APRENDENDO SOBRE O DPLYR: COMO O DPLYR PODE SER EFICIENTE E INTUITIVO

Luana Sílvia dos Santos

10 de novembro de 2018

ABG CONSULTORIA ESTATÍSTICA

Introdução

- O dplyr foi desenvolvido por Hadley Wickham, Romain François, Lionel Henry e Kirill Müller
- É um pacote utilizado para manipulação de dados
- Ele é bastante rápido e apresenta uma sintaxe intuitiva
- Suas funções são feitas em forma de verbo, sendo auto-explicativas
- Ele faz todas as tarefas básicas de manipulação de dados: filtrar, ordenar, agregar, sumarizar, criar variáveis, juntar dois ou mais bancos, etc

Operador *Pipe*

- Podemos fazer várias operações em sequência com o auxílio do operador %>% (*pipe operator*)
- Ao invés de escrever f (x), podemos escrever x %>% f()
- Uma receita que tenha as seguintes instruções: junte os ingredientes, misture e leve ao forno
- Forma usual: **forno(misture(junte(ingredientes)))**
- Com o *pipe*: **ingredientes %>% junte %>% misture %>% forno**

Principais Funções

- **mutate()** cria novas variáveis a partir de outras existentes
- **select()** seleciona variáveis a partir dos seus nomes
- **filter()** seleciona linhas a partir dos seus valores
- **summarise()** reduz vários valores em um único sumário
- **group_by ()** faz as operações por grupo
- **join ()** junta dois bancos de dados
- **arrange()** muda a ordem das linhas

Banco de Dados IMC

- Banco de dados contendo gênero, peso, altura e IMC de 500 pessoas
- Disponível no kaggle
- O objetivo é comparar o IMC entre os gêneros

```
> dados1<- read.csv("C:/Users/Luana/Desktop/Palestra R ladies/dados1.csv")
> head(dados1)
  Gender Height Weight Index
1   Male    174     96     4
2   Male    189     87     2
3 Female    185    110     4
4 Female    195    104     3
5   Male    149     61     3
6   Male    189    104     3
```

Index:

0 - Extremely Weak

1 - Weak

2 - Normal

3 - Overweight

4 - Obesity

5 - Extreme Obesity

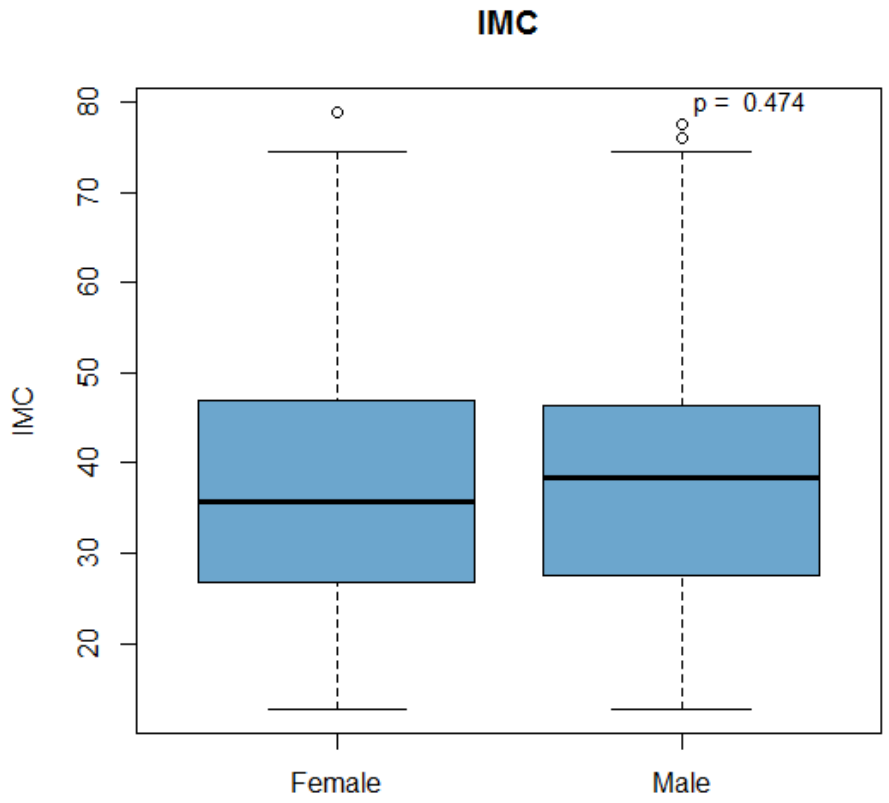
Função **mutate()**

- Essa função cria novas variáveis a partir de outras existentes

```
> # Vamos criar a variável IMC
> dt1<- mutate(dados1, IMC = dados1$Weight /(dados1$Height/100)^2)
> head(dt1)
Gender Height Weight Index      IMC
1     Male    174     96     4 31.70828
2     Male    189     87     2 24.35542
3   Female    185    110     4 32.14025
4   Female    195    104     3 27.35043
> # Agora usando o pipe
> dt1<- dados1 %>% mutate(IMC = Weight /(Height/100)^2)
> head(dt1)
Gender Height Weight Index      IMC
1     Male    174     96     4 31.70828
2     Male    189     87     2 24.35542
3   Female    185    110     4 32.14025
4   Female    195    104     3 27.35043
```

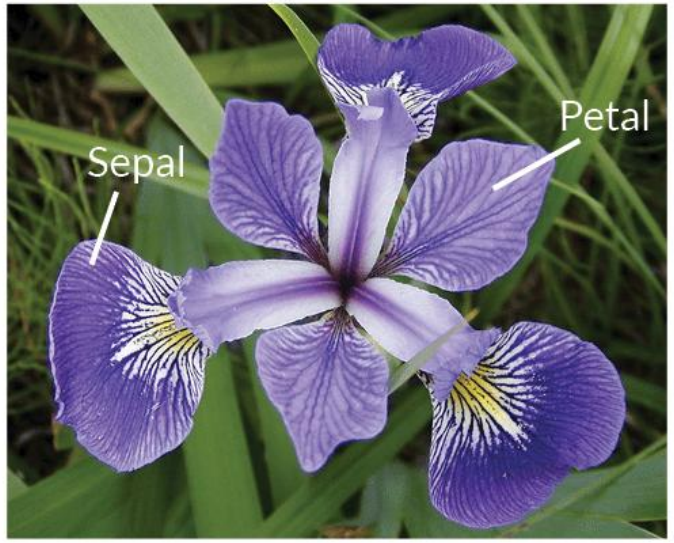
Função **mutate()**

```
> # Vamos verificar se o IMC é diferente entre os sexos
> boxplot(dt1$IMC ~ dt1$Gender, col="skyblue3", main="IMC", ylab="IMC")
> p<- wilcox.test(dt1$IMC ~ dt1$Gender, paired=F)$p.value
> text(2.2, 80, paste("p = ", round(p,3)), cex=0.9)
```



Banco de Dados *iris*

- Medidas de comprimento e largura da sépala e da pétala de 50 flores de 3 espécies de íris: *Iris setosa*, *versicolor* e *virginica,* disponível no R
- Objetivo: Comparar comprimento e largura das pétalas entre as espécies



Iris Versicolor

Iris Setosa

Iris Virginica

Função **select()**

- Essa função seleciona variáveis a partir dos seus nomes, usando as seguintes funções:
- **starts_with():** começa com um prefixo
- **ends_with():** termina com um prefixo
- **contains():** contém uma string
- **matches():** corresponde a uma expressão regular
- **everything():** todas as variáveis

Função **select()**

```
> dados2<- iris
> head(dados2)
  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
> # Selecionando apenas as colunas que começam com "Sepal"
> dt1<- dados2 %>% select(starts_with("Sepal"))
> head(dt1)
  Sepal.Length Sepal.Width
1          5.1         3.5
2          4.9         3.0
3          4.7         3.2
4          4.6         3.1
5          5.0         3.6
6          5.4         3.9
```

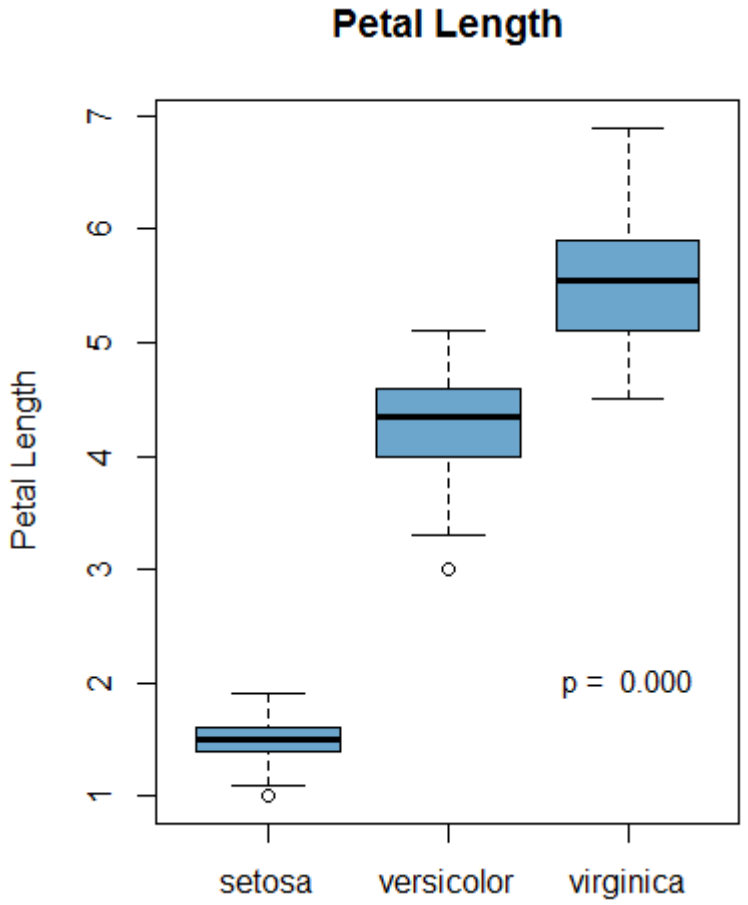
Função **select()**

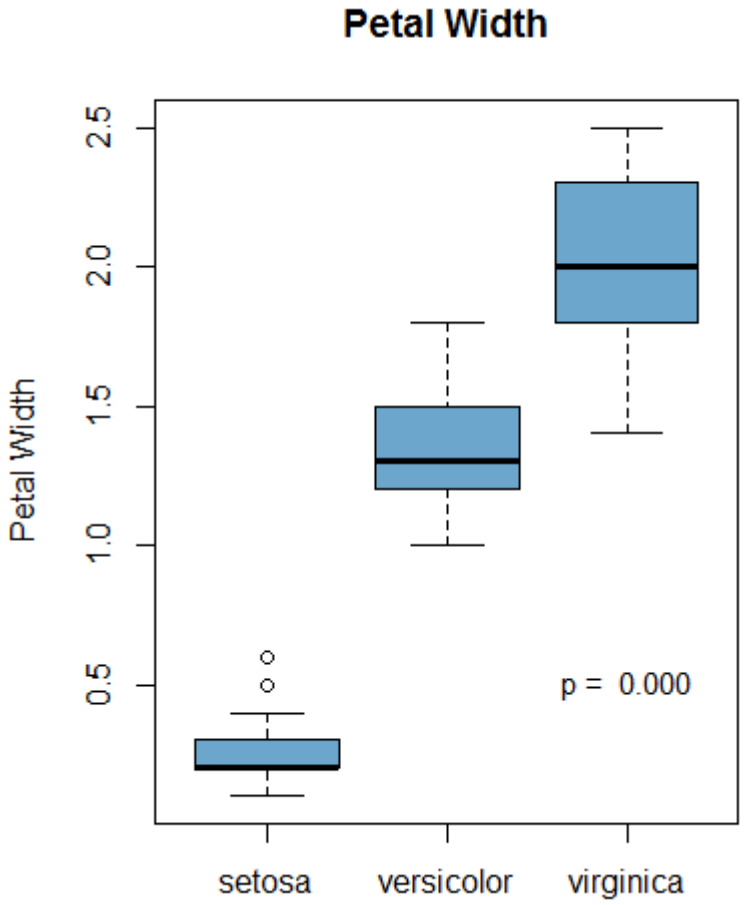
```
> # Selecionando todas as colunas, menos as que começam com "Sepal"
> dt2<- dados2 %>% select(-starts_with("Sepal"))
> head(dt2)
  Petal.Length Petal.Width Species
1          1.4         0.2  setosa
2          1.4         0.2  setosa
3          1.3         0.2  setosa
4          1.5         0.2  setosa
5          1.4         0.2  setosa
6  1.7         0.4  setosa

> boxplot(dt2$Petal.Length ~ dt2$Species, col="skyblue3", main="Petal
   Length", ylab="Petal Length")
p<- kruskal.test(dt2$Petal.Length ~ dt2$Species)$p.value

> boxplot(dt2$Petal.Width ~ dt2$Species, col="skyblue3", main="Petal Width",
+ ylab="Petal Width")
> p<- kruskal.test(dt2$Petal.Width ~ dt2$Species)$p.value
```

Função **select()**





Banco de Dados *starwars*

- Banco de dados do pacote dplyr contendo o nome, altura, peso, cor do cabelo, cor da pele, cor do olho, ano de nascimento (BBY = antes da batalha de Yavin), gênero (masculino, feminino, hermafrodita e nenhum), nome do mundo, espécie, filmes que apareceu, veículos que dirigiu e naves espaciais que dirigiu
- Objetivos: Fazer alguns filtros no banco de dados, calcular a média e o desvio padrão do peso e da altura de forma geral e por espécie

Explorando o banco de dados *starwars*

```
> # Vamos selecionar apenas 5 colunas de interesse
> dados3<- starwars %>% select(name, height, mass, hair_color, species)
> dados3
# A tibble: 87 x 5
   name               height  mass hair_color    species
   <chr>               <int> <dbl> <chr>         <chr>
 1 Luke Skywalker        172    77 blond         Human
 2 C-3PO                 167    75 <NA>          Droid
 3 R2-D2                  96    32 <NA>          Droid
 4 Darth Vader           202   136 none          Human
 5 Leia Organa           150    49 brown         Human
 6 Owen Lars             178   120 brown, grey   Human
 7 Beru Whitesun lars    165    75 brown         Human
 8 R5-D4                  97    32 <NA>          Droid
 9 Biggs Darklighter     183    84 black         Human
10 Obi-Wan Kenobi        182    77 auburn, white Human
# ... with 77 more rows
```

- O banco de dados está em formato tibble
- Uma reinvenção de um data frame
- Útil para bancos de dados grandes
- Muito usado no dplyr

Explorando o banco de dados *starwars*

- Para converter uma tibble em um data frame: data.frame()

```
> # Convertendo uma tibble em um data frame
> dados4<- data.frame(dados3)
> head(dados4)
              name height mass  hair_color species
1 Luke Skywalker    172   77       blond   Human
2          C-3PO    167   75        <NA>   Droid
3          R2-D2     96   32        <NA>   Droid
4    Darth Vader    202  136        none   Human
5    Leia Organa    150   49       brown   Human
6      Owen Lars    178  120 brown, grey   Human
```

Explorando o banco de dados *starwars*

- Para converter um data frame em uma tibble: as_tibble()

```
> # Convertendo um data frame em uma tibble
> dados5<- as_tibble(dados4)
> dados5
# A tibble: 87 x 5
   name               height  mass hair_color    species
   <chr>               <int> <dbl> <chr>         <chr>
 1 Luke Skywalker        172    77 blond         Human
 2 C-3PO                 167    75 <NA>          Droid
 3 R2-D2                  96    32 <NA>          Droid
 4 Darth Vader           202   136 none          Human
 5 Leia Organa           150    49 brown         Human
 6 Owen Lars             178   120 brown, grey   Human
 7 Beru Whitesun lars    165    75 brown         Human
 8 R5-D4                  97    32 <NA>          Droid
 9 Biggs Darklighter     183    84 black         Human
10 Obi-Wan Kenobi        182    77 auburn, whi… Human
# ... with 77 more rows
```

Função **filter()**

- Essa função seleciona linhas a partir dos seus valores
- Para selecionar apenas humanos

```
> # Filtrando apenas humanos
> dados5 %>% filter(species == "Human")
# A tibble: 35 x 5
   name               height  mass hair_color    species
   <chr>               <int> <dbl> <chr>         <chr>
 1 Luke Skywalker        172    77 blond         Human
 2 Darth Vader           202   136 none          Human
 3 Leia Organa           150    49 brown         Human
 4 Owen Lars             178   120 brown, grey   Human
 5 Beru Whitesun lars    165    75 brown         Human
 6 Biggs Darklighter     183    84 black         Human
 7 Obi-Wan Kenobi        182    77 auburn, white Human
 8 Anakin Skywalker      188    84 blond         Human
 9 Wilhuff Tarkin        180    NA auburn, grey  Human
10 Han Solo              180    80 brown         Human
# ... with 25 more rows
```

Função **filter()**

- Para selecionar observações sem cor do cabelo **e** peso > 100

```
> # Filtrando sem cor do cabelo e peso > 100 kg
> dados5 %>% filter(hair_color == "none" & mass >100)
# A tibble: 5 x 5
  name            height  mass hair_color species
  <chr>            <int> <dbl> <chr>      <chr>
1 Darth Vader        202   136 none       Human
2 IG-88              200   140 none       Droid
3 Bossk              190   113 none       Trandoshan
4 Dexter Jettster    198   102 none       Besalisk
5 Grievous           216   159 none       Kaleesh
```

Função **filter()**

- Para selecionar observações sem cor do cabelo **ou** peso > 100

```
> # Filtrando sem cor do cabelo ou peso > 100 kg
> dados5 %>% filter(hair_color == "none" |  mass >100)
# A tibble: 42 x 5
   name                  height  mass hair_color  species
   <chr>                  <int> <dbl> <chr>       <chr>
 1 Darth Vader              202   136 none        Human
 2 Owen Lars                178   120 brown, grey Human
 3 Chewbacca                228   112 brown       Wookiee
 4 Jabba Desilijic Tiure    175  1358 <NA>        Hutt
 5 Jek Tono Porkins         180   110 brown       Human
 6 IG-88                    200   140 none        Droid
 7 Bossk                    190   113 none        Trandoshan
 8 Lobot                    175    79 none        Human
 9 Ackbar                   180    83 none        Mon Calamari
10 Nien Nunb                160    68 none        Sullustan
# ... with 32 more rows
```

Função **summarise()**

- Essa função reduz vários valores em um único sumário
- Para calcular a média e o desvio padrão do peso e da altura:

```
> # Média e devio padrão do peso e da altura
> dados5 %>% summarise(n=n(),
+  mean_height=mean(height, na.rm=T),
+  sd_height=sd(height, na.rm=T),
+  mean_mass=mean(mass, na.rm=T),
+  sd_mass=sd(mass, na.rm=T))
# A tibble: 1 x 5
      n mean_height sd_height mean_mass sd_mass
  <int>       <dbl>     <dbl>     <dbl>   <dbl>
1    87        174.      34.8      97.3    169.
```

Funções **filter()** e **summarise()**

- Para calcular média e desvio padrão do peso e da altura dos humanos:

```
> # Média e devio padrão do peso e da altura dos humanos
> dados5 %>% filter(species == "Human") %>%
+      summarise(n=n(),
+   mean_height=mean(height, na.rm=T),
+   sd_height=sd(height, na.rm=T),
+   mean_mass=mean(mass, na.rm=T),
+   sd_mass=sd(mass, na.rm=T))
# A tibble: 1 x 5
      n mean_height sd_height mean_mass sd_mass
  <int>       <dbl>     <dbl>     <dbl>   <dbl>
1    35        177.      12.5      82.8    19.4
```

Funções **filter()** e **summarise()**

- Para calcular média e desvio padrão do peso e da altura das outras espécies:

```
> # Média e devio padrão do peso e da altura das outras espécies
> dados5 %>% filter(species != "Human") %>%
+      summarise(n=n(),
+    mean_height=mean(height, na.rm=T),
+    sd_height=sd(height, na.rm=T),
+    mean_mass=mean(mass, na.rm=T),
+    sd_mass=sd(mass, na.rm=T))
# A tibble: 1 x 5
      n mean_height sd_height mean_mass sd_mass
  <int>       <dbl>     <dbl>     <dbl>   <dbl>
1    47        174.      43.6      108.    217.
```

Funções **group_by()** e **summarise()**

- Essa função faz as operações por grupo
- Para calcular média e desvio padrão do peso e da altura por espécie:

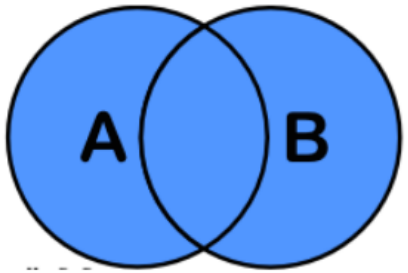
```
> # Média e devio padrão do peso e da altura por espécie
> dados5$species2<- ifelse(dados5$species=="Human", "Human", "Other")
> dados5 %>% group_by(species2) %>%
+      summarise(n=n(),
+   mean_height=mean(height, na.rm=T),
+   sd_height=sd(height, na.rm=T),
+   mean_mass=mean(mass, na.rm=T),
+   sd_mass=sd(mass, na.rm=T))
# A tibble: 3 x 6
  species2     n mean_height sd_height mean_mass sd_mass
  <chr>    <int>       <dbl>     <dbl>     <dbl>   <dbl>
1 Human       35        177.      12.5      82.8    19.4
2 Other       47        174.      43.6     108.    217.
3 <NA>         5        160       42.7      48      NA
```

Banco de Dados *Disney*

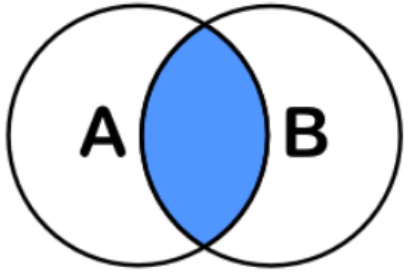
- Ambos os bancos foram retirados do data.world
- O primeiro contém 579 filmes e as variáveis nome do filme, gênero, total ganho e total ganho corrigido pela inflação
- O segundo contém 78 filmes da Disney e as variáveis nome do filme e diretor
- Objetivos: Unir os dois bancos para verificar quais diretores trouxeram mais lucro

Função **join()**

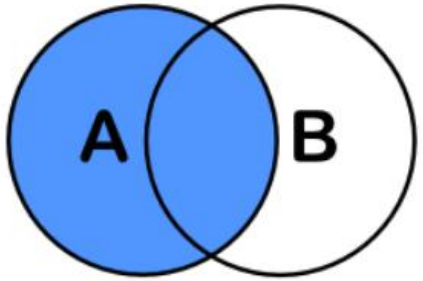
- Essa função junta dois bancos de dados, sendo usada da seguinte forma:
- **full_join():** Junta os dados, retendo todas as linhas.
- **inner_join():** Junta os dados, retendo as linhas que aparecem em ambos
- **left_join():** Junta os dados mantendo as linhas de A



full_join(A,B)



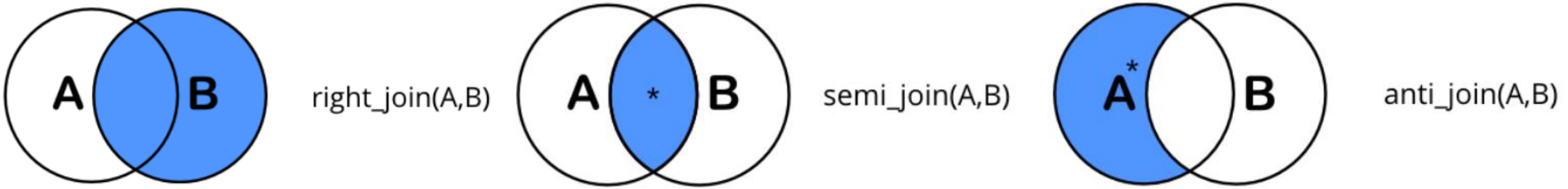
inner_join(A,B)



left_join(A,B)

Função **join()**

- Essa função junta dois bancos de dados, sendo usada da seguinte forma:
- **right_join():** Junta os dados mantendo as linhas de B
- **semi_join():** Todas as linhas de A que apareçam em B
- **anti_join():** Todas as linhas de A que não apareçam em B



Função **join()**

```
> dados6<- read.csv2("C:/Users/Luana/Desktop/Palestra R ladies/dados6.csv")
> head(dados6)
                              name     genre total_gross adjusted_gross
1 Snow White and the Seven Dwarfs   Musical   184925485     5228953251
2                      Pinocchio Adventure    84300000     2188229052
3                       Fantasia   Musical    83320000     2187090808
> dim(dados6)
[1] 579   4
>
> dados7<- read.csv2("C:/Users/Luana/Desktop/Palestra R ladies/dados7.csv")
> head(dados7)
                              name       director
1 Snow White and the Seven Dwarfs     David Hand
2                      Pinocchio Ben Sharpsteen
3                       Fantasia   full credits
> dim(dados7)
[1] 78  2
```

ABG CONSULTORIA ESTATÍSTICA

Função **join()**

```
> #Junta os dados retendo todas as linhas
> head(dados6 %>% full_join(dados7, by="name"))
                             name     genre total_gross adjusted_gross        director
1 Snow White and the Seven Dwarfs   Musical   184925485     5228953251      David Hand
2                      Pinocchio Adventure    84300000     2188229052  Ben Sharpsteen
3                       Fantasia   Musical    83320000     2187090808    full credits
> dim(dados6 %>% full_join(dados7, by="name"))
[1] 590   5
> #Junta os dados, retendo as linhas que aparecem em ambos
> head(dados6 %>% inner_join(dados7, by="name"))
                             name     genre total_gross adjusted_gross        director
1 Snow White and the Seven Dwarfs   Musical   184925485     5228953251      David Hand
2                      Pinocchio Adventure    84300000     2188229052  Ben Sharpsteen
3                       Fantasia   Musical    83320000     2187090808    full credits
> dim(dados6 %>% inner_join(dados7, by="name"))
[1] 67  5
```

Função **join()**

```
> #Junta os dados mantendo as linhas de A
> head(dados6 %>% left_join(dados7, by="name"))
                          name     genre total_gross adjusted_gross        director
1 Snow White and the Seven Dwarfs   Musical   184925485     5228953251      David Hand
2                  Pinocchio Adventure    84300000     2188229052  Ben Sharpsteen
3                   Fantasia   Musical    83320000     2187090808    full credits
> dim(dados6 %>% left_join(dados7, by="name"))
[1] 579   5
>
> #Junta os dados mantendo as linhas de B
> head(dados6 %>% right_join(dados7, by="name"))
                          name     genre total_gross adjusted_gross        director
1 Snow White and the Seven Dwarfs   Musical   184925485     5228953251      David Hand
2                  Pinocchio Adventure    84300000     2188229052  Ben Sharpsteen
3                   Fantasia   Musical    83320000     2187090808    full credits
> dim(dados6 %>% right_join(dados7, by="name"))
[1] 78  5
```

Função **join()**

```
> #Todas as linhas de A que apareçam em B
> head(dados6 %>% semi_join(dados7, by="name"))
                            name     genre total_gross adjusted_gross
1 Snow White and the Seven Dwarfs   Musical   184925485     5228953251
2                  Pinocchio Adventure    84300000     2188229052
3                   Fantasia   Musical    83320000     2187090808
> dim(dados6 %>% semi_join(dados7, by="name"))
[1] 67  4
>
> #Todas as linhas de A que não apareçam em B
> head(dados6 %>% anti_join(dados7, by="name"))
                        name     genre total_gross adjusted_gross
1          Song of the South Adventure    65000000     1078510579
2 20,000 Leagues Under the Sea Adventure    28200000      528279994
3  The Absent Minded Professor    Comedy    25381407      310094574
> dim(dados6 %>% anti_join(dados7, by="name"))
[1] 512   4
```

ABG CONSULTORIA ESTATÍSTICA

Função **join()**

```
> #Todas as linhas de B que não apareçam em A
> head(dados7 %>% anti_join(dados6, by="name"))
                  name         director
1                Dumbo  Ben Sharpsteen
2                Bambi       David Hand
3      Saludos Amigos      Jack Kinney
> dim(dados7 %>% anti_join(dados6, by="name"))
[1] 11  2
> #Junta os dados, retendo as linhas que aparecem em ambos
> dt8<- dados6 %>% inner_join(dados7, by="name")
> head(dt8)
                           name     genre total_gross adjusted_gross         director
1 Snow White and the Seven Dwarfs   Musical   184925485     5228953251       David Hand
2                   Pinocchio Adventure    84300000     2188229052  Ben Sharpsteen
3                     Fantasia   Musical    83320000     2187090808    full credits
```

Função **arrange()**

- Essa função muda a ordem das linhas

```
> #Soma agrupa por diretor, soma os valores e ordena do maior para o menor
> dt8 %>% group_by(director) %>%
+ summarise(total=sum(total_gross), ajust=sum(adjusted_gross)) %>%arrange(desc(ajust))
# A tibble: 42 x 3
   director                total      ajust
   <fct>                   <int>      <dbl>
 1 David Hand            184925485 5228953251
 2 Wolfgang Reitherman   421476209 2721259578
 3 Ben Sharpsteen         84300000 2188229052
 4 full credits           83320000 2187090808
 5 Ron Clements          840214815 1318949600
 6 John Lasseter         873182269 1292522618
 7 Andrew Stanton       1090633095 1267216038
 8 Hamilton Luske         93600000 1236035515
 9 Wilfred Jackson        85000000  920608730
10 Roger Allers          422780140  761640898
# ... with 32 more rows
```

Referências

- Hadley Wickham, Romain François, Lionel Henry and Kirill Müller (2018). dplyr: A Grammar of Data Manipulation. R package version 0.7.7.
- https://CRAN.R-project.org/package=dplyr
- https://analisereal.com/2015/09/07/introducao-ao-dplyr/
- https://www.kaggle.com/yersever/500-person-gender-height-weight-bodymassindex
- https://data.world/kgarrett/disney-character-success-00-16

OBRIGADA!

luana@abgconsultoria.com.br
https://www.linkedin.com/in/luana-sílvia-dos-santos

ABG CONSULTORIA ESTATÍSTICA

ABG CONSULTORIA ESTATÍSTICA