Introdução à Estatística usando o R: Seja bem-vind@ ao tidyverse

Profa Carolina & Prof Gilberto

Instituto de Matemática e Estatística
Universidade Federal da Bahia

Cronograma do curso

- Aula 1: Introdução ao R
- Aula 2: Estatística descritiva
- Aula 3: Probabilidade
- Aula 4: Intervalo de confiança
- Aula 5: Testes de hipóteses
- Aula 6: Regressão linear simples

Antes de começar

Abra o RStudio (editor que usaremos para aprender a usar R)

Instale os seguintes pacotes (caso ainda não tenha instalado: se der erro ao carregar, o pacote não está instalado)

- `install.packages('tidyverse')`
- `install.packages('MASS')`
- `install.packages('readxl')`
- `install.packages('openxlsx')`
- `install.packages('DescTools')`
- `devtools::install_github('gilberto-sassi/power')`

Carregue os pacotes necessários

- `library(MASS)`
- `library(readxl)`
- `library(xlsx)`
- `library(DescTools)`
- `library(tidyverse)`

Em que

- `install.package`: função que baixa (precisa de internet) e instala novos pacotes;
- `library`: carrega e prepara para uso pacotes instalados.

Origens históricas do R

História do S (precursor do R)

- R é uma linguagem derivada do S
- S foi desenvolvido em Fortran por John Chambers em 1976 no Bell Labs
- S foi desenvolvido para ser um ambiente de análise estatística
- Em 1988, a versão 4 (implementada em C) foi lançada (permitiu potabilidade entre sistemas operacionais)
- Filosofia do S: permitir que usuários possam analisar dados usando estatística com pouco conhecimento de programação

História do R

- Em 1991, Ross Ihaka e Robert Gentleman criaram o R na Nova Zelândia
- Em 1995, Ross e Robert liberam o R sob a licença "GNU General License", o que tornou o R um software livre
- Em 1997, The Core Group é criado para melhorar e controlar o código fonte do R

Características do R

- Constante melhoramento e atualização
- Portabilidade (roda em praticamente todos os sistemas operacionais)
- Grande comunidade de desenvolvedores que adicionam novas capacidades ao R através de pacotes
- Gráficos de maneira relativamente simples
- Interatividade
- Uma grande comunidade de usuários (especialmente útil para resolução de problemas)

Referência para aprender R

Onde baixar o R/RStudio:

- Baixe o R: https://cran.r-project.org/
- RStudio (IDE / Editor de R): https://www.rstudio.com/

Livros:

- Iniciante no R: Hands-On Programming with R: Write Your Own Functions and Simulations
- Intermediário no R: R for data science
- Avançado no R: Advanced R

Na internet:

- Material em português – Curso-R: http://material.curso-r.com/
- Nível intermediário de R – R for data science: http://r4ds.had.co.nz/
- Nível avançado de R – R Advanced: http://adv-r.had.co.nz/

O que fazer quando estiver em apuros?

- Documentação do R

```
help(mean) #pedindo ajuda pelo console
?mean #modo alternativo de pedir ajuda pelo console
```

- Programador mais experiente mais próximo
- Stack Overflow: https://pt.stackoverflow.com/
- Google

```
log('G')
```

```
## Error in log("G"): non-numeric argument to mathemat
```

Pesquisar no Google "`Error in log("G") : non-numeric argument to mathematical function`"

O RStudio

Componentes do RStudio

- Editor/Scripts: é onde escrevemos nossos códigos.
- Console: é onde rodamos o código e recebemos as saídas.
- Environment: painel com todos os objetos criados na sessão.
- Files: mostra os arquivos no diretório de trabalho. É possível navegar entre diretórios.
- Plots: painel onde os gráficos serão apresentados.
- Help: janela onde a documentação das funções serão apresentadas.
- History: painel com um histórico dos comandos rodados.
- Environment: Objetos criados.
- Packages: Gerenciador de pacotes do Editor RStudio.

Começando a usar o RStudio

1. Separe uma pasta para desenvolver a sua análise;
2. Crie um novo projeto nesta pasta;
3. Como rodar um código no R:
   - Selecione parte do código e clique em Crtl+Enter ou Crtl+R
   - Selecione parte do código e clique no botão "Run"
   - Digite no console o seguinte código: `source("nome do arquivo.R")`
   - Digite Crtl+Shift+R ou Crtl+Shift+S para rodar todo o arquivo ativo
4. Instalar pacotes e carregar pacotes:
   - através da interface gráfico do RStudio (Packages)
   - através do comando `install.packages("nome do pacote")`
   - para carregar pacotes: `library("nome do pacote")` ou `require("nome do pacote")`.

Operações aritméticas básicas para números

```
#Soma
1+1
```

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

```
#subtração
2-1
```

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

```
#divisão
3/2
```

```
## [1] 1.5
```

Operações aritméticas básicas para números

```
#pontenciação
2^3
```

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

```
#Resto da divisão de 5 por 3
5 %% 3
```

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

```
#parte inteira da divisão de 5 por 3
5 %/% 3
```

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

Figura 1: Divisão inteira %/% e operador resto %%.

$$
\begin{array}{r|l}
5 & 3 \\
\cline{2-2}
-3 & 1 = 5 \ \%\% \ 3 \\
\cline{1-1}
2 = 5 \ \%/\% \ 3 &
\end{array}
$$

Tipos básicos de dados no R

R é uma linguagem vetorial e matricial, e os objetos básicos são vetores, listas e matrizes. Números são vetores de comprimento 1.

Vetores são elementos no R caracterizados por todos os valores serem do mesmo tipo. Existem 5 tipos básicos de dados no R:

- Inteiro (Integer)

```
a <- 1L
typeof(a)
```

```
## [1] "integer"
```

- Número complexo (Complex)

```
a <- 2 + 5i
typeof(a)
```

```
## [1] "complex"
```

Tipos básicos de dados no R (continuação)

- Lógico (Logic)

```
a <- TRUE
typeof(a)
```

```
## [1] "logical"
```

- Número real (double)

```
a <- 1.3
typeof(a)
```

```
## [1] "double"
```

- Caracter (character)

```
a <- "Eu mesmo: Gilberto"
typeof(a)
```

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

Vetores no R

- Vetor numérico

```
a <- c(1, 2, 3)
print(a)
```

```
## [1] 1 2 3
```

```
class(a)
```

```
## [1] "numeric"
```

- Vetor caracter

```
a <- c("Gilberto", "Pereira", "Sassi")
print(a)
```

```
## [1] "Gilberto" "Pereira"  "Sassi"
```

```
class(a)
```

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

Matrizes no R

- Matriz numérica

```
(a <-  matrix(1:6, nrow = 2, ncol = 3) )
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
class(a)
```

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

- Matriz caracter

```
(a <- matrix(c('a','b','c','d'), nrow = 2, ncol = 2))
```

```
##      [,1] [,2]
## [1,] "a"  "c"
## [2,] "b"  "d"
```

```
class(a)
```

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

Operações com matrizes

- soma de matrizes (duas matrizes de mesma dimensão)

```
(A <- matrix(1:6, nrow = 3, ncol = 2))
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

```
B <- matrix(rep(0.1,6), nrow = 3, ncol = 2)
(C <- A + B)
```

```
##      [,1] [,2]
## [1,]  1.1  4.1
## [2,]  2.1  5.1
## [3,]  3.1  6.1
```

Operações com matrizes

- Transposição de matriz

```
(D <- t(A))
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

- Multiplicação de matrizes (quando possível)

```
(E <- A %*% t(A))
```

```
##      [,1] [,2] [,3]
## [1,]   17   22   27
## [2,]   22   29   36
## [3,]   27   36   45
```

Operações com matrizes

- Matriz identidade

```
(A <- diag(3))
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

- Matriz diagonal

```
(A <- diag(1:3))
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    2    0
## [3,]    0    0    3
```

Operações com matrizes

- Retirar a diagonal principal de um matriz

```
(A <- matrix(1:9, nrow = 3, ncol = 3))
```

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

```
(B <- diag(A))
```

```
## [1] 1 5 9
```

Operações com matrizes

- Retirar uma linha de um matriz

```
A <- matrix(1:9, nrow = 3, ncol = 3)
print(A[1, ]) #selecionar a linha 1
```

```
## [1] 1 4 7
```

```
print(A[,1]) #selecionar a coluna 1
```

```
## [1] 1 2 3
```

- Modificar o valor de um único elemento da matriz

```
A[1,3] <- 0.1 #Atribui 0,1 ao valor A[1,3]
print(A)
```

```
##      [,1] [,2] [,3]
## [1,]    1    4  0.1
## [2,]    2    5  8.0
## [3,]    3    6  9.0
```

Operações com matrizes

- Determinante da matriz

```
det(A)
```

```
## [1] 20.7
```

- Matriz inversa de uma matriz quadrada (`A %*% ginv(A) %*% A == A`) – precisa do pacote `MASS`

```
ginv(A)
```

```
##            [,1]       [,2]       [,3]
## [1,] -0.1449275 -1.7101449  1.5217391
## [2,]  0.2898551  0.4202899 -0.3768116
## [3,] -0.1449275  0.2898551 -0.1449275
```

Alguns comandos úteis para matrizes

Operador ou função	Descrição
$A * B$	multiplicação ponto-a-ponto
$A\%*\%B$	multiplicação matricial
$A\%o\%B$	multiplicação exterior $A \cdot B^\top$
$crossprod(A, B)$	$A \cdot B^\top$
$crossprod(A)$	$A \cdot A^\top$
$t(A)$	transposta da matriz
$diag(x)$	cria uma matriz diagonal igual a x
$diag(A)$	retira a diagonal da matriz
$diag(k)$	cria uma matriz identidade de ordem k
$ginv(A)$	matriz inversa de uma matriz do pacote MASS
$rowMeans(A)$	média por linhas
$rowSums(A)$	soma por linhas
$colMeans(A)$	médias por colunas
$colSums(A)$	soma por colunas

Valores especiais

- `NA` (Not Available): significa que o valor está faltante. Para verificar se um objeto é `NA`, usamos a função `is.na()`;
- `NaN` (Not a Number): significa que o resultado da operação envolvendo número não é um número. Para verificar se um objeto é `NaN`, usamos a função `is.nan()`;
- `Inf` (Infinito): significa que o valor numérico do objeto é maior que o limite que a máquina suporta. Para verificar se um objeto é `Inf`, usamos a função `is.infinite()`;
- `NULL` (Null): ausência de informação. `NA` está mais associada com ausência de informação em um conjunto de dados e `NULL` está associado com ausência de informação em programação. `is.null()` verifica se o objeto é `NULL`.

Listas

Podemos agregar quaisquer tipo de objeto em um único objeto chamado list.

```
#Exemplo
pedido <- list(pedido_id = 8001406,
               nome = "Fulano de Tal",
               cpf = "12345678900",
               itens = list(
                 list(descricao = "Ferrari",
                      frete = 0,
                      valor = 500000),
                 list(descricao = "Dolly",
                      frete = 1.5,
                      valor = 3.90)))
```

Listas

```
#retirar a ID do pedido
pedido$pedido_id
```

```
## [1] 8001406
```

```
#Quarto elemento da lista
pedido[3]
```

```
## $cpf
## [1] "12345678900"
```

```
#valor do quarto elemento da lista
pedido[[3]]
```

```
## [1] "12345678900"
```

Listas

```
#nome de um atributo da lista
pedido['nome']
```

```
## $nome
## [1] "Fulano de Tal"
```

```
#nomes da lista
names(pedido)
```

```
## [1] "pedido_id" "nome"      "cpf"       "itens"
```

tibble

Um data.frame é o mesmo que uma tabela do SQL ou um spreadsheet do Excel. Seus dados serão armazenados como um objeto `tibble`.

Um `tibble` é uma tabela, em que cada linha é um elemento ou indvíduo da amostra e cada coluna é uma variável.

- Operação com data.frame
  - `head()` - Mostra as primeiras 6 observações (linhas).
  - `tail()` - Mostra as últimas 6 observações (linhas).
  - `dim()` - Número de observações (linhas) e de variáveis (colunas).
  - `names()` - Os nomes das variáveis (colunas).
  - `add_column()` - adiciona uma variável (coluna) ao tibble.
  - `add_row()` ou `add_case` - adiciona novas observações (linhas) ao tibble.
  - `glimpse()` - sumário sobre o data-frame.

tibble

```
(dados <- tibble(temp = c(10,16,22),
                 especie = rep('trigo', 3),
                 germinacao = c(0,2,0)))
```

```
## # A tibble: 3 x 3
##    temp especie germinacao
##   <dbl> <chr>        <dbl>
## 1    10 trigo            0
## 2    16 trigo            2
## 3    22 trigo            0
```

```
head(dados, n= 1) #n primeiras observações da amostra
```

```
## # A tibble: 1 x 3
##    temp especie germinacao
##   <dbl> <chr>        <dbl>
## 1    10 trigo            0
```

tibble

```
tail(dados, n = 1) #n últimas observações da amostra
```

```
## # A tibble: 1 x 3
##    temp especie germinacao
##   <dbl> <chr>         <dbl>
## 1    22 trigo             0
```

```
#adicionando uma variável
(dados <- add_column(dados, inicio = c(0,0,0),
                     fim = c(10,15,5)))
```

```
## # A tibble: 3 x 5
##    temp especie germinacao inicio   fim
##   <dbl> <chr>         <dbl>  <dbl> <dbl>
## 1    10 trigo             0      0    10
## 2    16 trigo             2      0    15
## 3    22 trigo             0      0     5
```

```
names(dados) #nome das variáveis
```

```
## [1] "temp"       "especie"    "germinacao" "inicio"
```

tibble

```
(dados <- add_case(dados,
                   temp = 10,
                   especie = 'trigo',
                   germinacao = 7))
```

```
## # A tibble: 4 x 5
##    temp especie germinacao inicio   fim
##   <dbl> <chr>        <dbl>  <dbl> <dbl>
## 1    10 trigo            0      0    10
## 2    16 trigo            2      0    15
## 3    22 trigo            0      0     5
## 4    10 trigo            7     NA    NA
```

```
dim(dados) # número de observação e de variáveis
```

```
## [1] 4 5
```

Lendo os dados no R – arquivos excel

Para arquivos excel, usamos as funções `read_xls` e `read_xlsx` do pacote `readxl`.

Informações que precisam ser informadas:

- `path`: caminho completo até o arquivo excel;
- `sheet`: nome da planilha que será lida;
- `range`: delimita as células que serão lidas;
- `col_names`: argumento lógico. Se `TRUE`, a primeira linha de `range` é nome das variáveis.

```
df_iris <- read_xlsx(path = 'dados.xlsx',
                     sheet = 'Iris',
                         range = "A1:E101",
                         col_names = TRUE)
```

Lendo os dados no R

```
glimpse(df_iris)
```

```
## Rows: 100
## Columns: 5
## $ sepala_comp <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5
## $ sepala_larg <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3
## $ petala_comp <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1
## $ petala_larg <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0
## $ especie     <chr> "setosa", "setosa", "setosa", "setosa
```

Lendo os dados no R – arquivos de texto (csv ou txt)

Para arquivos de texto, usamos as funções `read_delim` do pacote `readr` (incluso no `tidyverse`).

Informações que precisam ser informadas:

- `file`: caminho completo até o arquivo `.txt` ou `.csv`;
- `col_names`: argumento lógico. Se `TRUE`, a primeira linha do arquivo `.txt` é o nome das variáveis.
- `delim`: caracter delimitador ou divisor das variáveis ou colunas.
- `locale`: opções para ler arquivos. Aqui no curso, vamos usar principalmente para especificar o sinal de decimal.

```
# lendo arquivos txt
df_iris <- read_delim(file = "iris.txt",col_names = TRUE,
            delim = "|", locale = locale(decimal_mark = ","))
```

```
## Parsed with column specification:
## cols(
##   sepala_comp = col_double(),
##   sepala_larg = col_double(),
##   petala_comp = col_double(),
##   petala_larg = col_double(),
##   especie = col_character()
## )
```

Lendo os dados no R – arquivos de texto (continuação)

```r
# lendo arquivos csv (formato europeu -- usado no Brasil)
df_iris <- read_csv2(file = "iris.csv", col_names = TRUE)
```

```
## Using ',' as decimal and '.' as grouping mark. Use read_delim() for mor
```

```
## Parsed with column specification:
## cols(
##   sepala_comp = col_double(),
##   sepala_larg = col_double(),
##   petala_comp = col_double(),
##   petala_larg = col_double(),
##   especie = col_character()
## )
```

Para salvar um `tibble`ou `matrix`, podemos usar a função `write.xlsx` do pacote `openxlsx`.

```r
write.xlsx(df_iris, 'nome_arquivo.xlsx')
```