Exercícios Aula 04

Pílula 1

1. Veja o seguinte pacote no github: genderBR. Como podemos baixá-lo ?
2. O código a seguir dá erro. Por quê?

```
a <- 23,1
```

3. Arrume o erro e responda: qual é a classe desse objeto?
4. A função `ceiling()` faz parte do `base R`. Ela recebe o argumento `x`, que recebe valores do tipo numéricos (seja do tipo *numeric*, seja do tipo *integer*). Se o número for quebrado, por exemplo 1,2, ele "arredonda para cima", transformando em 2.

   4a. Digite `ceiling(x = a)` no Console do R. Qual é o resultado encontrado? 4b. Digite `ceiling(x = b)` no Console do R. Qual é o resultado encontrado? Por quê? 4c. Digite `ceiling(y = a)` no Console do R. Qual é o resultado encontrado? Por quê?

5. O que você espera que seja "impresso" em cada caso.

   5a. Caso 1: Você espera que seja impresso 1 ou 2 ou outra opção?

```
a1 <- 1
a2 <- 2

a1 = a2

print(a1)
```

5b. Caso 2: Você espera que seja impresso 1 ou 2 ou outra opção?

```r
a1 <- 1
a2 <- 2

a1 = a2
a2 = a1

print(a2)
```

5c. Caso 3: Você espera que seja impresso 1 ou 2 ou outra opção?

```r
a1 <- 1
a2 <- 2

a2 = a1
a1 = a2

print(a1)
```

5d. Caso 4: Você espera que seja impresso 1 ou 2 ou outra opção?

```r
a1 <- 1
a2 <- 2

print(a1 == a2)
```

Pílula 2

1. Baixe o pacote cursoESMP.
2. Inspecione a base `cursoESMP::dados` com o `dplyr::glimpse()`. Quantas colunas existem? Existe alguma coluna do tipo `double` ou `integer`? Quantas linhas existem?
3. Qual é a função do `dplyr` que devemos usar em cada caso? `select()` ou `filter()` ? Faça os códigos utilizando a base `cursoESMP::dados`.

   A) Queremos manter apenas as linhas em que a classe é "Procedimento Comum Cível".
   B) Queremos manter apenas as colunas prcoesso, digital, assunto, classe, distribuicao.
   C) Queremos manter apenas as linhas em que digital for verdadeiro

4. Coloque em uma pipeline o seguinte código, criando **APENAS** a `base_final`, sem criar nenhuma base intermediária.

```r
base1 <- cursoESMP::dados

base2 <- dplyr::mutate(base1, classe_assunto = paste0(classe, " - ", assunto))

base_final <- dplyr::select(base2, -classe, -assunto)
```

Pílula 3

1. Vamos comparar o `dplyr::mutate()` com o `dplyr::summarise()`.

   A) Inspecione a base de dados `da` com o `dplyr::glimpse()`. Quantas colunas ela possui? E quantas linhas? Qual é o tipo de cada coluna?

   B) **Apenas olhe** o seguinte código. Utilizamos a função do `baseR` chamada `sqrt()`, que serve para fazer a raiz quadrada de uma variável numérica. Sabendo disso, o que você **espera** que essa função faça no `mutate()`? Haverá exclusão de coluna? E modificação? E adição? E no `summarise()`? Neste caso, haverá exclusão de coluna? E modificação? E adição?

   C) Agora rode o código. O que aconteceu em cada caso? Isso foi de acordo com a sua expectativa?

   D) Faz sentido usar o `sqrt()` junto do `summarise()`?

```r
set.seed(3)
da <- tibble::tibble(
  a = sample(letters, 1000, replace=T),
  b = sample(20:80, 1000, replace=T)
)

da |>
  dplyr::mutate(
    b2 = sqrt(b)
  )

da |>
  dplyr::summarise(
    b2 = sqrt(b)
  )
```

2. Vamos agora olhar para os comparadores do `baseR` e entender sua aplicação para o `mutate()` e para o `summarise()`.

   A) O que faz o comparador `==` ?

```r
set.seed(3)
vetor <- sample(letters, 100, replace=T)

vetor == "a"
```

B) O que faz a função `all()` ?

```r
set.seed(3)
vetor <- sample(letters, 100, replace=T)

all(vetor == "a")
```

C) O que faz a função `any()` ?

```r
set.seed(3)
vetor <- sample(letters, 100, replace=T)

any(vetor == "a")
```

D) Qual das três opções acima você acha mais apropriado utilizar no `summarise()`? E no `mutate()`? Crie exemplos utilizando a base `cursoESMP::dados`.

4. É possível usar o `group_by()` junto de `filter()` também.

A) Qual é a diferença entre o Código A e o Código B abaixo?

```r
set.seed(3)
base <- tibble::tibble(
  letras = sample(letters, 100, replace=T),
  valores = sample(30:80, 100, replace=T)
)

# Código A
base |>
  dplyr::filter(valores >= 60)

# Código B
base |>
  dplyr::group_by(letras) |>
  dplyr::filter(valores >= 60) |>
  dplyr::ungroup()
```

B) E agora? Qual é a diferença entre o Código A e o Código B abaixo?

```r
set.seed(3)
base <- tibble::tibble(
  letras = sample(letters, 100, replace=T),
  valores = sample(30:80, 100, replace=T)
)

# Código A
base |>
  dplyr::filter(all(valores >= 60))

# Código B
base |>
  dplyr::group_by(letras) |>
  dplyr::filter(all(valores >= 60)) |>
  dplyr::ungroup()
```