

R Básico

Aula 2 - Parte 2

João Matheus
Lineu Alberto

PET - Estatística
Universidade Federal do Paraná

PET-Estatística
2019
UFPR



PET-Estatística
UFPR

- 1 Tidyverse
- 2 Principais funcionalidades
- 3 Operador Pipe ($\%>\%$)
- 4 tibble
- 5 readr
- 6 Links úteis



PET-Estatística
UFPR



PET-Estatística
UFPR

1 Tidyverse

2 Principais funcionalidades

3 Operador Pipe ($\%>\%$)

4 tibble

5 readr

6 Links úteis



PET-Estatística UFPR



PET-Estatística
UFPR

O Tidyverse

- O tidyverse oferece uma reimplementação e extensão das funcionalidades básicas do R para manipulação e visualização de dados.
- É composto por 8 pacotes principais e diversos outros secundários.
- Foram planejados e construídos para trabalhar de forma conjunta.
- A gramática, argumentos e filosofia dos pacotes é mais intuitiva que o R base.
- Estas vantagens tornam o código mais simples de desenvolver e ler.



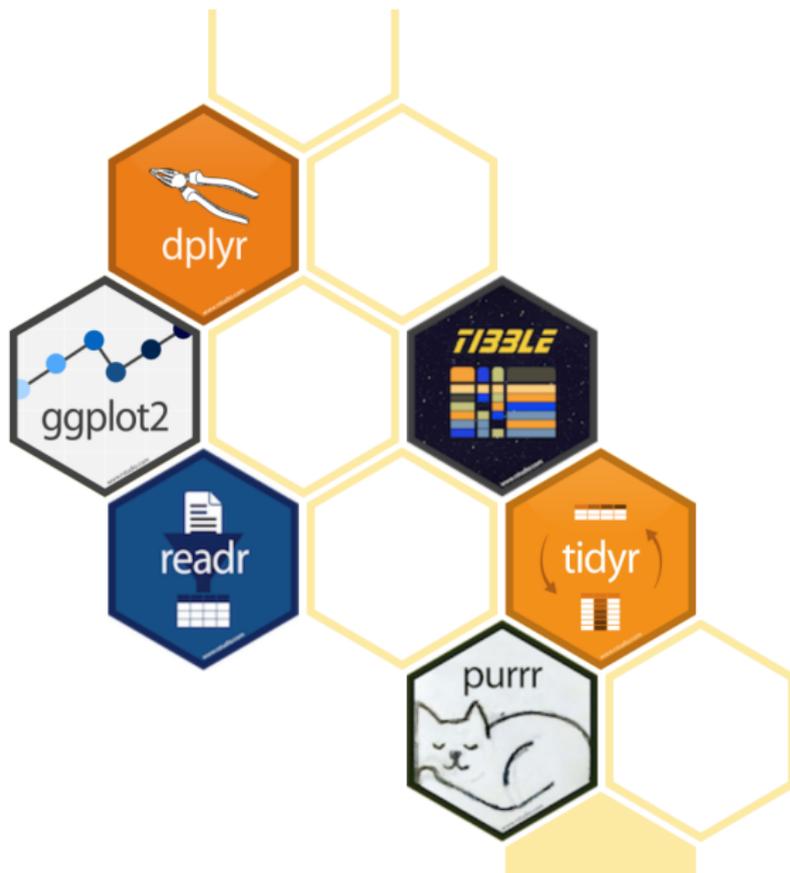
PET-Estatística
UFPR



PET-Estatística
UFPR



Pacotes tidyverse



- 1 Tidyverse
- 2 Principais funcionalidades**
- 3 Operador Pipe (`%>%`)
- 4 tibble
- 5 readr
- 6 Links úteis



PET-Estatística
UFPR

- Documentação: <https://tibble.tidyverse.org/>
- É uma reimplementação do Data Frame tradicional do R com melhorias.
- As mudanças dão maior consistência à manipulação de dados.
- Possui método `print()` mais informativo e conciso.
- Possui funções para criação de tibbles e operações básicas como adicionar linhas e colunas.



PET-Estatística
UFPR

- Documentação: <https://readr.tidyverse.org/>
- Pacote para importação de dados no formato texto.
- Dividido em funções de leitura, escrita e parsing.
- Possui diversas opções para controle de importação como: encoding, separador, decimal, aspas, comentários, etc.



- Documentação: <https://tidyr.tidyverse.org/>
- Pacote para ajustar conjuntos de dados para o formato tabular adequado (tidy).
- Possui funções para alterar disposição dos dados (formato largo e longo).
- Possui também funções para tratamento de valores ausentes (NA ou missing).



PET-Estatística
UFPR

- Documentação: <https://dplyr.tidyverse.org/>
- Pacote voltado para a manipulação de dados.
- Possui funções para adicionar colunas, selecionar, filtrar, rearranjar, sumarizar, renomear, etc.
- É um dos pacotes mais importantes e representativos no Tidyverse tendo em vista o tempo que se gasta na prática com a arrumação dos dados.



- Documentação: <https://ggplot2.tidyverse.org/>
- Pacote utilizado para a Geração de gráficos.
- Implementação baseada no **The Grammar of Graphics**.
- Leia a publicação no site do PET-Estatística UFPR: **Primeiros Passos com ggplot2**.



- Documentação: <https://forcats.tidyverse.org/>
- Pacote destinado à manipulação de variáveis categóricas.
- Permite renomear, reordenar, aglutinar níveis, etc.



PET-Estatística
UFPR

- Documentação: <https://stringr.tidyverse.org/>
- Pacote destinado à manipulação de strings.



PET-Estatística
UFPR

- Documentação: <https://purrr.tidyverse.org/>
- Pacote com funções para programação funcional.
- Trata-se de uma família apply melhorada.
- Possui funções para percorrer vetores, listas, colunas, linhas, etc.



PET-Estatística
UFPR

Outros pacotes tidyverse

- **broom**: Sumarização de informações importantes sobre modelos em tibbles ([link](#)).
- **haven**: Importação/exportação de conjuntos de dados com extensões oriundas de outros pacotes estatísticos como SAS, SPSS e Stata ([link](#)).
- **hms** e **lubridate**: Formatação de datas. Links: [hms](#) e [lubridate](#).
- **magrittr**: Oferece operadores para facilitar a escrita de código ([link](#)).
- **modelr**: Funções para auxílio no processo de modelagem ([link](#)).
- **readxl**: Importação e exportação de planilhas Excel ([link](#)).
- **rvest**: Funções para web scraping ([link](#)).

- 1 Tidyverse
- 2 Principais funcionalidades
- 3 Operador Pipe ($\%>\%$)**
- 4 tibble
- 5 readr
- 6 Links úteis



PET-Estatística
UFPR

Operador Pipe ($\%>\%$)

- O pacote **magrittr** é o pacote no qual o operador Pipe ($\%>\%$) está inserido.
- É um dos pacotes componentes do Tidyverse.
- O operador $\%>\%$ torna a leitura e escrita de códigos mais lógica e compreensível quando comparada ao R base, pois o código é estruturado da esquerda para a direita, evitando funções aglutinadas (uma dentro da outra) e minimizando a necessidade de variáveis locais.
- A melhor forma de entender o funcionamento é com exemplos:



- $x \%>\% f$ equivale a $f(x)$.
- $x \%>\% f(y)$ equivale a $f(x, y)$.
- $x \%>\% f \%>\% g \%>\% h$ equivale a $h(g(f(x)))$.

PET-Estatística
UFPR

Exemplo disponível em <https://www.curso-r.com/material/pipe/>

- Receita de bolo sem pipe:

```
esfrie(  
  asse(  
    coloque(  
      bata(  
        acrescente(  
          recipiente(  
            rep("farinha", 2),  
            "água", "fermento",  
            "leite", "óleo"), "farinha",  
            até = "macio"),  
          duração = "3min"),  
        lugar = "forma", tipo = "grande", untada = TRUE),  
        duração = "50min"), "geladeira", "20min")
```

Exemplo disponível em <https://www.curso-r.com/material/pipe/>

- Receita de bolo com pipe:

```
recipiente(rep("farinha", 2), "água", "fermento", "leite", "óleo") %>%  
acrescente("farinha", até = "macio") %>%  
bata(duração = "3min") %>%  
coloque(lugar = "forma", tipo = "grande", untada = TRUE) %>%  
asse(duração = "50min") %>%  
esfrie("geladeira", "20min")
```

PET-Estatística
UFPR

%>% - Exemplo com funções do R

- Sem pipe:

```
round(mean(1:100), digits = 1)
## [1] 50.5
```

- Com pipe:

```
1:100 %>%
  mean() %>%
  round(digits = 1)
## [1] 50.5
```

PEI-estatística
UFPR

- 1 Tidyverse
- 2 Principais funcionalidades
- 3 Operador Pipe (`%>%`)
- 4 tibble**
- 5 readr
- 6 Links úteis



PET-Estatística
UFPR

- Um Data Frame é uma estrutura de dados nativa do R para representação de dados tabulares. Lembre da aula 1, são vetores concatenados.
- O Tibble é uma reimplementação da estrutura básica do data frame com uma série de melhorias.
- É mais informativo, mais consistente, mais simples de converter, as células podem representar objetos complexos como vetores, matrizes, data frames, etc.



PET-Estatística
UFPR

- A principal diferença entre tibbles e data frames: `print()`.
- Se você printar um tibble serão mostradas apenas as 10 primeiras linhas e todas as colunas.
- Se você printar um dataframe TODO o data frame é printado.
- Vejamos as principais funcionalidades do Tibble.



PET-Estatística
UFPR

Criação de um tibble por especificação de colunas.

- A função `tibble()` é muito similar com a função `data.frame()`.
- Uma importante diferença é que utilizando a `tibble()`, uma terceira coluna pode ser calculada com base em duas anteriores.

```
tibble(x = c(10, 20,30),  
       y = c(1,2,3),  
       z = x+y)
```

```
## # A tibble: 3 x 3  
##       x     y     z  
##   <dbl> <dbl> <dbl>  
## 1     10     1    11  
## 2     20     2    22  
## 3     30     3    33
```

UFPR

Criação por especificação de linhas.

- Com a função `tribble()` é possível gerar um tibble especificando o conteúdo das linhas.

```
tribble(~col1, ~col2, ~col3,  
        "linha1", 2, 3.6,  
        "linha1", 1, 8.5)  
  
## # A tibble: 2 x 3  
##   col1    col2  col3  
##   <chr> <dbl> <dbl>  
## 1 linha1     2   3.6  
## 2 linha1     1   8.5
```

PEI-ESTADÍSTICA
UFPR

Coerção de vetores nomeados.

- A conversão de vetores nomeados se dá pela função `enframe()`.

```
notas <- c("Nome1" = 1, "Nome2" = 2, "Nome3" = 3, "Nome4" = 4)

notas %>%
  enframe(name = "aluno", value = "nota")

## # A tibble: 4 x 2
##   aluno  nota
##   <chr> <dbl>
## 1 Nome1     1
## 2 Nome2     2
## 3 Nome3     3
## 4 Nome4     4
```

Coerção de matrizes e dataframes.

- A conversão de matrizes e data frames é feita através da função `as_tibble()`.

```
matrix(1:12, ncol = 3) %>% as_tibble()
```

```
## # A tibble: 4 x 3
##   V1     V2     V3
##   <int> <int> <int>
## 1     1     5     9
## 2     2     6    10
## 3     3     7    11
## 4     4     8    12
```

PET-Estatística
UFPR

Coerção de matrizes e dataframes.

```
iris %>% as_tibble()
```

```
## # A tibble: 150 x 5
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
```

```
##           <dbl>         <dbl>         <dbl>         <dbl> <fct>
```

```
## 1           5.1           3.5           1.4           0.2 setosa
```

```
## 2           4.9           3             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             3.6           1.4           0.2 setosa
```

```
## 6           5.4           3.9           1.7           0.4 setosa
```

```
## 7           4.6           3.4           1.4           0.3 setosa
```

```
## 8           5             3.4           1.5           0.2 setosa
```

```
## 9           4.4           2.9           1.4           0.2 setosa
```

```
## 10          4.9           3.1           1.5           0.1 setosa
```

```
## # ... with 140 more rows
```

UFPR

- Dependendo do operador utilizado para fazer seleção condicional em um tibble o resultado pode ser um vetor ou um tibble, vejamos alguns casos.

```
tb <- iris[1:4, ] %>%  
as_tibble()
```

- Resulta em vetor:

```
tb$Petal.Length  
## [1] 1.4 1.4 1.3 1.5
```

```
tb[["Petal.Length"]]  
## [1] 1.4 1.4 1.3 1.5
```

```
tb[[3]]  
## [1] 1.4 1.4 1.3 1.5
```

- Resulta em tibble:

```
tb[, 3]
## # A tibble: 4 x 1
##   Petal.Length
##         <dbl>
## 1           1.4
## 2           1.4
## 3           1.3
## 4           1.5
```

```
tb[, "Petal.Length"]
## # A tibble: 4 x 1
##   Petal.Length
##         <dbl>
## 1           1.4
## 2           1.4
## 3           1.3
## 4           1.5
```

- Resulta em tibble:

```
tb[, c(3:4)]  
  
## # A tibble: 4 x 2  
##   Petal.Length Petal.Width  
##     <dbl>      <dbl>  
## 1     1.4        0.2  
## 2     1.4        0.2  
## 3     1.3        0.2  
## 4     1.5        0.2
```

```
tb[1, ]  
  
## # A tibble: 1 x 5  
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
##     <dbl>      <dbl>      <dbl>      <dbl> <fct>  
## 1     5.1        3.5         1.4        0.2 setosa
```

- Com a função `add_column()` é possível adicionar colunas a um tibble existente.

```
data.frame(col1 = c(1,2,3),  
           col2 = c(4,5,6)) %>%  
  as_tibble() %>%  
  add_column(col3 = c(7, 8, 9))
```

```
## # A tibble: 3 x 3  
##   col1 col2 col3  
##   <dbl> <dbl> <dbl>  
## 1     1     4     7  
## 2     2     5     8  
## 3     3     6     9
```

- Com a função `add_row()` é possível adicionar linhas a um tibble existente.

```
data.frame(col1 = c(1,2,3),  
           col2 = c(4,5,6)) %>%  
  as_tibble() %>%  
  add_row(col1 = 4, col2 = 7)
```

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

UFPR

Quando é necessário printar mais ou menos que as 10 primeiras linhas:

- Por default, ao printar um tibble apenas as 10 primeiras linhas são mostradas, caso haja a necessidade de mostrar mais ou menos que 10 linhas basta especificar os argumentos do print():

```
iris %>%
  as_tibble() %>%
  print(n = 1, width = Inf)

## # A tibble: 150 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##         <dbl>         <dbl>         <dbl>         <dbl> <fct>
## 1           5.1           3.5           1.4           0.2 setosa
## # ... with 149 more rows
```

UFPR

- 1 Tidyverse
- 2 Principais funcionalidades
- 3 Operador Pipe (`%>%`)
- 4 tibble
- 5 readr
- 6 Links úteis



PET-Estatística
UFPR

- Serve não só para leitura (importação) mas também para escrita (exportação).
- readr possui 6 funções de leitura para diferentes tipos de formatos:
 - 1 `read_csv()`: comma separated (CSV) files.
 - 2 `read_tsv()`: tab separated files.
 - 3 `read_delim()`: general delimited files.
 - 4 `read_fwf()`: fixed width files.
 - 5 `read_table()`: tabular files (dados tabulares separados por espaço).
 - 6 `read_log()`: web log files.



- Todas as read tem os mesmos argumentos:

```
function (file, col_names = TRUE, col_types = NULL, locale = default_locale(),  
na = c("", "NA"), quoted_na = TRUE, quote = " \", comment = "",  
trim_ws = TRUE, skip = 0, n_max = Inf, guess_max = min(1000, n_max), progress =
```

- O único argumento obrigatório é o caminho do arquivo.
- Os demais argumentos servem para um controle detalhado das opções de importação.
- As funções parsing são usadas para atribuir o tipo de valor apropriado durante a importação.



- Funções de importação do readr produzem tibbles.
- Comparado ao R básico, o readr:
 - 1 É consistente nos argumentos em diferentes funções.
 - 2 É mais rápido e mostra barra de progresso.
 - 3 Para exportação o readr possui funções de escrita em todos os formatos de leitura.
 - 4 Todas as write tem os mesmos argumentos.



PET-Estatística
UFPR

- Outro formato comum de armazenamento é nas planilhas excel, lembre-se que na primeira aula, em que utilizamos a função `read_excel()` do `readxl`.
- O **readxl** faz parte dos pacotes Tidyverse, portanto sua sintaxe parecida com os demais pacotes do universo.
- Demais formatos como bases SAS, SPSS e Stata podem ser importados para o R utilizando o pacote **haven**.



- 1 Tidyverse
- 2 Principais funcionalidades
- 3 Operador Pipe (`%>%`)
- 4 tibble
- 5 readr
- 6 Links úteis



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