Lição 2: Manipulação de Vetores e Dataframes

Nesta lição, vamos ver dois tipos de estruturas de dados que serão usados mais frequentemente nas próximas tarefas – vetores e dataframes. Além disso, vamos aprender a usar algumas funções úteis para inspecionar esses tipos de objeto. Para outros tipos de estruturas de dados, você pode consultar a excelente página de Hadley Wickham: http://adv-r.had.co.nz/Data-structures.html.

Vetores são a estrutura de dados mais básica do R. São estruturas unidimensionais de dados do mesmo tipo (numéricos, caracteres etc.). Na Lição 1 – Fundamentos, criamos alguns vetores: `x`, `y`, `z`, `aleatorio`, `numeros`. Também carregamos o dataframe `dados`. Rode as linhas de comando a seguir para deixá-los novamente disponíveis. Defina como diretório de trabalho aquele que contém a planilha DadosRT.csv.

```
# Recriar vetores da Lição 1

x <- 4 + 9
y <- x + 10
z <- "gato"
aleatorio <- c(x, y, 230)
numeros <- c(20:39)

# Definir diretório de trabalho

#setwd()

# Carregar dados

dados <- read_csv("DadosRT.csv",
                  col_types = cols(.default = col_factor(),
                                   cont.precedente = col_character(),
                                   ocorrencia = col_character(),
                                   cont.seguinte = col_character(),
                                   IDADE = col_integer(),
                                   INDICE.SOCIO = col_double(),
                                   FREQUENCIA = col_double()
                                   )
                  )
```

Para esta lição, vamos precisar mais adiante das funções do pacote tidyverse. Carregue-o para deixá-lo disponível nesta sessão.

```
library(tidyverse)
```

Para acessar elementos específicos dentro de um vetor, usamos o operador de indexação `[]` com um número dentro dos colchetes. Veja o que dá a linha de comando `aleatorio[3]`.

```
aleatorio[3]
## [1] 230
```

O R dá como resultado o terceiro elemento do vetor `aleatorio`. Peça ao R agora que mostre qual é o 15º elemento do vetor `numeros`.

```
numeros[15]
## [1] 34
```

Você se lembra qual era o valor de `x`? Olhe a aba Environment se necessário. Qual será o resultado de `numeros[x]`?

- O resultado é imprevisível
- Vai ocorrer um erro, porque `x` não é um número
- O R mostrará o 13º elemento do vetor `numeros`

Vamos checar o resultado de `numeros[x]`. Digite esse comando no *script*.

```
numeros[x]
## [1] 32
```

E se quisermos pedir ao R que mostre o 1º e o 3º elementos do vetor `aleatorio`? Uma função vista na Lição 1 vai ser útil aqui. Qual é ela?

- `c()`
- `getwd()`
- `library()`
- `read_csv()`
- `setwd()`

Vamos testar! Peça ao R que mostre o 1º e 3º elementos do vetor `aleatorio` com `aleatorio[c(1, 3)]`.

```
aleatorio[c(1, 3)]
```

```
## [1]  13 230
```

Também é possível exibir todos os elementos de um vetor *exceto* um ou alguns, com o operador de subtração `-`. Teste `numeros[-2]`.

```
numeros[-2]
```

```
##  [1] 20 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
```

O R retorna todos os elementos do vetor `numeros`, exceto o segundo elemento.

Na lição anterior, também vimos outro tipo de estrutura de dados, um dataframe. Diferentemente de um vetor, o dataframe é um conjunto bidimensional de dados, com linhas e colunas. Outro modo de entender o dataframe é como um conjunto de vetores de mesma extensão (mesmo número de colunas ou mesmo número de linhas).

Na Lição 1, havíamos carregado uma planilha dentro de um objeto chamado `dados`. Ele está disponível novamente para esta sessão do R. Digite `dados` para visualizar o tibble desse dataframe.

```
dados
```

```
## # A tibble: 9,226 × 20
##    VD         PARTICIPANTE SEXO.GENERO IDADE FAIXA.ETARIA ESCOLARID
ADE
##    <fct>      <fct>        <fct>       <int> <fct>        <fct>
##  1 retroflexo IvanaB       feminino       30 1a           fundament
al
##  2 retroflexo IvanaB       feminino       30 1a           fundament
al
##  3 retroflexo IvanaB       feminino       30 1a           fundament
al
##  4 tepe       IvanaB       feminino       30 1a           fundament
al
##  5 retroflexo IvanaB       feminino       30 1a           fundament
al
##  6 retroflexo IvanaB       feminino       30 1a           fundament
al
##  7 retroflexo IvanaB       feminino       30 1a           fundament
al
##  8 retroflexo IvanaB       feminino       30 1a           fundament
al
##  9 tepe       IvanaB       feminino       30 1a           fundament
al
## 10 retroflexo IvanaB       feminino       30 1a           fundament
al
## # … with 9,216 more rows, and 14 more variables: REGIAO <fct>,
## #   INDICE.SOCIO <dbl>, ORIGEM.PAIS <fct>, CONT.FON.PREC <fct>,
```

```
## #   CONT.FON.SEG <fct>, TONICIDADE <fct>, POSICAO.R <fct>,
## #   CLASSE.MORFOLOGICA <fct>, FREQUENCIA <dbl>, ESTILO <fct>,
## #   ITEM.LEXICAL <fct>, cont.precedente <chr>, ocorrencia <chr>,
## #   cont.seguinte <chr>
```

Veja que o R exibiu apenas as primeiras linhas de `dados` e omitiu as restantes. Se quisermos visualizar o conjunto completo de dados, é melhor fazê-lo em outra aba. Aplique a função `View()` a `dados` e veja o resultado.

```
View(dados)
```

N.B.: Resultado aqui omitido.

O R abre uma nova aba em Source e permite visualizar a planilha como no Excel ou no Calc. Bastante útil, não? O R também permite saber certas propriedades deste dataframe. O número de colunas pode ser visto com a função `ncol()`, que toma o dataframe como argumento. Digite `ncol(dados)`.

```
ncol(dados)
```

```
## [1] 20
```

Veja agora o número de linhas aplicando a função `nrow()` a `dados`.

```
nrow(dados)
```

```
## [1] 9226
```

A função `str()` (=structure) fornece uma visão global sobre o conjunto de dados. Aplique-a a `dados` para ver o resultado.

```
str(dados)
```

```
## spec_tbl_df [9,226 × 20] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ VD                : Factor w/ 2 levels "retroflexo","tepe": 1 1
1 2 1 1 1 1 2 1 ...
##  $ PARTICIPANTE      : Factor w/ 118 levels "IvanaB","HeloisaS",..:
1 1 1 1 1 1 1 1 1 1 ...
##  $ SEXO.GENERO       : Factor w/ 2 levels "feminino","masculino": 1
1 1 1 1 1 1 1 1 1 ...
##  $ IDADE             : int [1:9226] 30 30 30 30 30 30 30 30 30 30 .
..
##  $ FAIXA.ETARIA      : Factor w/ 3 levels "1a","3a","2a": 1 1 1 1 1
1 1 1 1 1 ...
##  $ ESCOLARIDADE      : Factor w/ 3 levels "fundamental",..: 1 1 1 1
1 1 1 1 1 1 ...
##  $ REGIAO            : Factor w/ 2 levels "periferica","central": 1
1 1 1 1 1 1 1 1 1 ...
##  $ INDICE.SOCIO      : num [1:9226] 2 2 2 2 2 2 2 2 2 2 ...
##  $ ORIGEM.PAIS       : Factor w/ 5 levels "mista","SPcapital",..: 1
1 1 1 1 1 1 1 1 1 ...
```

```
##  $ CONT.FON.PREC     : Factor w/ 7 levels “a”,”o”,”e”,”0”,..: 1 2 3
3 1 1 4 1 4 5 ...
##  $ CONT.FON.SEG      : Factor w/ 19 levels “ts”,”n”,”g”,”v”,..: 1 2
3 4 5 6 7 7 5 8 ...
##  $ TONICIDADE        : Factor w/ 2 levels “tonica”,”atona”: 1 2 2 2
1 1 1 1 2 1 ...
##  $ POSICAO.R         : Factor w/ 2 levels “medial”,”final”: 1 1 1 1
2 2 1 1 1 1 ...
##  $ CLASSE.MORFOLOGICA: Factor w/ 6 levels “substantivo”,..: 1 1 1 1
1 1 1 1 2 1 ...
##  $ FREQUENCIA        : num [1:9226] 1.34 0.16 0.22 0.44 1.94 1.94 0
.35 0.03 5.98 0.16 ...
##  $ ESTILO            : Factor w/ 4 levels “conversacao”,..: 1 1 1 1
1 1 1 1 1 1 ...
##  $ ITEM.LEXICAL      : Factor w/ 1151 levels “parte”,”jornal”,..: 1
2 3 4 5 5 6 7 8 9 ...
##  $ cont.precedente   : chr [1:9226] “do CEU é daquela” “eu não gost
o de” “não (es)to(u) entendendo a” “o que sei... num” ...
##  $ ocorrencia        : chr [1:9226] “parte <R>“ “jornal <R>“ “pergu
nta <R>“ “serviço <T>“ ...
##  $ cont.seguinte     : chr [1:9226] “que as perua(s) ia” “D1: mas d
igo assim” “D1: o que que/” “a não ser <A>“ ...
##  - attr(*, “spec”)=
##   .. cols(
##   ..   .default = col_factor(),
##   ..   VD = col_factor(levels = NULL, ordered = FALSE, include_na =
FALSE),
##   ..   PARTICIPANTE = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   SEXO.GENERO = col_factor(levels = NULL, ordered = FALSE, inc
lude_na = FALSE),
##   ..   IDADE = col_integer(),
##   ..   FAIXA.ETARIA = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   ESCOLARIDADE = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   REGIAO = col_factor(levels = NULL, ordered = FALSE, include_
na = FALSE),
##   ..   INDICE.SOCIO = col_double(),
##   ..   ORIGEM.PAIS = col_factor(levels = NULL, ordered = FALSE, inc
lude_na = FALSE),
##   ..   CONT.FON.PREC = col_factor(levels = NULL, ordered = FALSE, i
nclude_na = FALSE),
##   ..   CONT.FON.SEG = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   TONICIDADE = col_factor(levels = NULL, ordered = FALSE, incl
ude_na = FALSE),
##   ..   POSICAO.R = col_factor(levels = NULL, ordered = FALSE, inclu
de_na = FALSE),
##   ..   CLASSE.MORFOLOGICA = col_factor(levels = NULL, ordered = FAL
SE, include_na = FALSE),
##   ..   FREQUENCIA = col_double(),
##   ..   ESTILO = col_factor(levels = NULL, ordered = FALSE, include_
na = FALSE),
```

```
##   ..   ITEM.LEXICAL = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   cont.precedente = col_character(),
##   ..   ocorrencia = col_character(),
##   ..   cont.seguinte = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

O resultado de `str()` informa que `dados` é um dataframe com 9.226 linhas (observações) e 20 colunas (variáveis). Em seguida, indica, para cada variável, seu nome (que segue o símbolo `$`), seu tipo (num, Factor with N levels etc.) e seus primeiros dados. Veja que a função `str()` faz as vezes de `ncol()`, `nrow()` e do tibble, de modo que vamos usá-la frequentemente nas próximas lições. Sempre que carregar um dataframe com `read_csv()`, vale a pena checar se o dataframe foi carregado corretamente com `str()`.

Também vale conhecer a função `summary()`. Aplique-a a `dados` para ver o resultado.

```
summary(dados)
```

```
##           VD         PARTICIPANTE     SEXO.GENERO       IDADE
##  retroflexo:2611   VeraD   : 108   feminino :4565   Min.   :19.00
##  tepe      :6615   RodrigoC: 106   masculino:4661   1st Qu.:31.00
##                    RogerioA: 106                    Median :47.00
##                    MarcoP  : 105                    Mean   :46.82
##                    LucasS  : 105                    3rd Qu.:61.00
##                    RenataC : 105                    Max.   :83.00
##                    (Other) :8591
##  FAIXA.ETARIA      ESCOLARIDADE          REGIAO      INDICE.SOCIO
##  1a:3051      fundamental:1098   periferica:4881   Min.   :1.600
##  3a:2919      superior   :4729   central   :4345   1st Qu.:2.700
##  2a:3256      medio      :3399                     Median :3.100
##                                                    Mean   :3.075
##                                                    3rd Qu.:3.500
##                                                    Max.   :4.500
##
##       ORIGEM.PAIS   CONT.FON.PREC  CONT.FON.SEG   TONICIDADE
##  mista      :3201   a:1983        t      :1513   tonica:4611
##  SPcapital  :1992   o:1709        k      :1382   atona :4615
##  interior   :2562   e:2088        m      :1065
##  nordeste   : 611   0: 696        d      : 748
##  estrangeira: 860   3: 787        #      : 725
##                     u:1369        g      : 653
##                     i: 594        (Other):3140
##   POSICAO.R      CLASSE.MORFOLOGICA   FREQUENCIA
##  medial:7382   substantivo:4826     Min.   :0.010
##  final :1844   conj.prep  : 969     1st Qu.:0.130
##                adverbio   :  37     Median :0.620
##                verbo      :1181     Mean   :1.034
```

```
##                adjetivo   :1530     3rd Qu.:0.930
##                morf.inf   : 683     Max.   :5.980
##
##          ESTILO      ITEM.LEXICAL  cont.precedente
##  conversacao:5900   porque : 552   Length:9226
##  palavras   :1985   por    : 300   Class :character
##  jornal     : 592   perto  : 240   Mode  :character
##  depoimento : 749   erguer : 200
##                     irmã   : 193
##                     lugar  : 179
##                     (Other):7562
##   ocorrencia        cont.seguinte
##  Length:9226        Length:9226
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

`summary()` também fornece uma visão global de um conjunto de dados, com medidas estatísticas apropriadas para cada tipo de variável. Para variáveis fatoriais, como VD (que significa "variável dependente" e representa as variantes de /r/ em coda), é apresentado o número de ocorrências de cada variante. Para variáveis numéricas, como IDADE, são apresentadas medidas como valor mínimo, 1º quartil, média etc. Tudo isso vai ser objeto das Lições 3 a 7.

Note, no entanto, que o R computou as frequências das variáveis fatoriais e certas medidas estatísticas para as variáveis numéricas porque, ao importar os dados com `read_csv()`, essas variáveis foram definidas com essas características por meio de `col_factor()`, `col_integer()` etc. As últimas três colunas, cont.precedente, ocorrencia e cont.seguinte, que foram definidas como `col_character()` – por não serem variáveis de fato –, não apresentam essas medidas estatísticas! Perceba, então, que as medidas geradas por `summary()` dependem da natureza de cada coluna/variável!

Mas e se quisermos acessar elementos específicos de um dataframe, do modo como fizemos para os vetores anteriormente? Para isso, também usamos os colchetes `[]`. Entretanto, como dataframes são estruturas bidimensionais, precisamos de dois índices para localizar um elemento – um para a linha, e outro para a coluna. Memorize essa ordem: [L, C]!

Por exemplo, para acessar o elemento da 5ª linha da 6ª coluna do conjunto `dados`, a linha de comando é `dados[5, 6]`. Note que não há espaço entre o nome do objeto e o colchete aberto, e que os índices de linha e de coluna são separados por vírgula. Rode essa linha de comando agora.

```
dados[5, 6]
```

```
## # A tibble: 1 × 1
##   ESCOLARIDADE
##   <fct>
## 1 fundamental
```

Acesse o elemento da 11ª linha da 4ª coluna do conjunto dados.

```
dados[11, 4]
```

```
## # A tibble: 1 × 1
##   IDADE
##   <int>
## 1    30
```

Acesse agora os elementos da 1ª até a 10ª linha da 9ª coluna de `dados`.

```
dados[1:10, 9]
```

```
## # A tibble: 10 × 1
##    ORIGEM.PAIS
##    <fct>
##  1 mista
##  2 mista
##  3 mista
##  4 mista
##  5 mista
##  6 mista
##  7 mista
##  8 mista
##  9 mista
## 10 mista
```

Para acessar todos os dados de uma linha, basta deixar o espaço para o índice de coluna vazio. Digite `dados[3, ]` para ver o resultado. Note que ainda assim é necessário usar a vírgula!

```
dados[3, ]
```

```
## # A tibble: 1 × 20
##   VD         PARTICIPANTE SEXO.GENERO IDADE FAIXA.ETARIA ESCOLARIDA
DE
##   <fct>      <fct>        <fct>       <int> <fct>        <fct>
## 1 retroflexo IvanaB       feminino       30 1a           fundamenta
l
```

```
## # … with 14 more variables: REGIAO <fct>, INDICE.SOCIO <dbl>,
## #   ORIGEM.PAIS <fct>, CONT.FON.PREC <fct>, CONT.FON.SEG <fct>,
## #   TONICIDADE <fct>, POSICAO.R <fct>, CLASSE.MORFOLOGICA <fct>,
## #   FREQUENCIA <dbl>, ESTILO <fct>, ITEM.LEXICAL <fct>,
## #   cont.precedente <chr>, ocorrencia <chr>, cont.seguinte <chr>
```

De modo semelhante, se deixarmos o espaço para o índice de linha vazio e preenchermos apenas a coluna, o resultado será todos os elementos da coluna respectiva. Digite `dados[, 10]` para ver os elementos da 10ª coluna do dataframe.

```
dados[, 10]
```

```
## # A tibble: 9,226 × 1
##    CONT.FON.PREC
##    <fct>
##  1 a
##  2 o
##  3 e
##  4 e
##  5 a
##  6 a
##  7 0
##  8 a
##  9 0
## 10 3
## # … with 9,216 more rows
```

No R, sempre há diversas maneiras para se chegar a um mesmo resultado. Outro modo de acessar os elementos de uma coluna de um dataframe é por meio do símbolo `$`, que já vimos acima. Digite `dados$CONT.FON.PREC` para ver os elementos da coluna CONT.FON.PREC.

```
dados$CONT.FON.PREC
```

N.B.: Resultado aqui omitido.

A principal diferença entre `[]` e `$` é que, no último caso, usamos o *nome* da variável para acessar seus elementos. Isso pode ser útil quando se sabe o nome da variável mas não se lembra em que coluna ela está (o que provavelmente vai ser o caso com seus dados).

Muitas vezes temos o interesse em acessar elementos específicos de um dataframe para criar subconjuntos de dados. O pacote dplyr, que faz parte do conjunto de pacotes do tidyverse – que carregamos mais acima –, permite criar novos dataframes de um subconjunto de linhas ou colunas, a partir de certos critérios. Digamos que você queira

criar um subconjunto apenas com os dados de /r/ em coda que ocorrem em sílabas tônicas. Na coluna TONICIDADE de `dados`, esses dados estão identificados por "tonica".

A função `filter()` toma como primeiro argumento o conjunto de dados, e outro argumento que especifica a condição a ser preenchida. Digite `dados_tonicas <- filter(dados, TONICIDADE == "tonica")`. (Note que o sinal de igual, em R, é duplo: ==).

```
dados_tonicas <- filter(dados, TONICIDADE == "tonica")
```

Crie agora o subconjunto `dados_atonas`, que contém os dados de /r/ em coda que ocorrem em sílabas átonas (identificados por "atona").

```
dados_atonas <- filter(dados, TONICIDADE == "atona")
```

Em outros casos, pode ser que você queira especificar quais variantes não devem entrar no conjunto de dados. Imagine que, na análise, você decida retirar os dados de /r/ que ocorrem em verbos do infinitivo, como em "andar", "comer", uma vez que, nesse contexto, o /r/ é quase sempre apagado. A variável de interesse aqui é CLASSE.MORFOLOGICA e o código para verbos do infinitivo é "morf.inf". Vamos guardar o resultado em um novo dataframe chamado `dados_semInf`. E a informação mais importante de que você precisa para realizar essa operação: o símbolo para "não é igual a", no R, é `!=`.

```
dados_semInf <- filter(dados, CLASSE.MORFOLOGICA != "morf.inf")
```

Crie um novo subconjunto de dados que contém apenas os dados advindos de leituras (depoimento, jornal, palavras), ou seja, sem os dados "conversacao". A variável se chama ESTILO. Guarde o resultado em um novo dataframe chamado `dados_leitura`.

```
dados_leitura <- filter(dados, ESTILO != "conversacao")
```

Agora crie um subconjunto de dados, chamado `dados_menor2`, de palavras cuja FREQUENCIA é menor do que 2% do corpus. O operador lógico do R para "menor que" é `<`. Digite `dados_menor2 <- filter(dados, FREQUENCIA < 2)`.

```
dados_menor2 <- filter(dados, FREQUENCIA < 2)
```

Mais uma! Se temos interesse em palavras cuja frequência é *menor ou igual* a 2%, o operador lógico pertinente é <=. Crie um novo subconjunto, com essa condição, chamado dados_menor_igual2.

```
dados_menor_igual2 <- filter(dados, FREQUENCIA <= 2)
```

Para criar o subconjunto de /r/ em coda em sílabas tônicas, usamos as aspas para especificar "tonica". Para criar o subconjunto de palavras menos frequentes do que 2%, não usamos as aspas. Por quê?

- Porque queremos que o R entenda "2" como um caractere
- Porque "tonica" é uma sequência de caracteres e 2 é um elemento numérico
- Porque se digitarmos tonica sem as aspas, o R tratará a variante como numérica

Podemos ser ainda mais restritivos e criar um subconjunto de dados de /r/ em coda em sílabas tônicas que ocorrem em final de palavra. O operador lógico no R para juntar duas condições é &. Digite dados_tonica_final <- filter(dados, TONICIDADE == "tonica" & POSICAO.R == "final").

```
dados_tonica_final <- filter(dados, TONICIDADE == "tonica" & POSICAO.R
== "final")
```

Podemos também ser *menos* restritivos e criar um subconjunto que contenha os dados de /r/ em coda que estejam em sílabas tônicas *ou* em sílaba final. Em relação ao conjunto dados_tonica_final, esse novo conjunto de dados (sílabas tônicas *ou* posição final)...

- Provavelmente contém um número maior de dados do que dados_tonica_final
- Provavelmente contém o mesmo número de dados de dados_tonica_final
- Provavelmente contém um número menor de dados do que dados_tonica_final

A razão para que um novo subconjunto que tenha os dados em sílabas tônicas *ou* em posição final seja provavelmente maior é o fato de que o operador lógico "ou" retorna um resultado verdadeiro se qualquer uma das duas condições é preenchida e, portanto, um maior número de observações potencialmente preencherá o critério. No caso do operador lógico &, o resultado só é verdadeiro se ambas as condições forem verdadeiras,

o que faz com que a probabilidade de haver observações que preenchem ambos os critérios seja menor.

Vamos então criar o subconjunto de dados de /r/ em sílabas tônicas *ou* em sílabas finais. O operador lógico "ou", no R, é o símbolo |. Chame esse novo conjunto de dados_tonica_ou_final.

```
dados_tonica_ou_final <- filter(dados, TONICIDADE == "tonica" | POSICAO.R == "final")
```

Usamos até agora a função `filter()`, do pacote dplyr/tidyverse, para fazer subconjuntos de dados restringindo o número de linhas (=ocorrências) do dataframe. Também é possível selecionar as colunas de um conjunto de dados para criar um novo dataframe, por meio da função `select()`.

A função `select()`, de modo semelhante à função `filter()`, também toma como primeiro argumento o dataframe original. Os demais argumentos são as colunas/variáveis que você deseja incluir. Crie um novo df chamado `dados_varsociais1`, com as variáveis SEXO.GENERO, IDADE, ESCOLARIDADE e REGIAO.

```
dados_varsociais1 <- select(dados, SEXO.GENERO, IDADE, ESCOLARIDADE, REGIAO)
```

Também é possível usar o operador : para selecionar desde uma coluna até outra. Crie um novo dataframe, chamado `dados_varsociais2`, que contém as colunas a partir de SEXO.GENERO até REGIAO.

```
dados_varsociais2 <- select(dados, SEXO.GENERO:REGIAO)
```

Você pode comparar os dois dfs em Environment: enquanto `dados_varsociais1` tem quatro colunas, `dados_varsociais2` tem uma coluna a mais, pois também inclui a variável FAIXA.ETARIA. Se quiser, veja novamente o dataframe `dados` completo, aberto em uma das janelas em Source.

Nesta lição, você aprendeu uma série de funções para manipular conjuntos de dados no R, seja um vetor, seja um dataframe: `ncol()`, `nrow()`, `str()`, `summary()`, `filter()`, `select()` – as duas últimas funções, do pacote dplyr. Aprendeu também alguns operadores lógicos do R: ==, !=, <, <=, &, |.

Você pode estar com a sensação de que entendeu o conteúdo, mas que não vai se lembrar de usar a função adequada quando realmente precisar... A solução para isso é praticar! Faça os exercícios correspondentes a esta lição!

Para saber mais

Para saber mais sobre as funções do pacote dplyr, recomendo visitar a página https://dplyr.tidyverse.org/. Se quiser praticar mais, recomendo que você faça as lições 3 ("Sequences of Numbers"), 4 ("Vectors"), 6 ("Subsetting Vectors") e 7 ("Matrices and Data Frames") do curso R Programming, do swirl. Recomendo também a leitura do capítulo 2 de Gries (2019).

Exercícios

1. Carregue o pacote tidyverse na memória do R.
2. Defina o diretório de trabalho como aquele em que se encontra a planilha DadosRT.csv.
3. Carregue o conjunto de dados da planilha DadosRT num dataframe chamado `dadosR`. Use a função `read_csv()` para isso, e especifique as colunas como "factor", exceto IDADE, que deve ser especificada como "integer", e INDICE.SOCIO e FREQUENCIA, que devem ser especificadas como "double".
4. Inspecione a estrutura do dataframe `dadosR` com a função `str()`.
5. A função `str()` fornece uma visão global das propriedades de um dataframe e, como tal, permite checar rapidamente se um conjunto de dados foi corretamente carregado e se sua planilha não tem problemas. Qual é o número de linhas no dataframe `dadosR`?
6. Qual é o número de colunas no dataframe `dadosR`?
7. Quantos fatores têm as variáveis FAIXA.ETARIA e ORIGEM.PAIS, respectivamente?
    a. 3 e 5
    b. 2 e 105

c. 7 e 15

8. Que tipo de variável é IDADE?
    a. Factor
    b. int
    c. num
9. As três últimas colunas de `dadosR` mostram a ocorrência da palavra com /r/ em coda com algumas palavras do contexto precedente e seguinte. Elas servem como base para a codificação das demais colunas, mas não são variáveis em si. Crie um novo dataframe, chamado `dadosR2`, com a exclusão dessas três colunas.
10. As funções `head()` e `tail()` mostram as primeiras e as últimas ocorrências de um objeto. Elas admitem mais um argumento, que é o número de elementos que se deseja visualizar. Veja a documentação da função `head()` na Ajuda.
11. O número *default* da função `head()` é 6, mas caso você queira visualizar as 15 primeiras linhas, basta incluir 15 como segundo argumento da função. Aplique a função `head()` a `dadosR2` para visualizar as 15 primeiras linhas desse dataframe.
12. Visualize o elemento que está na 45ª linha da 10ª coluna de `dadosR2`.
13. Visualize os dados da 5ª coluna de `dadosR2`.
14. Visualize os dados da variável TONICIDADE de `dadosR2`.
15. Visualize os 6 primeiros dados da variável TONICIDADE de `dadosR2`.
16. Visualize os 20 primeiros dados da variável TONICIDADE de `dadosR2`.
17. Guarde os 20 primeiros dados da variável TONICIDADE de `dadosR2` em um objeto chamado `tonicidade20`.
18. Que tipo de estrutura de dado é `tonicidade20`?
    a. Um dataframe
    b. Um vetor
    c. Um vetor e um dataframe

19. Acesse o 10º elemento do vetor `tonicidade20`.
20. Acesse o 5º e o 3º elementos do vetor `tonicidade20`.
21. Acesse o intervalo do 5º ao 10º elemento do vetor `tonicidade20`.
22. Aplique a função `length()` para saber quantos dados há na coluna CONT.FON.SEG de `dadosR2`.
23. Na prática, a função `length()` aplicada a uma coluna de um dataframe fornece o número de observações/linhas, uma vez que todas as colunas de um dataframe têm a mesma extensão. Quais outras funções também informam o número de observações/linhas de um dataframe?
    a. `head()` e `str()`
    b. `nrow()` e `str()`
    c. `nrow()` e `summary()`
    d. `str()` e `summary()`
24. Na sequência, serão criados vários subconjuntos de dados com a função `filter()`. Use o dataframe `dadosR2` em todos os casos. Crie um novo dataframe chamado `dados_1a` que contém todos os dados dos falantes de 1ª faixa etária. A variável relevante se chama FAIXA.ETARIA e a variante é "1a".
25. Crie um subconjunto de dados chamado `dados_1a2a` que contém todos os dados dos falantes de 1ª e 2ª faixas etárias. Use um operador lógico nessa linha de comando.
26. Crie um subconjunto de dados chamado `dados_menor45` com os dados dos falantes que têm menos de 45 anos. Se precisar olhar o conjunto de dados, use a função `View()` ou `str()` para visualizar os nomes relevantes de variáveis e variantes.
27. Crie um subconjunto de dados chamado `dados_menor_igual45` com os dados dos falantes que têm 45 anos ou menos.
28. Crie um subconjunto de dados chamado `dados_maior_igual46` com os dados dos falantes que têm 46 anos ou mais.

29. Crie um subconjunto de dados chamado `dados_1a_central` com os dados dos falantes de primeira faixa etária e que vivem na região central da cidade.
30. Crie um subconjunto de dados chamado `dados_1a_ou_central` com os dados dos falantes de primeira faixa etária *ou* que vivem na região central da cidade.
31. Crie um subconjunto de dados com os critérios que desejar e guarde-o em um objeto. Use a função `filter()` para isso. Cheque na aba Environment se o objeto foi criado corretamente.
32. Crie um novo dataframe chamado `dadosR2_varsociais`, que contém as colunas VD, PARTICIPANTE, SEXO.GENERO, IDADE, FAIXA.ETARIA, ESCOLARIDADE, REGIAO, INDICE.SOCIO e ORIGEM.PAIS.