Insper EDUCAÇÃO EXECUTIVA ONLINE

Live Learning AO VIVO

Programação para Não Programadores

Aula 5

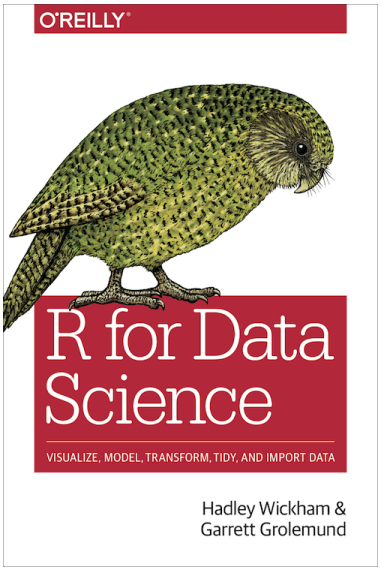
Prof. Magno Severino e Prof. Marina Muradian

27/04/2021

Objetivos de aprendizagem

- Conhecer as funções da biblioteca `tidyverse`.
- Aplicar operador pipe e os verbos (mutate, select, filter, summarise, arrange) para análise de dados.
- Obter estatísticas descritivas.

Referência

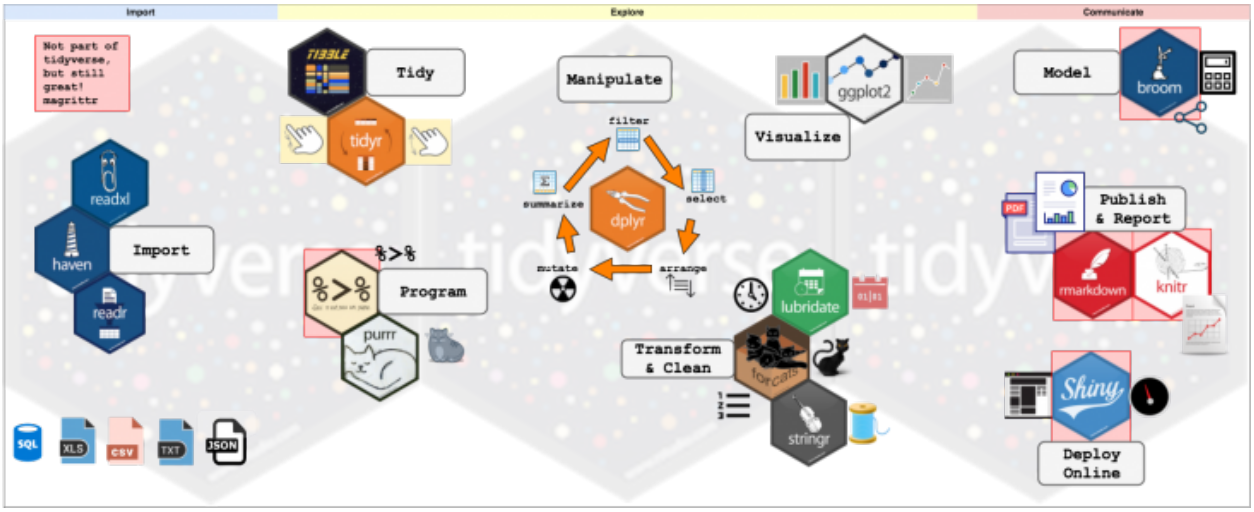


R For Data Science

Hadley Wickham e Garrett Grolemund

disponível em https://r4ds.had.co.nz/

`tidyverse`



fonte: https://www.storybench.org/getting-started-with-tidyverse-in-r/

`tidyverse` é uma coleção de bibliotecas criadas para o universo de data science. Todos os pacotes 'tidyverse' possuem a mesma gramática, estrutura de dados e filosofia:

Veja todos os pacotes disponíveis em:

https://www.tidyverse.org/

Vamos instalar o pacote `tidyverse`:

```
install.packages("tidyverse")
```

E ativar a biblioteca:

```
library(tidyverse)
```

O operador pipe (%>%) da library `magrittr`

Atalho no teclado: `Ctrl + Shift + M`

Passa o objeto do lado esquerdo como primeiro argumento (ou .argumento) da função do lado direito:

- **x** %>% f(y) é equivalente a f(**x**,y)
- **y** %>% f(x,.,z) é equivalente a f(x,**y**,z)

Na prática, vamos supor que queremos somar todos os elementos do `vetor` e em seguida tirar a raiz quadrada desta soma:

```
vetor <- c(20,40,60,80,200)

#raiz da soma
sqrt(sum(vetor))
```

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

Usando o pipe:

```
vetor %>% sum() %>% sqrt()
```

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

Transformação de Dados com `dplyr`

As cinco principais funções do `dplyr` são:

- `filter()`
- `arrange()`
- `select()`
- `mutate()`
- `summarize()`

Todos os verbos funcionam de maneira similar:

1. O primeiro argumento é um data frame
2. Os próximos argumentos descrevem o que fazer com o data frame
3. O resultado é um novo data frame

Filtrando linhas com `filter()`

Vamos voltar a utilizar a base de dados flights:

```
flights <- read.csv("flights.csv")
```

Sem pipe:

```
filter(flights, month == 1, day == 1)
```

```
## # A tibble: 842 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # ... with 832 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Com pipe:

```
flights %>% filter(month == 1, day == 1)
```

```
## # A tibble: 842 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # ... with 832 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Atrubuindo e imprimindo:

```
# Para atribuir e imprimir de uma só vez, coloque
# parênteses em volta da atribuição (os dois espaços
# depois dos parênteses abaixo não são necessários).

(  jan1 <- flights %>% filter(month == 1, day == 1)  )
```

```
## # A tibble: 842 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # ... with 832 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Podemos utilizar os operadores lógicos aprendidos nas primeiras aulas para filtrar aqui também.

Por exemplo, vamos filtar somente as observações que **não** são `NA` na coluna `arr_delay`:

```
flights %>% filter(!is.na(arr_delay))
```

```
## # A tibble: 327,346 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # ... with 327,336 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Origem = Kennedy ou Newark, Destino = Los Angeles:

```
flights %>% filter(origin %in% c("JFK", "EWR"), dest == "LAX")
```

```
## # A tibble: 16,174 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      558            600        -2      924
##  2  2013     1     1      628            630        -2     1016
##  3  2013     1     1      658            700        -2     1027
##  4  2013     1     1      702            700         2     1058
##  5  2013     1     1      743            730        13     1107
##  6  2013     1     1      828            823         5     1150
##  7  2013     1     1      829            830        -1     1152
##  8  2013     1     1      856            900        -4     1226
##  9  2013     1     1      859            900        -1     1223
## 10  2013     1     1      921            900        21     1237
## # ... with 16,164 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Arranjando linhas com `arrange()`

Organizando os dados segundo a ordem crescente da coluna `dep_delay`:

```
flights %>% arrange(dep_delay)
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013    12     7     2040           2123       -43       40
##  2  2013     2     3     2022           2055       -33     2240
##  3  2013    11    10     1408           1440       -32     1549
##  4  2013     1    11     1900           1930       -30     2233
##  5  2013     1    29     1703           1730       -27     1947
##  6  2013     8     9      729            755       -26     1002
##  7  2013    10    23     1907           1932       -25     2143
##  8  2013     3    30     2030           2055       -25     2213
##  9  2013     3     2     1431           1455       -24     1601
## 10  2013     5     5      934            958       -24     1225
## # ... with 336,766 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Agora usando a ordem decrescente:

```
flights %>% arrange(desc(dep_delay))
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     9      641            900      1301     1242
##  2  2013     6    15     1432           1935      1137     1607
##  3  2013     1    10     1121           1635      1126     1239
##  4  2013     9    20     1139           1845      1014     1457
##  5  2013     7    22      845           1600      1005     1044
##  6  2013     4    10     1100           1900       960     1342
##  7  2013     3    17     2321            810       911      135
##  8  2013     6    27      959           1900       899     1236
##  9  2013     7    22     2257            759       898      121
## 10  2013    12     5      756           1700       896     1058
## # ... with 336,766 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Se você fornecer mais de uma coluna, as demais colunas serão usadas sucessivamente para decidir os empates:

```
flights %>% arrange(desc(month), day)
```

```
## # A tibble: 336,776 x 19
```

```
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013    12     1       13           2359        14      446
##  2  2013    12     1       17           2359        18      443
##  3  2013    12     1      453            500        -7      636
##  4  2013    12     1      520            515         5      749
##  5  2013    12     1      536            540        -4      845
##  6  2013    12     1      540            550       -10     1005
##  7  2013    12     1      541            545        -4      734
##  8  2013    12     1      546            545         1      826
##  9  2013    12     1      549            600       -11      648
## 10  2013    12     1      550            600       -10      825
## # ... with 336,766 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Selecionando colunas com select()

Vamos supor que eu só queira utilizar colunas específicas da minha base: `carrier`, `year`, `month` e `day`

```
flights %>% select(carrier, year, month, day)
```

```
## # A tibble: 336,776 x 4
##    carrier  year month   day
##    <chr>   <int> <int> <int>
##  1 UA       2013     1     1
##  2 UA       2013     1     1
##  3 AA       2013     1     1
##  4 B6       2013     1     1
##  5 DL       2013     1     1
##  6 UA       2013     1     1
##  7 B6       2013     1     1
##  8 EV       2013     1     1
##  9 B6       2013     1     1
## 10 AA       2013     1     1
## # ... with 336,766 more rows
```

Selecionando todas as colunas, MENOS a coluna `year`:

```
flights %>% select(-year)
```

```
## # A tibble: 336,776 x 18
##    month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1     1     1      517            515         2      830            819
##  2     1     1      533            529         4      850            830
##  3     1     1      542            540         2      923            850
##  4     1     1      544            545        -1     1004           1022
##  5     1     1      554            600        -6      812            837
##  6     1     1      554            558        -4      740            728
##  7     1     1      555            600        -5      913            854
```

```
##  8     1     1      557            600        -3      709            723
##  9     1     1      557            600        -3      838            846
## 10     1     1      558            600        -2      753            745
## # ... with 336,766 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>
```

`everything()` é útil para mover as colunas de lugar:

```
flights %>% select(carrier, origin, everything())
```

```
## # A tibble: 336,776 x 19
##    carrier origin  year month   day dep_time sched_dep_time dep_delay
##    <chr>   <chr>  <int> <int> <int>    <int>          <int>     <dbl>
##  1 UA      EWR     2013     1     1      517            515         2
##  2 UA      LGA     2013     1     1      533            529         4
##  3 AA      JFK     2013     1     1      542            540         2
##  4 B6      JFK     2013     1     1      544            545        -1
##  5 DL      LGA     2013     1     1      554            600        -6
##  6 UA      EWR     2013     1     1      554            558        -4
##  7 B6      EWR     2013     1     1      555            600        -5
##  8 EV      LGA     2013     1     1      557            600        -3
##  9 B6      JFK     2013     1     1      557            600        -3
## 10 AA      LGA     2013     1     1      558            600        -2
## # ... with 336,766 more rows, and 11 more variables: arr_time <int>,
## #   sched_arr_time <int>, arr_delay <dbl>, flight <int>, tailnum <chr>,
## #   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>
```

Também podemos renomear uma coluna dentro do select:

```
#renomeando a coluna year para "ano":

flights %>% select("ano" = year, month, day)
```

```
## # A tibble: 336,776 x 3
##      ano month   day
##    <int> <int> <int>
##  1  2013     1     1
##  2  2013     1     1
##  3  2013     1     1
##  4  2013     1     1
##  5  2013     1     1
##  6  2013     1     1
##  7  2013     1     1
##  8  2013     1     1
##  9  2013     1     1
## 10  2013     1     1
## # ... with 336,766 more rows
```

Combinando `filter()`, `select()` e `arrange()`:

Suponha que nosso objetivo seja verificar qual a companhia aérea que mais atrasa nos vôos entre JFK e LAX:

```
flights %>%
    filter(origin == "JFK", dest == "LAX") %>%
    select(carrier, dep_delay) %>%
    arrange(desc(dep_delay))
```

```
## # A tibble: 11,262 x 2
##    carrier dep_delay
##    <chr>       <dbl>
##  1 DL            800
##  2 VX            634
##  3 VX            434
##  4 VX            413
##  5 VX            392
##  6 UA            364
##  7 AA            345
##  8 AA            334
##  9 VX            322
## 10 AA            321
## # ... with 11,252 more rows
```

Criando variáveis (colunas) com `mutate()`

Queremos criar uma variável que mostre a velocidade de cada voo:

```
flights %>%
    select(year:day, flight,  distance, air_time) %>%
    mutate(speed = distance / (air_time / 60))
```

```
## # A tibble: 336,776 x 7
##     year month   day flight distance air_time speed
##    <int> <int> <int>  <int>    <dbl>    <dbl> <dbl>
##  1  2013     1     1   1545     1400      227  370.
##  2  2013     1     1   1714     1416      227  374.
##  3  2013     1     1   1141     1089      160  408.
##  4  2013     1     1    725     1576      183  517.
##  5  2013     1     1    461      762      116  394.
##  6  2013     1     1   1696      719      150  288.
##  7  2013     1     1    507     1065      158  404.
##  8  2013     1     1   5708      229       53  259.
##  9  2013     1     1     79      944      140  405.
## 10  2013     1     1    301      733      138  319.
## # ... with 336,766 more rows
```

Note que você também pode se referir às variáveis que criou:

```
flights %>%
    select(year:day, flight,  distance, air_time) %>%
    mutate(hours = air_time / 60,
           speed = distance / hours)
```

```
## # A tibble: 336,776 x 8
##     year month   day flight distance air_time hours speed
##    <int> <int> <int>  <int>    <dbl>    <dbl> <dbl> <dbl>
##  1  2013     1     1   1545     1400      227 3.78   370.
##  2  2013     1     1   1714     1416      227 3.78   374.
##  3  2013     1     1   1141     1089      160 2.67   408.
##  4  2013     1     1    725     1576      183 3.05   517.
##  5  2013     1     1    461      762      116 1.93   394.
##  6  2013     1     1   1696      719      150 2.5    288.
##  7  2013     1     1    507     1065      158 2.63   404.
##  8  2013     1     1   5708      229       53 0.883  259.
##  9  2013     1     1     79      944      140 2.33   405.
## 10  2013     1     1    301      733      138 2.3    319.
## # ... with 336,766 more rows
```

`summarise()` colapsa a tabela toda em apenas uma linha

Vamos calcular o atraso médio de decolagem e a quantidade total de voos:

```
flights %>%
    filter(!is.na(dep_delay)) %>%
    summarise(mean_delay = mean(dep_delay), number_of_flights = n())
```

```
## # A tibble: 1 x 2
##   mean_delay number_of_flights
##        <dbl>             <int>
## 1       12.6            328521
```

`group_by()` muda a "unidade de análise" de toda a tabela para os grupos definidos

Vamos calcular o atraso médio de decolagem e a quantidade total de voos, mas agora analisando por companhia aérea:

```
flights %>%
    filter(!is.na(dep_delay)) %>%
    group_by(carrier) %>%
    summarise(mean_delay = mean(dep_delay), number_of_flights = n())
```

```
## # A tibble: 16 x 3
##    carrier mean_delay number_of_flights
##    <chr>        <dbl>             <int>
##  1 9E           16.7              17416
##  2 AA            8.59             32093
##  3 AS            5.80               712
##  4 B6           13.0              54169
##  5 DL            9.26             47761
##  6 EV           20.0              51356
##  7 F9           20.2                682
##  8 FL           18.7               3187
##  9 HA            4.90               342
## 10 MQ           10.6              25163
```

```
## 11 OO          12.6              29
## 12 UA          12.1           57979
## 13 US           3.78          19873
## 14 VX          12.9            5131
## 15 WN          17.7           12083
## 16 YV          19.0             545
```

DESAFIO:

Para cada destino, calcule: a (1) distância média dos voos e (2) o tempo de atraso médio na decolagem e (3) o número de voos na base e mostre tudo numa planilha só.