

R Básico

Aula 3 - Parte 1

João Matheus
Lineu Alberto

PET - Estatística
Universidade Federal do Paraná

PET-Estatística
2019
UFPR



PET-Estatística
UFPR

1 tidy

2 dplyr

3 Links úteis



PET-Estatística UFPR



PET-Estatística
UFPR

tidyr, dplyr e ggplot2

- São os pacotes de maior representatividade do conjunto pertencente ao tidyverse.
- Com esta trinca é possível:
 - Ajustar o conjunto de dados para o formato de interesse.
 - Manipular os dados (filtrar linhas, selecionar colunas, sumarizar informações, etc).
 - Produzir visualizações.



1 tidy

2 dplyr

3 Links úteis



PET-Estatística UFPR



PET-Estatística
UFPR

Dispõe de funções para formatação adequada dos dados para análise.
Principais funções:

- `gather()` - empilha.
- `spread()` - desempilha.
- `separate()` - separar caracteres.
- `unite()` - unir caracteres.
- `drop_na()` - retirar dados ausentes.
- `replace_na()` - substituir dados ausentes.



gather() e spread()

- São as funções do tidyr para empilhar e desempilhar os dados.
- É comum, em situações em que os dados são coletados ao longo do tempo, que cada coluna represente uma coleta.
- No formato ideal, espera-se que haja uma variável categórica que marque a ordem de coleta e uma variável com os valores observados na coleta.

```
td <- tibble(  
  id = rep(1:50),  
  jan = rnorm(50, rpois(1, (runif(1)*10))),  
  fev = rnorm(50, rpois(1, (runif(1)*10))),  
  mar = rnorm(50, rpois(1, (runif(1)*10))),  
  abr = rnorm(50, rpois(1, (runif(1)*10))),  
  mai = rnorm(50, rpois(1, (runif(1)*10))),  
  jun = rnorm(50, rpois(1, (runif(1)*10))),  
  jul = rnorm(50, rpois(1, (runif(1)*10))),  
  ago = rnorm(50, rpois(1, (runif(1)*10))),  
  set = rnorm(50, rpois(1, (runif(1)*10))),  
  out = rnorm(50, rpois(1, (runif(1)*10))),  
  nov = rnorm(50, rpois(1, (runif(1)*10))),  
  dez = rnorm(50, rpois(1, (runif(1)*10)))
```

gather() e spread()

Conjunto de dados no formato largo:

```
td

## # A tibble: 50 x 13
##       id   jan   fev   mar   abr   mai   jun   jul   ago   set   out
##   <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     1  3.19  5.69  8.71  12.7  3.42  5.81 -0.538 -0.220  4.39  4.87
## 2     2  1.23  7.34  9.53  12.6  3.73  4.78 -0.00635 -0.189  4.60  4.93
## 3     3  0.722  6.64  11.4  11.4  2.73  3.57 -0.963  1.45  2.03  5.85
## 4     4  3.01  5.92  9.11  12.9  3.41  5.27 -0.527  0.586  4.60  4.90
## 5     5  1.08  6.28  9.66  12.4  4.00  6.16 -1.79  1.22  3.04  4.47
## 6     6  2.48  6.22  7.64  11.4  5.00  4.52 -0.507 -1.45  3.93  5.07
## 7     7  0.790  7.93  7.31  12.7  3.31  4.62  1.44  1.66  5.25  3.90
## 8     8  1.92  6.02  8.87  13.0  2.91  5.84 -0.480 -1.11  3.18  6.17
## 9     9  2.84  7.24  8.46  13.5  2.88  4.32 -1.89  1.40  4.53  5.02
## 10    10 -0.369  6.46  10.2  12.0  4.48  3.02  1.61  0.511  3.45  5.74
## # ... with 40 more rows, and 2 more variables: nov <dbl>, dez <dbl>
```



gather() e spread()

Utilizando a função `gather()` para "empilhar" ou passar os dados para o formato longo:

```
td_gat <- td %>% gather(key = 'mes', value = 'valor', jan:dez)
td_gat
```

```
## # A tibble: 600 x 3
##       id mes      valor
##   <int> <chr> <dbl>
## 1     1  1 jan     3.19
## 2     2  2 jan     1.23
## 3     3  3 jan     0.722
## 4     4  4 jan     3.01
## 5     5  5 jan     1.08
## 6     6  6 jan     2.48
## 7     7  7 jan     0.790
## 8     8  8 jan     1.92
## 9     9  9 jan     2.84
## 10    10 10 jan    -0.369
## # ... with 590 more rows
```


gather() e spread()

Fazendo com que os dados retornem para o formato largo:

```
td_gat %>% spread(key=mes, value = valor)
```

```
## # A tibble: 50 x 13
##       id  abr   ago   dez  fev   jan   jul   jun  mai  mar  nov
##   <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     1     1  12.7 -0.220  2.50  5.69  3.19 -0.538  5.81  3.42  8.71  6.38
## 2     2     2  12.6 -0.189  2.08  7.34  1.23 -0.00635  4.78  3.73  9.53  6.22
## 3     3     3  11.4  1.45 -0.0845  6.64  0.722 -0.963  3.57  2.73  11.4  6.08
## 4     4     4  12.9  0.586  1.50  5.92  3.01 -0.527  5.27  3.41  9.11  5.98
## 5     5     5  12.4  1.22  2.91  6.28  1.08 -1.79  6.16  4.00  9.66  4.57
## 6     6     6  11.4 -1.45  3.69  6.22  2.48 -0.507  4.52  5.00  7.64  4.85
## 7     7     7  12.7  1.66  1.58  7.93  0.790  1.44  4.62  3.31  7.31  6.96
## 8     8     8  13.0 -1.11  2.33  6.02  1.92 -0.480  5.84  2.91  8.87  5.17
## 9     9     9  13.5  1.40  2.48  7.24  2.84 -1.89  4.32  2.88  8.46  7.74
## 10    10    10  12.0  0.511  3.41  6.46 -0.369  1.61  3.02  4.48  10.2  6.40
## # ... with 40 more rows, and 2 more variables: out <dbl>, set <dbl>
```



separate() e unite()

As funções `separate()` e `unite()` servem para manipulações simples de strings em um tibble. Servem para unir ou separar strings utilizando um delimitador.

```
td2 <- tibble(nome = c("Lineu Alberto", "João Matheus", "PET Estatística"))
td2

## # A tibble: 3 x 1
##   nome
##   <chr>
## 1 Lineu Alberto
## 2 João Matheus
## 3 PET Estatística
```

PET-Estatística
UFPR



separate() e unite()

Utilizando a função `separate()` para deixar um nome em cada coluna:

```
td3 <- td2 %>% separate(col = 'nome',  
                        into = c('primeiro_nome', 'segundo_nome'),  
                        sep = ' ')
```

```
td3
```

```
## # A tibble: 3 x 2  
##   primeiro_nome segundo_nome  
##   <chr>         <chr>  
## 1 Lineu         Alberto  
## 2 João          Matheus  
## 3 PET           Estatística
```

UFPR



PET-Estatística
UFPR

separate() e unite()

Utilizando a função `unite()` para retornar ao formato com o primeiro e segundo nome em uma mesma coluna:

```
td3 %>% unite(col = 'nome',  
             sep = ' ')
```

```
## # A tibble: 3 x 1  
##   nome  
##   <chr>  
## 1 Lineu Alberto  
## 2 João Matheus  
## 3 PET Estatística
```

PET-Estatística
UFPR



drop_na() e replace_na()

As funções `drop_na()` e `replace_na()` servem para tratamento de dados ausentes (missing data).

```
td4 <- tibble(col1 = 1:10, col2 = c(1,2,NA,4,NA,6,7,NA,9,10))
```

```
td4
```

```
## # A tibble: 10 x 2
```

```
##   col1 col2
```

```
##   <int> <dbl>
```

```
## 1     1     1
```

```
## 2     2     2
```

```
## 3     3    NA
```

```
## 4     4     4
```

```
## 5     5    NA
```

```
## 6     6     6
```

```
## 7     7     7
```

```
## 8     8    NA
```

```
## 9     9     9
```

```
## 10    10    10
```

drop_na() e replace_na()

Utilizando a `drop_na()` para retirar todas as linhas que contém NA do conjunto de dados:

```
td4 %>% drop_na()

## # A tibble: 7 x 2
##   col1  col2
##   <int> <dbl>
## 1     1     1
## 2     2     2
## 3     4     4
## 4     6     6
## 5     7     7
## 6     9     9
## 7    10    10
```

UFPR



PET-Estatística
UFPR

drop_na() e replace_na()

Utilizando a `replace_na()` para substituir valores ausentes por outro valor:

```
td4 %>% replace_na(list(col2 = 0))
```

```
## # A tibble: 10 x 2
```

```
##   col1  col2
```

```
##   <int> <dbl>
```

```
## 1     1     1
```

```
## 2     2     2
```

```
## 3     3     0
```

```
## 4     4     4
```

```
## 5     5     0
```

```
## 6     6     6
```

```
## 7     7     7
```

```
## 8     8     0
```

```
## 9     9     9
```

```
## 10    10    10
```

UFPR



PET-Estatística
UFPR

drop_na() e replace_na()

Utilizando a `replace_na()` para substituir valores ausentes por outro valor:

```
td4 %>% replace_na(list(col2 = 100000))
```

```
## # A tibble: 10 x 2
```

```
##   col1  col2
```

```
##   <int> <dbl>
```

```
## 1     1     1
```

```
## 2     2     2
```

```
## 3     3 100000
```

```
## 4     4     4
```

```
## 5     5 100000
```

```
## 6     6     6
```

```
## 7     7     7
```

```
## 8     8 100000
```

```
## 9     9     9
```

```
## 10    10    10
```

UFPR



PET-Estatística
UFPR

1 tidy

2 dplyr

3 Links úteis



PET-Estatística UFPR



PET-Estatística
UFPR

O dplyr é o pacote tidyverse destinado à manipulação. Ele torna as atividades de manipulação menos custosas devido à sintaxe centralizada das funções.

Principais funções:

- `filter()` - filtra linhas.
- `select()` - seleciona colunas.
- `arrange()` - ordena a base.
- `mutate()` - cria/modifica colunas.
- `group_by()` - agrupa a base.
- `summarise()` - sumariza a base.



Considere o conjunto de dados swiss, nativo do R:

```
swiss$names <- rownames(swiss)
dados <- swiss %>% as_tibble()
names(dados)
```

```
## [1] "Fertility"          "Agriculture"       "Examination"
## [4] "Education"         "Catholic"          "Infant.Mortality"
## [7] "names"
```

PET-Estatística UFPR



PET-Estatística
UFPR

```

dados

## # A tibble: 47 x 7
##   Fertility Agriculture Examination Education Catholic Infant.Mortality
## *   <dbl>         <dbl>         <int>     <int>     <dbl>         <dbl>
## 1    80.2           17             15        12        9.96          22.2
## 2    83.1           45.1            6         9        84.8          22.2
## 3    92.5           39.7            5         5        93.4          20.2
## 4    85.8           36.5            12         7        33.8          20.3
## 5    76.9           43.5            17        15         5.16          20.6
## 6    76.1           35.3             9         7        90.6          26.6
## 7    83.8           70.2            16         7        92.8          23.6
## 8    92.4           67.8            14         8        97.2          24.9
## 9    82.4           53.3            12         7        97.7           21
## 10   82.9           45.2            16        13        91.4          24.4
## # ... with 37 more rows, and 1 more variable: names <chr>

```

UFPR



PET-Estatística
UFPR

filter()

Utilizando a função `filter()` para selecionar linhas que atendem a uma condição:

```
filtro1 <- dados %>% filter(Fertility > 70)
```

```
filtro2 <- dados %>% filter(Examination > 12 & Agriculture > 30)
```

```
filtro3 <- dados %>% filter(Examination > 12 | Agriculture < 30)
```

```
filtro4 <- dados %>% filter(names %in% c("Courtelary", "Delemont", "Franches-Mnt"))
```



select()

Utilizando a função `select()` para selecionar colunas do conjunto de dados:

```
select1 <- dados %>% select(names, Education)
```

```
select2 <- dados %>% select(starts_with("e"))
```

```
select3 <- dados %>% select(Fertility, Agriculture:Education)
```

```
select4 <- dados %>% select(-Fertility, -names)
```



arrange()

A função `arrange()` serve para ordenar variáveis:

```
df <- letters %>% as_tibble()
```

```
df2 <- df %>% arrange(desc(value))  
df2$value
```

```
## [1] "z" "y" "x" "w" "v" "u" "t" "s" "r" "q" "p" "o" "n" "m" "l" "k" "j"  
## [18] "i" "h" "g" "f" "e" "d" "c" "b" "a"
```

```
df3 <- df2 %>% arrange(value)  
df3$value
```

```
## [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q"  
## [18] "r" "s" "t" "u" "v" "w" "x" "y" "z"
```



mutate()

A função `mutate()` tem como finalidade criar novas colunas a um conjunto de dados existente.

```
df2 <- tibble(col1 = rnorm(50), col2 = rpois(50, 1))
df2
```

```
## # A tibble: 50 x 2
##       col1  col2
##   <dbl> <int>
## 1 -0.397     1
## 2 -0.323     2
## 3  0.824     2
## 4 -0.296     0
## 5  0.431     0
## 6  0.293     0
## 7 -0.257     2
## 8  0.223     1
## 9 -1.77      3
## 10 -0.0454    0
## # ... with 40 more rows
```


mutate()

A utilização de um `mutate()` em uma coluna que já existe nos dados faz com que esta seja sobrescrita:

```
mutate1 <- df2 %>% mutate(col2 = letters[1:10] %>% rep(5))
mutate1
```

```
## # A tibble: 50 x 2
##   col1 col2
##   <dbl> <chr>
## 1 -0.397 a
## 2 -0.323 b
## 3  0.824 c
## 4 -0.296 d
## 5  0.431 e
## 6  0.293 f
## 7 -0.257 g
## 8  0.223 h
## 9 -1.77  i
## 10 -0.0454 j
## # ... with 40 more rows
```

mutate()

Uma nova coluna é acrescentada apenas escolhendo o nome e que valores esta coluna vai assumir:

```
mutate2 <- df2 %>% mutate(col3 = 1:nrow(df2))
mutate2
```

```
## # A tibble: 50 x 3
##       col1  col2  col3
##   <dbl> <int> <int>
## 1 -0.397     1     1
## 2 -0.323     2     2
## 3  0.824     2     3
## 4 -0.296     0     4
## 5  0.431     0     5
## 6  0.293     0     6
## 7 -0.257     2     7
## 8  0.223     1     8
## 9 -1.77      3     9
## 10 -0.0454    0    10
## # ... with 40 more rows
```

mutate()

Múltiplas colunas podem ser criadas e, tal como no tibble, as novas colunas podem ser combinações e operações aplicadas a colunas anteriores:

```
mutate3 <- df2 %>% mutate( col3 = col1 + col2,  
                           col4 = 1:nrow(df2),  
                           col5 = col1 * 10)
```

```
mutate3
```

```
## # A tibble: 50 x 5  
##       col1  col2    col3  col4    col5  
##       <dbl> <int>   <dbl> <int>   <dbl>  
##  1 -0.397     1  0.603     1  -3.97  
##  2 -0.323     2  1.68      2  -3.23  
##  3  0.824     2  2.82      3   8.24  
##  4 -0.296     0 -0.296     4  -2.96  
##  5  0.431     0  0.431     5   4.31  
##  6  0.293     0  0.293     6   2.93  
##  7 -0.257     2  1.74      7  -2.57  
##  8  0.223     1  1.22      8   2.23  
##  9 -1.77      3  1.23      9 -17.7  
## 10 -0.0454    0 -0.0454   10  -0.454  
## # ... with 40 more rows
```

summarise() e group_by()

A função `summarise()` serve para gerar tibbles de tamanho 1 com medidas resumo de determinadas colunas.

```
dados

## # A tibble: 47 x 7
##   Fertility Agriculture Examination Education Catholic Infant.Mortality
## *   <dbl>         <dbl>         <int>         <int>         <dbl>         <dbl>
## 1     80.2           17             15             12           9.96          22.2
## 2     83.1           45.1            6              9           84.8          22.2
## 3     92.5           39.7            5              5           93.4          20.2
## 4     85.8           36.5            12             7           33.8          20.3
## 5     76.9           43.5            17             15           5.16          20.6
## 6     76.1           35.3            9              7           90.6          26.6
## 7     83.8           70.2            16             7           92.8          23.6
## 8     92.4           67.8            14             8           97.2          24.9
## 9     82.4           53.3            12             7           97.7           21
## 10    82.9           45.2            16             13           91.4          24.4
## # ... with 37 more rows, and 1 more variable: names <chr>
```

summarise() e group_by()

Média de uma variável:

```
dados %>%  
  summarise(mean_fert = mean(Fertility))  
  
## # A tibble: 1 x 1  
##   mean_fert  
##   <dbl>  
## 1      70.1
```

PET-Estatística UFPR



PET-Estatística
UFPR

summarise() e group_by()

Média e mediana no mesmo summarise:

```
dados %>%  
  summarise(  
    media_fert = mean(Fertility),  
    mediana_fert = median(Fertility))  
  
## # A tibble: 1 x 2  
##   media_fert mediana_fert  
##   <dbl>         <dbl>  
## 1      70.1           70.4
```

PET-Estatística
UFPR



PET-Estatística
UFPR

summarise() e group_by()

Média, mediana e desvio padrão de 3 variáveis diferentes:

```
dados %>%  
  summarise(  
    media_fert = mean(Fertility),  
    mediana_agr = median(Agriculture),  
    sd_cat = sd(Catholic))  
  
## # A tibble: 1 x 3  
##   media_fert mediana_agr sd_cat  
##   <dbl>         <dbl> <dbl>  
## 1      70.1          54.1  41.7
```

PET-ESTADÍSTICA
UFPR



PET-Estatística
UFPR

summarise() e group_by()

Medida descritiva sumarizada por grupo, utilizando o group_by:

```
dados %>%  
  group_by(names) %>%  
  summarise(mean_fert = mean(Fertility))
```

```
## # A tibble: 47 x 2  
##   names      mean_fert  
##   <chr>      <dbl>  
## 1 Aigle      64.1  
## 2 Aubonne   66.9  
## 3 Avenches  68.9  
## 4 Boudry    70.4  
## 5 Broye     83.8  
## 6 Conthey   75.5  
## 7 Cossonay 61.7  
## 8 Courtelary 80.2  
## 9 Delemont  83.1  
## 10 Echallens 68.3  
## # ... with 37 more rows
```


summarise e group_by

Medida descritiva sumarizada por grupo, utilizando o group_by:

```
df2%>%  
  group_by(col2) %>%  
  summarise(mean_col1 = mean(col1))
```

```
## # A tibble: 5 x 2  
##   col2 mean_col1  
##   <int>   <dbl>  
## 1     0  -0.243  
## 2     1  -0.148  
## 3     2   0.0171  
## 4     3  -0.677  
## 5     4  -2.67
```

UFPR



PET-Estatística
UFPR

As funções `sample_n()` e `sample_frac()` servem para amostrar um percentual ou um número fixo de linhas do conjunto de dados:

```
dados %>% sample_n(10)
```

```
dados %>% sample_frac(0.1)
```

PET-Estatística UFPR



As funções do tipo join servem para cruzar bases por uma coluna chave.

```
tb1 <- tibble(codigo = c("001", "002", "003", "004", "005",  
                        "006", "007", "008", "009", "010"),  
             ocorrencia = rbinom(10,1, prob = 0.5),  
             n = rpois(10,2))
```

tb1

```
## # A tibble: 10 x 3
```

```
##   codigo ocorrencia     n  
##   <chr>      <int> <int>  
## 1 001          0     4  
## 2 002          1     2  
## 3 003          0     3  
## 4 004          1     0  
## 5 005          1     1  
## 6 006          1     4  
## 7 007          0     2  
## 8 008          1     3  
## 9 009          0     3  
## 10 010         1     0
```

join

```
tb2 <- tibble(code = c("001", "022", "003", "004", "005",  
                      "000", "007", "018", "079", "010"),  
             year = c(2010, 2011, 2015, 1999, 2000,  
                    1996, 1998, 1955, 1971, 2001))
```

tb2

```
## # A tibble: 10 x 2  
##   code  year  
##   <chr> <dbl>  
## 1 001    2010  
## 2 022    2011  
## 3 003    2015  
## 4 004    1999  
## 5 005    2000  
## 6 000    1996  
## 7 007    1998  
## 8 018    1955  
## 9 079    1971  
## 10 010    2001
```

inner_join()

A função `inner_join()` gera um novo conjunto de dados apenas para os casos em que a chave é verificada nos dois arquivos:

```
inner_join(tb1, tb2, by = c("codigo"="code"))
```

```
## # A tibble: 6 x 4
##   codigo ocorrencia     n year
##   <chr>      <int> <int> <dbl>
## 1 001          0     4 2010
## 2 003          0     3 2015
## 3 004          1     0 1999
## 4 005          1     1 2000
## 5 007          0     2 1998
## 6 010          1     0 2001
```

UFPR



PET-Estatística
UFPR

full_join()

A função `full_join()` une todas as linhas e todas as colunas, independente da chave ocorrer nos dois arquivos. Nos casos em que não há correspondência é gerado NA.

```
full_join(tb1, tb2, by = c("codigo"="code"))
```

```
## # A tibble: 14 x 4
##   codigo ocorrencia     n year
##   <chr>      <int> <int> <dbl>
## 1 001          0     4 2010
## 2 002          1     2   NA
## 3 003          0     3 2015
## 4 004          1     0 1999
## 5 005          1     1 2000
## 6 006          1     4   NA
## 7 007          0     2 1998
## 8 008          1     3   NA
## 9 009          0     3   NA
## 10 010         1     0 2001
## 11 022         NA    NA 2011
## 12 000         NA    NA 1996
## 13 018         NA    NA 1955
## 14 079         NA    NA 1971
```

left_join()

No `left_join()` todas as chaves do primeiro arquivo são mantidas e acrescenta-se valores nos casos em que há compatibilidade de chave. As chaves que são observadas apenas no segundo conjunto de dados são descartados:

```
left_join(tb1, tb2, by = c("codigo"="code"))
```

```
## # A tibble: 10 x 4
##   codigo ocorrencia     n year
##   <chr>     <int> <int> <dbl>
## 1 001         0     4 2010
## 2 002         1     2  NA
## 3 003         0     3 2015
## 4 004         1     0 1999
## 5 005         1     1 2000
## 6 006         1     4  NA
## 7 007         0     2 1998
## 8 008         1     3  NA
## 9 009         0     3  NA
## 10 010        1     0 2001
```

right_join()

No `right_join()` todas as chaves do segundo arquivo são mantidas e acrescenta-se valores nos casos em que há compatibilidade de chave. As chaves que são observadas apenas no primeiro conjunto de dados são descartados:

```
right_join(tb1, tb2, by = c("codigo"="code"))
```

```
## # A tibble: 10 x 4
##   codigo ocorrencia     n year
##   <chr>      <int> <int> <dbl>
## 1 001          0     4 2010
## 2 022         NA     NA 2011
## 3 003          0     3 2015
## 4 004          1     0 1999
## 5 005          1     1 2000
## 6 000         NA     NA 1996
## 7 007          0     2 1998
## 8 018         NA     NA 1955
## 9 079         NA     NA 1971
## 10 010         1     0 2001
```


1 tidy

2 dplyr

3 Links úteis



PET-Estatística UFPR



PET-Estatística
UFPR

- Página oficial do Tidyverse ([link](#)).
- R4DS ([link](#)).
- Material Curso-R ([link](#)).
- Material do curso sobre manipulação e visualização de dados com Tidyverse do professor Walmes Zeviani ([link](#)).

PET-Estatística
UFPR



PET-Estatística
UFPR