

# PROGRAMAÇÃO EM R

## AULA 02 : Manipulação de dados no R: da carga ao processo de transformação

# AGENDA

- Importação e Manipulação de arquivos no R;
  - Carregando arquivos em TXT;
  - Carregando arquivos CSV;
  - Carregando planilhas excel;
- *Data wrangling*: pré-processamento;
- Pacotes de manipulação e modelagem de dados no R;
- Limpeza, transformação e manipulação;
- Pipe %>%
- Remodelagem de dados;
- Reshape;
- Substituindo valores;
- Trabalhando com datas;
- Subconjuntos – Fatiamento.

# IMPORTAÇÃO E MANIPULAÇÃO DE ARQUIVOS NO R

- O primeiro passo de um projeto de *data science* é a carga de dados;
- Carga de arquivos .txt:
  - `read.table()` – faz a leitura de um arquivo em formato de tabela e carrega como dataframe;
  - `read_table()` - faz a leitura de um arquivo separado por espaço em formato de tabela e carrega como dataframe, não dá para mudar o sep (pacote readr);
  - `write.table()` – gravar os dados em arquivo;
- Carga de arquivos .csv:
  - `read.csv()` – carrega arquivos separados por vírgula e ponto como separador decimal (pacote utils);
  - `read.csv2()` - carrega arquivos separados por ponto e vírgula e vírgula como separador decimal (pacote utils);
  - `read_csv()` – mesmo objetivo do `read.csv` (pacote\_readr - carrega metadados);
- Carga de arquivos excel:
  - pacote readxl;
  - `read_excel()`;
  - `excel_sheets()` – listar planilhas;
  - carregar todas as planilhas de uma vez só: `lapply(excel_sheets(),read_excel, path)`.

# IMPORTAÇÃO E MANIPULAÇÃO DE ARQUIVOS NO R

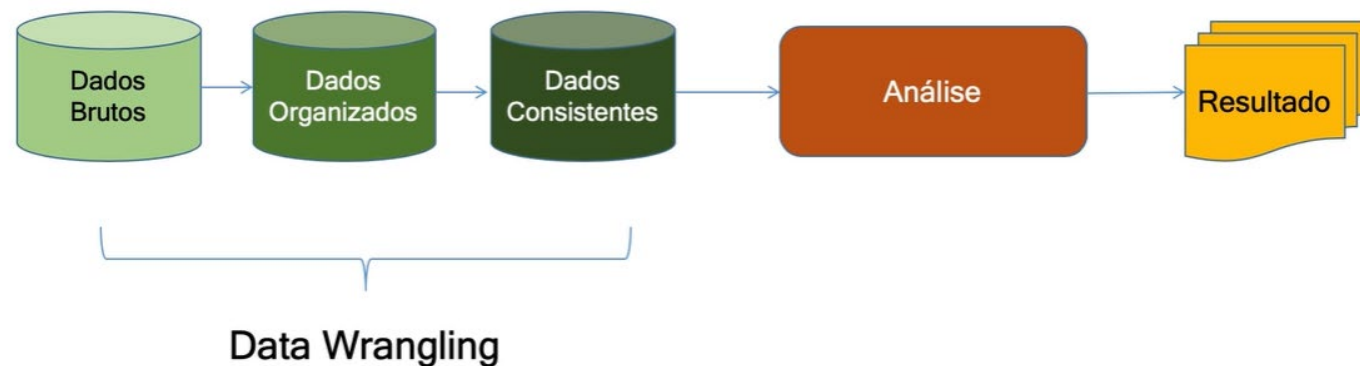
- Parâmetros comuns das funções de carga de dados:
  - header;
  - col.names;
  - na.string;
  - colClasses;
  - sep;
  - stringsAsFactors.

# IMPORTAÇÃO E MANIPULAÇÃO DE ARQUIVOS NO R

- Pontos relevantes no momento da carga de arquivos:
  - Geralmente, a primeira linha de cada arquivo é o cabeçalho com os nomes das colunas;
  - Para nomear as colunas utilize títulos curtos;
  - Evite o uso de caracteres especiais;
  - O tratamento de dados NA será feito no processo de limpeza.

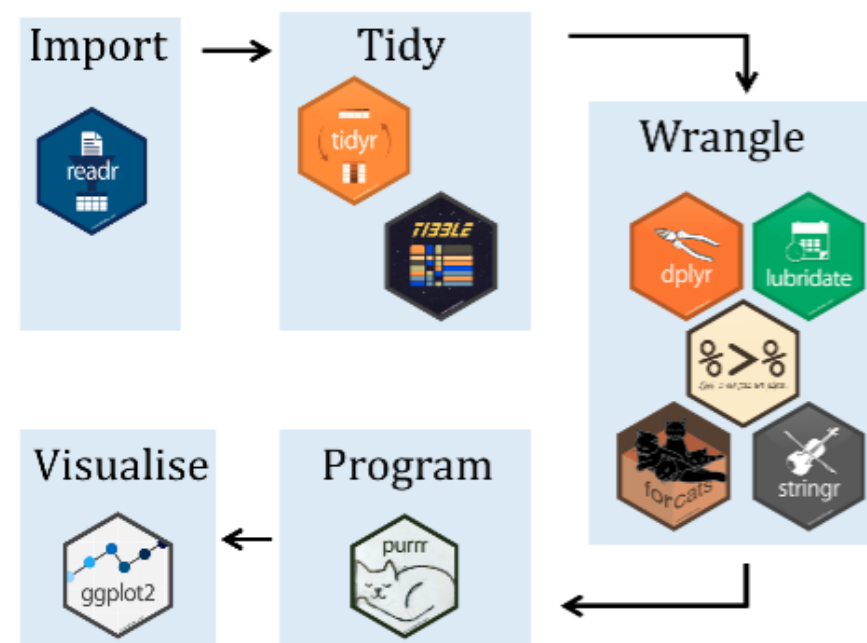
# DATA WRANGLING E PRÉ-PROCESSAMENTO

- O objetivo do processo de manipulação e dados é preparar dados para o processo de análise, já que raramente os dados virão prontos para serem utilizados para análise e ou modelagem.
- As tarefas do processo de manipulação de dados são:
  - Limpeza;
  - Processamento;
  - Organização;
  - Manipulação.



# PACOTE TYDYVERSE

- Conjuntos de pacotes criado pelo estatístico Wickham (2017) e seus colaboradores para realizar a leitura, escrita, manipulação e visualização de dados.



Etapas e pacotes do *tidyverse*

Fonte: <https://dataat.github.io/introducao-analise-de-dados-espaciais/r.html>

# PACOTES PARA MANIPULAÇÃO DE DADOS NO R

- O objetivo principal é garantir que cada variável esteja em coluna e cada observação deve estar em uma linha;

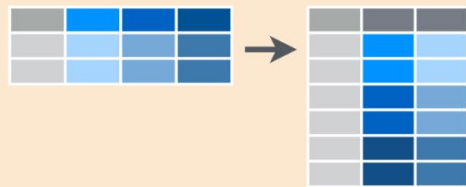


- Pacotes principais:
  - tidyr: modificar o formato dos dados;
    - separate(), spread(), gather();
  - dplyr: transformar os dados;
    - select(), filter(), group\_by, summarise(), join(), mutate(), arrange().



# PACOTES PARA MANIPULAÇÃO e MODELAGEM DE DADOS

- Funções `tydr()` - modificar o formato dos dados



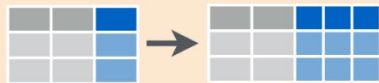
**`tidyr::gather(cases, "year", "n", 2:4)`**

Gather columns into rows.



**`tidyr::spread(pollution, size, amount)`**

Spread rows into columns.



**`tidyr::separate(storms, date, c("y", "m", "d"))`**

Separate one column into several.




**`tidyr::unite(data, col, ..., sep)`**

Unite several columns into one.


# PACOTES PARA MANIPULAÇÃO E MODELAGEM DE DADOS

## Funções dplyr() - transformar os dados


### Reshaping Data - Change the layout of a data set




**tidyr::gather(cases, "year", "n", 2:4)**  
Gather columns into rows.



**tidyr::spread(pollution, size, amount)**  
Spread rows into columns.



**tidyr::separate(storms, date, c("y", "m", "d"))**  
Separate one column into several.



**tidyr::unite(data, col, ..., sep)**  
Unite several columns into one.


**dplyr::data\_frame(a = 1:3, b = 4:6)**  
Combine vectors into data frame (optimized).

**dplyr::arrange(mtcars, mpg)**  
Order rows by values of a column (low to high).

**dplyr::arrange(mtcars, desc(mpg))**  
Order rows by values of a column (high to low).

**dplyr::rename(tb, y = year)**  
Rename the columns of a data frame.

### Subset Observations (Rows)



**dplyr::filter(iris, Sepal.Length > 7)**  
Extract rows that meet logical criteria.

**dplyr::distinct(iris)**  
Remove duplicate rows.

**dplyr::sample\_frac(iris, 0.5, replace = TRUE)**  
Randomly select fraction of rows.

**dplyr::sample\_n(iris, 10, replace = TRUE)**  
Randomly select n rows.

**dplyr::slice(iris, 10:15)**  
Select rows by position.

**dplyr::top\_n(storms, 2, date)**  
Select and order top n entries (by group if grouped data).

Logic in R - ?Comparison, ?base::Logic			
<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&,  , !, xor, any, all	Boolean operators


### Subset Variables (Columns)



**dplyr::select(iris, Sepal.Width, Petal.Length, Species)**  
Select columns by name or helper function.

Helper functions for select - ?select	
<b>select(iris, contains("x"))</b>	Select columns whose name contains a character string.
<b>select(iris, ends_with("Length"))</b>	Select columns whose name ends with a character string.
<b>select(iris, everything())</b>	Select every column.
<b>select(iris, matches("t"))</b>	Select columns whose name matches a regular expression.
<b>select(iris, num_range("x", 1:5))</b>	Select columns named x1, x2, x3, x4, x5.
<b>select(iris, one_of(c("Species", "Genus")))</b>	Select columns whose names are in a group of names.
<b>select(iris, starts_with("Sepal"))</b>	Select columns whose name starts with a character string.
<b>select(iris, Sepal.Length:Petal.Width)</b>	Select all columns between Sepal.Length and Petal.Width (inclusive).
<b>select(iris, -Species)</b>	Select all columns except Species.

### Summarise Data



**dplyr::summarise(iris, avg = mean(Sepal.Length))**  
Summarise data into single row of values.

**dplyr::summarise\_each(iris, funs(mean))**  
Apply summary function to each column.

**dplyr::count(iris, Species, wt = Sepal.Length)**  
Count number of rows with each unique value of variable (with or without weights).

**summary function**

Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:


<b>dplyr::first</b> First value of a vector.	<b>min</b> Minimum value in a vector.
<b>dplyr::last</b> Last value of a vector.	<b>max</b> Maximum value in a vector.
<b>dplyr::nth</b> Nth value of a vector.	<b>mean</b> Mean value of a vector.
<b>dplyr::n</b> # of values in a vector.	<b>median</b> Median value of a vector.
<b>dplyr::n_distinct</b> # of distinct values in a vector.	<b>var</b> Variance of a vector.
<b>IQR</b> IQR of a vector.	<b>sd</b> Standard deviation of a vector.

### Group Data


**dplyr::group\_by(iris, Species)**  
Group data into rows with the same value of Species.

**dplyr::ungroup(iris)**  
Remove grouping information from data frame.

**iris %>% group\_by(Species) %>% summarise(...)**  
Compute separate summary row for each group.



### Make New Variables



**dplyr::mutate(iris, sepal = Sepal.Length + Sepal.Width)**  
Compute and append one or more new columns.

**dplyr::mutate\_each(iris, funs(min\_rank))**  
Apply window function to each column.

**dplyr::transmute(iris, sepal = Sepal.Length + Sepal.Width)**  
Compute one or more new columns. Drop original columns.

**window function**

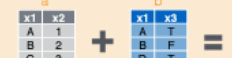
Mutate uses **window functions**, functions that take a vector of values and return another vector of values, such as:

<b>dplyr::lead</b> Copy with values shifted by 1.	<b>dplyr::cumall</b> Cumulative all
<b>dplyr::lag</b> Copy with values lagged by 1.	<b>dplyr::cumany</b> Cumulative any
<b>dplyr::dense_rank</b> Ranks with no gaps.	<b>dplyr::cummean</b> Cumulative mean
<b>dplyr::min_rank</b> Ranks. Ties get min rank.	<b>cumsum</b> Cumulative sum
<b>dplyr::percent_rank</b> Ranks rescaled to [0, 1].	<b>cummax</b> Cumulative max
<b>dplyr::row_number</b> Ranks. Ties got to first value.	<b>cummin</b> Cumulative min
<b>dplyr::ntile</b> Bin vector into n buckets.	<b>cumprod</b> Cumulative prod
<b>dplyr::between</b> Are values between a and b?	<b>pmax</b> Element-wise max
<b>dplyr::cume_dist</b> Cumulative distribution.	<b>pmin</b> Element-wise min

**iris %>% group\_by(Species) %>% mutate(...)**  
Compute new variables by group.



### Combine Data Sets



**Joining**

**dplyr::left\_join(a, b, by = "x1")**  
Join matching rows from b to a.

**dplyr::right\_join(a, b, by = "x1")**  
Join matching rows from a to b.

**dplyr::inner\_join(a, b, by = "x1")**  
Join data. Retain only rows in both sets.

**dplyr::full\_join(a, b, by = "x1")**  
Join data. Retain all values, all rows.

**Filtering Joins**

**dplyr::semi\_join(a, b, by = "x1")**  
All rows in a that have a match in b.

**dplyr::anti\_join(a, b, by = "x1")**  
All rows in a that do not have a match in b.



**Set Operations**

**dplyr::intersect(y, z)**  
Rows that appear in both y and z.

**dplyr::union(y, z)**  
Rows that appear in either or both y and z.

**dplyr::setdiff(y, z)**  
Rows that appear in y but not z.

**Binding**

**dplyr::bind\_rows(y, z)**  
Append z to y as new rows.

**dplyr::bind\_cols(y, z)**  
Append z to y as new columns.  
Caution: matches rows by position.

# LIMPEZA, TRANSFORMAÇÃO e MODELAGEM

- Read\_csv carrega no formato tibble (tabela dataframe);
- Olhando o conjunto de dados primeiramente: str() ou glimpse();
- Função mutate(): criar um nova variável no dataframe;
  - A função mutate() não inclui a variável no dataframe, ela apresenta o cálculo em tela;
- Função select(): selecionar colunas em conjunto de dados;
- Função filter(): selecionar dados a partir de uma ou mais condições (separar por vírgula);
- Função arrange(): ordenar um dataframe por uma ou mais colunas;
  - O padrão é a ordenação crescente, mas ao incluir a palavra desc, a ordenação passa a ser decrescente;
- Função summarize(): resumir (média, mínimo, máximo, total) o conjunto de dados;
- Função group by(): resumir os dados a partir de uma determinada coluna;
  - Geralmente a função group by é utilizada junto com o summarize;
- Função join(): combinar conjunto de dados;
- Função pivot(): mudar linhas e colunas, tendo maior usabilidade que a gather() e spread();
- Boa prática: as alterações realizadas devem ser armazenadas em outro dataframe, mantendo o original inalterado.

# PIPE %>% OPERADOR DE CONCATENAÇÃO

- Torna a escrita e leitura de códigos mais intuitiva e compreensível;
- Utiliza o valor resultante da expressão do lado esquerdo como primeiro argumento da função do lado direito;
- Deve-se utilizar para não escrever a todo momento o nome do dataframe;

## Código sem o PIPE

```
x <- c(1,2,3,4,5)  
sqrt(sum(x))
```



## Código com o PIPE

```
x %>% sum() %>% sqrt()
```

# MODELAGEM DE DADOS

- Mudança da forma (shape) dos dados;
  - Transformar o que é linha em coluna ou vice versa;
- Função `gather()`: transformar colunas em linhas (novas colunas, colunas que serão eliminadas)
- Função `spread()`: transformar linhas em colunas (oposto do `gather()`);
- Função `separate()`: separar uma coluna em várias colunas;
- Função `unite()`: unificar várias colunas em uma só (oposto do `separate()`).

# RESHAPE2

- Função melt(): transformar colunas em linhas (derreter)
- Função dcast(): transformar linhas em colunas(oposto do melt - fundir);

# SUBSTITUINDO VALORES

- Vamos ver como:
  - Substituir valores zerados;
  - Substituir valores NA de todo dataframe
  - Substituir valores NA de uma coluna específica de um dataframe;
  - Substituir valores NA pela média dos demais valores da coluna.

# TRABALHANDO COM DATAS

- Pacote recomendado: lubridate()
  - Funções:
    - year(): extrair ano;
    - month(): extrair mês;
    - day(): extrair dia;
    - week(): extrair semana;
    - wday(): extrair dia da semana;
    - second(): extrair os segundos;
    - minute(): extrair os minutos;
    - hour(): extrair a hora;
    - dmy() – transforma uma string no formato de data, considerando a sequência dia-mês-ano.
  - Criando intervalos:

```
inicio <- dmy("01-04-2023")  
fim <- dmy("30-10-2023")  
int<- interval(inicio, fim)  
int  
## [1] 2023-04-01 UTC--2023-10-30 UTC
```



# SUBCONJUNTO SATIAMENTO

- Em vetores:
  - Para retornar mais de um elemento: `v[c()]` – (pode colocar o índice ou o nome da coluna);
- Em matrizes:
  - Especificando o intervalo: `mat[:,]` ;
  - Também pode selecionar os elementos: `mat[c()]`;
- Em dataframes:
  - Selecionando pelo nome da coluna: `df$`;
  - Selecionando um conjunto de linhas e colunas: `df[c(),]`;
  - Selecionando linhas a partir de uma condição: `df[df$colname==10,]`;
  - Selecionando uma coluna a partir de uma variável: `df[,vars]`.