Manipulação de Dados - Parte II

Paulo Henrique S. Guimarães

1 Transformação de dados e comandos básicos

A manipulação de dados pode ser definida como o ato de transformar, reestruturar, limpar, agregar e juntar os dados. Para se ter uma noção da importância dessa fase, alguns estudiosos da área de Ciência de Dados costumam afirmar que 80% do trabalho do cientista de dados é encontrar uma boa fonte de dados, limpar e preparar os dados, sendo que os 20% restantes seriam o trabalho de aplicar modelos e realizar alguma análise propriamente dita.

1.1 Funções básicas

1.1.1 Tibbles

Os *Tibbles* são *data frames* mais "modernos", sendo um dos recursos unificadores do *tidyverse*.

Muitas vezes o uso de *tibbles* e *data frames* podem ser utilizados de forma intercambiável.

- Aplicação:

```
library(tibble)
dados <- as_tibble(iris)
dados
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            1.4         0.2 setosa
##  3          4.7         3.2          1.3         0.2 setosa
##  4          4.6         3.1          1.5         0.2 setosa
##  5          5           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # ... with 140 more rows
```

- **int** - valor inteiro;
- **dbl** - valor real (*doubles*);
- **chr** - caracter (*strings*);
- **dttm** - datas - tempos (uma data + um horário);
- **lgl** - vetor lógico;
- **fctr** - fator;
- **date** - datas.

1.1.2 Operador *pipe*

O conceito de *pipe* existe pelo menos desde os anos 1970. Este operador foi introduzido por Stefan Milton Bache no pacote *magrittr* e já existem diversos pacotes construidos para facilitar a sua utilização. Basicamente, o operador %>% usa o resultado do seu lado esquerdo como primeiro argumento da função do lado direito. De acordo com seu criador, o operador tinha o objetivo de simplificar comandos cujos resultados deveriam ser passados para outros comandos.

- Aplicação:

```
soma<-function(x) {x+5}
div<- function(x) {x/5}
x<- 1:10
div(soma(x))
```

```
##  [1] 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0
```

```
# ou da forma

library(magrittr)
x %>% soma() %>% div()
```

```
##  [1] 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0
```

A grande vantagem do *pipe* não é só enxergar quais funções são aplicadas primeiro, mas sim nos ajudar a programar *pipelines* (encanamento em inglês) de tratamentos de dados.

1.2 Pacote dplyr - principais funções

O dplyr permite que façamos um código facilmente legível e compreensível, justamente pelo fato de usar verbos e também de permitir o encadeamento, que faz com que a sequência do código seja mais próxima da maneira com que pensamos. Além disso, o pacote é bem rápido, ainda que não seja tão rápido quanto o *data.table* (que não tem uma sintaxe tão amigável).

O dplyr cobre praticamente todas as tarefas básicas da manipulação de dados: agregar, sumarizar,filtrar, ordenar, criar variáveis,juntar, dentre outras.

- Autores: Hadley Wickham, Romain Francois e RStudio.
- Página: http://www.rpubs.com/marcosfs2006/dplyr

Figure 1: Pacote dplyr.

- Principais funções do pacote:
- **filter** - seleciona observações por seus valores (filter());
- **arrange** - reordena as linhas (arrange());
- **select** - seleciona variáveis por seus nomes (select());
- **mutate** - cria novas variáveis com funções de variáveis existes (mutate());
- **summarize** - reúne muitos valores em um único resumo (summarize()).

Juntas, todas estas funções facilitam o encadeamento de vários passos simples para alcançar um resultado complexo.

```
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------ tidyverse 1.2.1 --

## v ggplot2 3.2.1     v purrr   0.3.2
## v tidyr   1.0.0     v dplyr   0.8.3
## v readr   1.3.1     v stringr 1.4.0
## v ggplot2 3.2.1     v forcats 0.4.0

## -- Conflicts --------------------------------------------------- tidyverse_conflicts() --
## x tidyr::extract()   masks magrittr::extract()
## x dplyr::filter()    masks stats::filter()
## x dplyr::lag()       masks stats::lag()
## x purrr::set_names() masks magrittr::set_names()
```

```
data(iris)
head(iris,10)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1           5.1         3.5          1.4         0.2  setosa
## 2           4.9         3.0          1.4         0.2  setosa
## 3           4.7         3.2          1.3         0.2  setosa
## 4           4.6         3.1          1.5         0.2  setosa
## 5           5.0         3.6          1.4         0.2  setosa
## 6           5.4         3.9          1.7         0.4  setosa
## 7           4.6         3.4          1.4         0.3  setosa
## 8           5.0         3.4          1.5         0.2  setosa
## 9           4.4         2.9          1.4         0.2  setosa
## 10          4.9         3.1          1.5         0.1  setosa
```

```
tail(iris)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 145          6.7         3.3          5.7         2.5 virginica
## 146          6.7         3.0          5.2         2.3 virginica
## 147          6.3         2.5          5.0         1.9 virginica
## 148          6.5         3.0          5.2         2.0 virginica
## 149          6.2         3.4          5.4         2.3 virginica
## 150          5.9         3.0          5.1         1.8 virginica
```

```r
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

```r
# Vamos filtar todas as flores da espécie setosa que tenham o tamanho da sépala entre 4.0 e 5.2.

iris[iris$Species == "setosa"  &  iris$Sepal.Length > 4.5 & iris$Sepal.Length < 5.0, ]
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 2           4.9         3.0          1.4         0.2  setosa
## 3           4.7         3.2          1.3         0.2  setosa
## 4           4.6         3.1          1.5         0.2  setosa
## 7           4.6         3.4          1.4         0.3  setosa
## 10          4.9         3.1          1.5         0.1  setosa
## 12          4.8         3.4          1.6         0.2  setosa
## 13          4.8         3.0          1.4         0.1  setosa
## 23          4.6         3.6          1.0         0.2  setosa
## 25          4.8         3.4          1.9         0.2  setosa
## 30          4.7         3.2          1.6         0.2  setosa
## 31          4.8         3.1          1.6         0.2  setosa
## 35          4.9         3.1          1.5         0.2  setosa
## 38          4.9         3.6          1.4         0.1  setosa
## 46          4.8         3.0          1.4         0.3  setosa
## 48          4.6         3.2          1.4         0.2  setosa
```

```r
# ou da forma:
filter(dados,Species == "setosa" & Sepal.Length > 4.5 & Sepal.Length < 5.0)
```

```
## # A tibble: 15 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          4.9         3            1.4         0.2 setosa
##  2          4.7         3.2          1.3         0.2 setosa
##  3          4.6         3.1          1.5         0.2 setosa
##  4          4.6         3.4          1.4         0.3 setosa
##  5          4.9         3.1          1.5         0.1 setosa
##  6          4.8         3.4          1.6         0.2 setosa
##  7          4.8         3            1.4         0.1 setosa
##  8          4.6         3.6          1           0.2 setosa
```

```
##  9          4.8         3.4          1.9         0.2 setosa
## 10          4.7         3.2          1.6         0.2 setosa
## 11          4.8         3.1          1.6         0.2 setosa
## 12          4.9         3.1          1.5         0.2 setosa
## 13          4.9         3.6          1.4         0.1 setosa
## 14          4.8         3            1.4         0.3 setosa
## 15          4.6         3.2          1.4         0.2 setosa
```

- Outro exemplo:

```
iris %>% filter(Sepal.Width == 3)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1           4.9           3          1.4         0.2     setosa
## 2           4.8           3          1.4         0.1     setosa
## 3           4.3           3          1.1         0.1     setosa
## 4           5.0           3          1.6         0.2     setosa
## 5           4.4           3          1.3         0.2     setosa
## 6           4.8           3          1.4         0.3     setosa
## 7           5.9           3          4.2         1.5 versicolor
## 8           5.6           3          4.5         1.5 versicolor
## 9           6.6           3          4.4         1.4 versicolor
## 10          6.7           3          5.0         1.7 versicolor
## 11          5.4           3          4.5         1.5 versicolor
## 12          5.6           3          4.1         1.3 versicolor
## 13          6.1           3          4.6         1.4 versicolor
## 14          5.7           3          4.2         1.2 versicolor
## 15          7.1           3          5.9         2.1  virginica
## 16          6.5           3          5.8         2.2  virginica
## 17          7.6           3          6.6         2.1  virginica
## 18          6.8           3          5.5         2.1  virginica
## 19          6.5           3          5.5         1.8  virginica
## 20          6.1           3          4.9         1.8  virginica
## 21          7.2           3          5.8         1.6  virginica
## 22          7.7           3          6.1         2.3  virginica
## 23          6.0           3          4.8         1.8  virginica
## 24          6.7           3          5.2         2.3  virginica
## 25          6.5           3          5.2         2.0  virginica
## 26          5.9           3          5.1         1.8  virginica
```

```
dados %>%
filter(Species == "setosa" & Sepal.Length > 4.5 & Sepal.Length < 5.0)
```

```
## # A tibble: 15 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          4.9         3            1.4         0.2 setosa
##  2          4.7         3.2          1.3         0.2 setosa
##  3          4.6         3.1          1.5         0.2 setosa
##  4          4.6         3.4          1.4         0.3 setosa
##  5          4.9         3.1          1.5         0.1 setosa
##  6          4.8         3.4          1.6         0.2 setosa
```

```
##  7          4.8         3            1.4         0.1 setosa
##  8          4.6         3.6          1           0.2 setosa
##  9          4.8         3.4          1.9         0.2 setosa
## 10          4.7         3.2          1.6         0.2 setosa
## 11          4.8         3.1          1.6         0.2 setosa
## 12          4.9         3.1          1.5         0.2 setosa
## 13          4.9         3.6          1.4         0.1 setosa
## 14          4.8         3            1.4         0.3 setosa
## 15          4.6         3.2          1.4         0.2 setosa
```

A seleção de linhas a partir das posições das mesmas pode ser feita com a função *slice()*.

```
dados %>%
slice(c(1, 3, 5))
```

```
## # A tibble: 3 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1          5.1         3.5          1.4         0.2 setosa
## 2          4.7         3.2          1.3         0.2 setosa
## 3          5           3.6          1.4         0.2 setosa
```

```
slice(dados,n()) # seleciona a última linha
```

```
## # A tibble: 1 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1          5.9           3          5.1         1.8 virginica
```

A seleção de valores distintos é feita com a função *distinct()*. Esta função é semelhante à função *unique()*.

```
dados1<-dados%>%
distinct(Species,Petal.Length)
head(dados1)
```

```
## # A tibble: 6 x 2
##   Species Petal.Length
##   <fct>          <dbl>
## 1 setosa           1.4
## 2 setosa           1.3
## 3 setosa           1.5
## 4 setosa           1.7
## 5 setosa           1.6
## 6 setosa           1.1
```

A seleção de amostras pode ser feita com as funções *sample_n()* e *sample_frac()*.

```
amostra<-dados %>%
sample_n(5)
print(amostra)
```

```
## # A tibble: 5 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1          4.9         2.5          4.5         1.7 virginica
## 2          6.3         2.5          4.9         1.5 versicolor
## 3          7.6         3            6.6         2.1 virginica
## 4          4.6         3.6          1           0.2 setosa
## 5          5.7         4.4          1.5         0.4 setosa
```

```
# ou então

sample_n(iris, 5)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 1          5.1         3.8          1.6         0.2    setosa
## 2          6.4         2.7          5.3         1.9 virginica
## 3          6.9         3.1          5.1         2.3 virginica
## 4          7.2         3.2          6.0         1.8 virginica
## 5          6.5         3.0          5.5         1.8 virginica
```

```
sample_n(iris,5, replace = TRUE) # com reposição
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1          5.8         2.6          4.0         1.2 versicolor
## 2          6.0         2.2          5.0         1.5  virginica
## 3          6.3         2.9          5.6         1.8  virginica
## 4          5.2         3.5          1.5         0.2     setosa
## 5          5.7         3.8          1.7         0.3     setosa
```

```
sample_frac(iris,0.1) # amostra de 10% das observações.
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1           4.9         3.6          1.4         0.1     setosa
## 2           5.0         3.6          1.4         0.2     setosa
## 3           5.8         2.7          4.1         1.0 versicolor
## 4           4.6         3.4          1.4         0.3     setosa
## 5           5.0         3.0          1.6         0.2     setosa
## 6           6.6         3.0          4.4         1.4 versicolor
## 7           5.8         2.6          4.0         1.2 versicolor
## 8           6.7         3.3          5.7         2.5  virginica
## 9           4.9         3.1          1.5         0.1     setosa
## 10          5.9         3.0          5.1         1.8  virginica
## 11          7.1         3.0          5.9         2.1  virginica
## 12          5.6         3.0          4.5         1.5 versicolor
## 13          5.2         3.4          1.4         0.2     setosa
## 14          6.3         2.5          5.0         1.9  virginica
## 15          5.5         4.2          1.4         0.2     setosa
```

1.2.1 Linhas ordenadas segundo um variável

```
dados2<-dados[order(dados$Petal.Length), ]
print(dados2)
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          4.6         3.6          1           0.2 setosa
##  2          4.3         3            1.1         0.1 setosa
##  3          5.8         4            1.2         0.2 setosa
##  4          5           3.2          1.2         0.2 setosa
##  5          4.7         3.2          1.3         0.2 setosa
##  6          5.4         3.9          1.3         0.4 setosa
##  7          5.5         3.5          1.3         0.2 setosa
##  8          4.4         3            1.3         0.2 setosa
##  9          5           3.5          1.3         0.3 setosa
## 10          4.5         2.3          1.3         0.3 setosa
## # ... with 140 more rows
```

Ou da forma:

```
dados %>%
  arrange(Petal.Length)
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          4.6         3.6          1           0.2 setosa
##  2          4.3         3            1.1         0.1 setosa
##  3          5.8         4            1.2         0.2 setosa
##  4          5           3.2          1.2         0.2 setosa
##  5          4.7         3.2          1.3         0.2 setosa
##  6          5.4         3.9          1.3         0.4 setosa
##  7          5.5         3.5          1.3         0.2 setosa
##  8          4.4         3            1.3         0.2 setosa
##  9          5           3.5          1.3         0.3 setosa
## 10          4.5         2.3          1.3         0.3 setosa
## # ... with 140 more rows
```

A função *select* pode ser utilizada para selecionar colunas de um *data frame*. Por exemplo, vamos selecionar colunas que comecem com s e terminem com h:

```
dados3 <- select(dados, starts_with('s'), ends_with('h'))
head(dados3)
```

```
## # A tibble: 6 x 5
##   Sepal.Length Sepal.Width Species Petal.Length Petal.Width
##          <dbl>       <dbl> <fct>          <dbl>       <dbl>
## 1          5.1         3.5 setosa           1.4         0.2
## 2          4.9         3   setosa           1.4         0.2
## 3          4.7         3.2 setosa           1.3         0.2
## 4          4.6         3.1 setosa           1.5         0.2
## 5          5           3.6 setosa           1.4         0.2
## 6          5.4         3.9 setosa           1.7         0.4
```

2 Criando e modificando variáveis em um conjunto de dados

```
dados4 <- iris %>% mutate(Dim = Sepal.Length/Petal.Length)
head(dados4)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species      Dim
## 1          5.1         3.5          1.4         0.2  setosa 3.642857
## 2          4.9         3.0          1.4         0.2  setosa 3.500000
## 3          4.7         3.2          1.3         0.2  setosa 3.615385
## 4          4.6         3.1          1.5         0.2  setosa 3.066667
## 5          5.0         3.6          1.4         0.2  setosa 3.571429
## 6          5.4         3.9          1.7         0.4  setosa 3.176471
```

```
dados5 <- dados4 %>% select(-Dim)
head(dados5)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

- Mais um exemplo - *dataset* **starwars**

```
library(dplyr)
glimpse(starwars)
```

```
## Observations: 87
## Variables: 13
## $ name       <chr> "Luke Skywalker", "C-3PO", "R2-D2", "Darth Vader", ...
## $ height     <int> 172, 167, 96, 202, 150, 178, 165, 97, 183, 182, 188...
## $ mass       <dbl> 77.0, 75.0, 32.0, 136.0, 49.0, 120.0, 75.0, 32.0, 8...
## $ hair_color <chr> "blond", NA, NA, "none", "brown", "brown, grey", "b...
## $ skin_color <chr> "fair", "gold", "white, blue", "white", "light", "l...
## $ eye_color  <chr> "blue", "yellow", "red", "yellow", "brown", "blue",...
## $ birth_year <dbl> 19.0, 112.0, 33.0, 41.9, 19.0, 52.0, 47.0, NA, 24.0...
## $ gender     <chr> "male", NA, NA, "male", "female", "male", "female",...
## $ homeworld  <chr> "Tatooine", "Tatooine", "Naboo", "Tatooine", "Alder...
## $ species    <chr> "Human", "Droid", "Droid", "Human", "Human", "Human...
## $ films      <list> [<"Revenge of the Sith", "Return of the Jedi", "Th...
## $ vehicles   <list> [<"Snowspeeder", "Imperial Speeder Bike">, <>, <>,...
## $ starships  <list> [<"X-wing", "Imperial shuttle">, <>, <>, "TIE Adva...
```

```
select(starwars, name, height, mass)
```

```
## # A tibble: 87 x 3
##    name               height  mass
##    <chr>               <int> <dbl>
```

```
##  1 Luke Skywalker        172    77
##  2 C-3PO                 167    75
##  3 R2-D2                  96    32
##  4 Darth Vader           202   136
##  5 Leia Organa           150    49
##  6 Owen Lars             178   120
##  7 Beru Whitesun lars    165    75
##  8 R5-D4                  97    32
##  9 Biggs Darklighter     183    84
## 10 Obi-Wan Kenobi        182    77
## # ... with 77 more rows
```

```
select(starwars, 1:3)
```

```
## # A tibble: 87 x 3
##    name               height  mass
##    <chr>               <int> <dbl>
##  1 Luke Skywalker        172    77
##  2 C-3PO                 167    75
##  3 R2-D2                  96    32
##  4 Darth Vader           202   136
##  5 Leia Organa           150    49
##  6 Owen Lars             178   120
##  7 Beru Whitesun lars    165    75
##  8 R5-D4                  97    32
##  9 Biggs Darklighter     183    84
## 10 Obi-Wan Kenobi        182    77
## # ... with 77 more rows
```

É possível remover colunas. Por exemplo, queremos selecionar todas as colunas do nosso *tibble*, exceto a coluna height:

```
select(starwars, -height)
```

```
## # A tibble: 87 x 12
##    name   mass hair_color skin_color eye_color birth_year gender homeworld
##    <chr> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>  <chr>
##  1 Luke~    77 blond      fair       blue            19   male   Tatooine
##  2 C-3PO    75 <NA>       gold       yellow         112   <NA>   Tatooine
##  3 R2-D2    32 <NA>       white, bl~ red             33   <NA>   Naboo
##  4 Dart~   136 none       white      yellow          41.9 male   Tatooine
##  5 Leia~    49 brown      light      brown           19   female Alderaan
##  6 Owen~   120 brown, gr~ light      blue            52   male   Tatooine
##  7 Beru~    75 brown      light      blue            47   female Tatooine
##  8 R5-D4    32 <NA>       white, red red             NA   <NA>   Tatooine
##  9 Bigg~    84 black      light      brown           24   male   Tatooine
## 10 Obi-~    77 auburn, w~ fair       blue-gray       57   male   Stewjon
## # ... with 77 more rows, and 4 more variables: species <chr>,
## #   films <list>, vehicles <list>, starships <list>
```

Lembrando que a função **filter** recebe uma ou mais condições lógicas e retorna as linhas do *tibble* que atendam o solicitado.

- Exemplos de aplicação

```
filter(starwars, is.na(hair_color))
```

```
## # A tibble: 5 x 13
##   name  height  mass hair_color skin_color eye_color birth_year gender
##   <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
## 1 C-3PO    167    75 <NA>       gold       yellow           112 <NA>
## 2 R2-D2     96    32 <NA>       white, bl~ red               33 <NA>
## 3 R5-D4     97    32 <NA>       white, red red               NA <NA>
## 4 Gree~    173    74 <NA>       green      black             44 male
## 5 Jabb~    175  1358 <NA>       green-tan~ orange           600 herma~
## # ... with 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

```
filter(starwars, is.na(hair_color), species == "Droid") # & (E) e | (OU)
```

```
## # A tibble: 3 x 13
##   name  height  mass hair_color skin_color eye_color birth_year gender
##   <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
## 1 C-3PO    167    75 <NA>       gold       yellow           112 <NA>
## 2 R2-D2     96    32 <NA>       white, bl~ red               33 <NA>
## 3 R5-D4     97    32 <NA>       white, red red               NA <NA>
## # ... with 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

```
select(filter(starwars, mass > 130), name:mass, homeworld)
```

```
## # A tibble: 5 x 4
##   name                  height  mass homeworld
##   <chr>                  <int> <dbl> <chr>
## 1 Darth Vader              202   136 Tatooine
## 2 Jabba Desilijic Tiure    175  1358 Nal Hutta
## 3 IG-88                    200   140 <NA>
## 4 Grievous                 216   159 Kalee
## 5 Tarfful                  234   136 Kashyyyk
```

```
starwars %>%
  filter(mass > 130) %>%
  select(name:mass, homeworld)
```

```
## # A tibble: 5 x 4
##   name                  height  mass homeworld
##   <chr>                  <int> <dbl> <chr>
## 1 Darth Vader              202   136 Tatooine
## 2 Jabba Desilijic Tiure    175  1358 Nal Hutta
## 3 IG-88                    200   140 <NA>
## 4 Grievous                 216   159 Kalee
## 5 Tarfful                  234   136 Kashyyyk
```

Com o **mutate** é possível criar novas colunas e essas novas colunas podem ser criadas em função das já existentes.

```r
starwars %>%
  mutate(altura_metros = height/100,
         IMC = mass/(altura_metros^2)) %>%
  select(name, IMC)
```

```
## # A tibble: 87 x 2
##    name                 IMC
##    <chr>              <dbl>
##  1 Luke Skywalker      26.0
##  2 C-3PO               26.9
##  3 R2-D2               34.7
##  4 Darth Vader         33.3
##  5 Leia Organa         21.8
##  6 Owen Lars           37.9
##  7 Beru Whitesun lars  27.5
##  8 R5-D4               34.0
##  9 Biggs Darklighter   25.1
## 10 Obi-Wan Kenobi      23.2
## # ... with 77 more rows
```

O **summarise (ou summarize)** permite que usemos funções de sumarização, ou seja, funções que recebem n elementos e retornam apenas 1 valor.

```r
starwars %>%
  summarise(max_altura = max(height, na.rm = T),
            massa_media = mean(mass, na.rm = T))
```

```
## # A tibble: 1 x 2
##   max_altura massa_media
##        <int>       <dbl>
## 1        264        97.3
```

Vamos usar o **arrange** para ordenar o tibble, de maneira crescente ou decrescente.

```r
starwars %>%
  mutate(altura_metros = height/100,
         IMC = mass/(altura_metros^2)) %>%
  select(name, height, mass, IMC) %>%
  arrange(IMC)
```

```
## # A tibble: 87 x 4
##    name          height  mass   IMC
##    <chr>          <int> <dbl> <dbl>
##  1 Wat Tambor       193    48  12.9
##  2 Adi Gallia       184    50  14.8
##  3 Sly Moore        178    48  15.1
##  4 Roos Tarpals     224    82  16.3
##  5 Padmé Amidala    165    45  16.5
##  6 Lama Su          229    88  16.8
##  7 Jar Jar Binks    196    66  17.2
##  8 Ayla Secura      178    55  17.4
```

```
##  9 Shaak Ti         178    57  18.0
## 10 Barriss Offee    166    50  18.1
## # ... with 77 more rows
```

```
starwars %>%
  mutate(altura_metros = height/100,
         IMC = mass/(altura_metros^2)) %>%
  select(name, height, mass, IMC) %>%
  arrange(-IMC) # Ou arrange(desc(IMC))
```

```
## # A tibble: 87 x 4
##    name                  height  mass   IMC
##    <chr>                  <int> <dbl> <dbl>
##  1 Jabba Desilijic Tiure    175  1358 443.
##  2 Dud Bolt                  94    45  50.9
##  3 Yoda                      66    17  39.0
##  4 Owen Lars                178   120  37.9
##  5 IG-88                    200   140  35
##  6 R2-D2                     96    32  34.7
##  7 Grievous                 216   159  34.1
##  8 R5-D4                     97    32  34.0
##  9 Jek Tono Porkins         180   110  34.0
## 10 Darth Vader              202   136  33.3
## # ... with 77 more rows
```

- Por fim, podemos realizar todas as operações por grupos. Para isso existe a função **group_by**. Faremos então o cálculo da maior altura e a massa média por espécie:

```
starwars %>%
  group_by(species) %>%
  summarise(max_altura = max(height, na.rm = T),
            massa_media = mean(mass, na.rm = T))
```

```
## # A tibble: 38 x 3
##    species   max_altura massa_media
##    <chr>          <int>       <dbl>
##  1 Aleena            79        15
##  2 Besalisk         198       102
##  3 Cerean           198        82
##  4 Chagrian         196       NaN
##  5 Clawdite         168        55
##  6 Droid            200        69.8
##  7 Dug              112        40
##  8 Ewok              88        20
##  9 Geonosian        183        80
## 10 Gungan           224        74
## # ... with 28 more rows
```

2.1 Merging

O pacote dplyr dispõe também de um conjunto de funções que possibilitam realizar o merging de dois data frames, de forma semelhante à função merge(). Estas funções são:

- inner_join()
- left_join()
- right_join()
- full_join()
- semi_join()
- anti_join()

2.1.1 Juntando duas tabelas em uma

Em muitas situações durante uma análise de dados é comum trabalhar com mais de uma tabela, sendo assim pode ser necessário usar ferramentas para as combinar em uma tabela em uma única.

O dplyr oferece várias funções para realizar a junção de duas tabelas, que são a família x_join(). Todas seguem a mesma sintaxe: x_join(x, y, by), em que x e y são os dois dataframes a serem juntados e by é um vetor de caracteres especificando a coluna que será usada como chave.

```
library(tidyverse)
library(dplyr)

vendedor <- tibble(id = c("A12", "A13", "A14", "A15"),
regiao = c("Sul", "Sudeste", "Oeste", "Norte"),
experiencia = c(5, 2, 12, 8))
vendas = tibble(id = c("A13", "A14", "A12", "A11"),
vendas = c(1200, 2500, 350, 1000))

left_join(vendedor, vendas)
```

```
## Joining, by = "id"
```

```
## # A tibble: 4 x 4
##   id    regiao  experiencia vendas
##   <chr> <chr>         <dbl>  <dbl>
## 1 A12   Sul               5    350
## 2 A13   Sudeste           2   1200
## 3 A14   Oeste            12   2500
## 4 A15   Norte             8     NA
```

```
left_join(vendedor, vendas, by = "id")
```

```
## # A tibble: 4 x 4
##   id    regiao  experiencia vendas
##   <chr> <chr>         <dbl>  <dbl>
## 1 A12   Sul               5    350
## 2 A13   Sudeste           2   1200
## 3 A14   Oeste            12   2500
## 4 A15   Norte             8     NA
```

A função left_join(x, y): retorna todas as observações em x, independentemente se existem correspondentes (de acordo com as especificações em by) ou não.

```r
inner_join(vendedor, vendas)
```

```
## Joining, by = "id"
```

```
## # A tibble: 3 x 4
##   id    regiao  experiencia vendas
##   <chr> <chr>         <dbl>  <dbl>
## 1 A12   Sul               5    350
## 2 A13   Sudeste           2   1200
## 3 A14   Oeste            12   2500
```

```r
# ou da forma:

vendedor %>% left_join(vendas)
```

```
## Joining, by = "id"
```

```
## # A tibble: 4 x 4
##   id    regiao  experiencia vendas
##   <chr> <chr>         <dbl>  <dbl>
## 1 A12   Sul               5    350
## 2 A13   Sudeste           2   1200
## 3 A14   Oeste            12   2500
## 4 A15   Norte             8     NA
```

A função inner_join(x, y): retornar apenas observações que correspondem tanto em x como em y. Note como o vendedor A15 não consta no dataframe final, pois ele não está presente no dataframe y.

```r
inner_join(vendedor, vendas)
```

```
## Joining, by = "id"
```

```
## # A tibble: 3 x 4
##   id    regiao  experiencia vendas
##   <chr> <chr>         <dbl>  <dbl>
## 1 A12   Sul               5    350
## 2 A13   Sudeste           2   1200
## 3 A14   Oeste            12   2500
```

```r
vendedor %>% inner_join(vendas)
```

```
## Joining, by = "id"
```

```
## # A tibble: 3 x 4
##   id    regiao  experiencia vendas
##   <chr> <chr>         <dbl>  <dbl>
## 1 A12   Sul               5    350
## 2 A13   Sudeste           2   1200
## 3 A14   Oeste            12   2500
```

A função full_join(x, y) é mais completa e retorna todas as observações presentes em x e y. full_join(vendedor, vendas, by = "id")

```
vendedor %>% full_join(vendas)
```

```
## Joining, by = "id"

## # A tibble: 5 x 4
##   id    regiao  experiencia vendas
##   <chr> <chr>         <dbl>  <dbl>
## 1 A12   Sul               5    350
## 2 A13   Sudeste           2   1200
## 3 A14   Oeste            12   2500
## 4 A15   Norte             8     NA
## 5 A11   <NA>             NA   1000
```

Outras funções:

```
right_join(vendedor, vendas, by = "id")
```

```
## # A tibble: 4 x 4
##   id    regiao  experiencia vendas
##   <chr> <chr>         <dbl>  <dbl>
## 1 A13   Sudeste           2   1200
## 2 A14   Oeste            12   2500
## 3 A12   Sul               5    350
## 4 A11   <NA>             NA   1000
```

```
# semi_join(vendedor,vendas)

# anti_join(vendedor,vendas)

# nest_join(vendedor,vendas)
```

Agora veja as seguintes funções para operações com conjuntos:

```
df1<-tibble(x = c(1:10), y = c("a","b","c","a","b","c","a","b","c","d"))
df1
```

```
## # A tibble: 10 x 2
##        x y
##    <int> <chr>
##  1     1 a
##  2     2 b
##  3     3 c
##  4     4 a
##  5     5 b
##  6     6 c
##  7     7 a
##  8     8 b
##  9     9 c
## 10    10 d
```

```r
df2<-tibble(x = c(5:14), y = c("a","a","a","a","a","a","a","a","a","a"))
df2
```

```
## # A tibble: 10 x 2
##        x y
##    <int> <chr>
##  1     5 a
##  2     6 a
##  3     7 a
##  4     8 a
##  5     9 a
##  6    10 a
##  7    11 a
##  8    12 a
##  9    13 a
## 10    14 a
```

```r
intersect(df1,df2)
```

```
## # A tibble: 1 x 2
##       x y
##   <int> <chr>
## 1     7 a
```

```r
union(df1,df2)
```

```
## # A tibble: 19 x 2
##        x y
##    <int> <chr>
##  1     1 a
##  2     2 b
##  3     3 c
##  4     4 a
##  5     5 b
##  6     6 c
##  7     7 a
##  8     8 b
##  9     9 c
## 10    10 d
## 11     5 a
## 12     6 a
## 13     8 a
## 14     9 a
## 15    10 a
## 16    11 a
## 17    12 a
## 18    13 a
## 19    14 a
```

```r
setdiff(df1,df2) # retorna observações em x, mas não em y.
```

```
## # A tibble: 9 x 2
##       x y
##   <int> <chr>
## 1     1 a
## 2     2 b
## 3     3 c
## 4     4 a
## 5     5 b
## 6     6 c
## 7     8 b
## 8     9 c
## 9    10 d
```

3 Referência

1) Livro: R for Data Science - Hadley Wickham & Garrett Grolemund. Alta Books, 2019.