Estatística básica usando o R: bem-vinde ao tidyverse

Parte 1

Introdução à linguagem R

Gilberto e Carolina

Sobre o curso

Preparando o ambiente

- Use, pelo menos, a versão `4.1` da linguagem `R`: cran.r-project.org
- **IDE** recomendadas: Rstudio e VSCode
  - Caso você queria usar o VSCode, instale a extensão da linguagem `R`: r-pack
- Neste curso usaremos o *framework* `tidyverse`
  - Instale usando o seguinte comando `install.packages('tidyverse')`
- Outras linguagens populares para análise de dados:
  - `python`: linguagem interpretada de próposito geral, contemporânea do `R`, simples e fácil de aprender. Podemos realizar análise de dados usando os pacotes `pandas` e `numpy`, mas, ao contrário do `R`, `python` não foi projetada e criada para análise de dados
  - `julia`: linguagem interpretada para análise de dados projeto pelo MIT, lançada em 2012, promete simplicidade e velocidade. Comunidade de usuários pequena, mas crescente.

A linguagem R

Uma introdução breve

O começo de tudo

S, o precursor da linguagem R:

- R é uma linguagem derivada do S
- S foi desenvolvido em `fortran` for **John Chambers** em 1976 no ***Bell Labs***
- S foi desenvolvido para a análise de dados
- **Filosofia do S:** permitir que usuários possam analisar dados usando estatística com pouco conhecimento de programação

Origem da linguagem R

- Em 1991, **Ross Ihaka** e **Robert Gentleman** criaram o R na Nova Zelândia
- Em 1996, **Ross** e **Robert** liberam o R sob a licença *GNU General License*
- Em 1997, **The Core Group** é criado para melhorar e controlar o código fonte do R

Motivos para 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 aqui*:** zen do R
- **Nível Iniciante:** R Tutorial na W3Schools
- **Nível Iniciante:** Hands-On Programming with R
- **Nível Intermediário:** R for Data Science
- **Nível Avançado:** Advanced R

Em pt-br

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

O que você pode 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

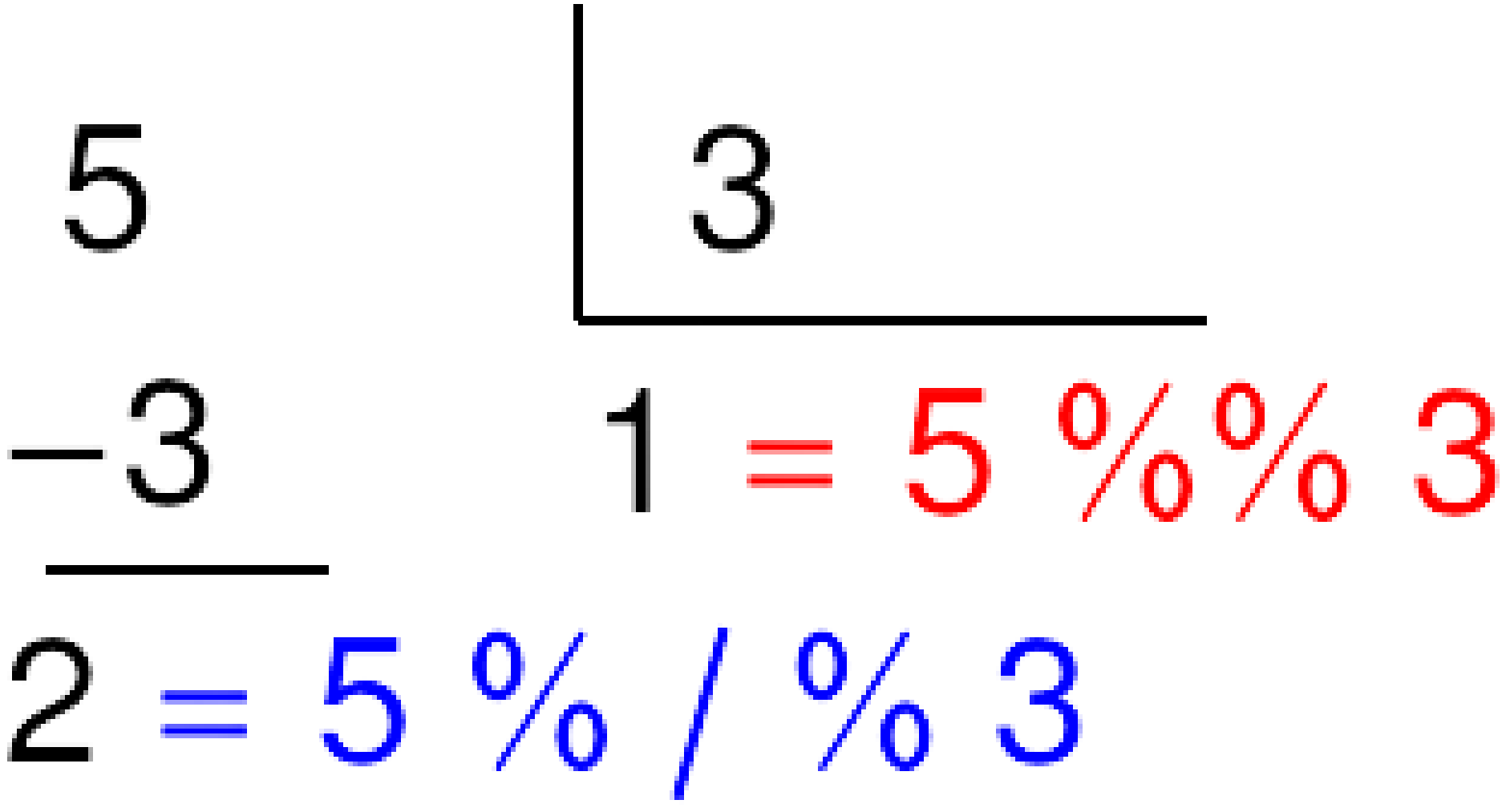
```
5 %% 3
```

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

Parte inteira da divisão

```
5 %/% 3
```

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


5
3
−3
1 = 5 %% 3
2 = 5 % / % 3

Estrutura de dados em 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 em R

Valores numéricos

Número inteiro

```
class(1L)
```

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

Número real

```
class(1.2)
```

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

Número complexo

```
class(1 + 1i)
```

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

Valores lógicos

Valor lógico

```
class(TRUE)
```

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


Valoes de texto (caracter ou string)

```
class("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] "" "" ""
```

Vetor de números reais

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

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

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

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

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

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


Vetor de números inteiros

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

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

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

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

Vetor de valores lógicos

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

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

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

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

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
```


Matriz

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
```

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
```


Operações com vetores

- Operações básicas (operação, substração, multiplicação e divisão) realizada em cada elemento do vetor
- *Slicing*: extrai 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

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

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

Operações com vetores

subtração

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

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

multiplicação

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

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

divisão

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

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

Operações com matrizes

- 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)

soma

```
matriz_a <- matrix(c(1, 2, 3, 4), nrow = 2)
matriz_b <- matrix(5:8, ncol = 2)
matriz_c <- matriz_a + matriz_b
matriz_c
```

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

Operações com matrizes

subtração

```
matriz_a <- matrix(c(1, 2, 3, 4), nrow = 2)
matriz_b <- matrix(5:8, ncol = 2)
matriz_c <- matriz_a - matriz_b
matriz_c
```

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

produto de Hadamard

- Para detalhes vide produto de Hadamard

```
matriz_a <- matrix(c(1, 2, 3, 4), nrow = 2)
matriz_b <- matrix(5:8, ncol = 2)
matriz_c <- matriz_a * matriz_b
matriz_c
```

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

Operações com matrizes

multiplicação de matrizes

- Para detalhes vide multiplicação de matrizes

```
matriz_c <- matriz_a %*% matriz_b
matriz_c
```

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

matriz inversa

- Para detalhes vide matriz inversa

```
matriz_a <- matrix(1:4, ncol = 2)
matriz_b <- solve(matriz_a)
matriz_b
```

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

```
matriz_a %*% matriz_b
```

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


Operações com matrizes

matriz transposta

- Para detalhes vide matriz transposta

```
matriz_a <- matrix(1:4, ncol = 2)
matriz_b <- t(matriz_a)
matriz_b
```

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

determinante

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

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

Operações com matrizes

solução de sistema de equações lineares

- Para detalhes vide sistema de equações lineares

```
b <- c(1, 2)
matriz_a <- matrix(1:4, nrow = 2)
x <- solve(matriz_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
matriz_a <- matrix(c(1, 1, 2, 3), nrow = 2)
ginv(matriz_a)
```

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

Operações com matrizes

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'`)

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

Tidy data

- Agrupamento de dados em tabela em que cada coluna é uma variável e cada linha é uma observação
- Criação 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"
```

Estrutura de Dados Heterogênea

operações básicas 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
```

Slicing a lista

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

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


Estrutura de Dados Heterogênea

acessando o valor de um elemento de 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 usando 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* (precisa falar imediatamente o que este objeto é ou faz)
- Use a convenção do `RStudio`:
  - Use apenas letras minúsculas, números e *underscore* (comece sempre com letras minúsculas)
  - Nomes de objetos precisam ser substantivos
  - Evite ao máximo os nomes que são usados por objetos que são *buit-in* do R
  - Coloque espaço depois da vírgula

Parênteses 1: guia de estilo no R

- Use a convenção do `RStudio`:
  - Não coloque espaço antes nem depois de parênteses. Exceções:
    - Coloque um espaço `( )` antes e depois de `if`, `for` ou `while`
    - Coloque um espaço depois de `( )` para funções.
  - Coloque espaço entre operadores básicos: `+`, `-`, `*`, `==` e outros. Exceção: `^`.

Para mais detalhes, consulte: guia de estilo do `tidyverse`.

Existem outros estilos (padrões) de codificação para a linguagem `R`, por exemplo guia de estilo do `google`. Mas o estilo mais usado e famoso é o *estilo de codificação do RStudio*.

Esolha um estilo e seja consistente! ~~Use apenas um estilo no se código~~

Parênteses 2: estrutura de diretórios

- `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`

Dados externos no R

Arquivos `.xlsx`, `.csv`, `.ods` e `.txt`

Arquivos `.xlsx` no `R`

- **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`

Arquivos `.xlsx`

```
library(readxl)
library(tidyverse)

tb_iris <- read_xlsx("data/raw/dados_iris.xlsx")

glimpse(tb_iris)
```

```
## Rows: 150
## Columns: 5
## $ comprimento_sepala <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4,
4.9, 5…
## $ largura_sepala     <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9,
3.1, 3…
## $ comprimento_petala <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4,
1.5, 1…
## $ largura_petala     <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2,
0.1, 0…
## $ especies           <chr> "setosa", "setosa", "setosa", "setosa",
"setosa", "…
```

Arquivos `.csv`

- **Pacote:** `readr` do `tidyverse`
- Parêmetros das funções `read_csv` e `read_csv2`: `path` - caminho até o arquivo

Padrão métrico versus padrão imperial inglês

- Se você mora ou está em um país que usa padrão *imperial inglês*:
  - colunas separadas por `,` e casa decimal indicada por `.`
- Se você mora ou está em um país que usa o sistema métrico:
  - colunas separadas por `;` e 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`

Arquivos `.csv`

```r
library(tidyverse)

dados_mtcars <- read_csv2("data/raw/mtcarros.csv")

glimpse(dados_mtcars)
```

```
## Rows: 32
## Columns: 11
## $ milhas_por_galao <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8,…
## $ cilindros        <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4,…
## $ cilindrada       <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.…
## $ cavalos_forca    <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 1…
## $ eixo             <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92,…
## $ peso             <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.19…
## $ velocidade       <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.0…
## $ forma            <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1,…
## $ transmissao      <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,…
## $ marchas          <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4,…
## $ carburadores     <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1,…
```


Arquivos *formato com comprimento fixo (fixed width format: fwf)*

- **Pacote:** `readr` do `tidyverse`
- Parêmetros das funções `read_fwf`:
  - `file`: caminho até o arquivo
  - `col_positions`: use a função `fwf_widths()` fornece as delimitações e os nomes das colunas
  - `col_types`: texto especificando o tipo de cada coluna: `c` para caracter, `d` para número real, `i` para inteiro, `l` para lógico, `D` para data e `T` para data e horário.

- **Variáveis:** `ano`, `id_equipe`, `id_liga`, `id_jogador` e `salario`
- **Larguras das variáveis:** 4, 3, 2, 9 e 8
- **Tipo de dados de cada variáveL:** `i`, `c`, `c`, `c` e `d`

```
df_salarios <- read_fwf(
  "data/raw/salarios.txt",
  col_positions = fwf_widths(
    c(4, 3, 2, 9, 8),
    col_names = c("ano", "id_equipe", "id_liga", "id_jogador", "salario")
  ),
  col_types = "icccd"
)
glimpse(df_salarios)
```

```
## Rows: 26,428
## Columns: 5
## $ ano        <int> 1985, 1985, 1985, 1985, 1985, 1985, 1985, 1985, 1985, 1985,…
## $ id_equipe  <chr> "ATL", "ATL", "ATL", "ATL", "ATL", "ATL", "ATL", "ATL", "AT…
## $ id_liga    <chr> "NL", "NL", "NL", "NL", "NL", "NL", "NL", "NL", "NL", "NL",…
## $ id_jogador <chr> "barkele01", "bedrost01", "benedbr01", "campri01", "ceronri…
## $ salario    <dbl> 870000, 550000, 545000, 633333, 625000, 800000, 150000, 483…
```

Arquivos `.ods`

- **Pacote:** `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`

Arquivos .ods

```
library(readODS)
library(tidyverse)
df_star_wars <- read_ods("data/raw/dados_star_wars.ods")

glimpse(df_star_wars)
```

```
## Rows: 87
## Columns: 14
## $ nome            <chr> "Luke Skywalker", "C-3PO", "R2-D2", "Darth Vader", "Le…
## $ altura          <dbl> 172, 167, 96, 202, 150, 178, 165, 97, 183, 182, 188, 1…
## $ massa           <dbl> 77.0, 75.0, 32.0, 136.0, 49.0, 120.0, 75.0, 32.0, 84.0…
## $ cor_do_cabelo   <chr> "Loiro", NA, NA, "Nenhum", "Castanho", "Castanho, Cinz…
## $ cor_da_pele     <chr> "Branca clara", "Ouro", "Branca, Azul", "Branca", "Cla…
## $ cor_dos_olhos   <chr> "Azul", "Amarelo", "Vermelho", "Amarelo", "Castanho", …
## $ ano_nascimento  <dbl> 19.0, 112.0, 33.0, 41.9, 19.0, 52.0, 47.0, NA, 24.0, 5…
## $ sexo_biologico  <chr> "Macho", "Nenhum", "Nenhum", "Macho", "Fêmea", "Macho"…
## $ genero          <chr> "Masculino", "Masculino", "Masculino", "Masculino", "F…
## $ planeta_natal   <chr> "Tatooine", "Tatooine", "Naboo", "Tatooine", "Alderaan…
## $ especie         <chr> "Humano", "Droide", "Droide", "Humano", "Humano", "Hum…
## $ filmes          <chr> "c(\"The Empire Strikes Back\", \"Revenge of the Sith\…
## $ veiculos        <chr> "c(\"Snowspeeder\", \"Imperial Speeder Bike\")", "char…
## $ naves_espaciais <chr> "c(\"X-wing\", \"Imperial shuttle\")", "character(0)",…
```

Salvando dados no R

Salvar no formato `.csv` sistema métrico

Pacotes: `readr`

```
library(readr)
df_guerra_estrelas <- dados_starwars |>
  select(nome, altura, massa, genero)
write_csv2(df_guerra_estrelas, file = "data/processed/df_guerra_estrelas.csv")
```


Salvar no formato `.xlsx`

Pacotes: `writexl`

```
library(writexl)
df_guerra_estrelas <- dados_starwars |>
  select(nome, altura, massa, genero)
write_xlsx(
  df_guerra_estrelas,
  path = "data/processed/df_guerra_estrelas.xlsx",
)
```


Salvar no formato `.xlsx`

Pacote: `readODS`

library(readODS) df_guerra_estrelas <- dados_starwars |> select(nome, altura, massa, genero)

write_ods(df_guerra_estrelas, path = “data/processed/df_guerra_estrelas.ods”) ```## Número de faltas

- Número de faltas até o dia 04/09/2022