interrogação seguido do nome da função.

2.3.1 Aplicando funções em vetores numéricos

```
# Criar um vetor, ou concatenar vetores: c()
alturas <- c(1.54, 1.92, 1.67, 1.72, 1.78)
alturas
```

```
## [1] 1.54 1.92 1.67 1.72 1.78
```

```
# Mínimo: min()
min(alturas)
```

```
## [1] 1.54
```

```
# Máximo: max()
max(alturas)
```

```
## [1] 1.92
```

```
# Somatório: sum()
sum(alturas)
```

```
## [1] 8.63
```

```
# Tamanho de um vetor: length()
length(alturas)
```

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

```
# Ordenar um vetor em ordem crescente: sort()
sort(alturas)
```

```
## [1] 1.54 1.67 1.72 1.78 1.92
```

```
# A função sort possui o argumento opcional decreasing = FALSE.
# Para ordenar a lista de forma decrescente, podemos alterá-lo.
sort(alturas, decreasing = TRUE)
```

```
## [1] 1.92 1.78 1.72 1.67 1.54
```

```r
# Criar uma sequencia de números: seq()
seq(from = 1, to = 50, by = 3)
```

```
##  [1]  1  4  7 10 13 16 19 22 25 28 31 34 37 40 43 46 49
```

```r
# Podemos substituir, por exemplo seq(from = x, to = y, by = 1)
3:15
```

```
##  [1]  3  4  5  6  7  8  9 10 11 12 13 14 15
```

2.4 Baixando e excutando pacotes externos

Até agora utilizamos apenas funções pré-instaladas no R. Para ter acesso a novos pacotes, será necessário instalá-los primeiro. Esse procedimento só é necessário ser feito uma vez, a não ser que o R precise ser reinstalado. Sempre que iniciar uma sessão será necessário carregar os pacotes que serão utilizados.

```r
# Instalando o tidyverse
install.packages("tidyverse")

# Carregando o tidyverse
library("tidyverse")
```

O `tidyverse` é um conjunto de pacotes que juntos fornecem uma ampla gama de ferramentas para análise de dados. Algumas dessas ferramentas apresentam versões otimizadas de funções já existentes no R.

O operador pipe (`%>%`) permite substituir o aninhamento de funções por uma cadeia de operação, o que torna os códigos mais legíveis. Veremos aplicações do pipe e de algumas dos principais pacotes e funções do `tidyverse` nos capítulos posteriores.

Capítulo 3

Importação

O R nos permite importar e utilizar em nossas análises uma série de diferentes tipos de arquivos. Isso pode ser feito por meio de funções nativas ou através de ferramentas disponíveis em pacotes criados por outros usuários. Por exemplo, o `tidyverse` possui os pacotes `readr`, que permite a leitura de arquivos como .csv, .fwf e .txt, e o pacote `readxl`, para leitura de arquivos Excel.

3.1 Arquivos separados por vírgula (.csv)

Comma-separated values é um formato de arquivo de texto onde as observações de uma tabela são representadas por linhas e as colunas são separadas por vírgula (no padrão americano) ou ponto-e-vírgula (no padrão europeu e brasileiro). No padrão separado por vírgula, o indicador decimal é o ponto; no padrãos separado por ponto-e-vírgula, a vírcula. É importante estar atento ao tipo de arquivo que estamos lidando.

O dataset `gapminder` contém dados econômicos e demográficos para diversos países a cada 5 anos de 1952 a 2007. O arquivo que usaremos como exemplo apresenta uma amostra contendo as informações disponíveis para o Brasil, Botsuana e Coréia do Sul. Ele foi salvo em .csv com padrão separado por ponto-e-vírgula.

```
# Primeira forma: ajustando os parâmetros da função read.csv
gapminderCountries <- read.csv("Aula1/gapminderCountries.csv",
                               sep = ";",
                               dec = ",")

# Segunda forma: utilizando a função read.csv2
gapminderCountries <- read.csv2("Aula1/gapminderCountries.csv")
```

```r
# Mostrando um pedaço do dataset
head(gapminderCountries)
```

```
##    country continent year lifeExp    pop gdpPercap
## 1 Botswana    Africa 1952  47.622 442308  851.2411
## 2 Botswana    Africa 1957  49.618 474639  918.2325
## 3 Botswana    Africa 1962  51.520 512764  983.6540
## 4 Botswana    Africa 1967  53.298 553541 1214.7093
## 5 Botswana    Africa 1972  56.024 619351 2263.6111
## 6 Botswana    Africa 1977  59.319 781472 3214.8578
```

Para maiores informações, leia a documentação das funções `read.csv` e `read.csv2`.

3.2 Arquivos Excel (.xls e .xlsx)

O Microsoft Excel é um dos *softwares* de planilha mais populares do mundo. São muito utilizados não só no meio corporativo, mas também no meio acadêmico. Apesar de limitações principalmente relacionadas a lidar com datas e tabelas muito grandes, é uma ferramenta poderosa e torna-se ainda melhor se aliada ao R.

O `tidyverse` não carrega automaticamente o pacote necessário para ler arquivos Excel, portanto, precisaremos carregar o `readxl`. Utilizaremos como exemplo uma amostra do dataset `gapminder`, mas desta vez contendo informações para todos os países no ano de 2007.

```r
# Carregando o pacote necessário
library(readxl)

# Importando o arquivo de interesse
gapminder2007 <- read_excel("Aula1/gapminder2007.xlsx")

# Mostrando um pedaço do dataset
head(gapminder2007)
```

```
## # A tibble: 6 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <chr>       <chr>     <dbl>   <dbl>    <dbl>     <dbl>
## 1 Afghanistan Asia       2007    43.8 31889923      975.
## 2 Albania     Europe     2007    76.4  3600523     5937.
## 3 Algeria     Africa     2007    72.3 33333216     6223.
```

```
## 4 Angola      Africa     2007    42.7 12420476     4797.
## 5 Argentina   Americas   2007    75.3 40301927    12779.
## 6 Australia   Oceania    2007    81.2 20434176    34435.
```

Capítulo 4

Visualização

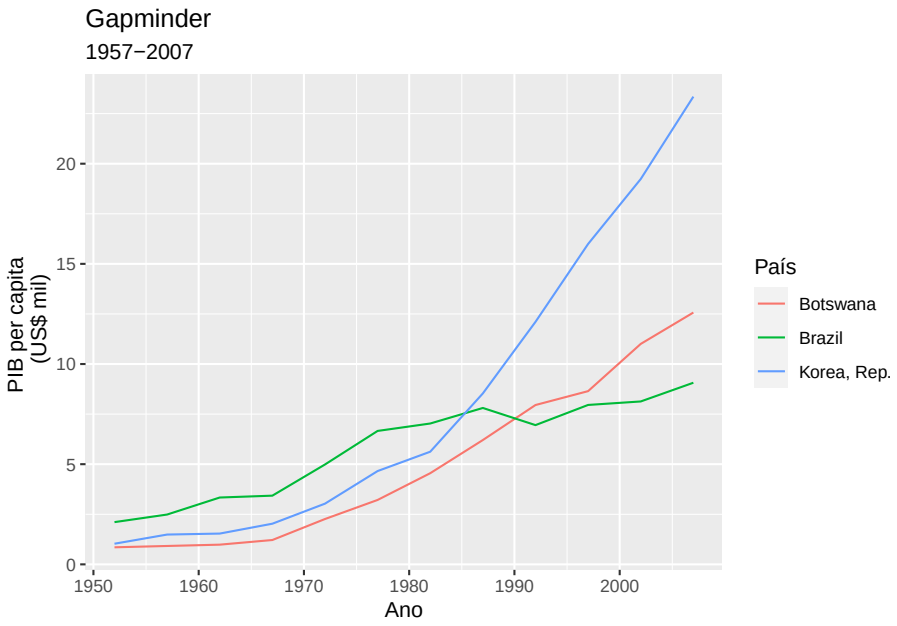
A visualização pode ser utilizada tanto para conhecer os dados com que iremos trabalhar, quanto para comunicar com ao público aquilo que descobrimos. Para tanto, o `tidyverse` fornece o pacote `ggplot2`, uma ferramenta poderossíma!

Ele permite construir desde gráficos simples até visualizações extremamente elaboradas, onde diversas variáveis são mapeadas em diferentes atributos de um gráfico. Somado a outras ferramentas, o `ggplot2` também pode ser utilizado para a construção de mapas.

4.1 Gráfico de linha

Gráficos de linhas são muito utilizados para mostrar tendências ao longo do tempo. Utilizando o dataset `gapminderCountries` que contruímos anteriormente, poderemos ver de que forma o PIB per capita de Brasil, Botsuana e Coreia do Sul se comportaram ao longo do tempo.

```
gapminderCountries %>%
  ggplot() +
  geom_line(aes(x = year,
                y = gdpPercap/10^3,
                color = country)) +
  labs(title = "Gapminder",
       subtitle = "1957-2007",
       color = "País") +
  xlab("Ano") +
  ylab("PIB per capita \n (US$ mil)")
```

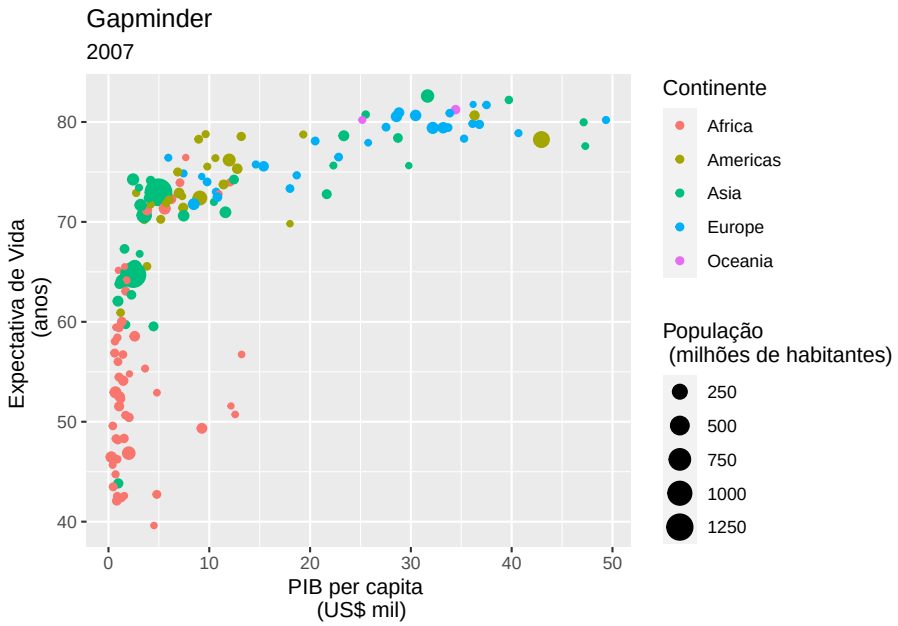


4.2 Gráfico de dispersão

Gráficos de dispersão permitem visualizar a relação entre duas variáveis numéricas, o que nos permite visualizar possíveis correlações. Utilizaremos o dataset `gapminder2007` para ver de que forma o PIB per capita está relacionado à expectativa de vida nos diferentes países do mundo em 2007.

Desta vez poderemos mostrar de que forma o `ggplot2` pode mapear diferentes atributos das observações em um mesmo gráfico. Em nosso exemplo, cada ponto representa um país, cada cor um continente, e o tamanho dos pontos é dado por sua população.

```
gapminder2007 %>%
  ggplot() +
  geom_point(aes(x = gdpPercap/10^3,
                 y = lifeExp,
                 color = continent,
                 size = pop/10^6)) +
  labs(title = "Gapminder",
       subtitle = "2007",
       color = "Continente",
       size = "População \n (milhões de habitantes)") +
  xlab("PIB per capita \n (US$ mil)") +
  ylab("Expectativa de Vida \n (anos)")
```


Gapminder
2007
Expectativa de Vida
(anos)
80
70
60
50
40
0
10
20
30
40
50
PIB per capita
(US$ mil)
Continente
Africa
Americas
Asia
Europe
Oceania
População
(milhões de habitantes)
250
500
750
1000
1250

Capítulo 5

Organização e Transformação

Capítulo 6

Modelagem

Capítulo 7

Próximos Passos