Introdução ao R

Maurício Bueno

2022-09-03

Table of Contents

# Introdução

O R é uma **linguagem** e um **ambiente** de programação dentro do qual se pode fazer uma infinidade de operações estatísticas e de processamento de dados. Foi concebido para ser colaborativo e gratuito. Qualquer pessoa pode criar uma função, ou um conjunto de funções (pacote) e disponibilizar a outros usuários… **gratuitamente**! Visite o [site](https://cran.r-project.org/) do R e baixe o programa para instalação em seu computador.

Em associação ao R, costuma-se usar o RStudio (baixe e instale clicando [aqui](https://www.rstudio.com/)), que facilita o uso do R ao disponibilizar um conjunto de quatro janelas que funcionam para as seguintes funções:

1. digitação de texto (superior, à esquerda),
2. display de resultados (console) (inferior, à esquerda),
3. visualizador de objetos (arquivos) gerados ao realizar um trabalho (superior, à direita),
4. informações gerais: arquivos, plotagens, pacotes instalados, help, e visualizador de imagens (inferior, à direita).

Para a digitação dos comandos, deve-se abrir um novo **Script** ou um **RMarkdown**.

Abre-se um novo **script** com a seguinte linha de comando: File + New File + R Script, cujo atalho é CTRL+SHIFT+N.

Abre-se um novo **RMarkdown** com a seguinte linha de comando: File + New File + R Markdown Fazendo isso, abrirá uma aba no quadrante superior esquerdo.

O **RMarkdown** aceita linhas de texto e linhas de código, mas as linhas de código devem ser escritas dentro de um **chunk**, que se abre com o comando CTRL+ALT+i.

Já o **Script** é um grande **chunk** e não aceita linhas de texto. Por isso, preferimos usar o RMarkdown.

Seja Script ou RMarkdown, você pode abrir quantas abas desejar.

É possível rodar um comando no R de duas formas:

* Selecionando-o e clicando em “Run” na aba superior da área de digitação.
* Colocar o cursor em qualquer ponto da linha de comando e apertar CTRL+ENTER.

## Instalação e ativação de pacotes

Algumas funções não precisam da instalação de pacotes, pois já vêm com o R. São funções básicas (base), como operações matemáticas, plotagem, estatísticas básicas, entre outras. Outras funções estão em pacotes que precisam ser **instalados**, por meio do comando install.packages(), e **ativados**, por meio do comando library(). Então, vamos instalar alguns pacotes que iremos utilizar durante o curso.

# Instalação dos pacotes  
  
if(!require(readxl)) install.packages("readxl",repos = "http://cran.us.r-project.org")  
  
if(!require(tidyverse)) install.packages("tidyverse",repos = "http://cran.us.r-project.org")  
if(!require(knitr)) install.packages("knitr",repos = "http://cran.us.r-project.org")  
if(!require(kableExtra)) install.packages("kableExtra", repos = "http://cran.us.r-project.org")  
if(!require(expss)) install.packages("expss", repos = "http://cran.us.r-project.org")  
  
if(!require(tibble)) install.packages("tibble")  
# Ativacao dos pacotes ====  
  
library(readxl)  
library(tidyverse)  
library(knitr)  
library(expss)

Todo pacote do R tem uma documentação, com informações sobre as funções que o pacote contém. Uma forma de encontrar essa documentação é no site <rdocumentation.org>.

As funções contidas dentro dos pacotes ser acessadas digitando-se um ponto de interrogação (?) antes do nome da função. Por exemplo, se quisermos a documentação da função **hist** da base do R, é só rodar o comando:

A documentação aparecerá na aba **Help** do quadrante de visualizações de informações do RStudio.

## Operadores básicos

Existem também alguns operadores importantes no R, que você deve conhecer:  
Um deles é a seta para a esquerda, formada com o sinal de *menor* e o traço de *menos*: <-.

Atalho para esse operador: pressione simultaneamente as teclas ALT e -

Esse operador indica que o resultado de uma operação será salva em um **objeto**.

Por exemplo: o comando abaixo diz “*faça a soma de 3 com 4 e guarde o resultado num objeto chamado* ***soma****”*. Esse objeto (soma) aparecerá no environment (quadrante superior direito do RStudio).

soma <- 3 + 4

O comando “#” permite que você faça comentários, sem que o R o entenda como comando. Por exemplo:

soma <- 3 + 4 # Esse comando calcula a soma de 3 + 4 e salva no objeto "soma"

O texto que aparece após a #, em outra cor (verde, azul, etc.), é ignorado pelo R. Esses são os comandos mais básicos. Há muitos outros atalhos…

| atalho | o.que.faz |
| --- | --- |
| CTRL + ENTER | Executa a linha selecionada |
| CTRL + SHIFT + C | Comenta e descomenta a linha |
| CTRL + 1 | Passa o cursor para o script |
| CTRL + 2 | Passa o cursor para o console |
| SETA PARA CIMA (no console) | acessa o histórico de comandos anteriores |
| CTRL + ALT + SETA PARA ESQUERDA OU DIREITA | Navega entre as abas de script abertas |
| CTRL + SHIFT + ENTER | Executa o Script inteiro |
| CTRL + S | Salva o Script |
| CTRL + L | Limpa o console |
| ALT + SHIFT + K | Ver a lista de atalhos |

## Definir diretório de trabalho

Para facilitar o trabalho com salvamento e recuperação de arquivos, é recomendável que se defina um diretório de trabalho, onde todos os arquivos referentes à análise de dados serão salvos.

# para verificar o diretório atual, execute o comando:   
getwd()

## [1] "C:/Users/jmhbu/OneDrive/Documentos/R/IntroR\_aula"

# OBS.: se for necessário, navegue até o local desejado utilizando a aba "files"  
  
# defina sua pasta de trabalho utilizando a função:   
setwd("C:/Users/jmhbu/OneDrive/Documentos/R/R\_basico/Intro\_R")

## Principais operadores

O R pode ser usado como calculadora…

A tabela a seguir apresenta os principais operadores que usamos nos códigos em R.

| Operador | Descrição |
| --- | --- |
| + | Adição |
| - | Subtração |
| \* | Multiplicação |
| / | Divisão |
| : | Sequência |
| ^ | Exponencial |
| sqrt | Raiz |
| == | Igualdade |
| > | Maior que |
| < | Menor que |
| <= | Menor ou igual |
| >= | Maior ou igual |
| ! | Não |
| & | E |
| | | Ou |

Exercitando…

# operações matemáticas  
  
3+4

## [1] 7

5-2

## [1] 3

4\*2

## [1] 8

9/3

## [1] 3

sqrt(9)

## [1] 3

2^3

## [1] 8

1:10

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

10:1

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

#3 = 3  
3 == 3

## [1] TRUE

3 > 2

## [1] TRUE

3 < 2

## [1] FALSE

3 >= 2

## [1] TRUE

3 >= 3

## [1] TRUE

3 <= 2:5

## [1] FALSE TRUE TRUE TRUE

3 + 4 >= 14/2

## [1] TRUE

!3 == 5

## [1] TRUE

1 < 2 & 2 < 3

## [1] TRUE

1 == 2 | 2 > 3

## [1] FALSE

## Operações lógicas

Algumas palavras são reservadas no R para operações lógicas. As principais são TRUE, FALSE e NA.

| Operadores.lógicos | Significados |
| --- | --- |
| NA | Not available - dado faltante/indiponível |
| NaN | Not a number - indefinições matemáticas como 0/0, log(-1) |
| Inf | Infinito - número muito grande, como 1/0 e 10^310 |
| NULL | Representa ausência de objeto |
| TRUE | Condição é verdadeira |
| FALSE | Condição é falsa |

## Criando objetos

Os objetos que podem ser criados no R são:

* listas
* vetores
* matrizes
* array
* dataframes

Vamos definir os que mais usamos em nossas análises.

### Lista

**Lista**: é uma coleção de informações.

minha\_lista <- list("s01","masculino",23,"ensino médio")  
class(minha\_lista)

## [1] "list"

# minha\_lista é um objeto do tipo lista, que guarda informações de diferentes tipos: id, sexo, idade, escolaridade  
# essas informações foram salvas num objeto do tipo lista, que pode ser visto no environment  
# informações não-numéricas aparecem entre "aspas" para o R reconhecer cada elemento dentro do objeto  
# as informações dessa lista poderiam ser de um sujeito num df, cada informação é de uma natureza  
# o comando "c", significa "concatenar". É usado para inserir mais de uma informação.

### Vetor

**Vetor**: é uma coleção de informações ou elementos **da mesma natureza**. Por exemplo, as variáveis que costumamos colocar nas colunas dos nossos bancos de dados são **vetores**.

meu\_vetor <- c("masculino","feminino","feminino","masculino","feminino")  
is.list(meu\_vetor)

## [1] FALSE

is.vector(meu\_vetor)

## [1] TRUE

is.data.frame(meu\_vetor)

## [1] FALSE

# meu\_vetor é um objeto do tipo vetor porque contem informações da mesma natureza. Poderiam ser informações sobre o sexo dos sujeitos em um banco de dados  
  
vetor1 <- c(1, 5, 3, -10)  
vetor2 <- c("a", "b", "c")  
  
class(vetor1)

## [1] "numeric"

class(vetor2)

## [1] "character"

# Se tentarmos misturar duas classes, o R vai apresentar um comportamento conhecido como coerção. Ele vai impor uma das classes aos objetos. Por exemplo:  
  
vetor <- c(1, 2, "a")  
  
vetor

## [1] "1" "2" "a"

class(vetor)

## [1] "character"

### Dataframe

**Dataframe**: é um conjunto de vetores. Nossos bancos de dados, no R, recebem o nome de dataframes.

# criar vetor sexo, em que 1 é masculino e 2 é feminino.  
sexo <- c(1,2,1,1,2)  
is.vector(sexo)

## [1] TRUE

# pode-se criar um vetor semelhante, mas com letras  
# quando uma informação textual for inserida num objeto, ela tem que ir entre aspas.  
sexo\_cod <- c("m","f","m","m","f") # vetor com informações sobre sexo  
  
# uma alternativa a isso é transformar a variável sexo em uma variável fator  
  
sexo <- factor(sexo, levels = c(1,2), labels = c("masculino", "feminino"))  
  
idade <- c(25,32,78,12,NA) # vetor com informações de idade  
  
# A partir dos dois vetores anteriores é possível criar um dataframe (conjunto de vetores)  
df <- data.frame(sexo\_cod,idade)  
  
# acrescentando a variável escolaridade  
escolaridade <- c("superior","medio","fundamental","fundamental","medio")  
  
cbind(df,escolaridade) # cbind() é uma função da base que anexa colunas.

## sexo\_cod idade escolaridade  
## 1 m 25 superior  
## 2 f 32 medio  
## 3 m 78 fundamental  
## 4 m 12 fundamental  
## 5 f NA medio

df <- cbind(df,escolaridade)   
  
# acrescentando a variável id  
id <- 1:5  
df <- cbind(id,df)  
  
# Se quiser inserir um "s" antes do número de cada sujeito, podemos usar a função paste(), que concatena duas partes da informação que constar em uma coluna. Essa função diz: concatene um s com valores de 1 a 5, separados por nada.  
  
paste("s",1:5,sep="\_")

## [1] "s\_1" "s\_2" "s\_3" "s\_4" "s\_5"

id <- paste("s",1:5,sep="")  
  
# uma forma de salvar os resultados em uma nova variável diretamente no banco de dados é usando a função $ (cifrão). Quando digitamos o $ em seguida ao nome do objeto (dataframe), o R abre, automaticamente, abre uma janela de opções com as variáveis que existem no dataframe. Ao selecionar "id", por exemplo, os novos dados serão salvos SOBRE os dados existentes. Caso se queira CRIAR uma nova variável, basta digitar o nome de uma variável inexistente no dataframe.  
  
df$id <- paste("s",1:5,sep="")  
  
# se quiser inserir um novo sujeito no dataframe  
  
suj6 <- list(id="s6",sexo\_cod="m",idade=34,escolaridade="superior")  
suj6 <- list("s6","m",34,"superior")  
  
  
rbind(df,suj6)

## id sexo\_cod idade escolaridade  
## 1 s1 m 25 superior  
## 2 s2 f 32 medio  
## 3 s3 m 78 fundamental  
## 4 s4 m 12 fundamental  
## 5 s5 f NA medio  
## 6 s6 m 34 superior

df <- rbind(df,suj6)  
df

## id sexo\_cod idade escolaridade  
## 1 s1 m 25 superior  
## 2 s2 f 32 medio  
## 3 s3 m 78 fundamental  
## 4 s4 m 12 fundamental  
## 5 s5 f NA medio  
## 6 s6 m 34 superior

# inserir a variável renda  
# seq(from,to,by,lenth)  
  
df$renda <- seq(1000, by=500,length=6)  
# ou  
#renda <- seq(1000, by=500,length=6)  
#df <- cbind(df,renda)  
  
# inserir uma variável "bonus", referente a um abono de 500 reais a cada sujeito.  
# usamos a função rep(), que é semelhante à variável seq(), sem o argumento "by", porque a informação é sempre a mesma  
df$bonus <- rep(500,6)  
  
# ou  
#bonus <- rep(500,6)  
#df <- cbind(df,bonus)  
  
# calcular "renda total", que é a soma das variáveis renda e bonus  
df$"renda total" <- df$renda + df$bonus  
  
# inserir a variável "Estado Civil"  
df$"estado civil" <- c("casado/a","solteiro/a","viúvo/a","solteiro/a","casado/a","casado/a")  
df

## id sexo\_cod idade escolaridade renda bonus renda total estado civil  
## 1 s1 m 25 superior 1000 500 1500 casado/a  
## 2 s2 f 32 medio 1500 500 2000 solteiro/a  
## 3 s3 m 78 fundamental 2000 500 2500 viúvo/a  
## 4 s4 m 12 fundamental 2500 500 3000 solteiro/a  
## 5 s5 f NA medio 3000 500 3500 casado/a  
## 6 s6 m 34 superior 3500 500 4000 casado/a

# Funções para visualização de dados

names(df)

## [1] "id" "sexo\_cod" "idade" "escolaridade" "renda"   
## [6] "bonus" "renda total" "estado civil"

str(df)

## 'data.frame': 6 obs. of 8 variables:  
## $ id : chr "s1" "s2" "s3" "s4" ...  
## $ sexo\_cod : chr "m" "f" "m" "m" ...  
## $ idade : num 25 32 78 12 NA 34  
## $ escolaridade: chr "superior" "medio" "fundamental" "fundamental" ...  
## $ renda : num 1000 1500 2000 2500 3000 3500  
## $ bonus : num 500 500 500 500 500 500  
## $ renda total : num 1500 2000 2500 3000 3500 4000  
## $ estado civil: chr "casado/a" "solteiro/a" "viúvo/a" "solteiro/a" ...

glimpse(df)

## Rows: 6  
## Columns: 8  
## $ id <chr> "s1", "s2", "s3", "s4", "s5", "s6"  
## $ sexo\_cod <chr> "m", "f", "m", "m", "f", "m"  
## $ idade <dbl> 25, 32, 78, 12, NA, 34  
## $ escolaridade <chr> "superior", "medio", "fundamental", "fundamental", "med~  
## $ renda <dbl> 1000, 1500, 2000, 2500, 3000, 3500  
## $ bonus <dbl> 500, 500, 500, 500, 500, 500  
## $ `renda total` <dbl> 1500, 2000, 2500, 3000, 3500, 4000  
## $ `estado civil` <chr> "casado/a", "solteiro/a", "viúvo/a", "solteiro/a", "cas~

head(df)

## id sexo\_cod idade escolaridade renda bonus renda total estado civil  
## 1 s1 m 25 superior 1000 500 1500 casado/a  
## 2 s2 f 32 medio 1500 500 2000 solteiro/a  
## 3 s3 m 78 fundamental 2000 500 2500 viúvo/a  
## 4 s4 m 12 fundamental 2500 500 3000 solteiro/a  
## 5 s5 f NA medio 3000 500 3500 casado/a  
## 6 s6 m 34 superior 3500 500 4000 casado/a

ncol(df)

## [1] 8

nrow(df)

## [1] 6

view(df)  
unique(df$"Estado Civil")

## NULL

unique(df$idade)

## [1] 25 32 78 12 NA 34

df\_copia <- df  
df <- df\_copia

# Exercícios 1

1. Pense em um número de 1 a 10. Verifique, usando o R, se esse número está entre 3 e 8 (incluindo o 3 e o 8).

Verifique, usando o R, se esse número **não está** entre 3 e 8 (excluindo o 3 e o 8).

Multiplique o número que você escolheu por 9. Pegue o resultado dessa multiplicação e some os algarismos (por exemplo, se deu 16, a soma dos algarismos é 1 + 6 = 7). Subtraia 5 desse resultado. Quanto deu?

ok, agora transforme esse resultado na letra correspondente do abecedário (por exemplo, a=1, b=2…). Pense num país que comece com essa letra! Agora, pegue a 5ª letra desse país e pense num animal que comece com essa letra. **Veja a resposta no final.**

1. Os quatro candidatos à presidência da república mais bem colocados nas pesquisas eleitorais são Lula (homem, 76 anos, Partido dos Trabalhadores), Bolsonaro (homem, 67 anos, Partido Liberal), Ciro Gomes (homem, 64 anos, Partido Democrático Trabalhista) e Simone Tebet (mulher, 52 anos, Movimento Democrático Brasileiro). Lula e Bolsonaro já exerceram a presiência da república, enquanto Ciro e Simone ainda não. Com essas informações, monte um dataframe chamdo brasil2022 contendo cinco colunas: id, nome, sexo, idade, partido e se exerceu a presidência anteriormente (sim ou não).
2. Quais são as classes das variáveis desse dataframe?
3. Acrescente ao dataframe brasil2022 uma linha com as informações do candidato Sílvio Santos, que é homem, tem 91 anos, nunca foi presidente anteriormente e não se tem informação sobre seu partido de afiliação.
4. Transforme as variáveis sexo e se já foi presidente em fatores.
5. Se sua resposta foi macaco, você errou, porque na Dinamarca não tem macacos!!! Mas parabéns, você realizou as contas corretamente no R.

# Funções para manipulação de dados

Muitas vezes, precisamos manipular os dados de um dataframe, seja para renomear colunas, selecionar linhas ou colunas, adicionar variáveis ou sujeitos, entre outras. Vejamos algumas delas.

## Alteração do nome de variáveis

# Alterar o nome da variável "sexo\_cod"  
saveRDS("df","df.rds")  
  
df <- rename(df,sexo = "sexo\_cod")  
  
# Para se alterar o nome de muitas variáveis, por exemplo, substituindo os espaços   
# em branco (" ") por underline ("\_"), pode-se usar a função gsub, da base do R.  
  
names(df) <- gsub(" ","\_",names(df))  
  
# essa função diz: vá nos nomes das variáveis de df e sibstitua os espaços em branco (" ") por underline ("\_")

## Selecionar linhas e colunas

# Selecionar linhas e colunas: dataframe[linhas,colunas]  
  
df[1:3, ] # seleção das três primeiras linhas

## id sexo idade escolaridade renda bonus renda\_total estado\_civil  
## 1 s1 m 25 superior 1000 500 1500 casado/a  
## 2 s2 f 32 medio 1500 500 2000 solteiro/a  
## 3 s3 m 78 fundamental 2000 500 2500 viúvo/a

df[ ,1:3] # seleção das três primeiras colunas

## id sexo idade  
## 1 s1 m 25  
## 2 s2 f 32  
## 3 s3 m 78  
## 4 s4 m 12  
## 5 s5 f NA  
## 6 s6 m 34

df[1:3,1:3] # seleção das três primeiras linhas e das três primeiras colunas

## id sexo idade  
## 1 s1 m 25  
## 2 s2 f 32  
## 3 s3 m 78

# Se que quisermos selecionar as linhas 1, 3 e 6, como fazemos?  
df[c(1,3,5), ]

## id sexo idade escolaridade renda bonus renda\_total estado\_civil  
## 1 s1 m 25 superior 1000 500 1500 casado/a  
## 3 s3 m 78 fundamental 2000 500 2500 viúvo/a  
## 5 s5 f NA medio 3000 500 3500 casado/a

# se quisermos selecionar as colunas id, idade e renda\_total  
df[ ,c(1,3,7)]

## id idade renda\_total  
## 1 s1 25 1500  
## 2 s2 32 2000  
## 3 s3 78 2500  
## 4 s4 12 3000  
## 5 s5 NA 3500  
## 6 s6 34 4000

df[ ,c("id","idade","renda\_total")]

## id idade renda\_total  
## 1 s1 25 1500  
## 2 s2 32 2000  
## 3 s3 78 2500  
## 4 s4 12 3000  
## 5 s5 NA 3500  
## 6 s6 34 4000

df[ ,c("renda\_total","idade","id")]

## renda\_total idade id  
## 1 1500 25 s1  
## 2 2000 32 s2  
## 3 2500 78 s3  
## 4 3000 12 s4  
## 5 3500 NA s5  
## 6 4000 34 s6

## Filtrar linhas inteiras

# Selecionar somente as linhas dos participantes do sexo masculino  
df[df$sexo == "m", ]

## id sexo idade escolaridade renda bonus renda\_total estado\_civil  
## 1 s1 m 25 superior 1000 500 1500 casado/a  
## 3 s3 m 78 fundamental 2000 500 2500 viúvo/a  
## 4 s4 m 12 fundamental 2500 500 3000 solteiro/a  
## 6 s6 m 34 superior 3500 500 4000 casado/a

# Selecionar somente as linhas dos participantes do sexo masculino com renda\_total maior ou igual a 2500 reais.  
df[df$sexo == "m" & df$renda\_total >= 2500, ]

## id sexo idade escolaridade renda bonus renda\_total estado\_civil  
## 3 s3 m 78 fundamental 2000 500 2500 viúvo/a  
## 4 s4 m 12 fundamental 2500 500 3000 solteiro/a  
## 6 s6 m 34 superior 3500 500 4000 casado/a

## Alterar o conteúdo dentro de uma variável

# Alterar o conteúdo dentro de uma variável  
  
df$sexo[df$sexo == "f"] <- 0  
df$sexo[df$sexo == "m"] <- 1

## Alterar o formato de variáveis

# visualisar que a variável sexo está como character, mas também foi possível perceber que outras variáveis também não estão no formato ideal. Vamos acertar todas elas.  
  
str(df)

## 'data.frame': 6 obs. of 8 variables:  
## $ id : chr "s1" "s2" "s3" "s4" ...  
## $ sexo : chr "1" "0" "1" "1" ...  
## $ idade : num 25 32 78 12 NA 34  
## $ escolaridade: chr "superior" "medio" "fundamental" "fundamental" ...  
## $ renda : num 1000 1500 2000 2500 3000 3500  
## $ bonus : num 500 500 500 500 500 500  
## $ renda\_total : num 1500 2000 2500 3000 3500 4000  
## $ estado\_civil: chr "casado/a" "solteiro/a" "viúvo/a" "solteiro/a" ...

# mudar variável sexo para factor (note que está como caractere)  
df$sexo <- as.numeric(df$sexo)  
df$sexo <- as.factor(df$sexo) # factor é uma variável nominal  
# df$sexo <- c("m","f","m","m","f","m")  
  
# df$idade <- as.numeric(df$idade)  
df$escolaridade <- as.factor(df$escolaridade)

## Atribuir níveis às variáveis de fator

# atribuir levels aos valores 0 e 1 da variável sexo  
levels(df$sexo) <- c("feminino","masculino")  
levels(df$escolaridade) <- c("fundamental","médio","superior")  
str(df)

## 'data.frame': 6 obs. of 8 variables:  
## $ id : chr "s1" "s2" "s3" "s4" ...  
## $ sexo : Factor w/ 2 levels "feminino","masculino": 2 1 2 2 1 2  
## $ idade : num 25 32 78 12 NA 34  
## $ escolaridade: Factor w/ 3 levels "fundamental",..: 3 2 1 1 2 3  
## $ renda : num 1000 1500 2000 2500 3000 3500  
## $ bonus : num 500 500 500 500 500 500  
## $ renda\_total : num 1500 2000 2500 3000 3500 4000  
## $ estado\_civil: chr "casado/a" "solteiro/a" "viúvo/a" "solteiro/a" ...

## Alterar a ordem das variáveis no dataframe

# Alterar a ordem das variáveis no banco de dados  
df <- df[ ,c("id","sexo","idade","escolaridade","estado\_civil",  
 "renda","bonus","renda\_total")]

# Exercícios 2

O *Internet Movie Database*, conhecido popularmente como IMDb, é uma base de dados de informação sobre cinema TV, música e games pertencente à Amazon. Vamos trabalhar com um arquivo que apresenta diversas informações de 1000 filmes. Para isso, siga os passos abaixo:

Salve o arquivo imdb.rds na pasta que você está utilizando para as aulas do R.

Importe o arquivo imdb.rds para dentro do R, salvando-o como imdb. Use o código de importação abaixo:

imdb <- readRDS("imdb.rds")  
# OBS: Esse código só irá funcionar se o arquivo estiver salvo na pasta em que vc está trabalhando.

Quantas linhas e colunas tem o dataframe imdb?

str(imdb)

## 'data.frame': 1000 obs. of 16 variables:  
## $ Series\_Title : chr "The Shawshank Redemption" "The Godfather" "The Dark Knight" "The Godfather: Part II" ...  
## $ Released\_Year: chr "1994" "1972" "2008" "1974" ...  
## $ Certificate : chr "A" "A" "UA" "A" ...  
## $ Runtime : chr "142 min" "175 min" "152 min" "202 min" ...  
## $ Genre1 : chr "Drama" "Crime" "Action" "Crime" ...  
## $ Genre2 : chr NA " Drama" " Crime" " Drama" ...  
## $ Genre3 : chr NA NA " Drama" NA ...  
## $ IMDB\_Rating : num 9.3 9.2 9 9 9 8.9 8.9 8.9 8.8 8.8 ...  
## $ Meta\_score : int 80 100 84 90 96 94 94 94 74 66 ...  
## $ Director : chr "Frank Darabont" "Francis Ford Coppola" "Christopher Nolan" "Francis Ford Coppola" ...  
## $ Star1 : chr "Tim Robbins" "Marlon Brando" "Christian Bale" "Al Pacino" ...  
## $ Star2 : chr "Morgan Freeman" "Al Pacino" "Heath Ledger" "Robert De Niro" ...  
## $ Star3 : chr "Bob Gunton" "James Caan" "Aaron Eckhart" "Robert Duvall" ...  
## $ Star4 : chr "William Sadler" "Diane Keaton" "Michael Caine" "Diane Keaton" ...  
## $ No\_of\_Votes : int 2343110 1620367 2303232 1129952 689845 1642758 1826188 1213505 2067042 1854740 ...  
## $ Gross : chr "28,341,469" "134,966,411" "534,858,444" "57,300,000" ...

nrow(imdb)

## [1] 1000

ncol(imdb)

## [1] 16

Mostre os filmes desse banco de dados foram dirigidos por Woody Allen?

imdb[imdb$Director == "Woody Allen",c("Series\_Title","Director")]

## Series\_Title Director  
## 415 Annie Hall Woody Allen  
## 527 Crimes and Misdemeanors Woody Allen  
## 529 Hannah and Her Sisters Woody Allen  
## 534 Manhattan Woody Allen  
## 744 Midnight in Paris Woody Allen  
## 836 The Purple Rose of Cairo Woody Allen  
## 838 Zelig Woody Allen  
## 847 Love and Death Woody Allen  
## 930 Match Point Woody Allen

Exiba um dataframe que mostra os títulos dos filmes dirigidos por Steven Spielberg que foram lançados até o ano 2000?

imdb[imdb$Director == "Steven Spielberg" & imdb$Released\_Year <= 2000,  
 c("Released\_Year","Director","Series\_Title")]

## Released\_Year Director Series\_Title  
## 8 1993 Steven Spielberg Schindler's List  
## 25 1998 Steven Spielberg Saving Private Ryan  
## 73 1981 Steven Spielberg Raiders of the Lost Ark  
## 169 1989 Steven Spielberg Indiana Jones and the Last Crusade  
## 263 1993 Steven Spielberg Jurassic Park  
## 416 1975 Steven Spielberg Jaws  
## 683 1985 Steven Spielberg The Color Purple  
## 689 1982 Steven Spielberg E.T. the Extra-Terrestrial  
## 833 1987 Steven Spielberg Empire of the Sun  
## 989 1977 Steven Spielberg Close Encounters of the Third Kind

Exiba um dataframe que mostre os titulos dos filmes dirigidos por Steven Spielberg ou James Cameron, que são do gênero drama ou ação **(Somente em Genre1)**.

imdb[(imdb$Director == "James Cameron" | imdb$Director == "Steven Spielberg") & (imdb$Genre1 == "Drama" | imdb$Genre1 == "Action"), c(1,5,10)]

## Series\_Title Genre1 Director  
## 25 Saving Private Ryan Drama Steven Spielberg  
## 45 Terminator 2: Judgment Day Action James Cameron  
## 73 Raiders of the Lost Ark Action Steven Spielberg  
## 107 Aliens Action James Cameron  
## 169 Indiana Jones and the Last Crusade Action Steven Spielberg  
## 263 Jurassic Park Action Steven Spielberg  
## 411 The Terminator Action James Cameron  
## 624 Avatar Action James Cameron  
## 653 Titanic Drama James Cameron  
## 683 The Color Purple Drama Steven Spielberg  
## 833 Empire of the Sun Action Steven Spielberg  
## 891 Bridge of Spies Drama Steven Spielberg  
## 952 Minority Report Action Steven Spielberg  
## 989 Close Encounters of the Third Kind Drama Steven Spielberg

imdb <-   
rename(imdb,c(título = "Series\_Title",  
 lançamento = "Released\_Year",  
 certificado = "Certificate",  
 duração = "Runtime",  
 gênero1 = "Genre1",  
 gênero2 = "Genre2",  
 gênero3 = "Genre3",  
 imdb = "IMDB\_Rating",  
 meta\_escore = "Meta\_score",  
 diretor = "Director",  
 ator1 = "Star1",  
 ator2 = "Star2",  
 ator3 = "Star3",  
 ator4 = "Star4",  
 votos = "No\_of\_Votes",  
 custo = "Gross"))  
  
names(imdb)

## [1] "título" "lançamento" "certificado" "duração" "gênero1"   
## [6] "gênero2" "gênero3" "imdb" "meta\_escore" "diretor"   
## [11] "ator1" "ator2" "ator3" "ator4" "votos"   
## [16] "custo"

Vamos fazer mais alguns exercícios com um banco de dados sobre a população mundial. Salve o arquivo world\_populations.cvs no diretório que você está utilizando para esta disciplina. Em seguida, importe esse banco de dados para dentro do R usando o código abaixo:

# Importar banco de dados de um arquivo .csv para um dataframe no R chamado world\_population  
wp <- read.csv("world\_population.csv", encoding = "UTF-8")

Crie um dataframe chamado wp1 que permita identificar as capitais de todos os países da América do Sul (países e capitais)

Para colocar a coluna “Country” em ordem alfabética, podemos usar a função order() dentro dos colchetes. Veja:

wp1[order(wp1$Country),]

## Country Capital  
## 9 Argentina Buenos Aires  
## 25 Bolivia Sucre  
## 28 Brazil Brasilia  
## 41 Chile Santiago  
## 43 Colombia Bogota  
## 57 Ecuador Quito  
## 65 Falkland Islands Stanley  
## 70 French Guiana Cayenne  
## 87 Guyana Georgetown  
## 162 Paraguay Asunción  
## 163 Peru Lima  
## 200 Suriname Paramaribo  
## 224 Uruguay Montevideo  
## 228 Venezuela Caracas

Selecione informações do dataframe wp que permita comparar os casos de Brazil, China, Rússia e Índia quanto às seguintes variáveis: população em 2020, densidade por km2, taxa de crescimento e porcentagem da população mundial. Salve essa seleção em um novo objeto chamado wp2.

Ordene o dataframe wp2 pelas suas variáveis. Dica, você pode usar o argumento decreasing = TRUE dentro da função order() para ordenar a coluna de forma decrescente. Teste essa possibilidade.