# Programação Estatística

## Introdução ao R - Manipulação de dados

Rachid Muleia, PhD in Statistics

Universidade Eduardo Mondlane
Faculdade de Ciências
Departamento de Matemática e Informática

2023-03-18

# 1 Manipulação de dados

# Manipulação de dados

# Objectivos

No final desta sessão, deverá saber:

- Organizar os dados no formato desejável;
- Criar novas variáveis;
- Filtrar linhas e seleccionar colunas;
- Juntar várias bases de dados numa só base (merge);
- Usar algumas funções da livraria `dplyr` para manipular dados;

# Manipulação de dados - Reformulação de dados

```
fish_data<- read.csv("Fish_survey.csv", header = TRUE,row.names

head(fish_data, n=10)
    X   Site    Month Transect Trout Perch Stickleback
1   1 River1  January        1    10     5          28
2   2 River1  January        2     0    13          42
3   3 River1  January        3     8    19           9
4   4 River2  January        1     3     5          72
5   5 River2  January        2     2     9          33
6   6 River2  January        3    15    24          65
7   7 River1 February        1     2    12          47
8   8 River1 February        2     7     3          69
9   9 River1 February        3     0     0          23
10 10 River2 February        1    11     8          89
fish_data<-fish_data[,-1]
```

# Manipulação de dados - Reformulação de dados

Para reformulação de dados iremos usar a livraria `tidyr` ou `tidyverse`

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

# Manipulação de dados - Reformulação de dados

```
library(tidyr)
fish_survey_long <- gather(fish_data, Species, Abundance, 4:6)
head(fish_survey_long,n=10)
     Site    Month Transect Species Abundance
1  River1  January        1   Trout        10
2  River1  January        2   Trout         0
3  River1  January        3   Trout         8
4  River2  January        1   Trout         3
5  River2  January        2   Trout         2
6  River2  January        3   Trout        15
7  River1 February        1   Trout         2
8  River1 February        2   Trout         7
9  River1 February        3   Trout         0
10 River2 February        1   Trout        11
```

Este representação é designada de long format

# Manipulação de dados - Reformulação de dados

Para converter os dados de volta podemos usar a função `spread` da livraria `tidyr`

```
library(tidyr)
fish_survey_wide<- spread(fish_survey_long, Species, Abundance)
head(fish_survey_wide,n=8)
    Site    Month Transect Perch Stickleback Trout
1 River1    April        1    23           5     1
2 River1    April        2    17          54     2
3 River1    April        3     4          31    15
4 River1 February        1    12          47     2
5 River1 February        2     3          69     7
6 River1 February        3     0          23     0
7 River1  January        1     5          28    10
8 River1  January        2    13          42     0
```

Este representação é designada de wide format

# Junção de base de dados

Para a combinação de dados iremos usar as seguintes funções:

- `rbind()` para juntar linhas;
- `cbind()` para juntar colunas;
- `merge()` para juntar bases de dados com base numa chave de identificação;

# rbind()

A função `rbind()` retorna a junção de linhas de duas bases de dados, que têm mesmas variáveis (colunas).

| Name | City | Country |
|---|---|---|
| Lenna | San Francisco | US |
| Malcom | New York | US |
| Akiko | Tokyo | Japan |

+

| Name | City | Country |
|---|---|---|
| Lenna | San Francisco | US |
| Thomas | London | UK |
| Diane | Chicago | US |

| Name | City | Country |
|---|---|---|
| Lenna | San Francisco | US |
| Malcom | New York | US |
| Akiko | Tokyo | Japan |
| Lenna | San Francisco | US |
| Thomas | London | UK |
| Diane | Chicago | US |

# rbind()

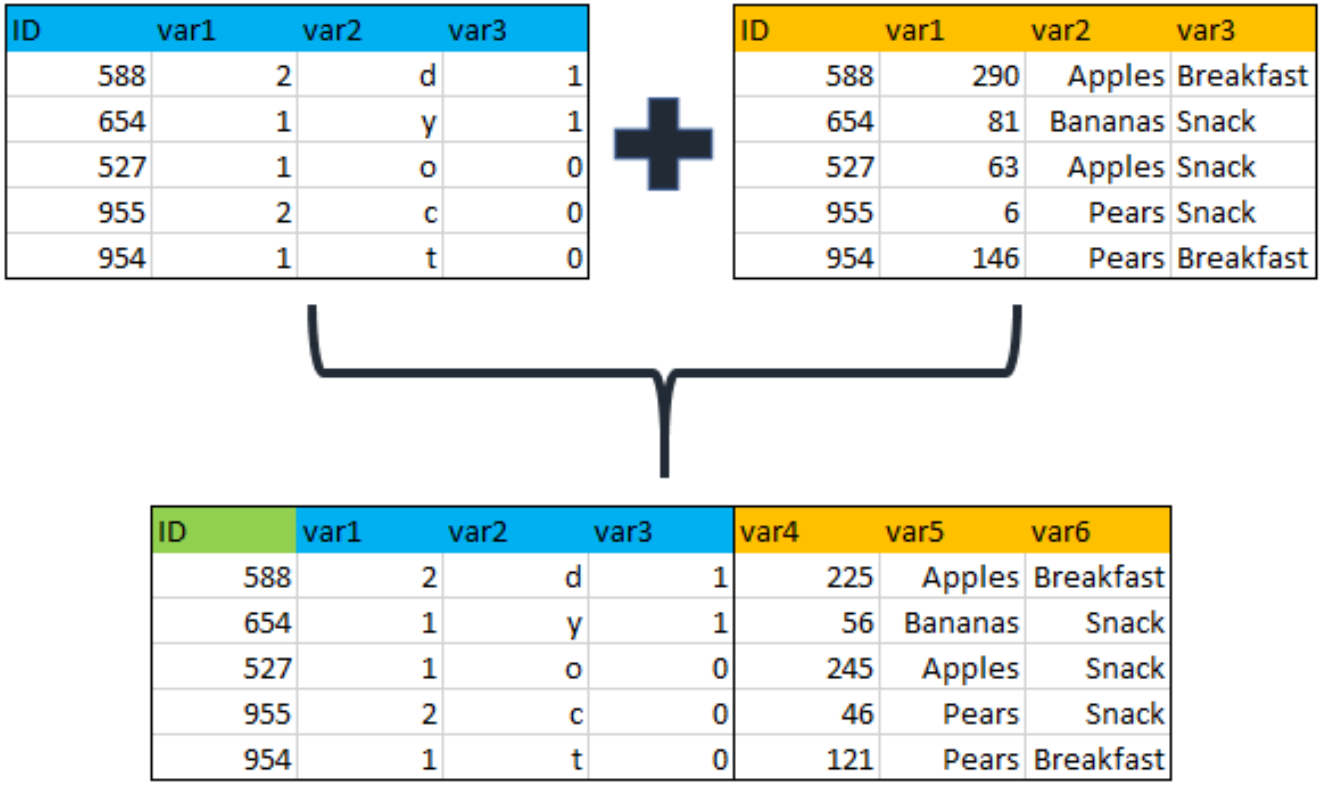
```
> data_1=data.frame(Name=c('Lenna','Malcom',"Akiko"),
+                   City=c("San Francisco","New York", "Tokyo"), Country=c("US","US","Japan"))
> data_2=data.frame(Name=c("Lenna","Thomas","Diane"),
+                   City=c("San Franscico", "London", "Chicago"), Country=c("US","UK","US"))
> data_1
    Name          City Country
1  Lenna San Francisco      US
2 Malcom      New York      US
3  Akiko         Tokyo   Japan
> data_2
    Name          City Country
1  Lenna San Franscico      US
2 Thomas        London      UK
3  Diane       Chicago      US
```

# rbind()

```
> data_new=rbind(data_1, data_2)
> data_new
    Name          City Country
1  Lenna San Francisco      US
2 Malcom      New York      US
3  Akiko         Tokyo   Japan
4  Lenna San Franscico      US
5 Thomas        London      UK
6  Diane       Chicago      US
```

# cbind()

Retorna a junção de colunas de duas bases de dados, quando estas têm o mesmo número de linhas.



| ID | var1 | var2 | var3 |
|---|---|---|---|
| 588 | 2 | d | 1 |
| 654 | 1 | y | 1 |
| 527 | 1 | o | 0 |
| 955 | 2 | c | 0 |
| 954 | 1 | t | 0 |

+

| ID | var1 | var2 | var3 |
|---|---|---|---|
| 588 | 290 | Apples | Breakfast |
| 654 | 81 | Bananas | Snack |
| 527 | 63 | Apples | Snack |
| 955 | 6 | Pears | Snack |
| 954 | 146 | Pears | Breakfast |

| ID | var1 | var2 | var3 | var4 | var5 | var6 |
|---|---|---|---|---|---|---|
| 588 | 2 | d | 1 | 225 | Apples | Breakfast |
| 654 | 1 | y | 1 | 56 | Bananas | Snack |
| 527 | 1 | o | 0 | 245 | Apples | Snack |
| 955 | 2 | c | 0 | 46 | Pears | Snack |
| 954 | 1 | t | 0 | 121 | Pears | Breakfast |

# cbind()

```
> data_1 <- data.frame(ID=c(588,654,527,955,954), # primeira coluna
+                      var1=c(2,1,1,2,1),         # segunda coluna
+                      var2=c('d','y','o','c','t'))# terceira coluna
> data_2 <- data.frame(ID=c(588,654,527,955,954),
+                      var3=c(290,81,63,6,146),
+                    var4=c('Apples','Bananas','Apples','Pears','Pears'),
+                    var5=c('Breakfast','Snack','Snack','Snack','Breakfast'))
> data_3 <- cbind(data_1,data_2)
> data_3
   ID var1 var2  ID var3    var4      var5
1 588    2    d 588  290  Apples Breakfast
2 654    1    y 654   81 Bananas     Snack
3 527    1    o 527   63  Apples     Snack
4 955    2    c 955    6   Pears     Snack
5 954    1    t 954  146   Pears Breakfast
```

# Merge/Join

- Merge permite unir duas bases dados com base numa identificação(chave-ID).
- Existem vários tipos de merge:
  - left join
  - right outer
  - inner join
  - full join

# Left join

*left join* - retorna todos elementos comuns nas duas tabelas (base de dados) e os restantes elementos na tabela a esquerda.

**A**

| OWNER | PET |
|---|---|
| Alice | Snowy |
| Bob | Mittens |
| Carol | Mittens |
| Dan | Goldie |
| Erin | Pancho |

**B**

| PET | SPECIES |
|---|---|
| Fido | Dog |
| Goldie | Goldfish |
| Mittens | Cat |
| Rex | Dog |
| Snowy | Dog |

| OWNER | PET | SPECIES |
|---|---|---|
| Alice | Snowy | Dog |
| Bob | Mittens | Cat |
| Carol | Mittens | Cat |
| Dan | Goldie | Goldfish |
| Erin | Pancho | NA |

# Left join

```
> data_1 <- data.frame(owner = c('Alice', 'Bob','Carol','Dan','Erin'),
+                      pet = c('Snowy','Mittens','Mittens', 'Goldie','Pancho'))
> data_2 <- data.frame(pet=c('Fido','Goldie','Mittens','Rex','Snowy'),
+                      species =c('Dog','Goldfish','Cat','Dog','Dog'))
> data_3 = merge(data_1,data_2, by.x='pet', by.y='pet', all.x=TRUE)
> data_3
      pet owner  species
1  Goldie   Dan Goldfish
2 Mittens   Bob      Cat
3 Mittens Carol      Cat
4  Pancho  Erin     <NA>
5   Snowy Alice      Dog
```

# Right join

**right join**- retorna todas as linhas comum nas duas tabelas e todas as restantes linhas da tabela a direita.

**A**

| OWNER | PET |
|---|---|
| Alice | Snowy |
| Bob | Mittens |
| Carol | Mittens |
| Dan | Goldie |
| Erin | Pancho |

**B**

| PET | SPECIES |
|---|---|
| Fido | Dog |
| Goldie | Goldfish |
| Mittens | Cat |
| Rex | Dog |
| Snowy | Dog |

| OWNER | PET | SPECIES |
|---|---|---|
| Alice | Snowy | Dog |
| Bob | Mittens | Cat |
| Carol | Mittens | Cat |
| Dan | Goldie | Goldfish |
| Erin | Pancho | NA |

# Right join

```
> data_1 <- data.frame(owner = c('Alice', 'Bob','Carol','Dan','Erin'),
+                      pet = c('Snowy','Mittens','Mittens', 'Goldie','Pancho'))
> data_2 <- data.frame(pet=c('Fido','Goldie','Mittens','Rex','Snowy'),
+                      species =c('Dog','Goldfish','Cat','Dog','Dog'))
> data_3 = merge(data_1,data_2, by.x='pet', by.y='pet', all.y=TRUE)
> data_3
      pet owner  species
1    Fido  <NA>      Dog
2  Goldie   Dan Goldfish
3 Mittens   Bob      Cat
4 Mittens Carol      Cat
5     Rex  <NA>      Dog
6   Snowy Alice      Dog
```

# Natural join

**natural join** retorna todas as linhas comuns nas duas tabelas (base de dados).

**A**

| OWNER | PET |
|---|---|
| Alice | Snowy |
| Bob | Mittens |
| Carol | Mittens |
| Dan | Goldie |
| Erin | Pancho |

**B**

| PET | SPECIES |
|---|---|
| Fido | Dog |
| Goldie | Goldfish |
| Mittens | Cat |
| Rex | Dog |
| Snowy | Dog |

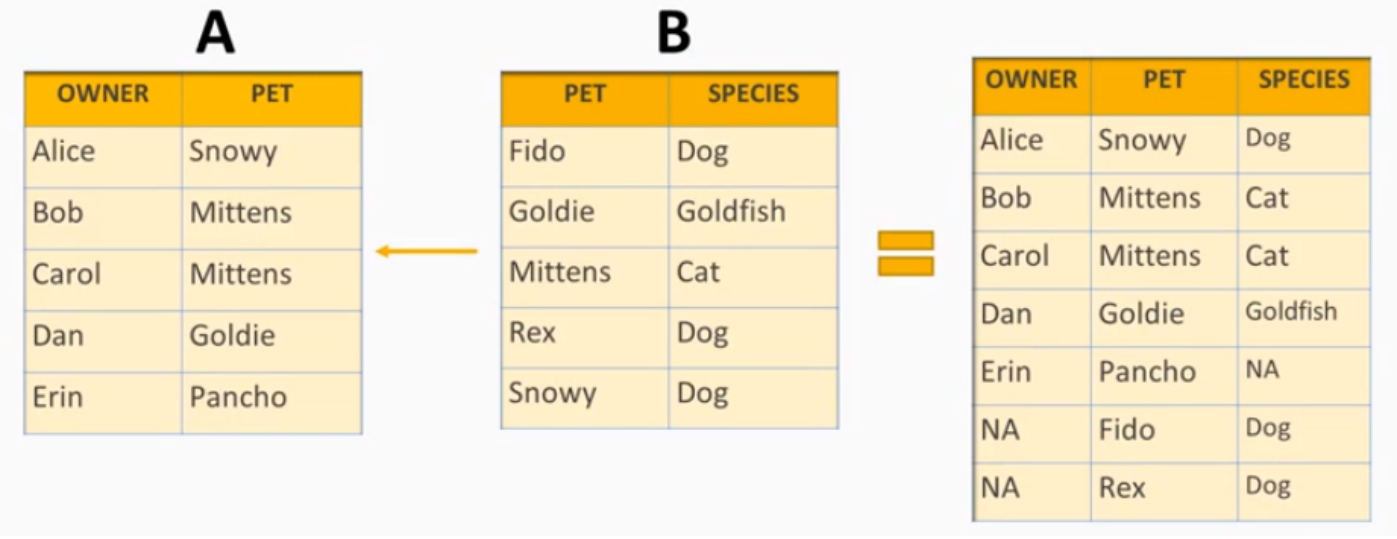
| OWNER | PET | SPECIES |
|---|---|---|
| Alice | Snowy | Dog |
| Bob | Mittens | Cat |
| Carol | Mittens | Cat |
| Dan | Goldie | Goldfish |

# Natural join

```
> data_1 <- data.frame(owner = c('Alice', 'Bob','Carol','Dan','Erin'),
+                      pet = c('Snowy','Mittens','Mittens', 'Goldie','Pancho'))
> data_2 <- data.frame(pet=c('Fido','Goldie','Mittens','Rex','Snowy'),
+                      species =c('Dog','Goldfish','Cat','Dog','Dog'))
> data_3 = merge(data_1,data_2, by.x='pet', by.y='pet')
> data_3
      pet owner  species
1  Goldie   Dan Goldfish
2 Mittens   Bob      Cat
3 Mittens Carol      Cat
4   Snowy Alice      Dog
```

# Full join

**full join** - retorna todas as linhas das tabelas unidas sejam elas correspondentes ou não.



**A**

| OWNER | PET |
|---|---|
| Alice | Snowy |
| Bob | Mittens |
| Carol | Mittens |
| Dan | Goldie |
| Erin | Pancho |

**B**

| PET | SPECIES |
|---|---|
| Fido | Dog |
| Goldie | Goldfish |
| Mittens | Cat |
| Rex | Dog |
| Snowy | Dog |

| OWNER | PET | SPECIES |
|---|---|---|
| Alice | Snowy | Dog |
| Bob | Mittens | Cat |
| Carol | Mittens | Cat |
| Dan | Goldie | Goldfish |
| Erin | Pancho | NA |
| NA | Fido | Dog |
| NA | Rex | Dog |

# Full Join

```
> data_1 <- data.frame(owner = c('Alice', 'Bob','Carol','Dan','Erin'),
+                      pet = c('Snowy','Mittens','Mittens', 'Goldie','Pancho'))
> data_2 <- data.frame(pet=c('Fido','Goldie','Mittens','Rex','Snowy'),
+                      species =c('Dog','Goldfish','Cat','Dog','Dog'))
> data_3 = merge(data_1,data_2, by.x='pet', by.y='pet', all=TRUE)
> data_3
      pet owner  species
1    Fido  <NA>      Dog
2  Goldie   Dan Goldfish
3 Mittens   Bob      Cat
4 Mittens Carol      Cat
5  Pancho  Erin     <NA>
6     Rex  <NA>      Dog
7   Snowy Alice      Dog
```

# Manipulaçao de dados- Adição de variáveis na base de dados

No R, temos três formas de adicionar uma variável à uma base dados

Usando `$`

- `data_frame$nova_varial<- conteudo_nova_variavel`

Usando o perador `[]`

- `data_frame[, "nova_varial"]<- conteudo_nova_variavel`

Usando a função `mutate()` da livraria `dplyr`

# Manipulaçao de dados- Adição de variáveis na base de dados

```
bird_data<- read.csv("Bird_Behaviour.csv", header = TRUE,
stringsAsFactors=FALSE)
str(bird_data)
'data.frame':   144 obs. of  8 variables:
 $ X          : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Ind        : chr  "PD1" "PD1" "PD1" "PD2" ...
 $ Species    : chr  "Passer_domesticus" "Passer_domesticus" "Passer_domesticus" "Passer_domesticus" ...
 $ Sex        : chr  "male" "male" "male" "female" ...
 $ Year       : int  2013 2014 2015 2013 2014 2015 2013 2014 2015 2013 ...
 $ FID        : int  5 2 8 10 10 6 4 10 8 2 ...
 $ Disturbance: int  8 40 30 35 15 6 49 11 9 44 ...
 $ Fledglings : int  1 4 4 3 0 2 0 3 2 2 ...
bird_data$log_FID<-rep(NA, dim(bird_data)[1])
bird_data$log_FID<-log(bird_data$FID)
```

# Manipulaçao de dados- Adição de variáveis na base de dados

Sepação de uma variável em duas variáveis

- Pode ser feita usando a função `separate()` da livraria `tidyr`

```
bird_data1 <- separate(bird_data, Species,
c("Genus","Species"), sep="_", remove=TRUE)

head(bird_data1)
  X Ind  Genus    Species    Sex Year FID Disturbance Fledglings   log_FID
1 1 PD1 Passer domesticus   male 2013   5           8          1 1.6094379
2 2 PD1 Passer domesticus   male 2014   2          40          4 0.6931472
3 3 PD1 Passer domesticus   male 2015   8          30          4 2.0794415
4 4 PD2 Passer domesticus female 2013  10          35          3 2.3025851
5 5 PD2 Passer domesticus female 2014  10          15          0 2.3025851
6 6 PD2 Passer domesticus female 2015   6           6          2 1.7917595
```

# Manipulaçao de dados- Adição de variáveis na base de dados

Combinar informação de duas variaveis em uma (concatenação)

- Pode ser feita usando a função `unite()` da livraria `tidyr`
- Também podemos usar a função `paste()` do R

```
bird_data2<- unite(bird_data1, "Genus_Species",
c(Genus, Species), sep="_", remove=TRUE)

bird_data1$Genus_Species1<-paste(bird_data1$Genus, bird_data1$Species,sep='_')

head(bird_data2)
  X Ind     Genus_Species    Sex Year FID Disturbance Fledglings   log_FID
1 1 PD1 Passer_domesticus   male 2013   5           8          1 1.6094379
2 2 PD1 Passer_domesticus   male 2014   2          40          4 0.6931472
3 3 PD1 Passer_domesticus   male 2015   8          30          4 2.0794415
4 4 PD2 Passer_domesticus female 2013  10          35          3 2.3025851
5 5 PD2 Passer_domesticus female 2014  10          15          0 2.3025851
6 6 PD2 Passer_domesticus female 2015   6           6          2 1.7917595
```

# Manipulaçao de dados- `dplyr`

Para a manipulação de dados, iremos usar as funções da livraria `dplyr`.

- `select`: retornar um subconjunto das colunas de uma base de dados.
- `filter`: extrair um subconjunto de linhas de uma base de dados com base em condições lógicas.
- `arrange`: reordenar linhas de um quadro de dados.
- `rename`: renomeia as as variáveis de uma base de dados.
- `mutate`: adiciona novas variáveis / colunas ou transforma variáveis existentes.
- `summarise`/ `summarize`: gera estatísticas sumárias de diferentes variáveis na base de dados.
- %>% : operador `pipe` usado para ligar várias acções.

# select

Para ilustrar o processo de manipulação de dados, vamos usar a base de dados `Chicago`.

```
> chicago <- readRDS("chicago.rds")
```

Antes de fazer qualquer operação, podemos inspeccionar base usando a função `str()` ou `dim()`.

```
> dim(chicago)
[1] 6940    8
> str(chicago)
'data.frame':   6940 obs. of  8 variables:
 $ city      : chr  "chic" "chic" "chic" "chic" ...
 $ tmpd      : num  31.5 33 33 29 32 40 34.5 29 26.5 32.5 ...
 $ dptp      : num  31.5 29.9 27.4 28.6 28.9 ...
 $ date      : Date, format: "1987-01-01" "1987-01-02" ...
 $ pm25tmean2: num  NA NA NA NA NA NA NA NA NA NA ...
 $ pm10tmean2: num  34 NA 34.2 47 NA ...
 $ o3tmean2  : num  4.25 3.3 3.33 4.38 4.75 ...
 $ no2tmean2 : num  20 23.2 23.8 30.4 30.3 ...
```

# select

Suponha que se pretende seleccionar as 3 primeiras colunas.

```
> library(dplyr)
> names(chicago)[1:3]
[1] "city" "tmpd" "dptp"
> sel_col <- select(chicago, city:dptp)
> head(sel_col)
  city tmpd   dptp
1 chic 31.5 31.500
2 chic 33.0 29.875
3 chic 33.0 27.375
4 chic 29.0 28.625
5 chic 32.0 28.875
6 chic 40.0 35.125
```

# select

Para alem de seleccionar, pode-se omitir as variáveis, bastando para tal usar o sinal menos `-`.

```
> select(chicago, -(city:dptp))
```

Este código indica que devemos incluir todas as variáveis excepto as variáveis `city` à `dptp`.

# select

A função `select ()` também permite uma sintaxe especial que permite especificar nomes de variáveis com base em padrões.Por exemplo, se você quiser manter todas as variáveis que terminam com um "2", poderíamos fazer:

```
> sel_col <- select(chicago, ends_with('2'))
> str(sel_col)
'data.frame':   6940 obs. of  4 variables:
 $ pm25tmean2: num  NA NA NA NA NA NA NA NA NA NA ...
 $ pm10tmean2: num  34 NA 34.2 47 NA ...
 $ o3tmean2  : num  4.25 3.3 3.33 4.38 4.75 ...
 $ no2tmean2 : num  20 23.2 23.8 30.4 30.3 ...
```

Ou, se quiséssemos manter todas as variáveis que começam com um "d", poderíamos fazer:

```
> sel_col <- select(chicago, starts_with('d'))
> str(sel_col)
'data.frame':   6940 obs. of  2 variables:
 $ dptp: num  31.5 29.9 27.4 28.6 28.9 ...
 $ date: Date, format: "1987-01-01" "1987-01-02" ...
```

# filter()

Extrair as linhas na base de dadoschicago onde os níveis de poluição são maiores que 30.

```
> sel_fil <- filter(chicago,pm25tmean2 > 30)
> str(sel_fil)
'data.frame':   194 obs. of  8 variables:
 $ city      : chr  "chic" "chic" "chic" "chic" ...
 $ tmpd      : num  23 28 55 59 57 57 75 61 73 78 ...
 $ dptp      : num  21.9 25.8 51.3 53.7 52 56 65.8 59 60.3 67.1 ...
 $ date      : Date, format: "1998-01-17" "1998-01-23" ...
 $ pm25tmean2: num  38.1 34 39.4 35.4 33.3 ...
 $ pm10tmean2: num  32.5 38.7 34 28.5 35 ...
 $ o3tmean2  : num  3.18 1.75 10.79 14.3 20.66 ...
 $ no2tmean2 : num  25.3 29.4 25.3 31.4 26.8 ...
```

# filter()

Podemos também estar interessados em filtrar todas as linhas que tem níveis de poluição maiores que 30 e temperatura maior que 80F.

```
> sel_fil <- filter(chicago,pm25tmean2 > 30 & tmpd > 80)
> str(sel_fil)
'data.frame':   17 obs. of  8 variables:
 $ city      : chr  "chic" "chic" "chic" "chic" ...
 $ tmpd      : num  81 81 82 84 85 84 82 82 81 82 ...
 $ dptp      : num  71.2 70.4 72.2 72.9 72.6 72.6 67.4 63.5 70.4 66.2 ...
 $ date      : Date, format: "1998-08-23" "1998-09-06" ...
 $ pm25tmean2: num  39.6 31.5 32.3 43.7 38.8 ...
 $ pm10tmean2: num  59 50.5 58.5 81.5 70 66 80.5 65 64 72.5 ...
 $ o3tmean2  : num  45.9 50.7 33 45.2 38 ...
 $ no2tmean2 : num  14.3 20.3 33.7 27.4 27.6 ...
```

# arrange()

A função `arrange()` permite ordenar os dados segundo uma coluna/variável.

Por exemplo, podemos ordenar os dados em função da data

```
> chicago <- arrange(chicago, date)
> head(chicago, n=10)
   city tmpd   dptp       date pm25tmean2 pm10tmean2  o3tmean2 no2tmean2
1  chic 31.5 31.500 1987-01-01         NA   34.00000  4.250000  19.98810
2  chic 33.0 29.875 1987-01-02         NA         NA  3.304348  23.19099
3  chic 33.0 27.375 1987-01-03         NA   34.16667  3.333333  23.81548
4  chic 29.0 28.625 1987-01-04         NA   47.00000  4.375000  30.43452
5  chic 32.0 28.875 1987-01-05         NA         NA  4.750000  30.33333
6  chic 40.0 35.125 1987-01-06         NA   48.00000  5.833333  25.77233
7  chic 34.5 26.750 1987-01-07         NA   41.00000  9.291667  20.58171
8  chic 29.0 22.000 1987-01-08         NA   36.00000 11.291667  17.03723
9  chic 26.5 29.000 1987-01-09         NA   33.28571  4.500000  23.38889
10 chic 32.5 27.750 1987-01-10         NA         NA  4.958333  19.54167
```

# arrange()

Para ordenar de forma decrescente é só fazer o uso da função `desc()`.

```
> chicago <- arrange(chicago, desc(date))
> head(chicago, n=10)
   city tmpd dptp       date pm25tmean2 pm10tmean2  o3tmean2 no2tmean2
1  chic   35 30.1 2005-12-31   15.00000       23.5  2.531250  13.25000
2  chic   36 31.0 2005-12-30   15.05714       19.2  3.034420  22.80556
3  chic   35 29.4 2005-12-29    7.45000       23.5  6.794837  19.97222
4  chic   37 34.5 2005-12-28   17.75000       27.5  3.260417  19.28563
5  chic   40 33.6 2005-12-27   23.56000       27.0  4.468750  23.50000
6  chic   35 29.6 2005-12-26    8.40000        8.5 14.041667  16.81944
7  chic   35 32.1 2005-12-25    6.70000        8.0 14.354167  13.79167
8  chic   37 35.2 2005-12-24   30.77143       25.2  1.770833  31.98611
9  chic   41 32.6 2005-12-23   32.90000       34.5  6.906250  29.08333
10 chic   22 23.3 2005-12-22   36.65000       42.5  5.385417  33.73026
```

# rename()

A função `rename()` permite alterar o nome da coluna/variável.

Por exemplo, vamos alterar o nome `date` para `data` e `pm25tmean2` para `nivel_de_poluicao`.

```
> chicago <- rename(chicago, data = date,  nivel_de_poluicao = pm25tmean2)
> head(chicago, n=10)
   city tmpd dptp       data nivel_de_poluicao pm10tmean2  o3tmean2 no2tmean2
1  chic   35 30.1 2005-12-31          15.00000       23.5  2.531250  13.25000
2  chic   36 31.0 2005-12-30          15.05714       19.2  3.034420  22.80556
3  chic   35 29.4 2005-12-29           7.45000       23.5  6.794837  19.97222
4  chic   37 34.5 2005-12-28          17.75000       27.5  3.260417  19.28563
5  chic   40 33.6 2005-12-27          23.56000       27.0  4.468750  23.50000
6  chic   35 29.6 2005-12-26           8.40000        8.5 14.041667  16.81944
7  chic   35 32.1 2005-12-25           6.70000        8.0 14.354167  13.79167
8  chic   37 35.2 2005-12-24          30.77143       25.2  1.770833  31.98611
9  chic   41 32.6 2005-12-23          32.90000       34.5  6.906250  29.08333
10 chic   22 23.3 2005-12-22          36.65000       42.5  5.385417  33.73026
```

# mutate()

A função `mutate()` existe para calcular transformações de variáveis em uma base de dados. Vamos converter a temperatura para graus celsius $(32^{\circ}F - 32) \times \frac{5}{9}$

```
> chicago <- mutate(chicago, temp_c = (tmpd-32)*5/9)
> head(chicago, n=10)
   city tmpd dptp       data nivel_de_poluicao pm10tmean2  o3tmean2 no2tmean2
1  chic   35 30.1 2005-12-31          15.00000       23.5  2.531250  13.25000
2  chic   36 31.0 2005-12-30          15.05714       19.2  3.034420  22.80556
3  chic   35 29.4 2005-12-29           7.45000       23.5  6.794837  19.97222
4  chic   37 34.5 2005-12-28          17.75000       27.5  3.260417  19.28563
5  chic   40 33.6 2005-12-27          23.56000       27.0  4.468750  23.50000
6  chic   35 29.6 2005-12-26           8.40000        8.5 14.041667  16.81944
7  chic   35 32.1 2005-12-25           6.70000        8.0 14.354167  13.79167
8  chic   37 35.2 2005-12-24          30.77143       25.2  1.770833  31.98611
9  chic   41 32.6 2005-12-23          32.90000       34.5  6.906250  29.08333
10 chic   22 23.3 2005-12-22          36.65000       42.5  5.385417  33.73026
      temp_c
1   1.666667
2   2.222222
3   1.666667
4   2.777778
5   4.444444
6   1.666667
7   1.666667
8   2.777778
9   5.000000
10 -5.555556
```

# mutate

A função `mutate()` permite criar uma variável sujeita a uma condição.

Por exemplo, crie uma variável níveis de poluição que assume valor 'baixo' poluição $<$ 30 e 'alto' caso contrário.

```
> chicago <- mutate(chicago,
+                   pol_ref = if_else(nivel_de_poluicao <30,'baixo','alto'))
> head(chicago)
  city tmpd dptp       data nivel_de_poluicao pm10tmean2  o3tmean2 no2tmean2
1 chic   35 30.1 2005-12-31          15.00000       23.5  2.531250  13.25000
2 chic   36 31.0 2005-12-30          15.05714       19.2  3.034420  22.80556
3 chic   35 29.4 2005-12-29           7.45000       23.5  6.794837  19.97222
4 chic   37 34.5 2005-12-28          17.75000       27.5  3.260417  19.28563
5 chic   40 33.6 2005-12-27          23.56000       27.0  4.468750  23.50000
6 chic   35 29.6 2005-12-26           8.40000        8.5 14.041667  16.81944
    temp_c pol_ref
1 1.666667   baixo
2 2.222222   baixo
3 1.666667   baixo
4 2.777778   baixo
5 4.444444   baixo
6 1.666667   baixo
```

# group_by()

A função group_by() é usada para gerar estatísticas sumárias nos estratos/grupos definidos por uma variável. Por exemplo, neste conjunto de dados de poluição do ar, você pode querer saber qual é o nível médio anual de PM2.5. Então o estrato é o ano.

```
> library(lubridate)
> chicago <- mutate(chicago, ano = year(data))
> anos <- group_by(chicago,ano )
> summarise(anos, pm25 = mean(nivel_de_poluicao, na.rm = TRUE),
+           o3 = max(o3tmean2, na.rm = TRUE),
+           no2 = median(no2tmean2, na.rm = TRUE))
# A tibble: 19 x 4
     ano  pm25    o3   no2
   <dbl> <dbl> <dbl> <dbl>
 1  1987 NaN    63.0  23.5
 2  1988 NaN    61.7  24.5
 3  1989 NaN    59.7  26.1
 4  1990 NaN    52.2  22.6
 5  1991 NaN    63.1  21.4
 6  1992 NaN    50.8  24.8
 7  1993 NaN    44.3  25.8
 8  1994 NaN    52.2  28.5
 9  1995 NaN    66.6  27.3
10  1996 NaN    58.4  26.4
11  1997 NaN    56.5  25.5
12  1998  18.3  50.7  24.6
13  1999  18.5  57.5  24.7
14  2000  16.9  55.8  23.5
15  2001  16.9  51.8  25.1
16  2002  15.3  54.9  22.7
17  2003  15.2  56.2  24.6
18  2004  14.6  44.5  23.4
19  2005  16.2  58.8  22.6
```

# group_by() , summarise() e %>%

Para mostrar mais aplicação destas duas funções, vamos usar a base de dados `iris`

- calcular o comprimento médio, máximo e mínimo das pétalas por espécie.

```
> data("iris")
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

# group_by() , summarise() e %>%

```
> iris %>% group_by(Species) %>%
+   summarize(mean(Petal.Length), max(Petal.Length), min(Petal.Length))
# A tibble: 3 x 4
  Species    `mean(Petal.Length)` `max(Petal.Length)` `min(Petal.Length)`
  <fct>                     <dbl>               <dbl>               <dbl>
1 setosa                     1.46                 1.9                 1
2 versicolor                 4.26                 5.1                 3
3 virginica                  5.55                 6.9                 4.5
```

# Exercícios - tente resolver, vai ajudar !!!

1) Quantos agregados têm acesso a electricidade?

2) Quantos vivem em casas modernas: - Uma casa é moderna se: -Chão: adobe, parquet, cimento, mosaico - Paredes: cimento, blocos, madeira - Tecto: Metal/zinco, madeira, telhas,

3) Transforme a idade registada em anos para idade em meses.

4) A dimensão da rede foi registada em metros quadrados, converta os valores para centímetros quadrados.

5) Seleccione todos indivíduos que professam a religião católica ou islámica.

6) Seleccione indivíduos que professam religião católica, cujo principal tipo de pesca seja arrasto ou rede de emalhar.

7) Determine o número de pescadores por cada tipo de pesca.

8) Calcule o número de respondentes por cada método de conservação e tipo de pesca.