Análise de dados com R

Aula 02 – Mão na massa

- Recapitulando aula anterior
- Comandos e operações básicas
- Vetores e Data Frames
- Pacotes e funções
- Importação de dados com *rio*
- Manuseio dos dados com *tidyverse*

Recapitulando

Google Colab

- Alternativa para programação em R
- Capacidade gratuita limitada, mas suficiente para trabalhos com pouco volume de dados

Comandos básicos

- # : para comentar no código, deixando a linha não executável
- "<-" ou "=" : para atribuir valor a objetos
- Tipos de valores:
  - num: numéricos
  - chr: caracteres
  - logi: lógicos

Manuseio de objetos

- vetores: objetos que guardam valores de mesmo tipo em uma dimensão
  - Acesso aos valores usando colchetes [posição]

- data.frame: objetos que guardam base de dados estruturadas
  - Acesso aos valores usando colchetes [linha, coluna]

Manuseio de objetos

- Funções: estruturadas com um nome, seguido de parênteses e argumentos de *input*
  - data.frame(vetor1 , vetor2)

Unidade mínima

vetores

1. Vetores são objetos que guardam informações de um tipo (num, chr ou logi).

no R, é definito por:

vetor_num = **c**(1,2,3,...,n)

vetor_chr = **c**("A","B","C",...,"n")

vetor_log = **c**(T,F,T,...,F)

vetor_num

1
2
3
...
n

2. O vetor tem uma dimensão,
Com valores do tipo numérico,
pode ser representado por uma reta

3. Para localizar valores em um vetor, basta usar Colchetes e a posição do valor:

vetor_num[3]

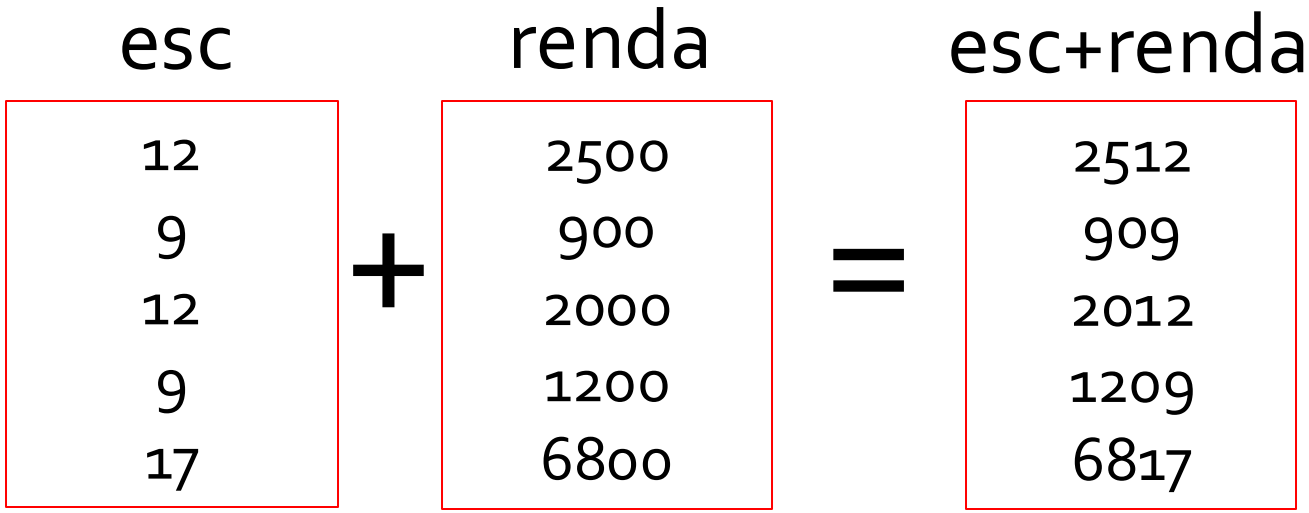
> 3

Passo pra trás

Operações com vetores

Podemos fazer operações com os vetores

esc = c(12, 9, 12, 9,17)
renda = c(2500, 900,2000,1200,6800)



Dados estruturados

Data frames

Quando os vetores têm igual tamanho, podem compor um data.frame

educ	rend	raca
12	2500	branca
9	900	indígena
12	2000	parda
9	1200	preta
17	6800	branca

Definição no R:

meus_dados <- **data.frame**(educ, rend, raca)

Dados estruturados

Data frames

manuseio

Localização e manuseio de valores guardados em um data.frame

Subset: acessar informações em um data frame

meus_dados[Linha, Coluna]

Ou

meus_dados$Coluna

meus_dados$Coluna [Linha]

Operações fudamentais

Testes lógicos

Na aula anterior vimos o R como calculadora. Hoje veremos também **operações lógicas**.

A parte de lógica consiste em testes lógicos cujas respostas pode ser **TRUE** ou **FALSE**

Os testes lógicos consistem em:

1. Se algo é maior que outro, usando o símbolo " **>** "
2. Se algo é menor que outro, usando o símbolo " **<** "

3. Se algo é igual a outro, usando " **==** "
4. Se algo é diferente de outro, usando " **!=** "

5. Se algo está contido em outro, usando **%in%**
6. Se algo não está contido em outro, usando a exclamação " **!** " com o **%in%**

(isso não é tão simples, veja a estrutura nos exemplos)

Também podemos combinar as relações com:

- Maior ou igual " **>=** "
- Menor ou igual " **<=** "

Recapitulando

Lógica

Temos também operações relacionais do tipo **OU**, **E, IN**

Para operações **OU,** usamos a barra vertical:

2 < 0 | 2 == 2 : dois é menor que zero **OU** dois é igual a dois?

Para operações **E,** usamos a o símbolo **&**:
2 < 0 & 2 == 2 : dois é menor que zero **E** dois é igual a dois?

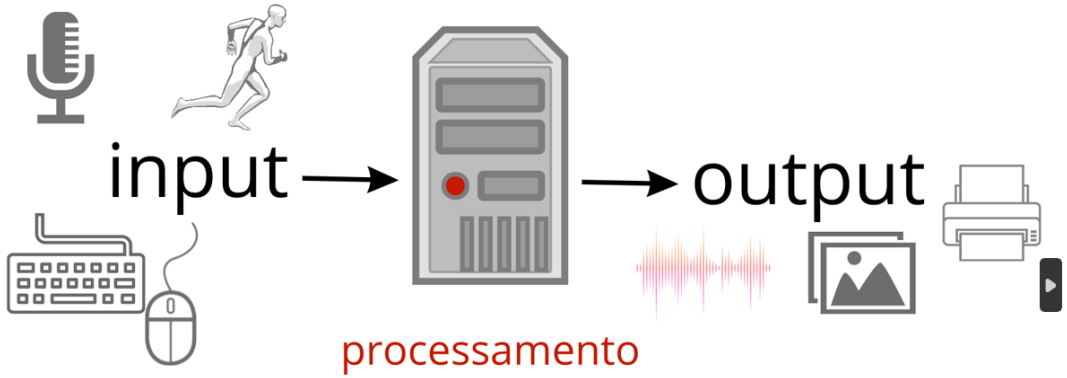
Para operações **IN,** usamos a o símbolo **%in%**:
2 %in% c(1,2,3,4,5) : dois está contido no conjunto 1,2,3,4,5?

Recapitulando

Funções

No R, as funções são fundamentais por realizarem os procedimentos que precisamos.

Uma função é definida por valores que entram (input), pelo processamento e por valores que saem (output)



A programação está aqui!

Softwares já programados processam informações por cliques. Softwares de programação conduz o processamento.

Funções

No R, escrevemos funções assim:

function(parâmetro) { procedimento }

Exemplo:

function(x) { x+2}

Para registrar a função, devemos salvar como objeto na memória

myFunction = function(x) { x+2}

Pacotes

Os pacotes são conjuntos de funções programadas.

No R, temos um universo de pacote disponíveis programados pela comunidade, mas quando fazemos o download, temos só funções de base.

Por isso, temos que instalar pacotes externos usando:

install.packages(nome do pacote)

Além disso, sempre que formos usar o pacote, devemos carregá-lo na memória.

library(nome do pacote)

Só para informar: pacotes direto do GitHub
install.package("devtools")
library("devtools")
install_github("nome_do_usuário/nome_do_pacote")

Pacotes & funções

Muitas vezes, precisamos saber quais são as funções que um pacote possui, e quais argumentos de *input* tais funções devem receber.

Para essa inspeção de pacote e funções, podemos acessar a documentação recorrendo à janela "Help", usando os seguintes recursos:

help(package = "nome_pacote"): para saber sobre o pacote

??pacote::função : para saber sobre a função

Setup

Setup

- A instalação e carregamento de pacotes sempre deve ser feita no início do *script*, compondo a parte de setup. Como os comandos são executados conforme a ordem em que aparecem no *script*, devemos começar com o setup do nosso ambiente de trabalho.

- Há três configurações de setup que devemos incorporar no nosso *script*:
  - Gestão da memória;
  - Gestão das funções e pacotes;
  - Gestão do diretório.

Memória local

- funções muito úteis para limpar a memória local (o que ajuda no processamento).
  - **rm**( ) : de "remove", para remover objetos da memória
  - **ls**( ) : de "list", para listar objetos guardados na memória
  - **gc**( ) : de "Garbage Clean", para limpeza da lixeira e liberação de memória.

Pacotes e funções

- Após a limpeza da memória, devemos inserir comandos para instalar (se necessário) e carregar na memória os pacotes de funções externas que serão utilizados.
  - **install.packages**( ) : para instalar pacotes
  - **library**( ) : para carregar pacotes na memória

Diretório de trabalho

- Após gerir a memória e as funções, definimos nosso diretório de trabalho. Essa etapa é muito importante, porque o R trabalha tendo como referência o diretório principal definido para acessar arquivos e navegar pelo computador. Para isso, utilizamos as funções:

  - **getwd**( ) : de "Get Working Directory", para obter diretório principal no qual está trabalhando

  - **setwd**( ) : de "Get Working Directory", para definir o diretório principal

Como iniciar um *script*

```
# Título do script: Como iniciar um script
# Autor: Victor G Alcantara

# Setup ------------------------------------------------------------------------------------------------------------

# Gestão da memória
rm( list = ls() ) # Remove todos os objetos listados na memória
gc() # Faz a limpeza da lixeira da memória

# Gestão de funções e pacotes
# install.packages("tidyverse") # Instala o pacote tidyverse como exemplo
library(tidyverse) # Carrega o pacote tidyverse como exemplo

# Gestão do diretório
getwd() # Verifica diretório principal de trabalho
setwd(dir = "C:/Users/13477365/Documents/book-dadoscomr") # Define diretório principal de trabalho
# Atenção à posição das barras! Deve estar como "/"
```

Importando dados

Questões técnicas

Questões técnicas

Extensões

Os formatos de arquivos mais comuns são:

CSV – Comma Separated Values : arquivos separados por vírgulas

CSV2 : adequado ao alfabeto latino

XLS e **XLSX** : Excel

SAV : SPSS

DTA: STATA

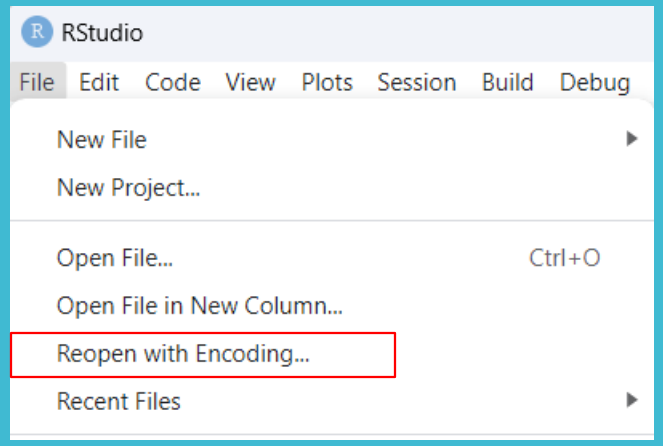
Especificamos para o computador o formato do nosso arquivo pela extensão acrescia após o ponto

arquivo.csv

arquivo.xlsx

Questões técnicas

Encoding



Encoding é como as informações são codificadas para o formato binário.

Como sabemos, o computador lê 0 e 1. Portanto, toda informação (incluso alfabetos) é codificada nesses valores. Alguns encodings (principalmente os padrão inglês) não têm códigos para representar o alfabeto latino e sinais específicos. Por isso, caso tenhamos alfabeto latino, temos que especificar.

read.csv(arquivo , fileEncoding = "Latin1")

Isso para lidar com valores latinos nos dados. No código, eventualmente temos que abrir especificando o encoding também.

Import e manuseio de dados

importação
R Input Output

Pacote **RIO**

```
meus_dados <- import( "endereço_do_arquivo" )
```

visualização
estrutura do dado

Funções úteis para visualizar o dado

dim( ): retorna as dimensões da base de dados (linhas x colunas)

names( ): retorna o nome das colunas/variáveis

head( ): retorna os primeiros valores da base de dados (default = 5, mas passível de alteração)

tail( ): retorna os últimos valores da base de dados (default = 5, mas passível de alteração)

str( ): retorna a estrutura da base de dados, indicando as dimensões e o tipo de cada variável

R base
Operações básicas

Manuseio dos dados com R base: seleção e filtro

Seleção de variáveis

meus_dados[, 1] ou meus_dados[, 2:3]

meus_dados[, "raca"] ou meus_dados[, c("rend", "raca")]

Filtro de casos

meus_dados[, "raca"] == "branca"

meus_dados$raca == "branca"

Renomear variáveis

colnames(dados) <- novos_nomes

Pacote **TIDYVERSE**

tidyverse
Operações básicas

Funções ou **Verbos** importantes:

Seleção de variáveis

meus_dados %>% **select**(raca)

Filtro de casos

meus_dados %>% **filter**(raca == "branca")

Renomear variáveis

meus_dados %>% **rename**("renda_dom" == rend)

Recodificar/mudar variáveis

meus_dados %>% **mutate**(renda2 = rend + 500)

importação
R Input Output

Pacote **RIO**

export(objeto, “endereço_do_arquivo”)