Estatística Computacional

Universidade Federal da Bahia

Gilberto Pereira Sassi

Tópico 1

Preparando o ambiente

- Você precisa de um computador para acompanhar as aulas
- Usaremos nas aulas: colab.research.google.com/#language=r e o rstudio.cloud
- Recomendo instalar o `R` com versão pelo `4.1`: cran.r-project.org
- **IDE** recomendadas: *RStudio* e *VSCode*
  - Caso você queira usar o *VSCode*, instale a extensão da linguagem `R`: `ikuyadeu.r`
- Neste curso, usaremos o *framework* **tidyverse**:
  - Instale o framework a partir do repositório CRAN: `install.packages("tidyverse")`
- Outras linguagens interessantes (se tivéssemos tempo): `python` e `julia`
  - `python`: linguagem interpretada de próposito geral, contemporânea do `R`, simples e fácil de aprender
  - `julia`: linguagem interpretada para análise de dados, lançada em 2012, promete simplicidade e velocidade
- **Conselho não solicitado:** Aprendam `HTML`, `CSS` e `javascript`

O começo de tudo

O precursor do R: S

- 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
- 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

Porque usar **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
- Um grande comunidade de usuários (especialmente útil para resolução de problemas)

Onde estudar fora de aula?

Livros

- **Nível *cheguei agora no rolê***: zen do R
- **Nível Iniciante:** R Tutorial na W3Schools
- **Nível Iniciante:** Hands-On Programming with R
- **Nível Intermediária:** R for Data Science
- **Nível Avançado:** Advanced R

Em pt-br

- Curso-R: material.curso-r.com

O que você fazer quando estiver em apuros?

- check a documentação do `R`:

```
help(mean)
?mean
```

- Peça ajuda a um programador mais experiente
- Consulte o pt.stackoverflow.com
- Use ferramentas de busca como o google e duckduckgo.com

```
log("G")
```

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

- Na ferramenta de busca, pesquise por `Error in log("G"): non-numeric argument to mathematical function`

Operações básicas

Soma

```
1 + 1
```

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

Substração

```
2 - 1
```

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

Divisão

```
3 / 2
```

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

Operações básicas

Potenciação

```
2^3
```

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

Resto da divisão e parte inteira da divisão

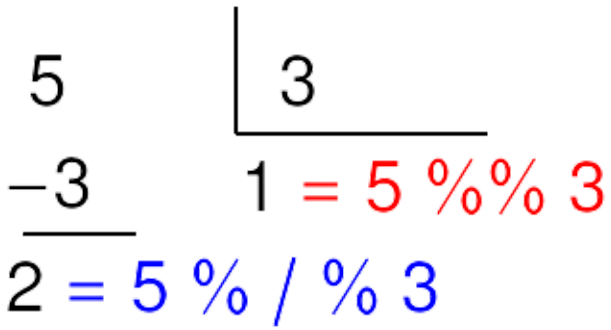
```
5 %% 3
```

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

Parte inteira da divisão

```
5 %/% 3
```

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


5
3
−3
1 = 5 %% 3
2 = 5 % / % 3

Estrutura de dados no R

- **Estrutura de dados:** atomic `vector` (a estrutura de dados mais básico no `R`), `matrix`, `array`, `list` e `data.frame` (`tibble` no `tidyverse`)
- **Tipo de dados:** caracter (`character`), número real (`double`), número inteiro (`integer`), número complexo (`complex`) e lógico (`logical`)
- **Estrutura de dados Homogênea:** `vector`, `matrix` e `array`
- **Estrutura de dados Heterôgenea:** `list` e `data.frame` (`tibble` no `tidyverse`)

Tipo de dados no R

Número inteiro

```
typeof(1L)
```

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

Número real

```
typeof(1.2)
```

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

Número complexo

```
typeof(1 + 1i)
```

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

Tipo de dados no R

Número lógico

```
typeof(TRUE)
```

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

Caracter

```
typeof("Gilberto")
```

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

Estrutura de Dados Homogênea

Vetor

- Agrupamento de valores de mesmo tipo em um único objeto
- Criação de vetores: `c(...)` e `vector('<tipo de dados>', <comprimento do vetor>)`, `vector()` é bastante usado em *laços de repetição*, que veremos na semana 4, o operador `:` e `seq(from = a, to = b, by = c)`

Vetor de caracteres

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

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

```
b <- vector("character", 3)
b
```

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

Estrutura de Dados Homogênea

Vetor de número real

```
a  <- c(0.2, 1.35)
a
```

```
## [1] 0.20 1.35
```

```
b <- vector("double", 3)
b
```

```
## [1] 0 0 0
```

```
d <- seq(from = 1, to = 3.5, by = 0.5)
d
```

```
## [1] 1.0 1.5 2.0 2.5 3.0 3.5
```

Estrutura de Dados Homogênea

Vetor de número inteiro

```
a  <- c(1L, 2L)
a
```

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

```
b <- vector("integer", 3)
b
```

```
## [1] 0 0 0
```

```
d <- 1:4
d
```

```
## [1] 1 2 3 4
```

Estrutura de Dados Homogênea

Vetor de número inteiro

```
a  <- c(TRUE, FALSE)
a
```

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

```
b <- vector("logical", 3)
b
```

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

Estrutura de Dados Homogênea

Matriz

- Agrupamento de valores de mesmo tipo em um único objeto de dimensão 2
- Criação de vetores: `matrix(..., nrow = <integer>, ncol = <integer>)` ou `diag(<vector>)`

Matriz de caracteres

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

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

Matriz de números reais

```
a <- matrix(seq(from = 0, to = 1.5, by = 0.5), nrow = 2)
a
```

```
##      [,1] [,2]
## [1,]  0.0  1.0
## [2,]  0.5  1.5
```

Estrutura de Dados Homogênea

Matriz de inteiros

```
a <- matrix(1L:4L, nrow = 2)
a
```

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

Matriz de valores lógicos

```
a <- matrix(c(TRUE, F, F, T), nrow = 2)
a
```

```
##       [,1]  [,2]
## [1,]  TRUE FALSE
## [2,] FALSE  TRUE
```

Estrutura de Dados Homogênea

Array

- Agrupamento de valores de mesmo tipo em um único objeto em duas ou mais dimensões
- Criação de vetores: `array(..., dim = <vector of integers>)`

```
dados_matriz_1 <- 10:13
dados_matriz_2  <- 14:17
resultado <- array(c(dados_matriz_1, dados_matriz_2), dim = c(2, 2, 2))
resultado
```

```
## , , 1
##
##      [,1] [,2]
## [1,]   10   12
## [2,]   11   13
##
## , , 2
##
##      [,1] [,2]
## [1,]   14   16
## [2,]   15   17
```

Estrutura de Dados Homogênea

Operações em Vetores númericos (`double`, `integer` e `complex`)

- Operações básicas (operação, substração, multiplicação e divisão) realizada em cada elemento do vetor
- *Slicing*: extrae parte de um vetor (não precisa ser vetor numérico)

Slicing

```
a <- c("a", "b", "c", "d", "e", "f", "g", "h", "i")
a[1:5] # selecionado todos os elementos entre o primeiro e o quinta
```

```
## [1] "a" "b" "c" "d" "e"
```

Adição (vetores números)

```
a <- 1:5
b <- 6:10
a + b
```

```
## [1]  7  9 11 13 15
```

Estrutura de Dados Homogênea

Substração (vetores numéricos)

```
a <- 1:5
b <- 6:10
b - a
```

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

Multiplicação (vetores numéricos)

```
a <- 1:5
b <- 6:10
b * a
```

```
## [1]  6 14 24 36 50
```

Divisão (vetores numéricos)

```
a <- 1:5
b <- 6:10
b / a
```

```
## [1] 6.000000 3.500000 2.666667 2.250000 2.000000
```

Estrutura de Dados Homogênea

Operações em Matrizes númericas (`double`, `integer` e `complex`)

- Operações básicas (operação, substração, multiplicação e divisão) realizada em cada elemento das matrizes
- Multiplicação de matrizes (vide multiplicação de matrizes), inversão de matrizes (vide inversão de matrizes), matriz transposta (vide matriz transposta), determinante (vide determinante de uma matriz) e solução de sistema de equações lineares (vide sistema de equações lineares)

Estrutura de Dados Homogênea

Soma de matrizes

```
A <- matrix(c(1, 2, 3, 4), nrow = 2)
B <- matrix(5:8, ncol = 2)
C <- A + B
C
```

```
##      [,1] [,2]
## [1,]    6   10
## [2,]    8   12
```

Soma de substração

```
A <- matrix(c(1, 2, 3, 4), nrow = 2)
B <- matrix(5:8, ncol = 2)
C <- B - A
C
```

```
##      [,1] [,2]
## [1,]    4    4
## [2,]    4    4
```

Estrutura de Dados Homogênea

Multiplicação ponto-a-ponto ou produto de Hadamard

- Para detalhes vide produto de Hadamard

```
A <- matrix(c(1, 2, 3, 4), nrow = 2); B <- matrix(5:8, ncol = 2)
C <- A * B
C
```

```
##      [,1] [,2]
## [1,]    5   21
## [2,]   12   32
```

Multiplicação de matrizes

- Para detalhes vide multiplicação de matrizes

```
C <- A %*% B
C
```

```
##      [,1] [,2]
## [1,]   23   31
## [2,]   34   46
```

Estrutura de Dados Homogênea

Matriz inversa

- Para detalhes vide matriz inversa

```
A <- matrix(1:4, ncol = 2)
B <- solve(A)
B
```

```
##      [,1] [,2]
## [1,]   -2  1.5
## [2,]    1 -0.5
```

```
A %*% B
```

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

Estrutura de Dados Homogênea

Matriz transposta

- Para detalhes vide matriz transposta

```
A <- matrix(1:4, ncol = 2)
B <- t(A)
B
```

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

Determinante de uma matriz

- Para detalhes vide determinante

```
A <- matrix(1:4, ncol = 2)
det(A)
```

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

Estrutura de Dados Homogênea

Solução de sistema de equações lineares

- Para detalhes vide sistema de equações lineares

```
b <- c(1, 2); A <- matrix(1:4, nrow = 2)
x <- solve(A, b)
x
```

```
## [1] 1 0
```

Matriz inversa generalizada

- G é a matriz inversa generalizada de A se $A \cdot G \cdot A = A$. Para detalhes vide matriz inversa generalizada

```
library(MASS) # ginv é uma função do pacote MASS
A <- matrix(c(1, 1, 2, 3), nrow = 2)
ginv(A)
```

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

Estrutura de Dados Homogênea

Outras operações com matrizes

Operador ou função	Descrição
`A %o% B`	produto diádico $A \cdot B^T$
`crossprod(A, B)`	$A \cdot B^T$
`crossprod(A)`	$A \cdot A^T$
`diag(x)`	retorna uma matrix diagonal com diagonal igual a `x` (`class(x) == 'numeric'`)
`diag(A)`	retorna um vetor com a diagona de A (`class(A) == 'matrix'`)
`diag(k)`	retorna uma matriz diagona de ordem k (`class(k) == 'numeric'`)
`rowMeans(A)`	retorna um vetor com as médias das linhas
`colMeans(A)`	retorna um vetor com as médias das colunas

Estrutura de Dados Heterogênea

Lista

- Agrupamento de valores de tipos diversos e estrutura de dados
- Criação de listas: `list(...)` e `vector("list", <comprimento da lista>)`

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

Estrutura de Dados Heterogênea

- Agrupamento de dados em tabela em que: cada coluna é uma variável; cada linha é uma observação
- Criaçcao de `tibble`: `tibble(...)` e `tribble(....)`

`tibble` (data frame)

```
library(tidyverse) # carregando o framework tidyverse
a <- tibble(variavel_1 = c(1, 2), variavel_2 = c("a", "b"))
glimpse(a)
```

```
## Rows: 2
## Columns: 2
## $ variavel_1 <dbl> 1, 2
## $ variavel_2 <chr> "a", "b"
```

```
a
```

```
## # A tibble: 2 × 2
##   variavel_1 variavel_2
##        <dbl> <chr>
## 1          1 a
## 2          2 b
```

Estrutura de Dados Heterogênea

Operações em um `tibble`

Vamos ver o uso dessas funções depois de aprender a carregar os dados no R.

Função	Descrição
`head()`	Mostra as primeiras linhas de um `tibble`
`tail()`	Mostra as últimas linhas de um `tibble`
`glimpse()`	Impressão de informações básicas dos dados
`add_case()` ou **`add_row()`**	Adiciona uma nova observação

Estrutura de Dados Heterogênea

Concatenação de listas

```
a  <- list("a", "b")
b <- list(1, 2)
d <- c(a, b)
d
```

```
## [[1]]
## [1] "a"
##
## [[2]]
## [1] "b"
##
## [[3]]
## [1] 1
##
## [[4]]
## [1] 2
```

Estrutura de Dados Heterogênea

Slicing a lista

```
d[1:2]
```

```
## [[1]]
## [1] "a"
##
## [[2]]
## [1] "b"
```

Acessando o valor de elmento em uma lista

```
d[[2]] # acessando o segundo elemento da lista d
```

```
## [1] "b"
```

Acessando o valor de elmento em uma lista pela chave

```
d <- list(chave_1 = 1, chave_2 = "docente")
d$chave_2 # retorna o valor
```

```
## [1] "docente"
```

Estrutura de Dados Heterogênea

Slicing uma lista por chaves

```
d <- list(chave_1 = 1, chave_2 = "docente", chave_3 = list("olá"))
d[c("chave_2", "chave_3")] # funciona como slicing
```

```
## $chave_2
## [1] "docente"
##
## $chave_3
## $chave_3[[1]]
## [1] "olá"
```

Enumerando chaves em um lista

```
d <- list(c(1, 2, 3), chave_1 = 1, chave_2 = "docente", chave_3 = list("olá"))
names(d)
```

```
## [1] ""         "chave_1" "chave_2" "chave_3"
```

Valores especiais em **R**

Valores especiais	Descrição	Função para identificar
`NA` (Not Available)	Valor faltante.	`is.na()`
`NaN` (Not a Number)	Resultado do cálculo indefinido.	`is.nan()`
`Inf` (Infinito)	Valor que excede o valor máximo que sua máquina aguenta.	`is.inf()`
`NULL` (Nulo)	Valor indefinido de expressões e funções (diferente de `NaN` e `NA`)	`is.null()`

Parênteses 1: guia de estilo no R

- Nome de um objeto precisa ter um *significado*. Esse significado precisa falar imediatamente o que este objeto é ou faz ~sua bisavó precisa entender o que este objeto é ou faz~
- Use a convenção do `R`:
  - Use apenas letras minúsculas, números e *underscore* (comece sempre com letras minúsculas)
  - Nomes de objetos precisam ser substantivos e precisam descrever o que este objeto é ou faz (seja conciso, direto e significativo)
  - Evite ao máximo os nomes que são usados por objetos que são *buit-in* do `R`
  - Coloque espaço depois da vírgula
  - Não coloque espaço antes nem depois de parênteses. Exceção: Coloque um espaço `()` antes e depois de `if`, `for` ou `while`, e coloque um espaço depois de `()`.
  - Coloque espaço entre operadores básicos: `+`, `-`, `*`, `==` e outros. Exceção: `^`.
- Para mais detalhes, consulte: guia de estilo do `tidyverse`

Parênteses 2: estrutura de diretórios

- Mantenha uma estrutura consistente de diretórios em seus projetos.
- Eu uso a seguinte estrutura:
  - `data`: diretório para armazenar seus conjuntos de dados
    - `raw`: dados brutos
    - `processed`: dados processados
  - `scripts`: código fonte do seu projeto
  - `figures`: figuras criadas no seu projeto
  - `output`: outros arquivos que não são figuras
  - `previous`: arquivos da versão anterior do projeto
  - `notes`: notas de reuniões e afins
  - `relatorio` (ou `artigos`): documento final de seu projeto
  - `documents`: livros, artigos e qualquer coisa que são referências em seu projeto
- Para mais detalhes, consulte esse guia do curso-r: diretórios e `.Rproj`

Lendo dados no **R**

Leitura de arquivos no formato `xlsx` ou `xls`

- **Pacote:** `readxl` do `tidyverse` (instale com o comando `install.packages('readxl')`)
- Parêmetros das funções `read_xls` (para ler arquivos `.xls`) e `read_xlsx` (para ler arquivos `.xlsx`):
  - `path`: caminho até o arquivo
  - `sheet`: especifica a planilha do arquivo que será lida
  - `range`: especifica uma área de uma planilha para leitura. Por exemplo: `B3:E15`
  - `col_names`: Argumento lógico com valor padrão igual a `TRUE`. Indica se a primeira linha tem o nome das variáveis
- Para mais detalhes, consulte a documentação oficial do *tidyverse*: documentação de `read_xl`

Lendo dados no **R**

Leitura de arquivos no formato `xlsx` ou `xls`

```
library(tidyverse)
library(readxl)
dados_iris <- read_xlsx("data/raw/iris.xlsx")

head(dados_iris, n = 4)
```

```
## # A tibble: 4 × 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <chr>
## 1          5.1         3.5          1.4         0.2 setosa
## 2          4.9         3            1.4         0.2 setosa
## 3          4.7         3.2          1.3         0.2 setosa
## 4          4.6         3.1          1.5         0.2 setosa
```

Lendo dados no **R**

Leitura de arquivos no formato `csv`

- **Pacote:** `readr` do `tidyverse` (instale com o comando `install.packages('readr')`)
- Parêmetros das funções `read_csv` e `read_csv2`:
  - `path`: caminho até o arquivo

Padrão imperial inglês versus o resto do planeta

- Se você mora ou está em países que usam o padrão *imperial inglês*:
  - colunas separadas por `,`
  - casa decimal indicada por `.`
- Se você mora ou estão em países que usam o sistema métrico:
  - colunas separadas por `;`
  - casa decimal por `,`

Preste atenção em como o seus dados estão armazenados!

Para mais detalhes, consulte a documentação oficial do *tidyverse*: documentação de `read_r`

Lendo dados no R

Leitura de arquivos no formato csv

```
library(tidyverse)
library(readr)
dados_mtcars <- read_csv2("data/raw/mtcars.csv")
```

```
## ℹ Using "','" as decimal and "'.'" as grouping mark. Use `read_delim()` for more control.
```

```
##
## ── Column specification ────────────────────────────────────────────────────────
## cols(
##   mpg = col_double(),
##   cyl = col_double(),
##   disp = col_double(),
##   hp = col_double(),
##   drat = col_double(),
##   wt = col_double(),
##   qsec = col_double(),
##   vs = col_double(),
##   am = col_double(),
##   gear = col_double(),
##   carb = col_double()
## )
```

Lendo dados no **R**

Leitura de arquivos no formato `ods`

- **Pacote:** `readODS` (instale com o comando `install.packages('readODS')`)
- Parêmetros das funções `read_ods`:
- `path`: caminho até o arquivo
  - `sheet`: especifica a planilha do arquivo que será lida
  - `range`: especifica uma área de uma planilha para leitura. Por exemplo: `B3:E15`
  - `col_names`: Argumento lógico com valor padrão igual a `TRUE`. Indica se a primeira linha tem o nome das variáveis
- Para mais detalhes, consulte a documentação do *readODS*: documentação de `readODS`

Lendo dados no R

Leitura de arquivos no formato ods

```
library(tidyverse)
library(readODS)
dados_toothgrowth <- read_ods("data/raw/ToothGrowth.ods")

glimpse(dados_toothgrowth)
```

```
## Rows: 60
## Columns: 3
## $ len  <dbl> 4.2, 11.5, 7.3, 5.8, 6.4, 10.0, 11.2, 11.2, 5.2, 7.0, 16.5, 16.5, 15.2, 17.3, 2
## $ supp <chr> "VC", "VC", "VC", "VC", "VC", "VC", "VC", "VC", "VC", "VC", "VC", "VC", "VC", "
## $ dose <dbl> 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
```

Salvando dados no R

Salvar no formato `.csv` (sistema métrico)

```
library(readr)
write_csv2(dados_toothgrowth, file = "data/processed/dados_csv2.csv")
```

Salvar no formato `.xlsx`

```
library(writexl)
write_xlsx(dados_toothgrowth, path = "data/processed/dados_xlsx.xlsx")
```

Salvar no formato ods

```
library(readODS)
write_ods(dados_toothgrowth, path = "data/processed/dados_ods.ods")
```