ME115 - Linguagem R

Atividade Prática 07 - Gabarito

1º semestre de 2023

Introdução

Nessa atividade, exploraremos os seguintes tópicos:

1. Criação de *tibbles* usando o pacote `tibble`;
2. Formatação de banco de dados usando o pacote `tidyr`.

Antes de iniciar a atividade instale, se necessário, e carregue os pacotes `tibble`, `tidyr` e `reshape2`.

```
library(tibble)
library(tidyr)
library(reshape2)
```

Atividade

1. Crie o conjunto de dados chamado `nba` que seja um *tibble* com as colunas a seguir. Esse conjunto de dados está no formato *tidy*? Se não estiver, explique.

```
player <- c("James", "Durant", "Curry", "Harden", "Paul", "Wade")
team <- c("CLEOH", "GSWCA", "GSWCA", "HOUTX", "HOUTX", "CLEOH")
day1points <- c(25, 23, 30, 41, 26, 20)
day2points <- c(24, 25, 33, 45, 26, 23)
```

Solução:

```
nba <- tibble(player, team, day1points, day2points)
head(nba)
```

```
## # A tibble: 6 x 4
##   player team  day1points day2points
##   <chr>  <chr>      <dbl>      <dbl>
## 1 James  CLEOH         25         24
## 2 Durant GSWCA         23         25
## 3 Curry  GSWCA         30         33
## 4 Harden HOUTX         41         45
## 5 Paul   HOUTX         26         26
## 6 Wade   CLEOH         20         23
```

Não, pois temos duas colunas para a variável pontos(`day1points` e `day2points`), o correto seria transformar o conjunto de dados de forma que exista uma variável dia e pontos.

2. Transforme o conjunto de dados `nba` para o formato *tidy*, usando cada uma das funções abaixo. Note que as colunas `player` e `teams` estão fixas.

- `melt()` do pacote `reshape2` e armazene-o no objeto `nba_melt`.
- `gather()` do pacote `tidyr` e armazene-o no objeto `nba_gather`.
- `pivot_longer()` do pacote `tidyr` e armazene-o no objeto `nba_long`.

Solução:

```
nba_melt <- melt(nba, id.vars = -c(3,4), variable.name = "day", value.name = "points")
head(nba_melt)
```

```
##   player  team        day points
## 1  James CLEOH day1points     25
## 2 Durant GSWCA day1points     23
## 3  Curry GSWCA day1points     30
## 4 Harden HOUTX day1points     41
## 5   Paul HOUTX day1points     26
## 6   Wade CLEOH day1points     20
```

```
nba_melt1 <- melt(nba, id.vars = 1:2, variable.name = "day", value.name = "points")
head(nba_melt1)
```

```
##   player  team        day points
## 1  James CLEOH day1points     25
## 2 Durant GSWCA day1points     23
## 3  Curry GSWCA day1points     30
## 4 Harden HOUTX day1points     41
## 5   Paul HOUTX day1points     26
## 6   Wade CLEOH day1points     20
```

```
nba_gather <- gather(nba, key = day, value = points, c(day1points, day2points))
head(nba_gather)
```

```
## # A tibble: 6 x 4
##   player team  day        points
##   <chr>  <chr> <chr>       <dbl>
## 1 James  CLEOH day1points     25
## 2 Durant GSWCA day1points     23
## 3 Curry  GSWCA day1points     30
## 4 Harden HOUTX day1points     41
## 5 Paul   HOUTX day1points     26
## 6 Wade   CLEOH day1points     20
```

```
nba_long <- pivot_longer(nba, cols = 3:4, names_to = "day", values_to = "points")
head(nba_long)
```

```
## # A tibble: 6 x 4
##   player team  day        points
##   <chr>  <chr> <chr>       <dbl>
## 1 James  CLEOH day1points     25
## 2 James  CLEOH day2points     24
## 3 Durant GSWCA day1points     23
## 4 Durant GSWCA day2points     25
## 5 Curry  GSWCA day1points     30
## 6 Curry  GSWCA day2points     33
```

```
nba_long1 <- pivot_longer(nba, cols = starts_with("day"),
                          names_to = "day", values_to = "points")
head(nba_long1)
```

```
## # A tibble: 6 x 4
##   player team  day        points
##   <chr>  <chr> <chr>       <dbl>
## 1 James  CLEOH day1points     25
```

```
## 2 James  CLEOH day2points     24
## 3 Durant GSWCA day1points     23
## 4 Durant GSWCA day2points     25
## 5 Curry  GSWCA day1points     30
## 6 Curry  GSWCA day2points     33
```

3. As funções `pivot_wider()` e `spread()` são equivalentes. Ambas pegam diferentes valores de uma variável e os espalha em diferentes colunas. Imagine isso como o reverso de `pivot_longer()` e `gather()`, respectivamente. Então, transforme os objetos `nba_long` e `nba_gather` de volta para o formato *wide*.

   Solução:

```
nba_wider <- pivot_wider(nba_long, names_from = day, values_from = points)
head(nba_wider)
```

```
## # A tibble: 6 x 4
##   player team  day1points day2points
##   <chr>  <chr>      <dbl>      <dbl>
## 1 James  CLEOH         25         24
## 2 Durant GSWCA         23         25
## 3 Curry  GSWCA         30         33
## 4 Harden HOUTX         41         45
## 5 Paul   HOUTX         26         26
## 6 Wade   CLEOH         20         23
```

```
nba_spread <- spread(nba_gather, day, points)
head(nba_spread)
```

```
## # A tibble: 6 x 4
##   player team  day1points day2points
##   <chr>  <chr>      <dbl>      <dbl>
## 1 Curry  GSWCA         30         33
## 2 Durant GSWCA         23         25
## 3 Harden HOUTX         41         45
## 4 James  CLEOH         25         24
## 5 Paul   HOUTX         26         26
## 6 Wade   CLEOH         20         23
```

4. A função `separate()` pega valores dentro de uma coluna e os separa. Como você provavelmente já viu, a coluna `team` é um pouco estranha. Obviamente, há mais de uma variável lá e temos que consertá-la. O time e o estado dos EUA são nossas variáveis, então temos que criar duas colunas, uma para o time (`team`) e outra para o estado (`state`). O que o argumento `sep` representa?

   Solução:

```
separate(nba_long, col = team, into = c("team", "state"), sep = 3)
```

```
## # A tibble: 12 x 5
##    player team  state day        points
##    <chr>  <chr> <chr> <chr>       <dbl>
##  1 James  CLE   OH    day1points     25
##  2 James  CLE   OH    day2points     24
##  3 Durant GSW   CA    day1points     23
##  4 Durant GSW   CA    day2points     25
##  5 Curry  GSW   CA    day1points     30
##  6 Curry  GSW   CA    day2points     33
##  7 Harden HOU   TX    day1points     41
##  8 Harden HOU   TX    day2points     45
```

```
##  9 Paul   HOU   TX    day1points     26
## 10 Paul   HOU   TX    day2points     26
## 11 Wade   CLE   OH    day1points     20
## 12 Wade   CLE   OH    day2points     23
```

5. Crie o banco de dados `wide` como mostrado a seguir:

   Solução:

```
wide <- tibble(ID = 1:10,
               Face.1 = c(411, 723, 325, 456, 579, 612, 709, 513, 527, 379),
               Face.2 = c(123, 300, 400, 500, 600, 654, 789, 906, 413, 567),
               Face.3 = c(1457, 1000, 569, 896, 956, 2345, 780, 599, 1023, 678))
head(wide)
```

```
## # A tibble: 6 x 4
##      ID Face.1 Face.2 Face.3
##   <int>  <dbl>  <dbl>  <dbl>
## 1     1    411    123   1457
## 2     2    723    300   1000
## 3     3    325    400    569
## 4     4    456    500    896
## 5     5    579    600    956
## 6     6    612    654   2345
```

   Utizando esse conjuto de dados:

   a. Organize o banco de dados `wide` e guarde-o no objeto `long`, criando as colunas `Face` e `ResponseTime` usando a função `pivot_longer()`.

      Solução:

```
long <- pivot_longer(wide, cols = 2:4, names_to = "Face",
                     values_to = "ResponseTime")
head(long)
```

```
## # A tibble: 6 x 3
##      ID Face   ResponseTime
##   <int> <chr>         <dbl>
## 1     1 Face.1          411
## 2     1 Face.2          123
## 3     1 Face.3         1457
## 4     2 Face.1          723
## 5     2 Face.2          300
## 6     2 Face.3         1000
```

```
long <- pivot_longer(wide, cols = starts_with("Face"), names_to= "Face",
                     values_to = "ResponseTime")
head(long)
```

```
## # A tibble: 6 x 3
##      ID Face   ResponseTime
##   <int> <chr>         <dbl>
## 1     1 Face.1          411
## 2     1 Face.2          123
## 3     1 Face.3         1457
## 4     2 Face.1          723
## 5     2 Face.2          300
## 6     2 Face.3         1000
```

b. Usando a função `separate()`, separe o número da palavra "Face" na coluna `Face`, guardando a palavra "Face" na coluna `Target` e o número na coluna `Number`. Guarde o novo *data frame* no objeto `long_separate`.

Solução:

```
long_separate <- separate(long, Face, into = c("Target", "Number"), sep = "\\.")
head(long_separate)
```

```
## # A tibble: 6 x 4
##      ID Target Number ResponseTime
##   <int> <chr>  <chr>         <dbl>
## 1     1 Face   1               411
## 2     1 Face   2               123
## 3     1 Face   3              1457
## 4     2 Face   1               723
## 5     2 Face   2               300
## 6     2 Face   3              1000
```

```
long_separate <- separate(long, Face, c("Target", "Number"))
head(long_separate)
```

```
## # A tibble: 6 x 4
##      ID Target Number ResponseTime
##   <int> <chr>  <chr>         <dbl>
## 1     1 Face   1               411
## 2     1 Face   2               123
## 3     1 Face   3              1457
## 4     2 Face   1               723
## 5     2 Face   2               300
## 6     2 Face   3              1000
```

c. Usando a função `unite()` no objeto `long_separate`, junte o conteúdo de `Target` e `Number` e guarde em `Face`. Guarde o novo *data frame* no objeto `long_unite`.

Solução:

```
long_unite <- unite(long_separate, Face, Target, Number, sep = ".")
head(long_unite)
```

```
## # A tibble: 6 x 3
##      ID Face   ResponseTime
##   <int> <chr>         <dbl>
## 1     1 Face.1          411
## 2     1 Face.2          123
## 3     1 Face.3         1457
## 4     2 Face.1          723
## 5     2 Face.2          300
## 6     2 Face.3         1000
```

d. Utizando o objeto `long_unite` volte ao formato `wide` usando a função `pivot_wider()` e guarde o novo formato no objeto `back_to_wide`.

Solução:

```
back_to_wide <- pivot_wider(long_unite, names_from = Face,
                            values_from = ResponseTime)
head(back_to_wide)
```

```
## # A tibble: 6 x 4
##      ID Face.1 Face.2 Face.3
##   <int>  <dbl>  <dbl>  <dbl>
## 1     1    411    123   1457
## 2     2    723    300   1000
## 3     3    325    400    569
## 4     4    456    500    896
## 5     5    579    600    956
## 6     6    612    654   2345
```

6. Os dados `iris` não estão no formato *tidy*. Por quê? No último slide da aula, ficou a tarefa de colocarmos os dados `iris` no formato *tidy*. Para isso:

   - usar a função `pivot_longer()` para agrupar as colunas 1 a 4 em uma única coluna, com os valores em uma coluna chamada `Medida`;
   - usar a função `separate()` para separar a parte da planta e a dimensão em duas colunas.

   Solução:

```
iris_tidy <- pivot_longer(iris, cols = -Species, names_to = "Parte",
                          values_to ="Medida" )
head(iris_tidy)
```

```
## # A tibble: 6 x 3
##   Species Parte        Medida
##   <fct>   <chr>         <dbl>
## 1 setosa  Sepal.Length    5.1
## 2 setosa  Sepal.Width     3.5
## 3 setosa  Petal.Length    1.4
## 4 setosa  Petal.Width     0.2
## 5 setosa  Sepal.Length    4.9
## 6 setosa  Sepal.Width     3
```

```
iris_tidy <- separate(iris_tidy, Parte, into = c("Parte","Dimensao"), sep = '\\.')
head(iris_tidy)
```

```
## # A tibble: 6 x 4
##   Species Parte Dimensao Medida
##   <fct>   <chr> <chr>     <dbl>
## 1 setosa  Sepal Length      5.1
## 2 setosa  Sepal Width       3.5
## 3 setosa  Petal Length      1.4
## 4 setosa  Petal Width       0.2
## 5 setosa  Sepal Length      4.9
## 6 setosa  Sepal Width       3
```

Agradecimento

O material foi produzido pela Profa. Tatiana Benaglia para o curso de ME115.