Estruturas de dados e manipulação avançada

Métodos Quantitativos Aplicados à Ciência Política

Frederico Bertholini

28.set.2020

7 select

8 filter

9 mutate

10 summarise

11 arrange

12 tydyr

13 Joins

14 Limpeza

data.frame

data.frame

- Um data.frame é o mesmo que uma tabela do SQL ou uma planilha Excel
- seus dados provavelmente serão importados para um objeto data.frame
- data.frame's são listas especiais em que todos os elementos possuem o mesmo comprimento.
- Cada elemento dessa lista pode ser pensado como uma coluna da tabela - ou como uma variável. Uso do '$'
- Seu comprimento representa o número de linhas - ou seja, de observações

data.frame

- Como data.frames's são listas, suas colunas podem ser de classes diferentes. Essa é a grande diferença entre data.frame's e matrizes.

Funções úteis:

```
head() # Mostra as primeiras 6 linhas.
tail() # Mostra as últimas 6 linhas.
dim() # Número de linhas e de colunas.
names() # Os nomes das colunas (variáveis).
str() # Estrutura do data.frame. Mostra, entre outras cois
cbind() # Acopla duas tabelas lado a lado.
rbind() # Empilha duas tabelas.
```

Pacotes

O que são pacotes

- O R possui diversas funções já instaladas dentro da sua programação
- Exemplos são sum(), length(), class(), c()
- Outras, porém, devem ser instaladas para que possam ser utilizadas pelos usuários
- A maneira com a qual instalamos novas funções, não definidas anteriormente no software, é através de pacotes
- Pacotes concentram diversas funções para diversas demandas
  Importação de dados;
  Organização de banco de dados;
  Análises estatísticas específicas;
  Gráficos diferenciados;

O que são pacotes

- A instalação de qualquer pacote pode ser feita por dentro do R
- Para isso, porém, é preciso primeiro conexão com a internet, já que o R busca o novo pacote no repositório de pacotes
- A função para instalar pacote, portanto, é install.packages()
- O nome da nova função deve vir, primeiramente entre parenteses
- Podemos começar instalando o pacote para importação de bases de dados: foreign

```
install.packages("foreign")
```

- Após alguns segundos, e algumas mensagens no console, a instalação será efetivada

Ativar pacotes

- Cada pacote, inclusive o foreign, tem uma documentação disponível na internet
- Nessa documentação estão disponíveis as funções que o pacote possui, além do nome do seu criador
- As função não ficam disponíveis assim que o pacote termina a instalação
- Para ativar as funções do pacote, é preciso utilizar a função library()

```
library(foreign)
```

- Repare, que uma vez instalado, o nome do pacote não precisa mais estar entre aspas

Ativar pacotes

- Portanto, para começarmos o procedimento de instalação do pacote, seguimos os passos a seguir:
  Caso não esteja instalado, instalar o pacote através da função install.packages();
  Para ativar o pacote, utilizar a função library() sem as aspas no nome do pacote instalado

```
install.packages("foreign")

library(foreign)
```

- Uma vez instalado o pacote, não é preciso instalar mais a não ser que você reinstale o R

Importação de dados

Passo a passo

- A importação é uma das tarefas que demandam mais atenção no R
- É preciso ter um conhecimento prévio de como sua base externa está constituída
- Outra informação importante é a extensão do arquivo da base
- Primeiramente, a informação que deve ser dada ao software é onde está a base - diretório de trabalho
- A função necessária é setwd() que define o diretório da sua seção no R
- Dentro da função, iremos inserir o local do arquivo
  Em caso de Windows, inverta as barras ou duplique;
  Não se esqueça das aspas;

```
setwd("/Volumes/Macintosh HD/MQCP_IPOL_2020/Slides/aula 03
```

Passo a passo

- Esse diretório definido significa que as bases e os gráficos produzidos serão enviados para essa pasta
- Finalmente, vamos importar as bases de dados
- Primeiro, vamos importar a base de extensão txt com o nome baserm
- Não é preciso de pacote para esse procedimento

```
lines <- readLines("dados/baserm.txt")

baserm <- read.table(text = lines, sep = '\t')
```

- Repare que definimos a base dentro das aspas e com a extensão
- Na segunda linha, o primeiro argumento é o texto, o segundo argumento trata de como os dados estão separados, geralmente txt vem separado assim

Passo a passo

- No pacote foreign, a forma mais genérica de importação da base é o read.table()
- Entretanto, o pacote apresenta uma série de especialidades, a depender da extensão em questão
- Para CSV, vimos que tem a read.csv(). Já para dta, base de origem do stata, temos a função read.dta()
- O pacote foreign não possui a extensão xlsx e xls, extensão muito encontrada e comum entre as bases de dados disponíveis
- Para isso, vamos instalar um novo pacote readxl

```
install.packages("readxl")
```

- Esse pacote disponibiliza as funções read_xls() e read_xlsx()

Passo a passo

- Vamos ativar as funções disponíveis no pacote readxl com a função library()

```
library(readxl)
```

- Vamos importar a base controle_cgu_municípios.xlsx

```
cgu <-
read_xlsx("dados/controle_cgu_municípios.xlsx")
```

- Repare que acessamos apenas a primeira página da base
- Para acessarmos a segunda páginas, utilizamos o argumento sheet=2

```
cgu <-
read_xlsx("dados/controle_cgu_municípios.xlsx",
sheet = 2)
```

- Alguns sinais de alerta surgem, porém não se trata de erro

Importação por pacote

- Por fim, podemos importar dados através de pacotes
- Após ativar o pacote "ElectionsBR", a função legend_fed() fica disponível para a importação
- Como toda a função, ou quase todas, possui argumentos importantes:
  - 'year =' se refere ao ano de extração
  - 'uf =' se refere à UF

Importação

- Vamos começar importando dados de coalizões pré-eleitorais (coligações) do DF em 2018, nos retornando um objeto em 'tbl_df' e data frame

```
library(electionsBR)
```

```
##
## To cite electionsBR in publications, use: citation('ele

## To learn more, visit: http://electionsbr.com
```

```
leg_df_2018 <- legend_fed(year = 2018,uf="DF")
```

```
## Processing the data...

## Warning: `as.tbl()` is deprecated as of dplyr 1.0.0.
## Please use `tibble::as_tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this war
```

Visualizando a base

- Primeira coisa importante de se informar é a classe desses objetos
- Temos 4 objetos: baserm, cgu, educacao e pnad2018

```
class(baserm)
```

```
## [1] "data.frame"
```

```
class(cgu)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
class(leg_df_2018)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

Visualizando a base

- Uma visão completa da base é o comando View()
- Entretanto, cuidado, dependendo do tamanho da base, podemos travar o software

```
View(baserm)
```

- Repare no V maiúsculo, lembre-se que o R é bastante sensível na sua linguagem
- O View() abre uma nova aba com a base no formato de grade
- Podemos, assim, visualizar a base de dados na forma mais intuitiva

Visualizando a base

- Porém, para bases como a pnad2018, por exemplo, sabemos que é grande demais para sua visualização ser feita através do View()
- Algumas funções podem nos ajudar nessa tarefa
- A primeira é o dim, que as dimensões da base

```
dim(leg_df_2018)
```

```
## [1] 251  23
```

- O primeiro valor sempre retrata o número de linhas, ou observações, enquanto o segundo valor apresenta o número de colunas, ou variáveis
- A função ncol() e length() também indicam quantas colunas, ou variáveis estão presentes na base

Visualizando a base

- Outra função importante na visualização de bases de dados é a lista de nomes
- A função names() descreve as variáveis presentes na base
- Isso facilita no momento de selecionar as variáveis que entrarão na análise de vocês

```
names(cgu)
```

```
##  [1] "COD.IBGE7"                "REGIÃO"
##  [3] "UF"                       "PORTE"
##  [5] "MUNICÍPIO"                "falha"
##  [7] "tempo_falha_02"           "tempo_falha_01"
##  [9] "reicidência_falha"        "ano_eleitoral"
## [11] "PERCENT_ganhador_2000"    "PERCENT_ganhador_2004"
## [13] "PERCENT_ganhador_2008"    "PERCENT_ganhador_2012"
## [15] "PERCENT_ganhador_médio"   "competição_pol_alta"
## [17] "Ideologia1"               "Ideologia2"
## [19] "Ideologia3"               "ideologia 4"
## [21] "ideologia_media"          "IDHM_2000"
```

Visualizando a base

- Outra função possível é o str()
- Essa função apresenta o nome das variáveis, a classe de cada uma delas e os primeiros valores

```
str(leg_df_2018)
```

```
## tibble [251 x 23] (S3: tbl_df/tbl/data.frame)
##  $ DATA_GERACAO        : chr [1:251] "23/09/2020" "23/0
##  $ HORA_GERACAO        : 'hms' num [1:251] 17:31:43 17:
##   ..- attr(*, "units")= chr "secs"
##  $ ANO_ELEICAO         : num [1:251] 2018 2018 2018 201
##  $ COD_TIPO_ELEICAO    : num [1:251] 2 2 2 2 2 2 2 2 2
##  $ NM_TIPO_ELEICAO     : chr [1:251] "ELEIÇÃO ORDINÁRIA
##  $ NUM_TURNO           : num [1:251] 1 1 1 1 1 1 1 1 1
##  $ COD_ELEICAO         : num [1:251] 297 297 297 297 29
##  $ DESCRICAO_ELEICAO   : chr [1:251] "Eleições Gerais E
##  $ DATA_ELEICAO        : chr [1:251] "07/10/2018" "07/1
##  $ SIGLA_UF            : chr [1:251] "DF" "DF" "DF" "DF
##  $ SIGLA_UE            : chr [1:251] "DF" "DF" "DF" "DF
```

Visualizando a base

- Finalmente, a função head() e tail()
- A primeira função apresenta os primeiros valores de uma base de dados

```
head(baserm,2)
```

```
##   sigla cod.ibge  estado anoeleitoral1990 anoeleitoral1
## 1    AC       12    Acre                1
## 2    AL       27 Alagoas                1
##   anoeleitoral1993 anoeleitoral1994 anoeleitoral1995 an
## 1                0                1                0
## 2                0                1                0
##   anoeleitoral1997 anoeleitoral1998 anoeleitoral1999 an
## 1                0                1                0
## 2                0                1                0
##   anoeleitoral2001 anoeleitoral2002 anoeleitoral2003 an
## 1                0                1                0
## 2                0                1                0
##   anoeleitoral2005 anoeleitoral2006 anoeleitoral2007 an
```

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")
```

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`

Arquivos binários

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

Bancos de dados

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

Tidy data e janitor

Veremos mais à frente, mas janitor::clean_names() é uma ferramenta tidy

```
library(janitor)
```

```
##
## Attaching package: 'janitor'

## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

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("dados/decisoes.rds")
glimpse(decisoes)
```

```
## Rows: 11,731
## Columns: 9
## $ `ID Decisão`     <chr> "11094999", "11093733", "11093
## $ n_processo       <chr> "0057003-20.2017.8.26.0000", "
## $ `Classe/Assunto` <chr> "Habeas Corpus / Homicídio Sim
## $ Município        <chr> "Cosmópolis", "São Paulo", "Ri
## $ Câmara           <chr> "3ª Câmara de Direito Criminal
## $ `Data decisão`   <chr> "19/12/2017", "19/12/2017", "1
## $ `Data registro`  <chr> "19/12/2017", "19/12/2017", "1
## $ Juiz             <chr> "Luiz Antonio Cardoso", "Luiz
## $ `txt decisão`    <chr> NA, NA, NA, "Execução Penal -
```

```
decisoes <- read_rds("dados/decisoes.rds") %>%
  janitor::clean_names() # com dois pontos eu não preciso
```

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 Prud
##  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                          n_processo
##    <chr>                                   <chr>
##  1 Habeas Corpus / Homicídio Simples       0057003-20.2017.
##  2 Habeas Corpus / Roubo                   0052762-03.2017.
##  3 Habeas Corpus / DIREITO PENAL           0055169-79.2017.
##  4 Agravo de Execução Penal / Pena Pr~     9000580-82.2017.
##  5 Mandado de Segurança / Crimes do S~     0052938-79.2017.
##  6 Agravo de Execução Penal / Pena Pr~     9000723-79.2017.
##  7 Apelação / Tráfico de Drogas e Con~     0003276-86.2015.
##  8 Agravo de Execução Penal / Livrame~     9000298-11.2017.
##  9 Apelação / Tráfico de Drogas e Con~     0004653-39.2015.
## 10 Habeas Corpus / Furto Qualificado       2221930-66.2017.
## # ... 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
```

Operações

- selecione as colunas que acabam com "cisao".

```
decisoes %>%
  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 - Impe
##  6 11093320   14/12/2017   "Execução Penal - Apuração d
##  7 11091506   14/12/2017   "Tráfico de entorpecentes -
##  8 11093326   14/12/2017   "Execução Penal - Pedido de
##  9 11092475   14/12/2017   "Tráfico de entorpecentes -
## 10 11093773   19/12/2017    <NA>
## # ... with 11,721 more rows
```

Operações

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

```
decisoes %>%
  select(-classe_assunto, -txt_decisao)
```

```
## # A tibble: 11,731 x 7
##    id_decisao n_processo   municipio  camara   data_dec
##    <chr>      <chr>        <chr>      <chr>    <chr>
##  1 11094999   0057003-20.~ Cosmópolis 3ª Câma~ 19/12/20
##  2 11093733   0052762-03.~ São Paulo  3ª Câma~ 19/12/20
##  3 11093677   0055169-79.~ Ribeirão ~ 3ª Câma~ 19/12/20
##  4 11093270   9000580-82.~ Araçatuba  8ª Câma~ 14/12/20
##  5 11093374   0052938-79.~ São Paulo  8ª Câma~ 14/12/20
##  6 11093320   9000723-79.~ President~ 8ª Câma~ 14/12/20
##  7 11091506   0003276-86.~ Bertioga   8ª Câma~ 14/12/20
##  8 11093326   9000298-11.~ Taubaté    8ª Câma~ 14/12/20
##  9 11092475   0004653-39.~ Aparecida  8ª Câma~ 14/12/20
```

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
##  2 0052938-79.2017.8.26.0000 11093374   São Paulo Grass
##  3 2214049-38.2017.8.26.0000 11093604   São Paulo Luiz
##  4 2227499-48.2017.8.26.0000 11093642   São Paulo Luiz
##  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(dat
```

```
## # 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
##  8 11091385   Araçatuba 27/11/2017   Ivo de Almeida
```

Mais ação


```
decisoes %>%
  select(juiz) %>%
  # filtra juízes 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.

- filtre apenas casos em que id_decisao não é NA

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

```
## # A tibble: 65 x 9
##    id_decisao n_processo classe_assunto municipio camar
##    <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, and 3 more variables: data_reg
```

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

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

```
## # A tibble: 314 x 9
##    id_decisao n_processo classe_assunto municipio camar
##    <chr>      <chr>      <chr>          <chr>     <chr>
##  1 11107242   0009617-6~ Apelação / Ro~ São Paulo 2ª Câ
##  2 11107425   2227593-9~ Habeas Corpus~ Iepê      2ª Câ
##  3 11107492   0076977-2~ Embargos de D~ São Paulo 2ª Câ
##  4 11107361   0012191-3~ Agravo de Exe~ Campinas  2ª Câ
##  5 11107383   2218460-2~ Habeas Corpus~ Sorocaba  2ª Câ
##  6 11107331   0006928-6~ Agravo de Exe~ Sorocaba  2ª Câ
##  7 11107651   0000297-5~ Apelação / Tr~ Junqueir~ 2ª Câ
##  8 11107485   2225548-1~ Habeas Corpus~ Nazaré P~ 2ª Câ
##  9 11107335   0006934-7~ Agravo de Exe~ Sorocaba  2ª Câ
## 10 11107340   0006682-6~ Agravo de Exe~ Sorocaba  2ª Câ
```

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
```

- 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

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

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


```
## `summarise()` ungrouping output (override with `.groups

## # A tibble: 315 x 5
##    municipio              n media_anos min_anos max_an
##    <chr>              <int>      <dbl>    <dbl>    <db
##  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 2.0203%
##  2 Francisco Orlando           226 1.9265%
##  3 Diniz Fernando              198 1.6878%
##  4 Walter da Silva             183 1.5600%
##  5 De Paula Santos             182 1.5514%
##  6 Machado de Andrade          182 1.5514%
##  7 Newton Neves                180 1.5344%
##  8 Leme Garcia                 179 1.5259%
```

+ 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
## 10 Figueiredo Gonçalves        176 0.0150
```

arrange

arrange

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

- Quem são os cinco relatores mais prolixos?

– Dica: use `str_length()` – Lembre-se da função `head()`

```
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()
```

```
## `summarise()` ungrouping output (override with `.groups
```

```
## # 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
```

tydyr

Alterando o formato de dados

Até agora, estudamos os principais ferramentas de transformação de dados do `dplyr`. Agora vamos aumentar nossa caixa-de-ferramentas com `tidyr`

- Carregando uma nova base de dados, que completa a de decisões.

```
processos <- read_rds("dados/processos_nested.rds")
```

Fomato tidy

- Hadley Wickham http://r4ds.had.co.nz/tidy-data.html

Funções do pacote

- Enquanto o `dplyr` faz recortes na base (com `filter()`e `select()`) e adições simples (`mutate()`, `summarise()`), o `tidyr` mexe no **formato** da tabela (`gather()`, `spread()`) e faz modificações menos triviais.
- As funções do `tidyr` geralmente vêm em pares com seus inversos:
  - `gather()` e `spread()`, -> substituídas por `pivot_longer` e `pivot_wider`
  - `nest()` e `unnest()`,
  - `separate()` e `unite()`

Onde estamos

http://r4ds.had.co.nz/wrangle-intro.html

gather()

- gather() empilha o banco de dados
- pivot_longer empilha de um jeito ainda mais fácil

```
decisoes %>%
  filter(!is.na(id_decisao)) %>%
  select(id_decisao:data_registro) %>%
  # 1. nome da coluna que vai guardar os nomes de colunas
  # 2. nome da coluna que vai guardar os valores das colun
  # 3. seleção das colunas a serem empilhadas
  gather(key="variavel", value="valor", -id_decisao) %>%
  arrange(id_decisao)
```

```
## # A tibble: 69,996 x 3
##    id_decisao variavel       valor
##    <chr>      <chr>          <chr>
##  1 11026431   n_processo     0000009-51.2015.8.26.0546
##  2 11026431   classe_assunto Apelação / Tráfico de Drog
##  3 11026431   municipio      Itapira
```

pivot_longer

Base relig_income do tidyr 3 variáveis:

- religion, nas linhas
- income, nas colunas e
- count, nas células

```
tidyr::relig_income
```

```
## # A tibble: 18 x 11
##    religion `<$10k` `$10-20k` `$20-30k` `$30-40k` `$40-
##    <chr>      <dbl>     <dbl>     <dbl>     <dbl>     <
##  1 Agnostic      27        34        60        81
##  2 Atheist       12        27        37        52
##  3 Buddhist      27        21        30        34
##  4 Catholic     418       617       732       670
##  5 Don't k~      15        14        15        11
##  6 Evangel~     575       869      1064       982
##  7 Hindu          1         9         7         9
##  8 Histori~     228       244       236       238
```

```
tidyr::relig_income %>%
  pivot_longer(!religion,
               names_to = "income", # diz a varável onde e
               values_to = "count" # diz a variável onde e
               )
```

```
## # A tibble: 180 x 3
##    religion income             count
##    <chr>    <chr>              <dbl>
##  1 Agnostic <$10k                 27
##  2 Agnostic $10-20k               34
##  3 Agnostic $20-30k               60
##  4 Agnostic $30-40k               81
##  5 Agnostic $40-50k               76
##  6 Agnostic $50-75k              137
##  7 Agnostic $75-100k             122
##  8 Agnostic $100-150k            109
##  9 Agnostic >150k                 84
## 10 Agnostic Don't know/refused    96
```

spread()

- spread() espalha uma variável nas colunas e preenche com outra variável
- Função inversa de gather
- Bem mais fácil com pivot_wider

```
decisoes %>%
  filter(!is.na(id_decisao)) %>%
  select(id_decisao:data_registro) %>%
  gather(key, value, -id_decisao) %>%
  # 1. coluna a ser espalhada
  # 2. valores da coluna
  spread(key, value)
```

```
## # A tibble: 11,666 x 7
##    id_decisao camara classe_assunto data_decisao data_r
##    <chr>      <chr>  <chr>          <chr>         <chr>
##  1 11026431   5ª Câ~ Apelação / Tr~ 30/11/2017   01/12/
##  2 11026432   5ª Câ~ Apelação / Fu~ 30/11/2017   01/12/
```

pivot_wider


```
decisoes %>%
  filter(!is.na(id_decisao)) %>%
  select(id_decisao:data_registro) %>%
  pivot_longer(!id_decisao,
               names_to = "tipo", # diz a varável onde ent
               values_to = "info" # diz a variável onde en
               ) %>%
  # 1. coluna a ser espalhada
  # 2. valores da coluna
  pivot_wider(names_from = "tipo",
              values_from = "info")
```

```
## # A tibble: 11,666 x 7
##    id_decisao n_processo classe_assunto municipio camar
##    <chr>      <chr>      <chr>          <chr>     <chr>
##  1 11094999   0057003-2~ Habeas Corpus~ Cosmópol~ 3ª Câ
##  2 11093733   0052762-0~ Habeas Corpus~ São Paulo 3ª Câ
##  3 11093677   0055169-7~ Habeas Corpus~ Ribeirão~ 3ª Câ
##  4 11093270   9000580-8~ Agravo de Exe~ Araçatuba 8ª Câ
```

- Qual juiz julga a maior proporção de processos que tratam de drogas

– Dica: construa um `data.frame` contendo as colunas juiz, n_processos_drogas, n_processos_n_drogas e total_processos, remodelando os dados para haver um juiz por linha e utilizando `spread()`

Resolução


```
## `summarise()` regrouping output by 'juiz' (override wit

## # A tibble: 65 x 5
## # Groups:   juiz [65]
##    juiz                  droga n_droga total proporcao
##    <chr>                 <dbl>   <dbl> <dbl>     <dbl>
##  1 Airton Vieira            23     131   154     0.149
##  2 Alcides Malossi Junior   23      72    95     0.242
##  3 Alexandre Almeida        41     122   163     0.252
##  4 Amaro Thomé              36      96   132     0.273
##  5 Andrade Sampaio          35      79   114     0.307
##  6 Angélica de Almeida       2       6     8     0.25
##  7 Antonio Tadeu Ottoni      0       1     1     0
##  8 Bandeira Lins             0       2     2     0
##  9 Camargo Aranha Filho     32     109   141     0.227
## 10 Camilo Léllis            32     133   165     0.194
## # ... with 55 more rows
```

Exercício

- Qual quantidade mensal de decisões por juiz?
- Dica: use `data_decisao` `dmy()` e `month()`

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|coc
    droga=case_when(
      droga==TRUE ~ "droga",
      droga==FALSE ~ "n_droga"
    )) %>%
  group_by(juiz,droga) %>%
  summarise(n=n()) %>%
  spread(droga,n,fill = 0) %>%
  mutate(total=droga+n_droga,
         proporcao=droga/total)
```

Resultado


```
## `summarise()` regrouping output by 'juiz' (override wit

## # A tibble: 65 x 5
## # Groups:   juiz [65]
##    juiz                  droga n_droga total proporcao
##    <chr>                 <dbl>   <dbl> <dbl>     <dbl>
##  1 Airton Vieira            23     131   154     0.149
##  2 Alcides Malossi Junior   23      72    95     0.242
##  3 Alexandre Almeida        41     122   163     0.252
##  4 Amaro Thomé              36      96   132     0.273
##  5 Andrade Sampaio          35      79   114     0.307
##  6 Angélica de Almeida       2       6     8     0.25
##  7 Antonio Tadeu Ottoni      0       1     1     0
##  8 Bandeira Lins             0       2     2     0
##  9 Camargo Aranha Filho     32     109   141     0.227
## 10 Camilo Léllis            32     133   165     0.194
## # ... with 55 more rows
```

Exemplo para o ggplot

Unindo e separando colunas

- `unite` junta duas ou mais colunas usando algum separador (_, por exemplo).
- `separate` faz o inverso de `unite`, e uma coluna em várias usando um separador.

Exemplo de separação de colunas

- Olhe os valores da variável classe_assunto

Exemplo de separação de colunas

- Vamos separar a coluna classe_assunto em duas colunas
- coluna classe e coluna assunto
- Existe separador? -> sim, /
- Usei count apenas em assunto

Em ação


```
decisoes %>%
  select(n_processo, classe_assunto) %>%
  separate(classe_assunto, c('classe', 'assunto'), sep = '
           extra = 'merge', fill = 'right') %>%
  count(assunto, sort = TRUE)

## count é um jeito resumido de usar group_by() %>% summar
```

Em ação


```
## # A tibble: 152 x 2
##    assunto                                    n
##    <chr>                                  <int>
##  1 Tráfico de Drogas e Condutas Afins      2441
##  2 Pena Privativa de Liberdade             1106
##  3 Roubo Majorado                          1093
##  4 Furto Qualificado                        838
##  5 Roubo                                    780
##  6 Progressão de Regime                     607
##  7 Furto                                    450
##  8 Receptação                               353
##  9 Homicídio Qualificado                    329
## 10 Crimes de Trânsito                       322
## # ... with 142 more rows
```

List columns: `nest()` e `unnest()`

`nest()` e `unnest()` são operações inversas e servem para tratar dados complexos, como o que temos em `processos`

```
d_partes <- processos %>%
  select(n_processo, partes) %>%
  unnest(partes)
```

As list columns são uma forma condensada de guardar dados que estariam em múltiplas tabelas. Por exemplo, uma alternativa à colocar as `partes` numa list column seria guardar a tabela `d_partes` separadamente.

```
glimpse(d_partes)
```

```
## Rows: 37,579
## Columns: 5
## $ n_processo <chr> "0000003-71.2016.8.26.0073", "000000
## $ id         <int> 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2,
## $ name       <chr> "JOSE MARIA JUSTINO NETO", "Defensor
## $ part       <chr> "Apelante", "Apelante", "Apelado", "
## $ role       <chr> "Apelante", "Apelante", "Apelado", "
```

Duplicatas

Para retirar duplicatas, utilizar `distinct`. Ele considera apenas a primeira linha em que encontra um padrão para as combinações de variáveis escolhidas e descarta as demais.

```
decisoes %>%
  distinct(municipio)
```

```
## # A tibble: 315 x 1
##    municipio
##    <chr>
##  1 Cosmópolis
##  2 São Paulo
##  3 Ribeirão Preto
##  4 Araçatuba
##  5 Presidente Prudente
##  6 Bertioga
##  7 Taubaté
##  8 Aparecida
##  9 Jandira
```

Por coluna

Para manter as demais colunas, use `.keep_all=`:

```
decisoes %>%
  distinct(municipio, camara,
           .keep_all = TRUE)
```

```
## # A tibble: 2,760 x 9
##    id_decisao n_processo classe_assunto municipio camar
##    <chr>      <chr>      <chr>          <chr>     <chr>
##  1 11094999   0057003-2~ Habeas Corpus~ Cosmópol~ 3ª Câ
##  2 11093733   0052762-0~ Habeas Corpus~ São Paulo 3ª Câ
##  3 11093677   0055169-7~ Habeas Corpus~ Ribeirão~ 3ª Câ
##  4 11093270   9000580-8~ Agravo de Exe~ Araçatuba 8ª Câ
##  5 11093374   0052938-7~ Mandado de Se~ São Paulo 8ª Câ
##  6 11093320   9000723-7~ Agravo de Exe~ Presiden~ 8ª Câ
##  7 11091506   0003276-8~ Apelação / Tr~ Bertioga  8ª Câ
##  8 11093326   9000298-1~ Agravo de Exe~ Taubaté   8ª Câ
##  9 11092475   0004653-3~ Apelação / Tr~ Aparecida 8ª Câ
## 10 11093773   2221930-6~ Habeas Corpus~ Jandira   3ª Câ
```

janitor::get_dupes()

Use `janitor::get_dupes()` para averiguar os casos em que há repetição de combinações de colunas.

```
decisoes %>%
  get_dupes(n_processo)
```

```
## # A tibble: 114 x 10
##    n_processo dupe_count id_decisao classe_assunto muni
##    <chr>           <int> <chr>      <chr>           <chr
##  1 0000276-8~          2 11051087   Apelação / Tr~ Itap
##  2 0000276-8~          2 11093633   Embargos de D~ Itap
##  3 0000358-1~          2 11108278   Embargos de D~ São
##  4 0000358-1~          2 11028129   Apelação / Ro~ São
##  5 0002236-1~          2 11041351   Apelação / Co~ Nhan
##  6 0002236-1~          2 11041352   Apelação / Co~ Nhan
##  7 0004453-2~          2 11041132   Apelação / Tr~ São
##  8 0004453-2~          2 11093635   Embargos de D~ São
##  9 0004636-5~          3 11032094   Apelação / Tr~ Olím
## 10 0004636-5~          3 11032093   Apelação / Tr~ Olím
```

Joins

Dados relacionais

- Hadley Wickham http://r4ds.had.co.nz/relational-data.html

Principais funções

Para juntar tabelas, usar `inner_join`, `left_join`, `anti_join`, etc.

Visualizando

Exemplo de inner join:


```
decisoes %>%
  filter(data_registro == "18/01/2018", !is.na(id_decisao)
  select(id_decisao, n_processo) %>%
  inner_join(processos, "n_processo")
```

```
## # A tibble: 169 x 5
##    id_decisao n_processo                infos           pa
##    <chr>      <chr>                     <list>          <l
##  1 11109089   0003779-93.2015.8.26.~ <tibble [14 x ~ <t
##  2 11109088   3001293-25.2013.8.26.~ <tibble [13 x ~ <t
##  3 11108246   0063566-45.2015.8.26.~ <tibble [14 x ~ <t
##  4 11108245   0003528-84.2015.8.26.~ <tibble [14 x ~ <t
##  5 11109087   0008470-76.2015.8.26.~ <tibble [14 x ~ <t
##  6 11109086   0013767-62.2012.8.26.~ <tibble [14 x ~ <t
##  7 11109085   3019561-54.2013.8.26.~ <tibble [14 x ~ <t
##  8 11108348   0003072-91.2017.8.26.~ <tibble [11 x ~ <t
##  9 11108725   0009578-41.2017.8.26.~ <tibble [12 x ~ <t
## 10 11108347   3001116-52.2013.8.26.~ <tibble [12 x ~ <t
## # ... with 159 more rows
```

Exemplo de right join:


```
decisoes %>%
  filter(data_registro == "18/01/2018", !is.na(id_decisao)
  select(id_decisao, n_processo) %>%
  right_join(processos, "n_processo")
```

```
## # A tibble: 11,638 x 5
##    id_decisao n_processo                infos                pa
##    <chr>      <chr>                     <list>               <l
##  1 11109089   0003779-93.2015.8.26.~ <tibble [14 x ~ <t
##  2 11109088   3001293-25.2013.8.26.~ <tibble [13 x ~ <t
##  3 11108246   0063566-45.2015.8.26.~ <tibble [14 x ~ <t
##  4 11108245   0003528-84.2015.8.26.~ <tibble [14 x ~ <t
##  5 11109087   0008470-76.2015.8.26.~ <tibble [14 x ~ <t
##  6 11109086   0013767-62.2012.8.26.~ <tibble [14 x ~ <t
##  7 11109085   3019561-54.2013.8.26.~ <tibble [14 x ~ <t
##  8 11108348   0003072-91.2017.8.26.~ <tibble [11 x ~ <t
##  9 11108725   0009578-41.2017.8.26.~ <tibble [12 x ~ <t
## 10 11108347   3001116-52.2013.8.26.~ <tibble [12 x ~ <t
## # ... with 11,628 more rows
```

Limpeza

Duplicatas

Para retirar duplicatas, utilizar `distinct`. Ele considera apenas a primeira linha em que encontra um padrão para as combinações de variáveis escolhidas e descarta as demais.

```
decisoes %>%
  distinct(municipio)
```

```
## # A tibble: 315 x 1
##    municipio
##    <chr>
##  1 Cosmópolis
##  2 São Paulo
##  3 Ribeirão Preto
##  4 Araçatuba
##  5 Presidente Prudente
##  6 Bertioga
##  7 Taubaté
##  8 Aparecida
##  9 Jandira
## 10 Flórida Paulista
## # ... with 305 more rows
```

Por coluna

Para manter as demais colunas, use `.keep_all=`:

```
decisoes %>%
  distinct(municipio, camara,
           .keep_all = TRUE)
```

```
## # A tibble: 2,760 x 9
##    id_decisao n_processo classe_assunto municipio camar
##    <chr>      <chr>      <chr>          <chr>     <chr>
##  1 11094999   0057003-2~ Habeas Corpus~ Cosmópol~ 3ª Câ
##  2 11093733   0052762-0~ Habeas Corpus~ São Paulo 3ª Câ
##  3 11093677   0055169-7~ Habeas Corpus~ Ribeirão~ 3ª Câ
##  4 11093270   9000580-8~ Agravo de Exe~ Araçatuba 8ª Câ
##  5 11093374   0052938-7~ Mandado de Se~ São Paulo 8ª Câ
##  6 11093320   9000723-7~ Agravo de Exe~ Presiden~ 8ª Câ
##  7 11091506   0003276-8~ Apelação / Tr~ Bertioga  8ª Câ
##  8 11093326   9000298-1~ Agravo de Exe~ Taubaté   8ª Câ
##  9 11092475   0004653-3~ Apelação / Tr~ Aparecida 8ª Câ
## 10 11093773   2221930-6~ Habeas Corpus~ Jandira   3ª Câ
## # ... with 2,750 more rows, and 3 more variables: data_
## #   juiz <chr>, txt_decisao <chr>
```

janitor::get_dupes()

Use `janitor::get_dupes()` para averiguar os casos em que há repetição de combinações de colunas.

```
decisoes %>%
  get_dupes(n_processo)
```

```
## # A tibble: 114 x 10
##    n_processo dupe_count id_decisao classe_assunto muni
##    <chr>           <int> <chr>      <chr>           <chr
##  1 0000276-8~          2 11051087   Apelação / Tr~ Itap
##  2 0000276-8~          2 11093633   Embargos de D~ Itap
##  3 0000358-1~          2 11108278   Embargos de D~ São
##  4 0000358-1~          2 11028129   Apelação / Ro~ São
##  5 0002236-1~          2 11041351   Apelação / Co~ Nhan
##  6 0002236-1~          2 11041352   Apelação / Co~ Nhan
##  7 0004453-2~          2 11041132   Apelação / Tr~ São
##  8 0004453-2~          2 11093635   Embargos de D~ São
##  9 0004636-5~          3 11032094   Apelação / Tr~ Olím
## 10 0004636-5~          3 11032093   Apelação / Tr~ Olím
## # ... with 104 more rows, and 3 more variables: data_re
## #   juiz <chr>, txt_decisao <chr>
```

- Janitor exemplos
  http://sfirke.github.io/janitor/articles/janitor.html
- Missing e imputação
  https://www.analyticsvidhya.com/blog/2016/03/tutorial-powerful-packages-imputing-missing-values/
- Outliers (critérios, limpeza e gráficos)
- `stringi` e `stringr`