

# Pacotes tidyverse

## Aula 02

Frederico Bertholini

O Universo tidyverse

# Manifesto tidyverse

O tidyverse, também chamado por muitos de hadleyverse, é um conjunto de pacotes que, por compartilharem esses princípios do manifesto tidy, podem ser utilizados naturalmente em conjunto. Pode-se dizer que existe o R antes do tidyverse e o R depois do tidyverse.

Os princípios fundamentais do tidyverse são:

- ▶ Reutilizar estruturas de dados existentes.
- ▶ Organizar funções simples usando o pipe.
- ▶ Aderir à programação funcional.
- ▶ Projetado para ser usado por seres humanos.

# Manifesto tidy

- ▶ Tidy Tools Manifesto <https://cran.r-project.org/web/packages/tidyverse/vignettes/manifesto.html>
- ▶ Tidy data vignette <https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html>
- ▶ Tidy Data paper <http://vita.had.co.nz/papers/tidy-data.pdf>
- ▶ Conjunto de pacotes <https://www.tidyverse.org/packages/>

## Usando o pipe - O operador %>%

O operador %>% (pipe) foi uma das grandes revoluções recentes do R, tornando a leitura de códigos mais lógica, fácil e compreensível.

```
library(tidyverse)
library(magrittr)
```

# Ideia

A ideia do operador `%>%` (pipe) é bem simples: usar o valor resultante da expressão do lado esquerdo como primeiro argumento da função do lado direito.

- ▶ As duas linhas abaixo são equivalentes.

```
f(x, y)
```

```
x %>% f(y)
```

## E se aumentarmos o código?

Vamos calcular a raiz quadrada da soma dos valores de 1 a 4.

Primeiro, sem o pipe.

```
sqrt(sum(x))
```

```
## [1] 3.162278
```

Agora com o pipe.

```
x %>%  
  sum %>%  
  sqrt
```

```
## [1] 3.162278
```

E se realmente tivermos muitas funções aninhadas?

A utilização do pipe transforma um código confuso e difícil de ser lido em algo *simples e intuitivo*.



## Receita de bolo - sem pipe

Tente entender o que é preciso fazer.

```
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")
```

## Receita de bolo - com pipe

Desistiu? Agora veja como fica escrevendo com o %>%:

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

## Exercícios pipe

## Exercício

1. Reescreva a expressão abaixo utilizando o `%>%`.

```
round(mean(divide_by(sum(1:10),3)),digits = 1)
```

# Resolução

## Exercício

```
2 %>%  
  add(2) %>%  
  c(6, NA) %>%  
  mean(na.rm = T) %>%  
  equals(5)
```

# Resolução

Importação no tidyverse



# Importação com readr, readxl, haven e DBI

No tidyverse, geralmente

- ▶ Funções `read_<formato>` servem para ler um arquivo no formato `<formato>`
- ▶ Funções `write_<formato>` servem para escrever num arquivo com o formato `<formato>`

## Arquivos de texto

- ▶ csv, tsv, txt, ...
- ▶ Para esses aqui, usar o pacote `readr`
- ▶ Você também pode experimentar o `data.table::fread`

## 'readr' para textos

Exemplo:

```
read_csv("data/import/mtcars.csv")
```

```
data.table::fread("data/import/mtcars.csv")
```

# Arquivos binários

- ▶ .RData, .rds, .feather, .fst
- ▶ .dta (Stata), .sas7bdat (SAS), .sav (SPSS)
- ▶ Ler com readr, haven, feather, fst.

Exemplo:

```
read_rds("data/import/mtcars.rds")
```

# Bancos de dados

- ▶ MySQL, SQL Server, PostgreSQL, SQLite, ...
- ▶ Spark, MongoDB, Hive, ...
- ▶ Utilizar pacotes DBI e odbc

Pacotes dplyr e tidyr

## Conjunto de dados

Vamos trabalhar com a base decisoes, que contém decisões do Tribunal de Justiça de São Paulo

```
decisoes <- read_rds("CADS2018/Exercícios/dados/decisoes.rds")
glimpse(decisoes)
```

```
## Observations: 11,731
## Variables: 9
## $ id_decisao      <chr> "11094999", "11093733", "11093677"
## $ n_processo      <chr> "0057003-20.2017.8.26.0000", "0057003-20.2017.8.26.0000", "0057003-20.2017.8.26.0000"
## $ classe_assunto   <chr> "Habeas Corpus / Homicídio Simples", "Habeas Corpus / Homicídio Simples", "Habeas Corpus / Homicídio Simples"
## $ municipio        <chr> "Cosmópolis", "São Paulo", "Ribeirão Preto"
## $ camara           <chr> "3ª Câmara de Direito Criminal", "3ª Câmara de Direito Criminal", "3ª Câmara de Direito Criminal"
## $ data_decisao     <chr> "19/12/2017", "19/12/2017", "19/12/2017"
## $ data_registro    <chr> "19/12/2017", "19/12/2017", "19/12/2017"
## $ juiz             <chr> "Luiz Antonio Cardoso", "Luiz Antonio Cardoso", "Luiz Antonio Cardoso"
## $ txt decisao       <chr> NA, NA, NA, "Execução Penal - Cível"
```

## Características do dplyr

- ▶ A utilização é facilitada com o emprego do operador `%>%`
- ▶ No primeiro argumento colocamos o `data.frame` ou o `tibble`, e nos outros argumentos colocamos o que queremos fazer.



# As cinco funções principais do dplyr

- ▶ `select`: selecionar colunas
- ▶ `filter`: filtrar linhas
- ▶ `mutate`: criar colunas
- ▶ `summarise`: sumarizar colunas
- ▶ `arrange`: ordenar linhas

select

`select`

- ▶ Utilizar `starts_with(x)`, `contains(x)`, `matches(x)`, `one_of(x)`, etc.
- ▶ Possível colocar nomes, índices, e intervalos de variáveis com `:`.

## Em ação

```
decisoes %>%
```

```
  select(id_decisao, n_processo, municipio, juiz)
```

```
## # A tibble: 11,731 x 4
```

```
##   id_decisao n_processo      municipio
```

```
##   <chr>      <chr>      <chr>
```

```
## 1 11094999    0057003-20.2017.8.26.0000 Cosmópolis
```

```
## 2 11093733    0052762-03.2017.8.26.0000 São Paulo
```

```
## 3 11093677    0055169-79.2017.8.26.0000 Ribeirão Preto
```

```
## 4 11093270    9000580-82.2017.8.26.0032 Araçatuba
```

```
## 5 11093374    0052938-79.2017.8.26.0000 São Paulo
```

```
## 6 11093320    9000723-79.2017.8.26.0482 Presidente Prudente
```

```
## 7 11091506    0003276-86.2015.8.26.0075 Bertioga
```

```
## 8 11093326    9000298-11.2017.8.26.0625 Taubaté
```

```
## 9 11092475    0004653-39.2015.8.26.0028 Aparecida
```

```
## 10 11093773    2221930-66.2017.8.26.0000 Jandira
```

```
## # ... with 11,721 more rows
```

## Em ação

```
decisoes %>%
```

```
  select(classe_assunto:id_decisao, juiz)
```

```
## # A tibble: 11,731 x 4
```

```
##   classe_assunto
```

```
##   <chr>
```

```
## 1 Habeas Corpus / Homicídio Simples
```

```
## 2 Habeas Corpus / Roubo
```

```
## 3 Habeas Corpus / DIREITO PENAL
```

```
## 4 Agravo de Execução Penal / Pena Privativa de Liberdade
```

```
## 5 Mandado de Segurança / Crimes do Sistema Nacional de
```

```
## 6 Agravo de Execução Penal / Pena Privativa de Liberdade
```

```
## 7 Apelação / Tráfico de Drogas e Condutas Afins
```

```
## 8 Agravo de Execução Penal / Livramento Condicional
```

```
## 9 Apelação / Tráfico de Drogas e Condutas Afins
```

```
## 10 Habeas Corpus / Furto Qualificado
```

```
## # ... with 11,721 more rows
```

## Em ação

```
decisoes %>%  
  select(id_decisao, starts_with('data_'))
```

```
## # A tibble: 11,731 x 3  
##   id_decisao data_decisao data_registro  
##   <chr>      <chr>      <chr>  
## 1 11094999 19/12/2017 19/12/2017  
## 2 11093733 19/12/2017 19/12/2017  
## 3 11093677 19/12/2017 19/12/2017  
## 4 11093270 14/12/2017 19/12/2017  
## 5 11093374 14/12/2017 19/12/2017  
## 6 11093320 14/12/2017 19/12/2017  
## 7 11091506 14/12/2017 19/12/2017  
## 8 11093326 14/12/2017 19/12/2017  
## 9 11092475 14/12/2017 19/12/2017  
## 10 11093773 19/12/2017 19/12/2017  
## # ... with 11,721 more rows
```

## Exercício

- ▶ selecione as colunas que acabam com “cisao”.

## Resolução

```
decisoos %>%  
  select(ends_with("cisao"))
```

```
## # A tibble: 11,731 x 3
```

```
##   id_decisao data_decisao txt_decisao
```

```
##   <chr>         <chr>         <chr>
```

```
## 1 11094999    19/12/2017    <NA>
```

```
## 2 11093733    19/12/2017    <NA>
```

```
## 3 11093677    19/12/2017    <NA>
```

```
## 4 11093270    14/12/2017    "Execução Penal - Comutação
```

```
## 5 11093374    14/12/2017    "Mandado de segurança - Impet
```

```
## 6 11093320    14/12/2017    "Execução Penal - Apuração de
```

```
## 7 11091506    14/12/2017    "Tráfico de entorpecentes - A
```

```
## 8 11093326    14/12/2017    "Execução Penal - Pedido de I
```

```
## 9 11092475    14/12/2017    "Tráfico de entorpecentes -
```

```
## 10 11093773   19/12/2017    <NA>
```

```
## # ... with 11,721 more rows
```



## Exercício

- ▶ tire as colunas de texto = `'txt_decisao'` e classe/assunto = `'classe_assunto'`.
  - ▶ Dica: veja os exemplos de `?select` em `Drop variables ...`

## Resolução

```
decisoos %>%
```

```
  select(-classe_assunto, -txt_decisao)
```

```
## # A tibble: 11,731 x 7
```

```
##      id_decisao n_processo      municipio
```

```
##      <chr>      <chr>      <chr>
```

```
##    1 11094999    0057003-20.2017.8.26.0000 Cosmópolis
```

```
##    2 11093733    0052762-03.2017.8.26.0000 São Paulo
```

```
##    3 11093677    0055169-79.2017.8.26.0000 Ribeirão Preto
```

```
##    4 11093270    9000580-82.2017.8.26.0032 Araçatuba
```

```
##    5 11093374    0052938-79.2017.8.26.0000 São Paulo
```

```
##    6 11093320    9000723-79.2017.8.26.0482 Presidente Prudente
```

```
##    7 11091506    0003276-86.2015.8.26.0075 Bertiooga
```

```
##    8 11093326    9000298-11.2017.8.26.0625 Taubaté
```

```
##    9 11092475    0004653-39.2015.8.26.0028 Aparecida
```

```
##   10 11093773    2221930-66.2017.8.26.0000 Jandira
```

```
## # ... with 11,721 more rows
```

filter

## filter

- ▶ Use , ou & para “e” e | para “ou”.
- ▶ Condições separadas por vírgulas é o mesmo que separar por &.

## filter em ação

```
decisoes %>%  
  select(n_processo, id_decisao, municipio, juiz) %>%  
  filter(municipio == 'São Paulo')
```

```
## # A tibble: 2,446 x 4
```

##	n_processo	id_decisao	municipio	juiz
##	<chr>	<chr>	<chr>	<chr>
## 1	0052762-03.2017.8.26.0000	11093733	São Paulo	Luiz A
## 2	0052938-79.2017.8.26.0000	11093374	São Paulo	Grass
## 3	2214049-38.2017.8.26.0000	11093604	São Paulo	Luiz A
## 4	2227499-48.2017.8.26.0000	11093642	São Paulo	Luiz A
## 5	9002384-31.2017.8.26.0050	11093376	São Paulo	Grass
## 6	0021158-39.2015.8.26.0050	11091508	São Paulo	Grass
## 7	7005375-26.2015.8.26.0198	11091668	São Paulo	Grass
## 8	9002039-65.2017.8.26.0050	11094451	São Paulo	Grass
## 9	2203993-43.2017.8.26.0000	11094449	São Paulo	Grass
## 10	0099423-21.2016.8.26.0050	11091474	São Paulo	Grass

```
## # ... with 2,436 more rows
```

Dica: usar %in%

```
library(lubridate) # para trabalhar com as datas  
#`day(dmy(data_decisao))` pega o dia da decisão.
```

```
decisoes %>%  
  select(id_decisao, municipio, data_decisao, juiz) %>%  
  # municipio igual a campinas ou jaú, OU dia da decisão m  
  filter(municipio %in% c('Campinas', 'Jaú') | day(dmy(data
```

```
## # A tibble: 3,352 x 4
```

```
##   id_decisao municipio data_decisao juiz  
##   <chr>         <chr>      <chr>      <chr>  
## 1 11093272    Campinas  14/12/2017  Grassi Neto  
## 2 11093359    Campinas  07/12/2017  Grassi Neto  
## 3 11088333    Campinas  14/12/2017  Grassi Neto  
## 4 11093018     Jaú       28/11/2017  Ivan Sartori  
## 5 11089105     Jaú       14/12/2017  Ricardo Tucunduva  
## 6 11089111    Campinas  14/12/2017  Ricardo Tucunduva  
## 7 11091386    Santos    27/11/2017  Ivo de Almeida
```

## Mais ação

```
decisoos %>%  
  select(juiz) %>%  
  # filtra juizes que têm `Z` ou `z` no nome  
  filter(str_detect(juiz, regex("z", ignore_case = TRUE)))  
  # conta e ordena os juizes em ordem decrescente  
  count(juiz, sort = TRUE) %>%  
  head(5)
```

```
## # A tibble: 5 x 2  
##   juiz          n  
##   <chr>      <int>  
## 1 Gilberto Ferreira da Cruz    237  
## 2 Diniz Fernando             198  
## 3 Sérgio Mazina Martins       173  
## 4 Luiz Antonio Cardoso        163  
## 5 Rachid Vaz de Almeida       150
```

## Obs

A função `str_detect()` retorna `TRUE` se um elemento do vetor de textos é compatível com uma *expressão regular*. Estudaremos o pacote `stringr` e as funções `str_*` em outra aula.



## Exercício

- ▶ filtre apenas casos em que `id_decisao` não é NA

# Resolução

```
decisoos %>%  
  filter(is.na(id_decisao))
```

```
## # A tibble: 65 x 9
```

```
##   id_decisao n_processo classe_assunto municipio camara
```

```
##   <chr>      <chr>      <chr>      <chr>      <chr>
```

```
## 1 <NA>      <NA>      <NA>      <NA>      <NA>
```

```
## 2 <NA>      <NA>      <NA>      <NA>      <NA>
```

```
## 3 <NA>      <NA>      <NA>      <NA>      <NA>
```

```
## 4 <NA>      <NA>      <NA>      <NA>      <NA>
```

```
## 5 <NA>      <NA>      <NA>      <NA>      <NA>
```

```
## 6 <NA>      <NA>      <NA>      <NA>      <NA>
```

```
## 7 <NA>      <NA>      <NA>      <NA>      <NA>
```

```
## 8 <NA>      <NA>      <NA>      <NA>      <NA>
```

```
## 9 <NA>      <NA>      <NA>      <NA>      <NA>
```

```
## 10 <NA>     <NA>      <NA>      <NA>      <NA>
```

```
## # ... with 55 more rows
```

## Exercício

- ▶ filtre todas as decisões de 2018.
- Dica: função `lubridate::year()`

## Resolução

```
decisoos %>%  
  filter(year(dmy(data_decisao)) == 2018)
```

```
## # A tibble: 314 x 9
```

```
##   id_decisao n_processo      classe_assunto  
##   <chr>      <chr>      <chr>  
## 1 11107242    0009617-63.2016.8.26.0635  Apelação / Roubo  
## 2 11107425    2227593-93.2017.8.26.0000  Habeas Corpus /  
## 3 11107492    0076977-24.2016.8.26.0050  Embargos de Dec  
## 4 11107361    0012191-36.2017.8.26.0502  Agravo de Execu  
## 5 11107383    2218460-27.2017.8.26.0000  Habeas Corpus /  
## 6 11107331    0006928-63.2017.8.26.0521  Agravo de Execu  
## 7 11107651    0000297-54.2017.8.26.0311  Apelação / Tráf  
## 8 11107485    2225548-19.2017.8.26.0000  Habeas Corpus /  
## 9 11107335    0006934-70.2017.8.26.0521  Agravo de Execu  
## 10 11107340   0006682-67.2017.8.26.0521  Agravo de Execu  
## # ... with 304 more rows
```

mutate

## mutate

- ▶ Aceita várias novas colunas iterativamente.
- ▶ Novas variáveis devem ter o mesmo `length` que o `nrow` do `bd` original ou 1.

## mutate em ação

```
decisoes %>%  
  select(n_processo, data_decisao, data_registro) %>%  
  mutate(tempo = dmy(data_registro) - dmy(data_decisao))
```

```
## # A tibble: 11,731 x 4
```

##	n_processo	data_decisao	data_registro
##	<chr>	<chr>	<chr>
## 1	0057003-20.2017.8.26.0000	19/12/2017	19/12/2017
## 2	0052762-03.2017.8.26.0000	19/12/2017	19/12/2017
## 3	0055169-79.2017.8.26.0000	19/12/2017	19/12/2017
## 4	9000580-82.2017.8.26.0032	14/12/2017	19/12/2017
## 5	0052938-79.2017.8.26.0000	14/12/2017	19/12/2017
## 6	9000723-79.2017.8.26.0482	14/12/2017	19/12/2017
## 7	0003276-86.2015.8.26.0075	14/12/2017	19/12/2017
## 8	9000298-11.2017.8.26.0625	14/12/2017	19/12/2017
## 9	0004653-39.2015.8.26.0028	14/12/2017	19/12/2017
## 10	2221930-66.2017.8.26.0000	19/12/2017	19/12/2017
## #	... with 11,721 more rows		

## Exercício

- ▶ Crie uma coluna binária `drogas` que vale `TRUE` se no texto da decisão algo é falado de drogas e `FALSE` caso contrário. – Dica: `str_detect`

Obs.: Considere tanto a palavra 'droga' como seus sinônimos, ou algum exemplo de droga e retire os casos em que `txt_decisao` é vazio



## Resolução

```
decisoes %>%  
  filter(!is.na(txt_decisao)) %>%  
  mutate(txt_decisao = tolower(txt_decisao),  
         droga = str_detect(txt_decisao,  
                             "droga|entorpecente|psicotr[óo]pico|maconha|haxixe|coca  
dplyr::select(n_processo, droga)
```

```
## # A tibble: 6,933 x 2
```

##	n_processo	droga
##	<chr>	<lgl>
## 1	9000580-82.2017.8.26.0032	FALSE
## 2	0052938-79.2017.8.26.0000	FALSE
## 3	9000723-79.2017.8.26.0482	FALSE
## 4	0003276-86.2015.8.26.0075	TRUE
## 5	9000298-11.2017.8.26.0625	TRUE
## 6	0004653-39.2015.8.26.0028	TRUE
## 7	9000788-34.2017.8.26.0269	FALSE
## 8	9000673-53.2017.8.26.0482	FALSE

summarise

## summarise

- ▶ Retorna um vetor de tamanho 1 a partir de uma operação com as variáveis (aplicação de uma função).
- ▶ Geralmente é utilizado em conjunto com `group_by()`.
- ▶ Algumas funções importantes: `n()`, `n_distinct()`.

## Em ação

```
decisoes %>%
  select(n_processo, municipio, data_decisao) %>%
  #       pega ano da decisão
  mutate(ano_julgamento = year(dmy(data_decisao)),
         # pega o ano do processo 0057003-20.2017.8.26.000
         ano_proc = str_sub(n_processo, 12, 15),
         # transforma o ano em inteiro
         ano_proc = as.numeric(ano_proc),
         # calcula o tempo em anos
         tempo_anos = ano_julgamento - ano_proc) %>%
  group_by(municipio) %>%
  summarise(n = n(),
            media_anos = mean(tempo_anos),
            min_anos = min(tempo_anos),
            max_anos = max(tempo_anos))
```

## Resultado

```
## # A tibble: 315 x 5
```

##	municipio	n	media_anos	min_anos	max_anos
##	<chr>	<int>	<dbl>	<dbl>	<dbl>
##	1 Adamantina	17	0.765	0	
##	2 Aguaí	19	1.16	0	
##	3 Águas de Lindóia	5	1.4	0	
##	4 Agudos	8	3.25	0	
##	5 Altinópolis	7	0.857	0	
##	6 Americana	56	1.41	0	
##	7 Américo Brasiliense	9	1.56	0	
##	8 Amparo	9	2.11	0	
##	9 Andradina	41	0.707	0	
##	10 Angatuba	4	0.5	0	
##	... with 305 more rows				

## usando count()

A função `count()`, simplifica um `group_by %>% summarise %>% ungroup`:

```
decisoes %>%  
  count(juiz, sort = TRUE) %>%  
  mutate(prop = n / sum(n),  
         prop = scales::percent(prop))
```

```
## # A tibble: 100 x 3  
##   juiz                n prop  
##   <chr>             <int> <chr>  
## 1 Gilberto Ferreira da Cruz    237 2.02%  
## 2 Francisco Orlando          226 1.93%  
## 3 Diniz Fernando             198 1.69%  
## 4 Walter da Silva            183 1.56%  
## 5 De Paula Santos            182 1.55%  
## 6 Machado de Andrade         182 1.55%  
## 7 Newton Neves              180 1.53%
```

## + fácil ainda

mas sem formato %

```
decisoes %>%  
  count(juiz, sort = TRUE) %>%  
  mutate(prop = prop.table(n))
```

```
## # A tibble: 100 x 3
```

##	juiz	n	prop
##	<chr>	<int>	<dbl>
##	1 Gilberto Ferreira da Cruz	237	0.0202
##	2 Francisco Orlando	226	0.0193
##	3 Diniz Fernando	198	0.0169
##	4 Walter da Silva	183	0.0156
##	5 De Paula Santos	182	0.0155
##	6 Machado de Andrade	182	0.0155
##	7 Newton Neves	180	0.0153
##	8 Leme Garcia	179	0.0153
##	9 Grassi Neto	177	0.0151

arrange



## arrange

- ▶ Simplesmente ordena de acordo com as opções.
- ▶ Utilizar `desc()` para ordem decrescente ou o sinal de menos (-).

## Exercício

- ▶ Quem são os cinco relatores mais prolixos?
- Dica: use `str_length()` – Lembre-se da função `head()`

## Resolução

```
decisoes %>%  
  filter(!is.na(txt_decisao)) %>%  
  mutate(tamanho = str_length(txt_decisao)) %>%  
  group_by(juiz) %>%  
  summarise(n = n(),  
            tamanho_mediana = median(tamanho)) %>%  
  filter(n >= 10) %>%  
  arrange(desc(tamanho_mediana)) %>%  
  head()
```

```
## # A tibble: 6 x 3
```

##	juiz	n	tamanho_mediana
##	<chr>	<int>	<dbl>
## 1	Airton Vieira	154	3146.
## 2	Ely Amioka	81	1847
## 3	Grassi Neto	141	1675
## 4	Alcides Malossi Junior	95	1541
## 5	Cesar Augusto Andrade de Castro	77	1341