Manipulação de bases no R

Professor: Vitor Pereira

02/09/2022

Preâmbulo

- Apagando tudo no ambiente

```
rm(list=ls()) # apaga tudo no ambiente
```

- Instalar os pacotes de que precisarás
  - Pacotes padrão (CRAN) podem ser instalados com:

```
install.packages("devtools")
```

- Outros pacotes armazenaos no github precisam do Devtools::

```
devtools:: install_github("data-edu/dataedu")
```

- Depois precisamos carregar os pacotes
  - Notem que não precisamos das aspas

```
library(devtools)
```

Preâmbulo

- O pacote "pacman" já inspeciona quais pacotes tens instalados, instala o que não tiver sido instalado e já carrega os pacotes

```
pacman:: p_load(tidyverse, apaTables, sjPlot,
                readxl, dataedu, zelador, epiDisplay)
```

Para São Tomé e Príncipe, é importante acertar a hora do R:

```
Sys.setenv(TZ="UCT") # Acerta a hora de STP
```

Preâmbulo: Os caminhos

- Os caminhos servem para organizar o trabalho
  - São cruciais para trabalhar colaborativamente!
  - Para alterar de um computador para o outro, basta modificar a linha "root"

```
# Paths
root     <- "C:/Users/vitor/Dropbox (Personal)/Sao Tome e Principe/2022/
input    <- paste0(root, "input/")
output   <- paste0(root, "output/")
tmp      <- paste0(root, "tmp/")
code     <- paste0(root, "code/")
alunos   <- paste0(root, "alunos/")
setwd(root)
```

- Atenção:
  - As barras são para a direita!
  - As vezes o cominho fica muito longo e o R não consegue ler. Nesse caso, é melhor salvar a pasta do projeto em C:\

Preâmbulo: Último passo...

- Se precisarmos em algum momento passar toda a base para caractere (para poder fazer o append/rbind/apensar bases na vertical)

```
# Le todas as colunas e decide quais sao numericas
col_types <- readr::cols(.default = readr::col_character())
```

- Essa linha será aproveitada depois quando for converter de volta as colunas com números para fomato numérico. Não se esqueça de carregar o pacote readr

```
#transforma de volta as colunas numericas para o formato de numero
base_alunos <- readr::type_convert(base_alunos)
```

- Por último, separe visualmente cada parte do código

```
########################################
```

- Atenção: Comentem abundantemente seus códigos com o cardinal

Como abrir a base

- A forma de abrir a base dependerá do formato da base
- Arquivos em csv podem ser abertos através do pacote readr()

```
pacman:: p_load(readr)
iris <- read_csv("iris.csv")
```

- Arquivos de excel em xlsx podem ser abertos atravé do pacote readxl. Observe que precisamos nomear o ficheiro e a folha.

```
pacman::p_load(readxl)
basica<-read_excel(paste0(input, "BASE BÁSICO - inicio ano 2021-2022.xls
```

Como abrir a base

- Arquivos em stata precisam do pacote readstata13

```
pacman::p_load(readstata13)
dat <- read.dta13("TEAdataSTATA.dta")
```

- Arquivos em SPSS dependem do pacote

```
pacman::p_load(haven)
dataset = read_sav(path)
```

- Arquivos em txt

```
df <- read.table("dataset.txt", header=TRUE, sep=",")
```

Visualizando a base

- Os comandos View(), head(), str(), tab1() e summary() servem para inspecionar a base de dados

```
# vamos abrir o pacote
library(readr)
library(epiDisplay)
iris <- read_csv("iris.csv")
head(iris) #primeiras linhas
str(iris) # estrutura da base
View(iris) # abre a base para poder olhar
summary(iris) # faz um resumo da base
tab1(iris$sepal.length)
```

_ O str mostra a estrutura da base

- Não confundam summary com summarise. O summary() vai dar um resumo de cada variável
- O tab1 depende do pacote epiDisplay

Como limpar as bases

- Os nomes das colunas das bases não podem ter espaços, nem traços, nem caracteres especiais
- Para limpar esses nomes, basta utilizar o comando clean_names() do pacote janitor()

```
# Limpando os nomes das variáveis
pacman::p_load(janitor)
pre_survey <- clean_names(dataedu::pre_survey)
```

Como limpar as bases

- Os nomes de algumas colunas podem vir muito grandes ou ser pouco informativos. Para modificar, utilizamos o comando rename()

```
# renomear variaveis
pre_survey <- pre_survey %>%
  rename(
    q1 = q1maincellgroup_row1 ,
    q2 = q1maincellgroup_row2 ,
    q3 = q1maincellgroup_row3 ,
    q4 = q1maincellgroup_row4 ,
    q5 = q1maincellgroup_row5 ,
    q6 = q1maincellgroup_row6 ,
    q7 = q1maincellgroup_row7 ,
    q8 = q1maincellgroup_row8 ,
    q9 = q1maincellgroup_row9 ,
    q10 = q1maincellgroup_row10,
    usuario = opdata_username,
    curso = opdata_course_id
    )
```

Remover colunas e linhas vazias

- Para remover as colunas e linhas vazias, utilze o comando remove_empty(), do pacote janitor

```
# Remove as linhas e colunas vazias
 remove_empty(c("rows", "cols"))
```

Limpar o conteúdo das variáveis

- É possível retirar caracteres especiais
  - Dentro de uma pipe(%>%), coloque:

```
classe = gsub("ª", "", classe)) # Tira o "ª"
```

- Nesse exemplo, retiramos todos símbolos "ª" da variável classe, permitindo convert-la para números

Limpar o conteúdo das variáveis

- Também é possível modificar o conteúdo de uma variável.
- Para isso, vamos utilizar os comandos mutate() e replace() do pacote dplyr()

```
pacman::p_load(dplyr)
# Append- agregacao das bases, uma base em cima da outra
base_alunos <- base_alunos %>%
  mutate(distrito=replace(distrito, # corrige o distrito
                          distrito=="LOBATA","Lobata"))
```

Limpar o conteúdo das variáveis

- Também é possível utilizar o comando case_when () do mutate() e dplyr()
- Também podemos utilizar o mutate(case_when()) para criar novas variáveis. Exemplo:

```
base_alunos <- base_alunos %>%
  mutate(distorcao = case_when(
    idade-classe-5 >=2 ~ "Em distorção",
    idade-classe-5 < 2 ~ "Fora de distorção" ))
```

Manipulando os dados: O dplyr

- O pacote dplyr possui alguns comandos bastante importantes:

select() - seleciona colunas arrange() - ordena a base filter() - filtra linhas mutate() - cria/modifica colunas group_by() - agrupa a base summarise() - sumariza a base

Exemplos- dplyr: O mutate

```
- Mutate para criar uma nova variável categórica

measure_mean <- measure_mean %>%
  mutate( construto = case_when(
      questao %in% c("q1", "q4", "q5", "q8", "q10") ~ "interesse" ,
      questao %in% c("q2", "q6" , "q9") ~ "utilidade do curso",
      questao %in% c("q3", "q7") ~ "competencia percebida"))
```

Exemplos- dplyr: O select

- O comando select irá selecionar as colunas desejas

```
basica <- basica %>%
  dplyr::select(codigo_escola) %>%  # Ficamos apenas com o códiugo da es
```

- Se você quiser todas as colunas, exceto algumas, basta colocar o símbolo de menos (-) antes da variável que desejas excluir.

```
basica <- basica %>%
  dplyr::select(-codigo_escola) %>%  # ficamos com todas as colunas, exc
```

O group_by e o summarise

- O comando group_by() irá agrupar "virtualmente" a base de dados de acordo com os valores de uma coluna. Em geral, é utilizado logo antes de uma operação em que a base será reduzida/colapsada através do summarise

```
medias_construto <- measure_mean %>%
  group_by(construto) %>%
  summarise(
    # Média
    media_respostas = mean(resposta, na.rm=TRUE),
    # Mediana
    median_repostas = median(resposta, na.rm=TRUE),
    # Desvio Padrao
    desv_pad_repostas = sd(resposta, na.rm=TRUE),
    # Percentual de missings
    perc_missing    = mean(is.na(resposta),
    # Total de linhas
    total_respostas = n() ,
    # Número de valores distintos de escolas
    total_escolas= n_distinct(school) )
```

Transformando todas as colunas em texto

- As vezes , para fazer uma junção na vertical, precisamos passar todas as colunas para texto.
    - Por que? Porque o formato de cada coluna deve ser o mesmo. Se uma delas é diferente, teremos um erro.

```
# transforma tudo em texto
base_alunos %>%  base alunos %>%
  mutate(across(.fns = as.character)) # transformamos todas as colunas e
```

- Depois de feito o append/rbind, é importante voltar com as colunas numéricas

```
base_alunos <- bind_rows(basica, secundaria_7_9,
      secundaria_10_12) %>% # append das bases
# Converte de volta
  base_alunos <- readr::type_convert(base_alunos)
```