Universidade de Brasília

DEPARTAMENTO DE ESTATÍSTICA

16 de novembro de 2022

Lista 1: Computação eficiente (dados em memória)

Resolução - William Rappel - 22/006032

Computação em Estatística para dados e cálculos massivos

Tópicos especiais em Estatística 2

Prof. Guilherme Rodrigues

César Augusto Fernandes Galvão (aluno colaborador)

Gabriel Jose dos Reis Carvalho (aluno colaborador)

1. **As questões deverão ser respondidas em um único relatório *PDF* ou *html*, produzido usando as funcionalidades do *Rmarkdown* ou outra ferramenta equivalente.**
2. **O aluno poderá consultar materiais relevantes disponíveis na internet, tais como livros, *blogs* e artigos.**
3. **O trabalho é individual. Suspeitas de plágio e compartilhamento de soluções serão tratadas com rigor.**
4. **Os códigos *R* utilizados devem ser disponibilizados na integra, seja no corpo do texto ou como anexo.**
5. **O aluno deverá enviar o trabalho até a data especificada na plataforma Microsoft Teams.**
6. **O trabalho será avaliado considerando o nível de qualidade do relatório, o que inclui a precisão das respostas, a pertinência das soluções encontradas, a formatação adotada, dentre outros aspectos correlatos.**
7. **Escreva seu código com esmero, evitando operações redundantes, visando eficiência computacional, otimizando o uso de memória, comentando os resultados e usando as melhores práticas em programação.**

```
## Warning: package 'pacman' was built under R version 4.1.3
```

Nessa lista, utilizamos os pacotes `vroom` e `data.table` para analisar, com rapidez computacional e eficiente uso de memória, dados públicos sobre a vacinação contra a Covid-19.

Questão 1: leitura eficiente de dados

a) Utilizando códigos R, crie uma pasta (chamada *dados*) em seu computador e faça o *download* dos arquivos referentes aos estados do Acre, Alagoas, Amazonas e Amapá, disponíveis no endereço eletrônico a seguir. *https://opendatasus.saude.gov.br/dataset/covid-19-vacinacao/resource/5093679f-12c3-4d6b-b7bd-07694de54173?inner_span=True*

Dica: Veja os slides sobre *web scraping* disponibilizados na página da equipe na plataforma MS Teams, em *Materiais de estudo*, na aba *arquivos*; Eles permitem a imediata identificação dos endereços dos arquivos a serem baixados. Use *wi-fi* para fazer os downloads!

Solução

Primeiro, vamos criar a pasta dados, caso ela não exista.

```
folder <- 'dados'
if (!file.exists(folder)) dir.create(folder)
```

Em seguida, vamos realizar a leitura da página html do link de interesse.

```
url <- 'https://opendatasus.saude.gov.br/dataset/covid-19-vacinacao/resource/5093679f-12c3-4d6b-b7bd
page <- url %>%
  read_html()
```

Agora, pegamos a lista de endereços e o título de cada um. Checamos se possuem o mesmo comprimento.

```
links <- page %>%
  html_nodes('.notes a') %>%
  html_attr('href')
titles <- page %>%
  html_nodes('.notes a') %>%
  html_text()
length(links) == length(titles)
```

```
## [1] TRUE
```

Vamos filtrar apenas os links e títulos que envolvem os estados Acre, Alagoas, Amazonas e Amapá.

```
idx <- titles %>%
  str_detect('(AC)|(AL)|(AM)|(AP)')
links <- links[idx]
titles <- titles[idx]
file_names <- titles %>%
  str_to_lower() %>%
  str_remove_all('(dados)| ')
```

Agora, realizamos o download dos dados.

```
path <- folder %>%
  str_c('/', file_names, '.csv')
for (i in seq_along(links)) {
  if (!file.exists(path[i])) {
    download.file(links[i], path[i])
  }
}
```

b) Usando a função `p_load` (do pacote `pacman`), carregue o pacote `vroom` (que deve ser usado em toda a Questão 1) e use-o para carregar o primeiro dos arquivos baixados para o R (*Dados AC - Parte 1*). Descreva brevemente o banco de dados.

Solução

Vamos realizar a leitura do arquivo `ac-parte1.csv`.

```
ac_1 <- path[1] %>%
  vroom(
    delim=';',
    locale=locale('br', encoding='UTF-8'),
    num_threads=3
  )
```

```
## Rows: 552954 Columns: 32
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ";"
## chr  (23): document_id, paciente_id, paciente_enumSexoBiologico, paciente_ra...
## dbl   (7): paciente_idade, paciente_endereco_coIbgeMunicipio, paciente_ender...
## date  (2): paciente_dataNascimento, vacina_dataAplicacao
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
ac_1
```

```
## # A tibble: 552,954 x 32
##    document_id   paciente_id    paciente_idade paciente_dataNa~ paciente_enumSe~
##    <chr>         <chr>                   <dbl> <date>           <chr>
##  1 ae1d5016-9a2~ dedf6ad67744e~             35 1985-11-27       M
##  2 a4d56f24-edb~ 712bd3c45a233~             19 2003-01-07       M
##  3 ea5f8e83-555~ 1abd3ef52790a~             60 1961-05-06       F
##  4 a4ef4e1a-65e~ 0ef71951170e0~             49 1971-10-30       F
##  5 ab674854-f54~ b9273ea164faa~             53 1968-10-28       F
##  6 a520cf28-d8b~ d9f129e12366d~             40 1980-09-05       M
##  7 11ac2cd8-f42~ 85b6325ff58b6~             41 1979-08-06       F
##  8 a57515a2-883~ 3d2b51d3528e5~             21 2000-04-22       F
##  9 93137004-a77~ 0d3a1d96f82cf~             27 1993-09-17       M
## 10 a575c3c4-b77~ 828b5350ffba0~             34 1987-01-18       M
## # ... with 552,944 more rows, and 27 more variables:
## #   paciente_racaCor_codigo <chr>, paciente_racaCor_valor <chr>,
## #   paciente_endereco_coIbgeMunicipio <dbl>, paciente_endereco_coPais <dbl>,
## #   paciente_endereco_nmMunicipio <chr>, paciente_endereco_nmPais <chr>,
## #   paciente_endereco_uf <chr>, paciente_endereco_cep <dbl>,
## #   paciente_nacionalidade_enumNacionalidade <chr>,
## #   estabelecimento_valor <chr>, estabelecimento_razaoSocial <chr>, ...
```

O banco de dados apresenta 552.954 linhas e 32 colunas. Cada linha se refere a um paciente/indivíduo e cada coluna a alguma informação demográfica daquele indivíduo ou referente à vacinação do mesmo.

Das 32 colunas, 23 são do tipo `character`, 7 do tipo `double` e 2 do tipo `date`.

c) Qual é o tamanho total (em Megabytes) de todos os arquivos baixados (use a função `file.size`)? Qual é o espaço ocupado pelo arquivo *Dados AC - Parte 1* na memória do `R` (use a função `object.size`) e no Disco rígido (*HD*)? Comente os resultados.

Solução

Primeiro, calculamos o tamanho total da pasta contendo todos os arquivos baixados.

```
sum(file.size(path))/10^6
```

```
## [1] 8667.878
```

Logo, o tamanho total é de mais de 8 GB.

Em seguida, calculamos o espaço ocupado pelo arquivo `ac-parte1.csv` na memória do R e no Disco Rígido.

```
format(object.size(ac_1), units='Mb')
```

```
## [1] "255.9 Mb"
```

```
file.size(path[1])/10^6
```

```
## [1] 283.1486
```

O arquivo na memória do R ocupa um espaço de 256 MB, que é um valor inferior aos 283 MB ocupados no HD.

d) Repita o procedimento do item **b)**, mas, dessa vez, carregue para a memória apenas os casos em que a vacina aplicada foi a Janssen. Para tanto, faça a filtragem usando uma conexão `pipe()`. Observe que a filtragem deve ser feita durante o carregamento, e não após ele.

Quantos megabites deixaram de ser carregados para a memória RAM (ao fazer a filtragem durante a leitura, e não no próprio `R`)?

Solução

A coluna que informa a vacina aplicada é `vacina_fabricante_nome`.

```
table(ac_1$vacina_fabricante_nome)
```

```
##
##                                ASTRAZENECA
##                                       8657
##                          ASTRAZENECA/FIOCRUZ
##                                     176532
##                                    JANSSEN
##                                      19532
##                       Pendente Identifica??o
##                                        499
##                       Pendente Identificação
##                                      12106
##                                     PFIZER
##                                     242146
##                          PFIZER - PEDI?TRICA
##                                       5568
##                          PFIZER - PEDIÁTRICA
##                                       3973
## PFIZER MANUFACTURING BELGIUM NV - BELGICA
##                                          3
##                                    SINOVAC
##                                       1007
##                              SINOVAC/BUTANTAN
##                                      82931
```

Vamos utilizar o comando `pipe()` para realizar este filtro durante o carregamento.

```
command <- 'findstr -i "document JANSSEN" dados\\ac-parte1.csv'
ac_1_janssen <- command %>%
  pipe() %>%
  vroom(
    delim=';',
    locale=locale('br', encoding='UTF-8'),
    num_threads=3
  )
```

```
## Rows: 19532 Columns: 32
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ";"
## chr  (24): document_id, paciente_id, paciente_enumSexoBiologico, paciente_ra...
## dbl   (6): paciente_idade, paciente_endereco_coIbgeMunicipio, paciente_ender...
## date  (2): paciente_dataNascimento, vacina_dataAplicacao
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
ac_1_janssen
```

```
## # A tibble: 19,532 x 32
##    document_id   paciente_id    paciente_idade paciente_dataNa~ paciente_enumSe~
##    <chr>         <chr>                   <dbl> <date>           <chr>
##  1 a7b2e803-75a~ 3e35925a3512d~             30 1991-08-30       F
##  2 5636015b-b4f~ b1abdbb5151d5~             62 1959-11-12       F
##  3 a735be69-d8a~ f2516d6a6c60e~             32 1989-12-27       F
##  4 627c6216-ea6~ cd7847b709f88~             49 1972-08-20       M
##  5 aa6dce4a-eb8~ afa861484ff34~             31 1989-12-07       M
##  6 b9d0d725-6ed~ 68b345b2afd33~             30 1992-02-01       F
##  7 50766f6e-1d0~ 4125caa679211~             40 1981-03-01       M
##  8 24e4fe16-bdc~ fe8e31b6f96ad~             63 1959-03-18       F
##  9 a26ee707-996~ fc61b4711940c~             43 1978-01-24       F
## 10 18294c2e-ae8~ b538d47a5a867~             29 1992-06-29       M
## # ... with 19,522 more rows, and 27 more variables:
## #   paciente_racaCor_codigo <chr>, paciente_racaCor_valor <chr>,
## #   paciente_endereco_coIbgeMunicipio <dbl>, paciente_endereco_coPais <dbl>,
## #   paciente_endereco_nmMunicipio <chr>, paciente_endereco_nmPais <chr>,
## #   paciente_endereco_uf <chr>, paciente_endereco_cep <chr>,
## #   paciente_nacionalidade_enumNacionalidade <chr>,
## #   estabelecimento_valor <chr>, estabelecimento_razaoSocial <chr>, ...
```

Em seguida, vemos o espaço ocupado pelo objeto `dados_ac_1_jansenn`.

```
format(object.size(ac_1_janssen), units='Mb')
```

```
## [1] "9.8 Mb"
```

Dessa forma, ao fazer a filtragem durante a leitura, e não no próprio `R`, deixamos de carregar 251 MB.

e) Carregue para o `R` **todos** os arquivos da pasta de uma única vez (usando apenas um comando `R`, sem métodos iterativos), trazendo apenas os casos em que a vacina aplicada foi a Janssen.

Solução

Concatenamos o comando e executamos o carregamento de uma só vez.

```
command <- 'findstr -i "document JANSSEN" dados\\*.csv'
janssen <- command %>%
  pipe() %>%
  vroom(
    delim=';',
    locale=locale('br', encoding='UTF-8'),
    num_threads=3
    )
```

```
## Rows: 405322 Columns: 32
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ";"
## chr  (26): dados\ac-parte1.csv:"document_id", paciente_id, paciente_enumSexo...
## dbl   (4): paciente_idade, estabelecimento_municipio_codigo, vacina_categori...
## date  (2): paciente_dataNascimento, vacina_dataAplicacao
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
janssen <- janssen %>%
  filter(estabelecimento_uf != 'estabelecimento_uf')
janssen
```

```
## # A tibble: 405,311 x 32
##    `dados\\ac-part~ paciente_id paciente_idade paciente_dataNa~ paciente_enumSe~
##    <chr>            <chr>                <dbl> <date>          <chr>
##  1 "dados\\ac-part~ 3e35925a35~             30 1991-08-30      F
##  2 "dados\\ac-part~ b1abdbb515~             62 1959-11-12      F
##  3 "dados\\ac-part~ f2516d6a6c~             32 1989-12-27      F
##  4 "dados\\ac-part~ cd7847b709~             49 1972-08-20      M
##  5 "dados\\ac-part~ afa861484f~             31 1989-12-07      M
##  6 "dados\\ac-part~ 68b345b2af~             30 1992-02-01      F
##  7 "dados\\ac-part~ 4125caa679~             40 1981-03-01      M
##  8 "dados\\ac-part~ fe8e31b6f9~             63 1959-03-18      F
##  9 "dados\\ac-part~ fc61b47119~             43 1978-01-24      F
## 10 "dados\\ac-part~ b538d47a5a~             29 1992-06-29      M
## # ... with 405,301 more rows, and 27 more variables:
## #   paciente_racaCor_codigo <chr>, paciente_racaCor_valor <chr>,
## #   paciente_endereco_coIbgeMunicipio <chr>, paciente_endereco_coPais <chr>,
## #   paciente_endereco_nmMunicipio <chr>, paciente_endereco_nmPais <chr>,
## #   paciente_endereco_uf <chr>, paciente_endereco_cep <chr>,
## #   paciente_nacionalidade_enumNacionalidade <chr>,
## #   estabelecimento_valor <chr>, estabelecimento_razaoSocial <chr>, ...
```

```
rm(list=ls())
```

Questão 2: manipulação de dados

a) Utilizando o pacote `data.table`, repita o procedimento do item **1e)**, agora mantendo, durante a leitura, todas as vacinas e apenas as colunas `estabelecimento_uf`, `vacina_descricao_dose` e

`estabelecimento_municipio_codigo`. Use o pacote `geobr` para obter os dados sobre as regiões de saúde do Brasil (comando `geobr::read_health_region()`). O pacote `geobr` não está mais disponível para download no CRAN; Para instalá-lo, use o link https://cran.r-project.org/src/contrib/Archive/geobr/.

A tabela que relaciona o código do IBGE (`estabelecimento_municipio_codigo`, na tabela de vacinação) e o código de saúde (`code_health_region`, na tabela de regiões de saúde) está disponível pelo link https://sage.saude.gov.br/paineis/regiaoSaude/lista.php?output=html& e nos arquivos da lista.

Solução

Primeiro, realizamos o carregamento com a função `fread`.

```
janssen_dt <- fread(
  cmd='findstr ; dados\\*.csv',
  select=c('estabelecimento_uf', 'vacina_descricao_dose', 'estabelecimento_municipio_codigo'),
  encoding='UTF-8',
  sep=';'
)
janssen_dt <- janssen_dt[estabelecimento_uf != 'estabelecimento_uf']
janssen_dt[, estabelecimento_municipio_codigo := as.integer(estabelecimento_municipio_codigo)]
```

Em seguida, obtemos os dados das regiões de saúde do Brasil.

```
regioes_saude_br <- geobr::read_health_region() %>%
  as.data.table()
```

```
## Using year 2013
```

```
## Downloading: 780 B     Downloading: 780 B     Downloading: 1.7 kB     Downloading: 1.7 kB     Dow
```

```
regioes_saude_br[, geom := NULL]
regioes_saude_br[, code_health_region := as.integer(code_health_region)]
```

Por último, lemos a tabela que relaciona o código do IBGE com o código de saúde.

```
codigos <- fread(
  file='Tabela_codigos.csv',
  encoding='UTF-8',
  select=c('Cód IBGE', 'Cód Região de Saúde'),
  col.names=c('estabelecimento_municipio_codigo', 'code_health_region')
)
```

b) Junte (*join*) os dados da base de vacinações com o das regiões de saúde e descreva brevemente o que são as regiões (use documentação do governo, não se atenha à documentação do pacote). Em seguida, crie as variáveis descritas abaixo:

1. Quantidade de vacinados por região de saúde;
2. Condicionalmente, a *faixa de vacinação* por região de saúde (alta ou baixa, em relação à mediana da distribuição de vacinações).

Crie uma tabela com as 5 regiões de saúde com menos vacinados em cada *faixa de vacinação*.

Solução

A Resolução Nº 1 do Ministério da Saúde, datada de 29 de Setembro de 2011 define região de saúde como sendo: o espaço geográfico contínuo constituído por agrupamento de Municípios limítrofes, delimitado a partir de identidades culturais, econômicas e sociais e de redes de comunicação e infraestrutura de transportes compartilhados, com a finalidade de integrar a organização, o planejamento e a execução de ações e serviços de saúde.

Agora, vamos realizar o join e, em seguida, as agregações.

```
janssen_dt_full <- janssen_dt %>%
  merge(
    codigos,
    by.x='estabelecimento_municipio_codigo',
    by.y='estabelecimento_municipio_codigo',
    all.x=TRUE
  ) %>%
  merge(
    regioes_saude_br,
    by.x='code_health_region',
    by.y='code_health_region',
    all.x=TRUE
  )
janssen_dt_full
```

```
##           code_health_region estabelecimento_municipio_codigo
##        1:              12001                            120005
##        2:              12001                            120005
##        3:              12001                            120005
##        4:              12001                            120005
##        5:              12001                            120005
##       ---
## 17942941:              27010                            270710
## 17942942:              27010                            270710
## 17942943:              27010                            270710
## 17942944:              27010                            270710
## 17942945:              27010                            270710
##           estabelecimento_uf vacina_descricao_dose  name_health_region
##        1:                 AC               Reforço           Alto Acre
##        2:                 AC               2ª Dose           Alto Acre
##        3:                 AC               1ª Dose           Alto Acre
##        4:                 AC               1ª Dose           Alto Acre
##        5:                 AC               2ª Dose           Alto Acre
##       ---
## 17942941:                 AL               1ª Dose 10ª Região de Saúde
## 17942942:                 AL               Reforço 10ª Região de Saúde
## 17942943:                 AL               1ª Dose 10ª Região de Saúde
## 17942944:                 AL               1ª Dose 10ª Região de Saúde
## 17942945:                 AL               2ª Dose 10ª Região de Saúde
##           code_state abbrev_state name_state
##        1:         12           AC       Acre
##        2:         12           AC       Acre
##        3:         12           AC       Acre
##        4:         12           AC       Acre
##        5:         12           AC       Acre
##       ---
## 17942941:         27           AL    Alagoas
## 17942942:         27           AL    Alagoas
## 17942943:         27           AL    Alagoas
## 17942944:         27           AL    Alagoas
## 17942945:         27           AL    Alagoas
```

```
janssen_dt_full[, .(n = .N), by = name_health_region][, med := median(n)][, faixa_de_vacinacao := if
```

```
##          name_health_region      n    med faixa_de_vacinacao id
##  1:          Médio Amazonas 361708 361708               alta  1
##  2:           Área Sudoeste 377919 361708               alta  2
```

```
##  3:      6ª Região de Saúde 399733 361708              alta  3
##  4:      5ª Região de Saúde 408578 361708              alta  4
##  5: Juruá e Tarauacá/Envira 414869 361708              alta  5
##  6:              Área Norte 110379 361708             baixa  1
##  7:               Alto Acre 122967 361708             baixa  2
##  8:          Regional Purus 182562 361708             baixa  3
##  9:          Regional Juruá 201955 361708             baixa  4
## 10:      2ª Região de Saúde 256833 361708             baixa  5
```

c) Utilizando o pacote `dtplyr`, repita o procedimento do item **b)** (lembre-se das funções `mutate`, `group_by`, `summarise`, entre outras). Exiba os resultados.

Solução

Primeiro, os joins. Depois, as agregações.

```
janssen_dt %>%
  lazy_dt() %>%
  left_join(codigos, by='estabelecimento_municipio_codigo') %>%
  left_join(regioes_saude_br, by='code_health_region') %>%
  group_by(name_health_region) %>%
  summarise(n = n()) %>%
  mutate(faixa_de_vacinacao = if_else(n < median(n), 'baixa', 'alta')) %>%
  arrange(faixa_de_vacinacao, n) %>%
  group_by(faixa_de_vacinacao) %>%
  mutate(id = 1:n()) %>%
  filter(id <= 5) %>%
  as_tibble()
```

```
## # A tibble: 10 x 4
##    name_health_region           n faixa_de_vacinacao    id
##    <chr>                    <int> <chr>              <int>
##  1 Médio Amazonas          361708 alta                   1
##  2 Área Sudoeste           377919 alta                   2
##  3 6ª Região de Saúde      399733 alta                   3
##  4 5ª Região de Saúde      408578 alta                   4
##  5 Juruá e Tarauacá/Envira 414869 alta                   5
##  6 Área Norte              110379 baixa                  1
##  7 Alto Acre               122967 baixa                  2
##  8 Regional Purus          182562 baixa                  3
##  9 Regional Juruá          201955 baixa                  4
## 10 2ª Região de Saúde      256833 baixa                  5
```

d) Com o pacote `microbenchmark`, compare o tempo de execução dos itens **b)** e **c)**. Isso é, quando se adota o `data.table` e o `dtplyr`, respectivamente.

Extra: Inclua na comparação a execução usando o próprio `dplyr`. Para isso, primeiro converta os 3 objetos do item **a)** para a classe `tibble`.

Solução

Primeiro, criamos funções para realizar cada processo.

```
# data.table
func_dt <- function() {
  full <- janssen_dt %>%
  merge(
    codigos,
    by.x='estabelecimento_municipio_codigo',
    by.y='estabelecimento_municipio_codigo',
```

```
    all.x=TRUE
  ) %>%
  merge(
    regioes_saude_br,
    by.x='code_health_region',
    by.y='code_health_region',
    all.x=TRUE
  )
  full[, .(n = .N), by = name_health_region][, med := median(n)][, faixa_de_vacinacao := ifelse(n <
}

# dtplyr
func_dtplyr <- function() {
  janssen_dt %>%
    lazy_dt() %>%
    left_join(codigos, by='estabelecimento_municipio_codigo') %>%
    left_join(regioes_saude_br, by='code_health_region') %>%
    group_by(name_health_region) %>%
    summarise(n = n()) %>%
    mutate(faixa_de_vacinacao = if_else(n < median(n), 'baixa', 'alta')) %>%
    arrange(faixa_de_vacinacao, n) %>%
    group_by(faixa_de_vacinacao) %>%
    mutate(id = 1:n()) %>%
    filter(id <= 5) %>%
    as_tibble()
}

# dplyr
func_dplyr <- function() {
  x_tbl <- janssen_dt %>% as_tibble()
  y_tbl <- codigos %>% as_tibble()
  z_tbl <- regioes_saude_br %>% as_tibble()
  x_tbl %>%
    left_join(y_tbl, by='estabelecimento_municipio_codigo') %>%
    left_join(z_tbl, by='code_health_region') %>%
    group_by(name_health_region) %>%
    summarise(n = n()) %>%
    mutate(faixa_de_vacinacao = if_else(n < median(n), 'baixa', 'alta')) %>%
    arrange(faixa_de_vacinacao, n) %>%
    group_by(faixa_de_vacinacao) %>%
    mutate(id = 1:n()) %>%
    filter(id <= 5)
}
```

Em seguida, aplicamos a comparação com `microbenchmark`.

```
microbenchmark(
  data.table = func_dt(),
  dtplyr     = func_dtplyr(),
  dplyr      = func_dplyr(),
  times      = 10
)
```

```
## Unit: seconds
##        expr      min       lq     mean   median       uq      max neval
##  data.table 3.901226 4.555613 4.747729 4.770402 5.048737 5.355329    10
##      dtplyr 3.385758 3.526162 3.720573 3.635905 3.899209 4.326215    10
##       dplyr 3.371034 3.476967 3.756824 3.737075 3.909240 4.436961    10
```

Portanto, utilizando a mediana como critério, o tempo de execução do `dtplyr` foi o menor, seguido pelo `dplyr` e, por último, o `data.table`.