Métodos Quantitativos

Aula 03. Manipulação de dados no R

Pedro H. G. Ferreira de Souza

pedro.ferreira@ipea.gov.br

Mestrado Profissional em Políticas Públicas e Desenvolvimento

Instituto de Pesquisa Econômica Aplicada (Ipea)

26 set. 2022

Aula passada

- Causas de efeitos *vs.* efeitos de causas
- Causalidade como manipulação intencional $\rightarrow$ experimentos
- Problema fundamental da inferência causal é a impossibilidade de observarmos os resultados potenciais
  - Construção de contrafactuais para identificar ATE, ATT e ATU
  - SUTVA como pressuposto
- Quando a diferença de médias é um bom estimador do ATE?
  - SDO = ATE + Viés de seleção + Viés de efeitos heterogêneos
  - Para não ter vieses, $(Y^1, Y^0) \perp D \rightarrow$ apelo da alocação aleatória
- Guia prático de Kellstedt e Whitten
- Quase experimentos

Análise de dados na prática

Processo iterativo

- Explorar os dados
- Produzir estatísticas descritivas
- Estimar modelos
- Visualizar os resultados

Opções de *softwares* estatísticos e linguagens de programação

- **R** é gratuito, rápido, versátil, popular, com enorme comunidade...
- *Python*, *SAS*, *SPSS*, *Stata* e afins são alternativas razoáveis
- *Excel* **não** é boa opção para coisas importantes

Primeiros passos com o R

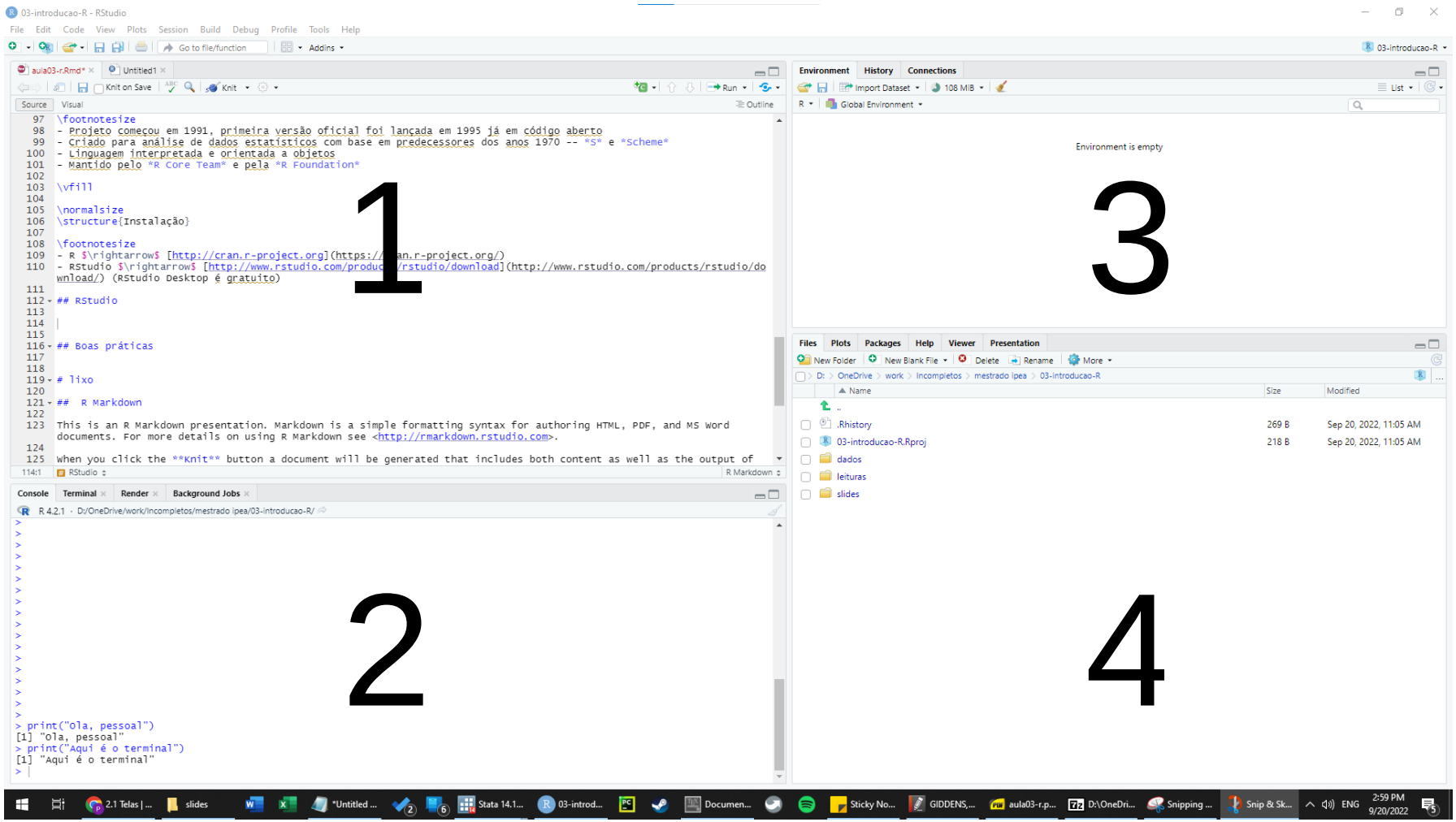
Breve histórico

- Projeto começou em 1991, primeira versão oficial foi lançada em 1995 já em código aberto
- Criado para análise de dados estatísticos inspirado em *S* e *Scheme* (linguagens dos anos 1970)
- Linguagem interpretada e orientada a objetos, mantida pelo *R Core Team* e pela *R Foundation*

Instalação

- R $\rightarrow$ http://cran.r-project.org
- RStudio $\rightarrow$ http://www.rstudio.com/products/rstudio/download (versão *Desktop* é gratuita)

Interface do RStudio



Como passar *inputs* para o R

Console

Input de instruções via linha de comando e exibição dos *outputs*.

É útil para testar coisas rápidas: cada linha é avaliada quando você aperta `enter`.

Script

É um arquivo de texto que te permite escrever e executar sequências de comandos, que ficam registradas permitindo replicabilidade.

- Para criar um script, clique em **File** → **New File** → **R Script**.
- No editor, **Run** (`ctrl+enter`) executa só a linha ou trecho selecionado e **Source** (`ctrl+shift+enter`) executa tudo.

R como calculadora

Execute os comandos abaixo no console ou em um script:

```
1 + 2 - 3 + 5*2 - 18/9
sqrt(4) + 2^3
7 %/% 2
7 %% 2
```

R como calculadora

Execute os comandos abaixo no console ou em um script:

```
1 + 2 - 3 + 5*2 - 18/9
sqrt(4) + 2^3
7 %/% 2
7 %% 2
```

```
## [1] 8
```

```
## [1] 10
```

```
## [1] 3
```

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

Projetos

Correspondem a pastas específicas com todos os arquivos para a análise, servindo para organizar tudo. Para criar um novo projeto:

1. Clique em **File** → **New Project** e escolha uma das opções:
   - **New directory** cria uma nova pasta para o seu projeto
   - **Existing directory** usa alguma pasta já existente
   - **Version control** faz integração com Github e plataformas afins
2. Copie arquivos para a pasta do projeto conforme a necessidade
3. Depois de criar seu projeto, clique em **File** → **New File** → **R Script**
   - Não se esqueça de salvar o script na pasta correta
4. Para reabrir o projeto em outras ocasiões, basta clicar duas vezes no arquivo com extensão `.proj` na pasta do projeto

Pacotes no R

São coleções de funções, dados e documentação que estendem as capacidades do R. Seu repositório central é o CRAN, mas é possível encontrá-los em outros lugares (e.g., Github), e/ou criar novos.

- O site do RStudio tem boas recomendações

Instalação, uso e atualização

```
# É preciso instalar os pacotes antes do primeiro uso
install.packages("PACOTE1")
install.packages(c("PACOTE2","PACOTE3"))
# Depois, carregar pacotes individualmente antes do uso
library(PACOTE1)
# Para atualizar, use update.packages
update.packages(c("PACOTE1","PACOTE2"))
```

Boas práticas

1. Crie pastas para cada novo projeto, de preferência com uma estrutura fixa, como, por exemplo:

```
Projeto XYZ\
|--- codigos\
|--- documentos\
|--- dados\
     |--- brutos\
     |--- limpos\
|--- resultados\
     |--- graficos\
     |--- tabelas\
|--- texto\
```

Boas práticas

2. Dê um nome intuitivo ao arquivo com o script principal para o R
   - Por exemplo, `executa.R` ou coisa do tipo
3. **Nunca** altere os arquivos brutos com os dados originais
   - Crie cópias quando necessário
4. Crie scripts organizados, bem documentados e replicáveis
   - Use o símbolo '#' para comentar o código e ajude seu eu-futuro
5. Evite copiar-e-colar resultados para não cometer erros bobos
   - Tente automatizar tudo; caso não seja possível, muito cuidado!

Boas práticas

6. Use e abuse do Google para tirar dúvidas
   - Pesquise vários termos relacionados até achar a solução
7. **Pense como uma máquina**
   - Sintaxe imperfeita provoca erros $\rightarrow$ cuidado com vírgulas, espaços e pontos e atenção com (), [] e {}
   - Instruções precisam ser claras, sequenciais, exatas, literais, sem ambiguidades
   - Decomponha tarefas grandes em uma sequência de instruçõòes simples e faça testes a cada etapa

Referências úteis

Em português

Curso-R, Ciência de Dados em R

IBPAD, Ciência de Dados com R - Introdução

R-Ladies São Paulo, Oficina de R - Básico e Oficina de R - Intermediário

Vanderlei Debastiani, Introdução ao R

Em inglês

Roger Peng, R Programming for Data Science

Rafael Irizarry, Introduction to Data Science

Hadley Wickham e Garret Grolemund, R for Data Science

Regra #6: Use e abuse do Google!

Objetivos

Objetivo primário

Criar um projeto organizado e aprender como obter dentro do R a lista com conteúdo da pasta de trabalho.

Objetivos secundários

Instalar e carregar pacotes, introduzir funções e objetos

Funções são sequências de comandos que tomam um ou mais parâmetros ou argumentos como *input* e geram *output(s)* único(s)

Objetos são nomes associados a conteúdos, com propriedade ou atributos específicos dependendo do conteúdo

Exercício

Execute no console:

```
pi
round(pi, 2)
pi_arredondado <- round(pi, digits = 2)
pi_arredondado + 10
```

Exercício

Execute no console:

```
pi
round(pi, 2)
pi_arredondado <- round(pi, digits = 2)
pi_arredondado + 10
```

```
## [1] 3.141593
```

```
## [1] 3.14
```

```
## [1] 13.14
```

Documentação de funções:

```
?round
help(round)
```

Instruções para o projeto

1. Crie uma **pasta vazia** em algum lugar no seu computador
2. Baixe o material de apoio da aula 03 no **Github**
   - https://github.com/phgfsouza/mq2022/
3. **Descompacte** o arquivo zip na pasta que você criou
4. Abra o **RStudio**, crie um novo projeto na pasta já existente
   - *File* $\rightarrow$ *New Project* $\rightarrow$ *Existing directory*
5. Crie um **novo script** e salve em `.\codigos\` na pasta do projeto
   - *File* $\rightarrow$ *New File* $\rightarrow$ *R Script*
   - Salve com algum nome do tipo `primeiro-script.R`

Obs: minha versão está em

`.\codigos\primeiro-script-comentado.r`

Estrutura desejada de pastas

```
## D:/OneDrive/work/Incompletos/mestrado ipea/03-
introducao-R
## +-- 03-introducao-R.Rproj
## +-- apoio
## |   \-- material-apoio.zip
## +-- atividade
## |   +-- atividade03-respostas.r
## |   \-- atividade03.r
## +-- codigo
## |   +-- atividade03-respostas.r
## |   +-- atividade03.r
## |   \-- primeiro-script-comentado.r
## +-- dados
## |   +-- brutos
## |   |   +-- datafolha_oxfam.csv
## |   |   \-- datafolha_oxfam_leiame.docx
## |   \-- limpos
```

Conferindo a estrutura de pastas dentro do R [1/2]

```
# Instalacao de pacotes (so precisa uma vez)
# install.packages(c("here", "fs"))

# Carregamento dos pacotes
library(fs)
library(here)

# As funcoes nativas getwd() e setwd():
getwd()
setwd("c:")
getwd()
```

(Continua)

Conferindo a estrutura de pastas dentro do R [2/2]

```
# O pacote 'here' é mais pratico:
here()
pasta_do_projeto <- here()
pasta_do_projeto

# Atributos do objeto que criamos
class(pasta_do_projeto)
str(pasta_do_projeto)

# Usando nosso objeto como input para obter a
# estrutura de pastas
dir_tree(pasta_do_projeto)
dir_tree(here())
```

Resultado esperado

```
## D:/OneDrive/work/Incompletos/mestrado ipea/03-
introducao-R
## +-- 03-introducao-R.Rproj
## +-- apoio
## |   \-- material-apoio.zip
## +-- atividade
## |   +-- atividade03-respostas.r
## |   \-- atividade03.r
## +-- codigo
## |   +-- atividade03-respostas.r
## |   +-- atividade03.r
## |   \-- primeiro-script-comentado.r
## +-- dados
## |   +-- brutos
## |   |   +-- datafolha_oxfam.csv
## |   |   \-- datafolha_oxfam_leiame.docx
## |   \-- limpos
```

Classes e coleções de objetos no R

O R possui cinco tipos básicos ou "atômicos" de objetos:

- **Character** → texto
- **Integer** → números inteiros
- **Numeric** → números racionais
- **Complex** → números complexos (raramente usado)
- **Logical** → booleano TRUE/FALSE

Classes e coleções de objetos no R

O R possui cinco tipos básicos ou "atômicos" de objetos:

- **Character** → texto
- **Integer** → números inteiros
- **Numeric** → números racionais
- **Complex** → números complexos (raramente usado)
- **Logical** → booleano TRUE/FALSE

Coleções de objetos podem ser:

- **Vetores**
- **Matrizes**
- Listas
- Fatores
- **Data frames** ou **tibbles**
- etc

Cuidados ao nomear objetos

1. O R diferencia entre maiúsculas e minúsculas
2. Nomes de objetos não podem conter espaços, somente letras, números, _ e .
3. Nomes de objetos têm que começar com letras
4. Evite usar acentos nos nomes
5. Dê nomes intuitivos aos objetos

Objetos da classe *character*

```
suco <- 'Coca-cola'
print(suco)
class(suco)
```

Objetos da classe *character*

```
suco <- 'Coca-cola'
print(suco)
class(suco)
```

```
## [1] "Coca-cola"
```

```
## [1] "character"
```

Objetos da classe *character*

```
suco <- 'Coca-cola'
print(suco)
class(suco)
```

```
## [1] "Coca-cola"
```

```
## [1] "character"
```

```
bebida <- paste(suco, "é muito melhor", sep = " ")
print(bebida)
Encoding(bebida) <- "latin1"
print(bebida)
```

Objetos da classe *character*

```
suco <- 'Coca-cola'
print(suco)
class(suco)
```

```
## [1] "Coca-cola"
```

```
## [1] "character"
```

```
bebida <- paste(suco, "é muito melhor", sep = " ")
print(bebida)
Encoding(bebida) <- "latin1"
print(bebida)
```

```
## [1] "Coca-cola é muito melhor"
```

```
## [1] "Coca-cola Ã© muito melhor"
```

Objetos da classe *character*

```
Suco <- "O lobo 'ama' o bolo."
print(suco)
print(Suco)
```

Objetos da classe *character*

```
Suco <- "O lobo 'ama' o bolo."
print(suco)
print(Suco)
```

```
## [1] "Coca-cola"
```

```
## [1] "O lobo 'ama' o bolo."
```

Objetos da classe *character*

```
Suco <- "O lobo 'ama' o bolo."
print(suco)
print(Suco)
```

```
## [1] "Coca-cola"
```

```
## [1] "O lobo 'ama' o bolo."
```

```
xpto42_alfa.0 <- "1982"
print(xpto42_alfa.0)
class(xpto42_alfa.0)
```

Objetos da classe *character*

```
Suco <- "O lobo 'ama' o bolo."
print(suco)
print(Suco)
```

```
## [1] "Coca-cola"
```

```
## [1] "O lobo 'ama' o bolo."
```

```
xpto42_alfa.0 <- "1982"
print(xpto42_alfa.0)
class(xpto42_alfa.0)
```

```
## [1] "1982"
```

```
## [1] "character"
```

Objetos das classes *integer* e *numeric*

```
a <- 2022L
b <- 2022
str(a)
str(b)
```

Objetos das classes *integer* e *numeric*

```
a <- 2022L
b <- 2022
str(a)
str(b)
```

```
##  int 2022
```

```
##  num 2022
```

Objetos das classes *integer* e *numeric*

```
a <- 2022L
b <- 2022
str(a)
str(b)
```

```
##  int 2022
```

```
##  num 2022
```

```
a <- (1 + 2 + sqrt(9) + (4/2)^2 + (26 %/% 5) + 27 %% 7) * 2
b <- a^(-1)
str(a)
str(b)
```

Objetos das classes *integer* e *numeric*

```
a <- 2022L
b <- 2022
str(a)
str(b)
```

```
##  int 2022
```

```
##  num 2022
```

```
a <- (1 + 2 + sqrt(9) + (4/2)^2 + (26 %/% 5) + 27 %% 7) * 2
b <- a^(-1)
str(a)
str(b)
```

```
##  num 42
```

```
##  num 0.0238
```

Objetos da classe *logical*

```
nome_aleatorio <- TRUE
nome_aleatorio <- nome_aleatorio == FALSE
teste1 <- a >= b & a > 0
teste2 <- a != b | b < 0
str(nome_aleatorio)
str(teste1)
str(teste2)
teste1 != teste2
```

Objetos da classe *logical*

```
nome_aleatorio <- TRUE
nome_aleatorio <- nome_aleatorio == FALSE
teste1 <- a >= b & a > 0
teste2 <- a != b | b < 0
str(nome_aleatorio)
str(teste1)
str(teste2)
teste1 != teste2
```

```
##  logi FALSE
```

```
##  logi TRUE
```

```
##  logi TRUE
```

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

Objetos da classe *logical*

```
is.character("vamos que vamos")
teste_ch <- is.character("vamos que vamos")
str(teste_ch)
```

Objetos da classe *logical*

```
is.character("vamos que vamos")
teste_ch <- is.character("vamos que vamos")
str(teste_ch)
```

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

```
##  logi TRUE
```

Objetos da classe *logical*

```
is.character("vamos que vamos")
teste_ch <- is.character("vamos que vamos")
str(teste_ch)
```

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

```
##  logi TRUE
```

```
ddd.df <- "061"
is.numeric(ddd.df)
is.logical(ddd.df)
is.character(ddd.df)
```

Objetos da classe *logical*

```
is.character("vamos que vamos")
teste_ch <- is.character("vamos que vamos")
str(teste_ch)
```

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

```
##  logi TRUE
```

```
ddd.df <- "061"
is.numeric(ddd.df)
is.logical(ddd.df)
is.character(ddd.df)
```

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

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

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

Conversão entre classes

```
str(as.integer("2022.5"))
str(as.numeric("2022.5"))
str(as.integer(ddd.df))
str(as.character(42))
str(as.logical(0))
str(as.logical(-0.1))
```

Conversão entre classes

```
str(as.integer("2022.5"))
str(as.numeric("2022.5"))
str(as.integer(ddd.df))
str(as.character(42))
str(as.logical(0))
str(as.logical(-0.1))
```

```
##  int 2022
##  num 2022
##  int 61
##  chr "42"
##  logi FALSE
##  logi TRUE
```

Erros: *NA* e *Inf*

```
as.numeric("ih, rapaz, danou-se")
as.numeric("TRUE")
10 + "oito"
1/0
```

Erros: *NA* e *Inf*

```
as.numeric("ih, rapaz, danou-se")
as.numeric("TRUE")
10 + "oito"
1/0
```

```
## Warning: NAs introduced by coercion

## [1] NA

## Warning: NAs introduced by coercion

## [1] NA

## Error in 10 + "oito": non-numeric argument to binary operator

## [1] Inf
```

Gerenciando o workspace

```
# Funções nativas ls() e rm()
ls()         # enumera os objetos do workspace
rm(ddd.df)  # apaga objetos do workspace
# Combinando as duas, podemos limpar o workspace
rm(list = ls())
ls()
```

Gerenciando o workspace

```
# Funções nativas ls() e rm()
ls()        # enumera os objetos do workspace
rm(ddd.df)  # apaga objetos do workspace
# Combinando as duas, podemos limpar o workspace
rm(list = ls())
ls()
```

```
##  [1] "a"                 "b"                 "bebida"
##  [5] "nome_aleatorio"    "pasta_do_projeto" "pi_arredondado"
##  [9] "Suco"              "teste_ch"          "teste1"
## [13] "xpto42_alfa.0"

##  chr [1:13] "a" "b" "bebida" "ddd.df" "nome_aleatorio" "pasta

## character(0)
```

Objetivos

Objetivo primário

Apresentar algumas das principais estruturas de dados do R: vetores, matrizes e *data frames*.

Falaremos em outras aulas de fatores e listas.

Objetivo secundário

Apresentar novas funções, em especial o pacote `dplyr`, que é muito útil e facilita a sintaxe.

Mostrar como carregar arquivos `csv` e afins no R.

Instruções

1. Crie um **novo script** em branco e salve em `.\codigos\`
   - *File* → *New File* → *R Script*
   - Salve com algum nome do tipo `segundo-script.r`
2. Adicione um cabeçalho no script contendo informações básicas
   - O cabeçalho é só um trecho comentado no topo do script indicando brevemente o objetivo, o autor, data etc
   - O objetivo disso é só criar o hábito de organização em vocês, porque vai ser muito útil no futuro
   - Se quiser um modelo, veja `.\codigos\primeiro-script-comentado.r`

Coleções de objetos

Objetos simples como os que vimos até agora não serem para muita coisa, mas eles podem ser combinados em **estruturas de dados** mais complexas.

O que veremos

Vetores → estrutura unidimensional homogênea

Matrizes → estruturas bidimensionais homogêneas

Data frames → estruturas bidimensionais reunindo vetores, fatores e/ou listas em colunas com o mesmo comprimento

Atributos de objetos

São metadados, isto é, informações descritivas associadas aos objetos e coleções de objetos.

As funções nativas **class()** e **str()** retornam alguns atributos, mas há outras opções, inclusive atributos modificáveis.

Outras funções nativas incluem **typeof()**, **names()**, **dim()** e **dimnames()**.

Alguns atributos podem ser visualizados de uma só vez pela função nativa **attributes()**.

Vetores numéricos

São conjuntos indexados de valores de objetos homogêneos criados por principalmente por **c()**

```
x <- 4
y <- c(8)
vetor_numerico <- c(1, x, y, 15, 16, 23, 42)
print(vetor_numerico)
is.vector(x)
is.vector(vetor_numerico)
```

```
## [1]  1  4  8 15 16 23 42
## [1] TRUE
## [1] TRUE
```

Vetores não numéricos

```
vetor_character <- c("domingo", "segunda", "terça",
                     "quarta", "quinta", "sexta",
                     "sabado")
print(vetor_character)
is.vector(vetor_character)
vetor_logical <- c(TRUE, FALSE, TRUE, FALSE, TRUE,
                   TRUE, FALSE)
print(vetor_logical)
is.vector(vetor_logical)
```

```
## [1] "domingo" "segunda" "terça"   "quarta"  "quinta"  "
## [1] TRUE
## [1]  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE
## [1] TRUE
```

Indexação de vetores

```
vetor_numerico
vetor_numerico[2]
vetor_numerico[4:7]
names(vetor_numerico) <- c("a", "b", "c", "d", "e", "f", "g")
vetor_numerico[c("b", "a")]
vetor_character
vetor_character[-c(1,7)]
```

Indexação de vetores

```
vetor_numerico
vetor_numerico[2]
vetor_numerico[4:7]
names(vetor_numerico) <- c("a", "b", "c", "d", "e", "f", "g")
vetor_numerico[c("b", "a")]
vetor_character
vetor_character[-c(1,7)]
```

```
## [1]  1  4  8 15 16 23 42
## [1] 4
## [1] 15 16 23 42
## b a
## 4 1
## [1] "domingo" "segunda" "terça"   "quarta"  "quinta"  "sexta"
## [1] "segunda" "terça"   "quarta"  "quinta"  "sexta"
```

Homogeneidade de vetores

```
c(1, 2, 3, 4, "cinco")
c(TRUE, FALSE, 5 , 0)
c(TRUE, NA, "zero", "zero", 7)
```

Homogeneidade de vetores

```
c(1, 2, 3, 4, "cinco")
c(TRUE, FALSE, 5 , 0)
c(TRUE, NA, "zero", "zero", 7)
```

```
## [1] "1"     "2"     "3"     "4"     "cinco"
## [1] 1 0 5 0
## [1] "TRUE" NA     "zero" "zero" "7"
```

Funções rep(), seq() e length()

```
x <- 1:10
print(x)
y <- rep(1, 10)
print(y)
z <- c( seq(1,10,2), seq(0, 9, 2))
print(z)
length(x) == 10
```

Funções rep(), seq() e length()

```
x <- 1:10
print(x)
y <- rep(1, 10)
print(y)
z <- c( seq(1,10,2), seq(0, 9, 2))
print(z)
length(x) == 10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

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

```
##  [1] 1 3 5 7 9 0 2 4 6 8
```

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

Operações com vetores

```
x * 10
x - y
x * z
(x / z)[5:7]
x %*% y
```

Operações com vetores

```
x * 10
x - y
x * z
(x / z)[5:7]
x %*% y
```

```
##  [1]  10  20  30  40  50  60  70  80  90 100
```

```
##  [1] 0 1 2 3 4 5 6 7 8 9
```

```
##  [1]  1  6 15 28 45  0 14 32 54 80
```

```
## [1] 0.5555556       Inf 3.5000000
```

```
##      [,1]
## [1,]   55
```

Simulando um dado com vetor numérico

Simulando um dado com vetor numérico

```
dado <- 1:6
sample(dado, size = 1)
sample(dado, size = 3, replace = TRUE)
for(i in c(100, 1000, 10000, 99999)){
  print(paste("Com", i, "lances,", "a média foi",
              mean(sample(dado, size = i, replace = TRUE))))
}
```

Simulando um dado com vetor numérico

```
dado <- 1:6
sample(dado, size = 1)
sample(dado, size = 3, replace = TRUE)
for(i in c(100, 1000, 10000, 99999)){
  print(paste("Com", i, "lances,", "a média foi",
              mean(sample(dado, size = i, replace = TRUE))))
}
```

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

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

```
## [1] "Com 100 lances, a média foi 3.4"
## [1] "Com 1000 lances, a média foi 3.486"
## [1] "Com 10000 lances, a média foi 3.5073"
## [1] "Com 99999 lances, a média foi 3.49205492054921"
```

Matrizes

São coleções bidimensionais de vetores com o mesmo comprimento

```
a <- 1:3
(primeira_matriz <- matrix(c(a, -2*a, a^2, rep(1,3)),
                           ncol = 4))
class(primeira_matriz)
dim(primeira_matriz)
```

Matrizes

São coleções bidimensionais de vetores com o mesmo comprimento

```
a <- 1:3
(primeira_matriz <- matrix(c(a, -2*a, a^2, rep(1,3)),
                           ncol = 4))
class(primeira_matriz)
dim(primeira_matriz)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1   -2    1    1
## [2,]    2   -4    4    1
## [3,]    3   -6    9    1
```

```
## [1] "matrix" "array"
```

```
## [1] 3 4
```

Matrizes não numéricas

```
row1 <- c("bom", "dia", "pessoal")
row2 <- c("como", "estao", "voces?")
row3 <- c("x", "y", "z")
(m <- cbind( rbind(row1, row2, row3), matrix(row3) ))
dim(m); nrow(m); ncol(m)
```

Matrizes não numéricas

```
row1 <- c("bom", "dia", "pessoal")
row2 <- c("como", "estao", "voces?")
row3 <- c("x", "y", "z")
(m <- cbind( rbind(row1, row2, row3), matrix(row3) ))
dim(m); nrow(m); ncol(m)
```

```
##      [,1]   [,2]    [,3]      [,4]
## row1 "bom"  "dia"   "pessoal" "x"
## row2 "como" "estao" "voces?"  "y"
## row3 "x"    "y"     "z"       "z"
```

```
## [1] 3 4
```

```
## [1] 3
```

```
## [1] 4
```

Indexação

```
colnames(m) <- c("C1", "C2", "C3", "C4")
rownames(m) <- c("R1", "R2", "R3")
m[3,2]
m[2:3, 3:4]
m["R1", ]
```

Indexação

```
colnames(m) <- c("C1", "C2", "C3", "C4")
rownames(m) <- c("R1", "R2", "R3")
m[3,2]
m[2:3, 3:4]
m["R1", ]
```

```
## [1] "y"
```

```
##    C3       C4
## R2 "voces?" "y"
## R3 "z"      "z"
```

```
##        C1        C2        C3        C4
##     "bom"     "dia" "pessoal"       "x"
```

Homogeneidade

Funcionamento igual ao de **vetores**:

```
numerico <- c(1, 2, 4, 8)
texto <- c("um", "dois", "quatro", "oito")
(matriz <- matrix( c(numerico, texto), nrow = 2))
```

Homogeneidade

Funcionamento igual ao de **vetores**:

```
numerico <- c(1, 2, 4, 8)
texto <- c("um", "dois", "quatro", "oito")
(matriz <- matrix( c(numerico, texto), nrow = 2))
```

```
##      [,1] [,2] [,3]   [,4]
## [1,] "1"  "4"  "um"   "quatro"
## [2,] "2"  "8"  "dois" "oito"
```

Operações

Funcionamento parecido com o de **vetores**:

```
X <- matrix(c(1, 40, 20, 30), byrow = TRUE,
            nrow = 2, ncol = 2)
Y <- t(X)
X * 10
X + Y
X * Y       # elemento a elemento
X %*% Y     # multiplicacao de matrizes
t(X)        # matriz transposta
solve(X)    # matriz inversa
```

Output omitido para poupar espaço.

Data frames

Principal estrutura para nós: são tabelas retangulares com colunas que podem ser heterogêneas, porém com o mesmo comprimento

- **Tibbles** são data frames moderninhos usados do pacote **tidyverse**

```
# Podemos criar data frames com a função... data.frame()
df <- data.frame(nome = c("Pedro", "Moacir", "Maria"),
                 sexo = c("M", "M", "F"),
                 idade = c(40, 34, 30))
df
dim(df)
str(df)
head(df, n = 1)
tail(df, n = 1)
```

Primeiro data frame

```
##     nome sexo idade
## 1  Pedro    M    40
## 2 Moacir    M    34
## 3  Maria    F    30
```

```
## [1] 3 3
```

```
## 'data.frame':    3 obs. of  3 variables:
##  $ nome : chr  "Pedro" "Moacir" "Maria"
##  $ sexo : chr  "M" "M" "F"
##  $ idade: num  40 34 30
```

```
##    nome sexo idade
## 1 Pedro    M    40
```

```
##    nome sexo idade
## 3 Maria    F    30
```

Importando dados

```
#install.packages("tidyverse")
library(tidyverse)
library(here)
oxfam.df <- readr::read_csv(here("dados", "brutos",
                                 "datafolha_oxfam.csv"))
dim(oxfam.df)
class(oxfam.df)
str(oxfam.df)
view(oxfam.df)
glimpse(oxfam.df)
```

Output omitido para poupar espaço.

Os pacotes `readr`, `readxl` e `haven` importam quase todos os formatos.

Indexação

```
# Outputs idênticos:
head(oxfam.df[1], n = 5)
oxfam.df[1:5, 1]
oxfam.df[1:5, "regiao"]
oxfam.df$regiao[1:5]
```

Indexação

```
# Outputs idênticos:
head(oxfam.df[1], n = 5)
oxfam.df[1:5, 1]
oxfam.df[1:5, "regiao"]
oxfam.df$regiao[1:5]
```

Em **data frames**, vocês podem ser referir a uma variável usando $, inclusive para criá-las, mas para apagá-las é um pouco diferente:

```
# Sintaxe nativa para criar coluna
oxfam.df$nova_variavel <- oxfam.df$regiao
dim(oxfam.df)
# Sintaxe nativa para apagar uma coluna
oxfam.df <- subset(oxfam.df, select = -c(nova_variavel))
```

dplyr

É um pacote de manipulação de dados da coleção **tidyverse** baseado no operador *pipe* %>% e em verbos explícitos:

filter() → filtro para selecionar linhas

select() → filtro para selecionar colunas

arrange() → ordena os dados de acordo com uma ou mais colunas

mutate() → cria ou modifica colunas

summarise() → sumário agregado por categorias de uma coluna

group_by() → agrupa linhas por categorias de uma coluna

filter

```
# Carregando pacotes (ja instalados na minha maquina)
library(tidyverse)
library(summarytools)
# Filtrando apenas regiao sudeste
filter_ex1 <- oxfam.df %>% filter(regiao == "Sudeste")
# Frequencia das regioes no novo data frame
filter_ex1 %>%
  freq(regiao, round.digits = 1, headings = FALSE,
       totals = FALSE, cumul = FALSE)
```

filter

```
# Carregando pacotes (ja instalados na minha maquina)
library(tidyverse)
library(summarytools)
# Filtrando apenas regiao sudeste
filter_ex1 <- oxfam.df %>% filter(regiao == "Sudeste")
# Frequencia das regioes no novo data frame
filter_ex1 %>%
  freq(regiao, round.digits = 1, headings = FALSE,
       totals = FALSE, cumul = FALSE)
```

```
##
##                  Freq   % Valid   % Total
## ------------- ------ --------- ---------
##       Sudeste    910     100.0     100.0
##          <NA>      0                 0.0
```

filter

```
# Somente homens pretos ou pardos
filter_ex2 <- oxfam.df %>%
                  filter(sexo=="Masculino" &
                         cor_raca %in% c("Preta", "Parda"))
filter_ex2 %>% with(ctable(x = sexo, y = cor_raca,
                           headings = FALSE, prop = "t"))
```

filter

```
# Somente homens pretos ou pardos
filter_ex2 <- oxfam.df %>%
                  filter(sexo=="Masculino" &
                               cor_raca %in% c("Preta", "Parda"))
filter_ex2 %>% with(ctable(x = sexo, y = cor_raca,
                           headings = FALSE, prop = "t"))
```

```
##
## ---------- --------- ------------ ------------ ------------
-
##              cor_raca        Parda        Preta        Total
##         sexo
##    Masculino             432 (74.1%)   151 (25.9%)   583 (100.0%)
##        Total             432 (74.1%)   151 (25.9%)   583 (100.0%)
## ---------- --------- ------------ ------------ ------------
-
```

filter

```
# Somente adultos entre 18 e 22 anos OU com renda
# familiar diferente de 'R$ 0 a 998';
filter_ex3 <- oxfam.df %>%
                  filter(between(idade, 18, 22) |
                               !renda_familiar=="R$ 0 a 998")
filter_ex3 %>% with(ctable(x = as.character(idade),
                           y = renda_familiar,
                           headings = FALSE, prop = "t"))
```

Output omitido para poupar espaço.

select

```
# Seleciona so colunas 'regiao', 'idade' e 'religiao'
select_ex1 <- oxfam.df %>%
                  select(regiao, idade, religiao)
glimpse(select_ex1)
```

select

```
# Seleciona so colunas 'regiao', 'idade' e 'religiao'
select_ex1 <- oxfam.df %>%
                  select(regiao, idade, religiao)
glimpse(select_ex1)
```

```
## Rows: 2,086
## Columns: 3
## $ regiao   <chr> "Sudeste", "Sudeste", "Sudeste", "Nort
## $ idade    <dbl> 74, 65, 54, 38, 38, 62, 48, 63, 52, 2
## $ religiao <chr> "Católica", "Católica", "Católica", "C
```

select

```
# Seleciona so colunas que começam com 'r'
select_ex2 <- oxfam.df %>% select(starts_with('r'))
glimpse(select_ex2)
```

select

```r
# Seleciona so colunas que começam com 'r'
select_ex2 <- oxfam.df %>% select(starts_with('r'))
glimpse(select_ex2)
```

```
## Rows: 2,086
## Columns: 4
## $ regiao           <chr> "Sudeste", "Sudeste", "Sudeste
## $ religiao         <chr> "Católica", "Católica", "Catól
## $ renda_individual <chr> "R$ 2995 a 4990", "R$ 2995 a 4
## $ renda_familiar   <chr> "R$ 2995 a 4990", "R$ 4991 a 9
```

select

```
# Exclui colunas que terminam com 'ao' ou contem 'p'
select_ex3 <- oxfam.df %>%
                  select(-ends_with('ao'),
                         -contains('p'))
glimpse(select_ex3)
```

select

```
# Exclui colunas que terminam com 'ao' ou contem 'p'
select_ex3 <- oxfam.df %>%
                  select(-ends_with('ao'),
                         -contains('p'))
glimpse(select_ex3)
```

```
## Rows: 2,086
## Columns: 5
## $ sexo             <chr> "Masculino", "Masculino", "Mas
## $ idade            <dbl> 74, 65, 54, 38, 38, 62, 48, 6
## $ cor_raca         <chr> "Branca", "Parda", "Parda", "P
## $ renda_individual <chr> "R$ 2995 a 4990", "R$ 2995 a 4
## $ renda_familiar   <chr> "R$ 2995 a 4990", "R$ 4991 a 9
```

arrange

```
# Combinando funções + introduzindo o arrange
arrange_ex1 <- oxfam.df %>%
                  filter(idade < 18) %>%
                    select(idade, sexo, regiao) %>%
                      arrange(sexo, desc(idade))
head(arrange_ex1, n = 5)
```

arrange

```
# Combinando funções + introduzindo o arrange
arrange_ex1 <- oxfam.df %>%
                  filter(idade < 18) %>%
                    select(idade, sexo, regiao) %>%
                      arrange(sexo, desc(idade))
head(arrange_ex1, n = 5)
```

```
## # A tibble: 5 x 3
##   idade sexo     regiao
##   <dbl> <chr>    <chr>
## 1    17 Feminino Sul
## 2    17 Feminino Sudeste
## 3    17 Feminino Norte
## 4    17 Feminino Sudeste
## 5    17 Feminino Nordeste
```

mutate

```
# Criando idade em meses
oxfam.df <- oxfam.df %>%
                  mutate(idade_meses = idade * 12)
oxfam.df %>%
  select(idade, idade_meses) %>%
    descr(stats = c("mean", "sd", "min", "max"))
```

mutate

```
# Criando idade em meses
oxfam.df <- oxfam.df %>%
                  mutate(idade_meses = idade * 12)
oxfam.df %>%
  select(idade, idade_meses) %>%
    descr(stats = c("mean", "sd", "min", "max"))
```

```
##
##                  idade   idade_meses
## ------------- ------- -------------
##          Mean   40.65         487.74
##       Std.Dev   16.48         197.77
##           Min   16.00         192.00
##           Max   87.00        1044.00
```

mutate

```
# Criando duas variaveis: indicador para idoso e
# razao idade/idade media
oxfam.df <- oxfam.df %>%
                  mutate(idoso = idade > 65,
                         razao_idade = idade / mean(idade))
oxfam.df %>%
  select(idoso, razao_idade) %>%
    descr(stats = c("mean", "sd", "min", "max"))
oxfam.df %>% freq(idoso, headings = FALSE)
```

Resposta no próximo slide.

mutate

```
##
##                 razao_idade
## ------------- -------------
##          Mean            1.00
##       Std.Dev            0.41
##           Min            0.39
##           Max            2.14
##
##                Freq   % Valid   % Valid Cum.   % Total   % Tot
## ---------- ------ --------- -------------- --------- ---
-----------
##       FALSE   1922     92.14          92.14     92.14
##        TRUE    164      7.86         100.00      7.86
##        <NA>      0                                0.00
##       Total   2086    100.00         100.00    100.00
```

Exercício #1

```
# a) Filtre so quem se acha na metade de baixo da
# distribuicao de renda (p13)

# b) Em seguida, crie a variável numerica 'media_top10'
# com a media das estimativas de renda necessaria para
# pertencer aos 10% mais ricos (p13)

# c) Depois, crie a variavel logical 'corrupcao' com
# valor TRUE se a pessoa achar que o combate 'a
# corrupcao eh a coisa mais importante para reduzir
# a desigualdade

# d) Por fim, mostre a tabela de frequencia dos
# resultados
```

Exercício #1

```
# A etapa (a) exige o uso do 'filter' (dplyr).
# A (b) e a (c) implicam uso do 'mutate' (dplyr).
exercicio1 <-
  oxfam.df %>%
    filter(...) %>%
      mutate(...)

# A (d) pode ser feita com o 'freq' (summarytools).
exercicio1 %>% freq(...)
```

Exercício #1

```
# Etapas (a), (b) e (c)
exercicio1 <-
  oxfam.df %>%
    filter(p13 >= 50) %>%
      mutate(media_10pct = mean(p19, na.rm = TRUE),
             corrupcao = p27e > p27a & p27e > p27b &
               p27e > p27c & p27e > p27d)

# A (d) pode ser feita com o 'freq' (summarytools).
exercicio1 %>% freq(corrupcao)
```

Exercício #1

```
## Frequencies
## exercicio1$corrupcao
## Type: Logical
##
##                Freq   % Valid   % Valid Cum.   % Total   % Tot
## ----------- ------ --------- -------------- --------- ---
-----------
##       FALSE    812     98.07          98.07     97.60
##        TRUE     16      1.93         100.00      1.92
##        <NA>      4                                0.48
##       Total    832    100.00         100.00    100.00
```

Exercício #2

```
# Para quem se acha nos 50% mais ricos, qual a media e a
# mediana de renda necessarias para estar nos 10% mais
# ricos? E para quem se acha nos 50% mais pobres?
```

Exercício #2

```
# Para quem se acha nos 50% mais ricos, qual a media e a
# mediana de renda necessarias para estar nos 10% mais
# ricos? E para quem se acha nos 50% mais pobres?
```

```
# 50% mais ricos
oxfam.df %>%
  filter(...) %>%
    select(...) %>%
      descr(...)
# 50% mais pobres
oxfam.df %>%
  filter(...) %>%
    select(...) %>%
      descr(...)
```

Exercício #2

```
# 50% mais ricos
oxfam.df %>%
  filter(p13 >= 50) %>%
    select(p19) %>%
      descr(stats = c('mean','med'))
```

Exercício #2

```
# 50% mais ricos
oxfam.df %>%
  filter(p13 >= 50) %>%
    select(p19) %>%
      descr(stats = c('mean','med'))
```

```
## Descriptive Statistics
## oxfam.df$p19
## N: 832
##
##                      Mean      Median
## --------- -------------- -----------
##       p19   16788866.88    30000.00
```

Exercício #2

```
# 50% mais pobres
oxfam.df %>%
  filter(p13 < 50) %>%
    select(p19) %>%
      descr(stats = c('mean','med'))
```

Exercício #2

```
# 50% mais pobres
oxfam.df %>%
  filter(p13 < 50) %>%
    select(p19) %>%
      descr(stats = c('mean','med'))
```

```
## Descriptive Statistics
## oxfam.df$p19
## N: 1254
##
##                      Mean      Median
## --------- -------------- -----------
##       p19   17221321.70   30000.00
```

Exercício #3

```
# Recodifique a p13 em décimos da distribuicao
# de renda usando o operador %/% e depois mostre
# a tabela de frequencias
```

Exercício #3

```
# Recodifique a p13 em décimos da distribuicao
# de renda usando o operador %/% e depois mostre
# a tabela de frequencias
```

```
oxfam.df %>%
  mutate(...) %>%
    freq(...)
```

Exercício #3

```
# Primeira tentativa
oxfam.df %>%
  mutate(p13dec = p13 %/% 10) %>%
    freq(p13dec)
```

Resultados no próximo slide.

Exercício #3

```
##
##                Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
## ----------- ------ --------- -------------- --------- ----------
----
##           0    438     21.00          21.00     21.00          21.00
##           1    210     10.07          31.06     10.07          31.06
##           2    155      7.43          38.49      7.43          38.49
##           3    196      9.40          47.89      9.40          47.89
##           4    255     12.22          60.12     12.22          60.12
##           5    509     24.40          84.52     24.40          84.52
##           6    137      6.57          91.08      6.57          91.08
##           7     87      4.17          95.25      4.17          95.25
##           8     53      2.54          97.79      2.54          97.79
##           9     18      0.86          98.66      0.86          98.66
##          10     28      1.34         100.00      1.34         100.00
##        <NA>      0                                0.00         100.00
##       Total   2086    100.00         100.00    100.00         100.00
```

Exercício #3:

```
# O que deu errado?
# Queriamos decimos, mas a tabela tem 11 grupos,
# alem do NA.

# Para consertar, vamos introduzir uma nova funcao do
# dplyr, a if_else().
# Vamos aproveitar tambem para corrigir a numeracao
# para ficar de 1 a 10, como eh convencional.

oxfam.df %>%
  mutate(p13 = if_else(p13 == 100, 99, p13),
         p13dec = (p13 %/% 10) + 1) %>%
    freq(p13dec)
```

Exercício #3

```
##
##                Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
## ----------- ------ --------- -------------- --------- ----------
----
##           1    438     21.00          21.00     21.00          21.00
##           2    210     10.07          31.06     10.07          31.06
##           3    155      7.43          38.49      7.43          38.49
##           4    196      9.40          47.89      9.40          47.89
##           5    255     12.22          60.12     12.22          60.12
##           6    509     24.40          84.52     24.40          84.52
##           7    137      6.57          91.08      6.57          91.08
##           8     87      4.17          95.25      4.17          95.25
##           9     53      2.54          97.79      2.54          97.79
##          10     46      2.21         100.00      2.21         100.00
##        <NA>      0                                0.00         100.00
##       Total   2086    100.00         100.00    100.00         100.00
```

Próxima aula

Atividades

Entrega da atividade #3 no Google Classroom

Leituras obrigatórias

Agresti 2018, cap. 3

Bussab e Morettin 2010, cap. 3 e 4

Leituras optativas

Huntington-Klein 2022, cap. 3 e 4

Kellstedt e Whitten 2018, cap. 6