Introdução ao Tidyverse

Tópicos Especiais em Meio Ambiente I

Prof. Claudiano Neto

2021-11-22

Carregando pacotes necessários

```
library(gapminder)
library(dplyr)
library(tidyverse)
```

Instalando e carregando pacote `gapminder`

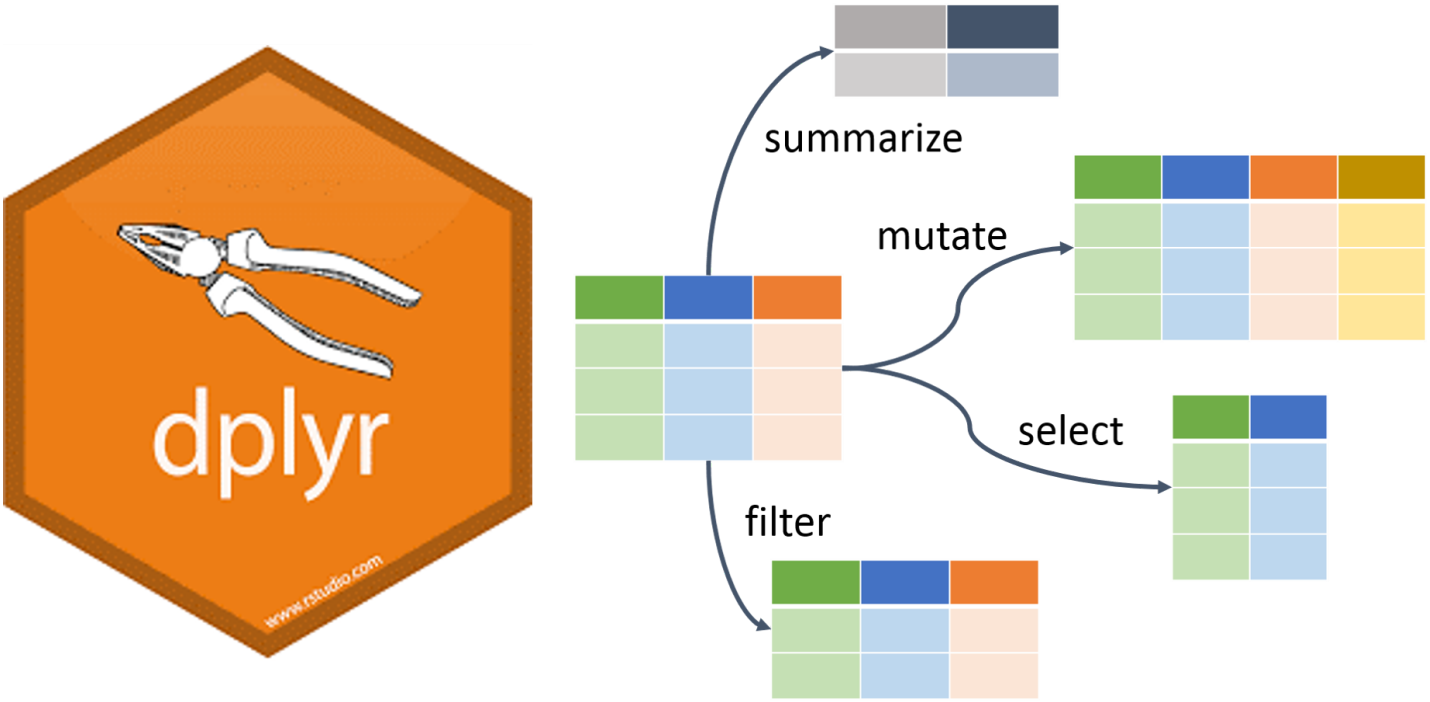
```
install.packages("gapminder")
library(gapminder)
```

```
gapminder # puxando a base de dados
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # ... with 1,694 more rows
```

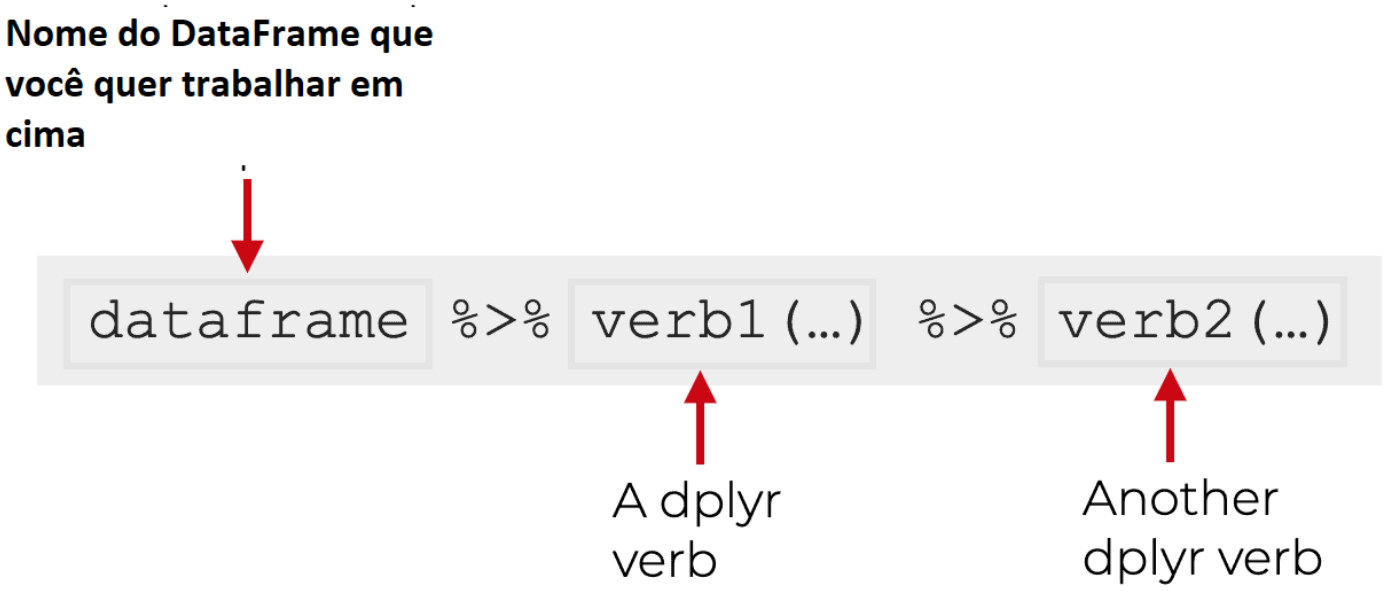
Conhecendo os verbos do Dplyr?

- Linhas: `filter()` , `slice()` , `arrange()`
- colunas: `select()` , `rename()` , `mutate()` , `relocate()`
- grupos de linhas: `summarise()` , `group_by()`

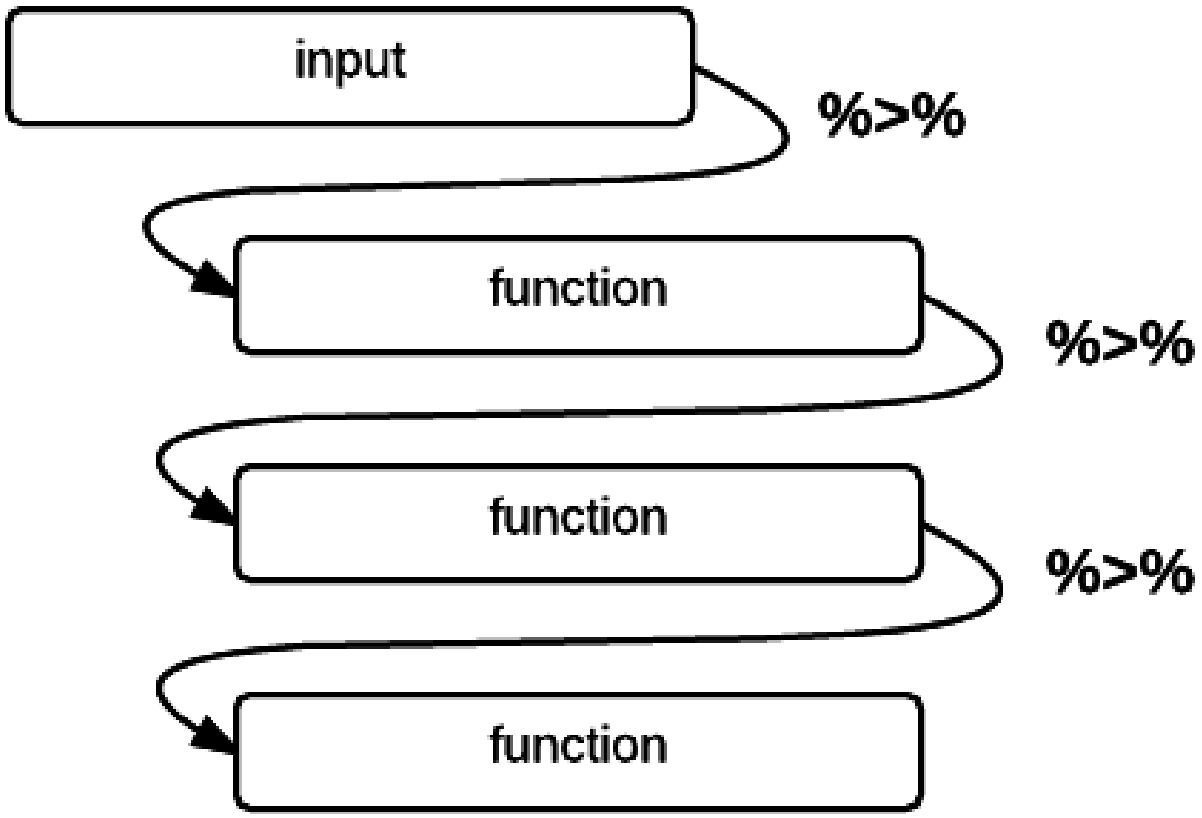


Aprendendo usar o Operador Pipe - %>%

O operador pipe tem a função de permitir o encadeamento de uma função após a outra sem precisar criar variáveis intermediárias ou utilizar parênteses intermináveis. A utilidade principal do pipe é melhorar a leitura do código.



Visto de outra forma, seria algo dessa forma.



E agora mostrando com um exemplo.

```r
starwars %>%
  mutate(bmi = mass/((height/100)^2)) %>%
  select(name, bmi, species) %>%
  group_by(species) %>%
  summarise(bmi = mean(bmi))
#> # A tibble: 38 x 2
#>    species     bmi
#>    <chr>     <dbl>
#>  1 Aleena     24.0
#>  2 Besalisk   26.0
#>  3 Cerean     20.9
#>  4 Chagrian   NA
#>  5 Clawdite   19.5
#>  6 Droid      NA
#>  7 Dug        31.9
#>  8 Ewok       25.8
#>  9 Geonosian  23.9
#> 10 Gungan     NA
#> # ... with 28 more rows
```

Vamos entender de ordenação dos dados

```r
tb <- tibble(x = 1:4,     # aqui estamos criando um objeto tibble
y = c(4, 7, 1, 3),
z = c(10, 10, 22, 22),
k = c(TRUE, FALSE, FALSE, TRUE),
u = c("A", "B", "A", "B"),
vazio = c("NA", 1 , 1, "NA"))
```

Pausa para uma dica - **`dput()`**

```
dput(tb) # vamos usar para recriar (de uma forma diferente) o objeto que estamos trabalh
ando
```

```
## structure(list(x = 1:4, y = c(4, 7, 1, 3), z = c(10, 10, 22,
## 22), k = c(TRUE, FALSE, FALSE, TRUE), u = c("A", "B", "A", "B"
## ), vazio = c("NA", "1", "1", "NA")), row.names = c(NA, -4L), class = c("tbl_df",
## "tbl", "data.frame"))
```

Ordenação crescente

```
tb %>% arrange(z, y) # observe que se trata de ordenação crescente
```

```
## # A tibble: 4 x 6
##       x     y     z k     u     vazio
##   <int> <dbl> <dbl> <lgl> <chr> <chr>
## 1     1     4    10 TRUE  A     NA
## 2     2     7    10 FALSE B     1
## 3     3     1    22 FALSE A     1
## 4     4     3    22 TRUE  B     NA
```

Ordenação descendente

```
tb %>% arrange(desc(z), y)
```

```
## # A tibble: 4 x 6
##       x     y     z k     u     vazio
##   <int> <dbl> <dbl> <lgl> <chr> <chr>
## 1     3     1    22 FALSE A     1
## 2     4     3    22 TRUE  B     NA
## 3     1     4    10 TRUE  A     NA
## 4     2     7    10 FALSE B     1
```

Como filtrar os dados - FILTER?

```
tb %>% filter(x > 2 | u == "A")  # lembre-se q barra vertical "|" significa OU
```

```
## # A tibble: 3 x 6
##       x     y     z k     u     vazio
##   <int> <dbl> <dbl> <lgl> <chr> <chr>
## 1     1     4    10 TRUE  A     NA
## 2     3     1    22 FALSE A     1
## 3     4     3    22 TRUE  B     NA
```

```
tb %>% filter(x > 2 & u == "A")  # lembre-se "&" significa OU
```

```
## # A tibble: 1 x 6
##       x     y     z k     u     vazio
##   <int> <dbl> <dbl> <lgl> <chr> <chr>
## 1     3     1    22 FALSE A     1
```

Como fatiar as linhas - SLICE?

```
tb %>% slice(1:3) # Somente as 3 primeiras linhas
```

```
## # A tibble: 3 x 6
##       x     y     z k     u     vazio
##   <int> <dbl> <dbl> <lgl> <chr> <chr>
## 1     1     4    10 TRUE  A     NA
## 2     2     7    10 FALSE B     1
## 3     3     1    22 FALSE A     1
```

```
tb %>% slice(-(3:5)) # somente as 2 primeiras linhas
```

```
## # A tibble: 2 x 6
##       x     y     z k     u     vazio
##   <int> <dbl> <dbl> <lgl> <chr> <chr>
## 1     1     4    10 TRUE  A     NA
## 2     2     7    10 FALSE B     1
```

Como selecionar variáveis/colunas?

Como selecionar as variáveis por listagem?

```
tb %>% select(x, y) # Inclusão.
```

```
## # A tibble: 4 x 2
##       x     y
##   <int> <dbl>
## 1     1     4
## 2     2     7
## 3     3     1
## 4     4     3
```

```
tb %>% select(-z) # Exclusão.
```

```
## # A tibble: 4 x 5
##       x     y k     u     vazio
##   <int> <dbl> <lgl> <chr> <chr>
## 1     1     4 TRUE  A     NA
## 2     2     7 FALSE B     1
## 3     3     1 FALSE A     1
## 4     4     3 TRUE  B     NA
```

```
tb %>% select(y:k) # Intervalo
```

```
## # A tibble: 4 x 3
##       y     z k
##   <dbl> <dbl> <lgl>
## 1     4    10 TRUE
## 2     7    10 FALSE
## 3     1    22 FALSE
## 4     3    22 TRUE
```

Como selecionar as variáveis por posição?

```
tb %>% select(3, 2, 1)
```

```
## # A tibble: 4 x 3
##       z     y     x
##   <dbl> <dbl> <int>
## 1    10     4     1
## 2    10     7     2
## 3    22     1     3
## 4    22     3     4
```

```
tb %>% select(-(1:2))
```

```
## # A tibble: 4 x 4
##       z k     u     vazio
##   <dbl> <lgl> <chr> <chr>
## 1    10 TRUE  A     NA
## 2    10 FALSE B     1
## 3    22 FALSE A     1
## 4    22 TRUE  B     NA
```

Como selecionar as variáveis por condição?

```
tb %>% select_if(is.numeric) # o que É numérico
```

```
## # A tibble: 4 x 3
##       x     y     z
##   <int> <dbl> <dbl>
## 1     1     4    10
## 2     2     7    10
## 3     3     1    22
## 4     4     3    22
```

```
tb %>% select_if(negate(is.numeric)) # o que NÃO é numérico
```

```
## # A tibble: 4 x 3
##   k     u     vazio
##   <lgl> <chr> <chr>
## 1 TRUE  A     NA
## 2 FALSE B     1
## 3 FALSE A     1
## 4 TRUE  B     NA
```

COMO *sortear* AS LINHAS?

```
tb %>% sample_n(size = 3) # por tamanho discreto
```

```
## # A tibble: 3 x 6
##       x     y     z k     u     vazio
##   <int> <dbl> <dbl> <lgl> <chr> <chr>
## 1     3     1    22 FALSE A     1
## 2     2     7    10 FALSE B     1
## 3     4     3    22 TRUE  B     NA
```

```
tb %>% sample_frac(size = 0.5) # por tamanho fracionado
```

```
## # A tibble: 2 x 6
##       x     y     z k     u     vazio
##   <int> <dbl> <dbl> <lgl> <chr> <chr>
## 1     1     4    10 TRUE  A     NA
## 2     3     1    22 FALSE A     1
```

Como realizar transformações - alterar e criar?

Para *modificar* uma variável, usamos *MUTATE()*

```
tb %>% mutate(x = x * 2,
              u = as_factor(u),
              j = median(x))
```

```
## # A tibble: 4 x 7
##       x     y     z k     u     vazio     j
##   <dbl> <dbl> <dbl> <lgl> <fct> <chr> <dbl>
## 1     2     4    10 TRUE  A     NA        5
## 2     4     7    10 FALSE B     1         5
## 3     6     1    22 FALSE A     1         5
## 4     8     3    22 TRUE  B     NA        5
```

Criando uma variável a partir de outra

```
tb %>% mutate(v = y * z^(x/4))
```

```
## # A tibble: 4 x 7
##       x     y     z k     u     vazio     v
##   <int> <dbl> <dbl> <lgl> <chr> <chr> <dbl>
## 1     1     4    10 TRUE  A     NA     7.11
## 2     2     7    10 FALSE B     1     22.1
## 3     3     1    22 FALSE A     1     10.2
## 4     4     3    22 TRUE  B     NA    66
```

Renomeando uma variável/coluna

```
# tb %>% rename(nomeNovo = NomeVelho)
tb %>% rename(nomeNovo = x)
```

```
## # A tibble: 4 x 6
##   nomeNovo     y     z k     u     vazio
##      <int> <dbl> <dbl> <lgl> <chr> <chr>
## 1        1     4    10 TRUE  A     NA
## 2        2     7    10 FALSE B     1
## 3        3     1    22 FALSE A     1
## 4        4     3    22 TRUE  B     NA
```

Como realizar medidas resumos?

Como agrupar por variáveis estratificadoras

```
tb %>% count(u) # lembre como era a variável "u"
```

```
## # A tibble: 2 x 2
##   u         n
##   <chr> <int>
## 1 A         2
## 2 B         2
```

Usando o “group_by” para agrupar variáveis

```
tb %>%
      group_by(u) %>%  # agrupando por u
      summarise(x_mean = mean(x), # summarizando
          y_range = max(y) - min(y),
          z_desv = sd(z))
```

```
## # A tibble: 2 x 4
##   u     x_mean y_range z_desv
##   <chr>  <dbl>   <dbl>  <dbl>
## 1 A          2       3   8.49
## 2 B          3       4   8.49
```

```
tb %>%
    group_by(u) %>%
    summarise_if(is.numeric, mean) # Se var for numérica, calcule média.
```

```
## # A tibble: 2 x 4
##   u         x     y     z
##   <chr> <dbl> <dbl> <dbl>
## 1 A         2   2.5    16
## 2 B         3   5      16
```

Como agrupar usando funções resumo vetoriais

```
tb %>%
    group_by(u) %>%
    summarise_if(is.numeric, funs(min, median, max))
```

```
## # A tibble: 2 x 10
##   u     x_min y_min z_min x_median y_median z_median x_max y_max z_max
##   <chr> <int> <dbl> <dbl>    <dbl>    <dbl>    <dbl> <int> <dbl> <dbl>
## 1 A         1     1    10        2      2.5       16     3     4    22
## 2 B         2     3    10        3      5         16     4     7    22
```

```
# O verbo FILTER permite selecionar uma parte especifica dos dados para realizar sua ana
lise. Deve ser precedido do *%>%* que significa (pegue o que estiver a frente e alimente
-o na proxima etapa).

gapminder %>%
  filter (year == 2007)
```

```
## # A tibble: 142 x 6
##    country     continent  year lifeExp       pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>     <int>     <dbl>
##  1 Afghanistan Asia       2007    43.8  31889923      975.
##  2 Albania     Europe     2007    76.4   3600523     5937.
##  3 Algeria     Africa     2007    72.3  33333216     6223.
##  4 Angola      Africa     2007    42.7  12420476     4797.
##  5 Argentina   Americas   2007    75.3  40301927    12779.
##  6 Australia   Oceania    2007    81.2  20434176    34435.
##  7 Austria     Europe     2007    79.8   8199783    36126.
##  8 Bahrain     Asia       2007    75.6    708573    29796.
##  9 Bangladesh  Asia       2007    64.1 150448339     1391.
## 10 Belgium     Europe     2007    79.4  10392226    33693.
## # ... with 132 more rows
```

Você poderia mudar o filtro para outra coisa

```
gapminder %>%
  filter(country == "United States")
```

```
## # A tibble: 12 x 6
##    country       continent  year lifeExp       pop gdpPercap
##    <fct>         <fct>     <int>   <dbl>     <int>     <dbl>
##  1 United States Americas   1952    68.4 157553000    13990.
##  2 United States Americas   1957    69.5 171984000    14847.
##  3 United States Americas   1962    70.2 186538000    16173.
##  4 United States Americas   1967    70.8 198712000    19530.
##  5 United States Americas   1972    71.3 209896000    21806.
##  6 United States Americas   1977    73.4 220239000    24073.
##  7 United States Americas   1982    74.6 232187835    25010.
##  8 United States Americas   1987    75.0 242803533    29884.
##  9 United States Americas   1992    76.1 256894189    32004.
## 10 United States Americas   1997    76.8 272911760    35767.
## 11 United States Americas   2002    77.3 287675526    39097.
## 12 United States Americas   2007    78.2 301139947    42952.
```

Podemos ter multiplas condições para o filtro, basta separa-las por “,” virgula.

```
gapminder %>%
  filter(year == 2007, country == "United States")
```

```
## # A tibble: 1 x 6
##   country       continent  year lifeExp       pop gdpPercap
##   <fct>         <fct>     <int>   <dbl>     <int>     <dbl>
## 1 United States Americas   2007    78.2 301139947    42952.
```

ARRANGE verbo

```
# O verbo "arrange" classifica uma tabela baseado em uma variavel (crescente ou decresce
nte). É muito util quando voce deseja conhecer os valores mais extremos do conjunto do b
anco de dados.

gapminder %>%
  arrange(gdpPercap)
```

```
## # A tibble: 1,704 x 6
##    country          continent  year lifeExp      pop gdpPercap
##    <fct>            <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Congo, Dem. Rep. Africa     2002    45.0 55379852      241.
##  2 Congo, Dem. Rep. Africa     2007    46.5 64606759      278.
##  3 Lesotho          Africa     1952    42.1   748747      299.
##  4 Guinea-Bissau    Africa     1952    32.5   580653      300.
##  5 Congo, Dem. Rep. Africa     1997    42.6 47798986      312.
##  6 Eritrea          Africa     1952    35.9  1438760      329.
##  7 Myanmar          Asia       1952    36.3 20092996      331
##  8 Lesotho          Africa     1957    45.0   813338      336.
##  9 Burundi          Africa     1952    39.0  2445618      339.
## 10 Eritrea          Africa     1957    38.0  1542611      344.
## # ... with 1,694 more rows
```

```
# Dentro do verbo arrange voce coloca a variavel que orientará a classificacao
```

Podemos organizar por ordem decrescente via arrange(desc())

```
gapminder %>%
  arrange(desc(gdpPercap))
```

```
## # A tibble: 1,704 x 6
##    country   continent  year lifeExp     pop gdpPercap
##    <fct>     <fct>     <int>   <dbl>   <int>     <dbl>
##  1 Kuwait    Asia       1957    58.0  212846   113523.
##  2 Kuwait    Asia       1972    67.7  841934   109348.
##  3 Kuwait    Asia       1952    55.6  160000   108382.
##  4 Kuwait    Asia       1962    60.5  358266    95458.
##  5 Kuwait    Asia       1967    64.6  575003    80895.
##  6 Kuwait    Asia       1977    69.3 1140357    59265.
##  7 Norway    Europe     2007    80.2 4627926    49357.
##  8 Kuwait    Asia       2007    77.6 2505559    47307.
##  9 Singapore Asia       2007    80.0 4553009    47143.
## 10 Norway    Europe     2002    79.0 4535591    44684.
## # ... with 1,694 more rows
```