



Universidade de Brasília

DEPARTAMENTO DE ESTATÍSTICA

16 de novembro de 2022

Lista 1: Computação eficiente (dados em memória)

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 íntegra, 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.

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:

```
options(timeout=600)

library(pacman)
library(rvest)
library(stringr)
library(dplyr)

dir.create('dados')

ufs_regex = "AC|AL|AM|AP"
ufs = c("AC", "AL", "AM", "AP")
parts = 1:3

ADDRESS = "https://opendatasus.saude.gov.br/dataset/covid-19-vacinacao/resource/5093679f-12c3-4d6b-b7bd-07694de54173?inner_span=True"

links = read_html(ADDRESS) %>%
  html_elements("li") %>%
  html_elements("a") %>%
  html_attr("href")

download_links = links[grepl(ufs_regex, links)]
paths = do.call(paste0, expand.grid(ufs, parts) %>% arrange(Var1, Var2))

for (i in seq_along(download_links)) {
  download.file(url = download_links[i],
               destfile = paste0('dados/', paths[i], '.csv'))
}
```

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:

```
p_load(vroom)
ac1 = vroom(file = "dados/AC1.csv",
            locale = locale("br", encoding = "UTF-8"),
            num_threads = 3) %>%
  mutate_at(vars(matches('paciente.*raca|vacina.*nome')), as.factor)
```

```
## Rows: 552545 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.
```

```
COLS = c('paciente_idade',
         'paciente_enumSexoBiologico',
         'paciente_racaCor_valor',
         'vacina_grupoAtendimento_nome',
         'vacina_nome',
         'vacina_dataAplicacao')
```

```
summary(ac1[, COLS])
```

```
## paciente_idade    paciente_enumSexoBiologico    paciente_racaCor_valor
## Min.   : 0.00    Length:552545          AMARELA      :160090
## 1st Qu.: 22.00    Class :character          BRANCA       : 60775
## Median : 35.00    Mode  :character          INDIGENA     : 11444
## Mean   : 36.85                    PARDA       :241472
## 3rd Qu.: 50.00                    PRETA       : 13848
## Max.   :121.00                    SEM INFORMACAO: 64915
## NA's    :1                      NA's        :    1
##
##                                     vacina_grupoAtendimento_nome
## Pessoas de 18 a 64 anos                                     :289185
## Pessoas de 12 a 17 anos                                     : 49024
## Pessoas de 5 a 11 anos                                       : 28953
## Hipertensão de difícil controle ou com complicações/lesão de órgão alvo: 22593
## Pessoas de 65 a 69 anos                                       : 16927
## (Other)                                                         :145022
## NA's                                                            :   841
##
##                                     vacina_nome    vacina_dataAplicacao
## COVID-19 PFIZER - COMIRNATY                :241984    Min.   :2021-01-17
## COVID-19 ASTRAZENEC/FIOCRUZ - COVISHIELD:176437    1st Qu.:2021-07-06
## COVID-19 SINOVA/BUTANTAN - CORONAVAC        : 82873    Median :2021-09-20
## COVID-19 PEDIÁTRICA - PFIZER COMIRNATY      : 20961    Mean   :2021-10-23
## COVID-19 JANSSEN - Ad26.COV2.S              : 19503    3rd Qu.:2022-01-31
## (Other)                                       : 10786    Max.   :2022-11-23
## NA's                                         :    1
```

O conjunto de dados consiste em registros de vacinação de pacientes, para o estado do Acre. Nota-se que a raça PARDA é a mais frequente e a idade média é de 36,85 anos. Além disso, a vacina de maior aplicação foi a COVID-19 PFIZER - COMINARTY. Há registros de vacinação desde 2021-01-17 até 2022-11-23.

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:

```
files = do.call(paste0, expand.grid('dados/', list.files('dados/')[-8]))
(dados_folder_size = sum(sapply(files, file.size)))
```

```
## [1] 7360741789
```

```
(ac1_memory_R = object.size(ac1))
```

```
## 255165840 bytes
```

```
(ac1_hd = 282951680)
```

```
## [1] 282951680
```

O tamanho total de todos os arquivos é de 7360741789 bytes, ler algo desse tamanho irá provavelmente tornar o R muito lento. Nota-se que o tamanho do dataset do Acre (Parte 1) é um pouco menor (255165840) quando comparado com o do disco rígido (282951680).

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

Solução:

```
janssen = vroom(file = pipe('findstr -i "document JANSSEN" dados\\AC1.csv'),
                 delim = ";",
                 locale = locale("br", encoding = "UTF-8"))
```

```
## Rows: 19503 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.
```

```
janssen_memory_R = object.size(janssen)
(ac1_memory_R - janssen_memory_R)
```

```
## 244913736 bytes
```

Nota-se que 244913736 bytes deixaram de ser carregados para a memória RAM.

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:

```
janssen_all = vroom(file = pipe("findstr JANSSEN dados\\*.csv"),
                     delim = ";",
                     locale = locale("br", encoding = "UTF-8"))
```

```
## New names:
## Rows: 402660 Columns: 32
## -- Column specification
## ----- Delimiter: ";" chr
```

```
## (24): dados\AC1.csv:"42f81bc5-1178-4fbe-a847-7c7aec6e69e1-i0b0", 6a512c... dbl
## (6): 24, 120020...8, 10, 120020...18, 2, 88 date (2): 1996-09-17, 2021-07-02
## 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.
## * '120020' -> '120020...8'
## * 'CRUZEIRO DO SUL' -> 'CRUZEIRO DO SUL...10'
## * 'AC' -> 'AC...12'
## * '120020' -> '120020...18'
## * 'CRUZEIRO DO SUL' -> 'CRUZEIRO DO SUL...19'
## * 'AC' -> 'AC...20'
```

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:

```
library(data.table)
library(geobr)
files = list.files(path='dados/', full.names = TRUE)
COLS = c('estabelecimento_uf',
         'vacina_descricao_dose',
         'estabelecimento_municipio_codigo')
covid_subset = rbindlist(lapply(files, fread, select = COLS))

health_region = read_health_region() %>% as.data.table()
```

```
## Downloading: 1.2 kB      Downloading: 1.2 kB      Downloading: 9.3 kB      Downloading: 9.3 kB      D
```

```
municipal_code = fread("Tabela_codigos.csv")
colnames(municipal_code) = c('x',
                             'uf',
                             'municipio',
                             'cod_IBGE',
                             'cod_regiao_saude',
                             'nome_regiao_saude')
```

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:

De acordo com a RESOLUÇÃO Nº 1, DE 29 DE SETEMBRO DE 2011, do MINISTÉRIO DA SAÚDE, considera-se Região de Saúde 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.

```
data_table_wrapper = function(){
  covid_subset_add = merge(covid_subset, municipal_code,
                           by.x = 'estabelecimento_municipio_codigo',
                           by.y = 'cod_IBGE',
                           all.x = TRUE)

  agg_region_vaccinated = covid_subset_add[, .N, by = nome_regiao_saude]
  vaccination_median = median(agg_region_vaccinated$N)
  agg_region_vaccinated = agg_region_vaccinated[,classification:=
                                                ifelse(N<=vaccination_median,
                                                "Baixa", "Alta")]

  agg_region_vaccinated = agg_region_vaccinated[order(N)]

  bottom_low = agg_region_vaccinated[classification == "Baixa"][1:5, ]
  bottom_high = agg_region_vaccinated[classification == "Alta"][1:5, ]

  return(list(bottom_low, bottom_high))
}

data_table_wrapper()
```

```
## [[1]]
##      nome_regiao_saude      N classification
## 1:      Área Norte 110280      Baixa
## 2:      Alto Acre 122800      Baixa
## 3:      Regional Purus 182205      Baixa
## 4:      Regional Juruá 201612      Baixa
## 5: 2ª Região de Saúde 256440      Baixa
##
## [[2]]
##      nome_regiao_saude      N classification
## 1:      Área Sudoeste 376821      Alta
## 2:      6ª Região de Saúde 399354      Alta
## 3:      5ª Região de Saúde 407997      Alta
## 4: Juruá e Tarauacá/Envira 414712      Alta
## 5:      Rio Negro e Solimões 456254      Alta
```

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:

```
library(dtplyr)
lazy_covid_subset = lazy_dt(covid_subset)

dtplyr_wrapper = function(x, y){
  lazy_agg_region_vaccinated = x %>%
    left_join(y,by = c("estabelecimento_municipio_codigo" = "cod_IBGE")) %>%
    group_by(nome_regiao_saude) %>%
    summarise(n = n()) %>%
    mutate(classification=
           if_else(n<=median(n),"Baixa","Alta")) %>%
```

```

    arrange(n) %>%
    as_tibble()

    bottom_low = lazy_agg_region_vaccinated %>%
      filter(classification=='Baixa') %>% head(n=5)
    bottom_high = lazy_agg_region_vaccinated %>%
      filter(classification=='Alta') %>% head(n=5)

    return(list(bottom_low, bottom_high))

  }

dtplyr_wrapper(lazy_covid_subset, municipal_code)

```

```

## [[1]]
## # A tibble: 5 x 3
##   nome_regiao_saude      n classification
##   <chr>                <int> <chr>
## 1 Área Norte          110280 Baixa
## 2 Alto Acre           122800 Baixa
## 3 Regional Purus      182205 Baixa
## 4 Regional Juruá      201612 Baixa
## 5 2ª Região de Saúde  256440 Baixa
##
## [[2]]
## # A tibble: 5 x 3
##   nome_regiao_saude      n classification
##   <chr>                <int> <chr>
## 1 Área Sudoeste        376821 Alta
## 2 6ª Região de Saúde   399354 Alta
## 3 5ª Região de Saúde   407997 Alta
## 4 Juruá e Tarauacá/Envira 414712 Alta
## 5 Rio Negro e Solimões  456254 Alta

```

d) Com o pacote `microbenchmark`, comparar 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:

```

library(microbenchmark)
tbl_covid_subset = covid_subset %>% as_tibble()
tbl_municipal_code = municipal_code %>% as_tibble()

microbenchmark(
  data_table = data_table_wrapper(),
  dtplyr = dtplyr_wrapper(lazy_covid_subset, municipal_code),
  dplyr = dtplyr_wrapper(tbl_covid_subset, tbl_municipal_code),
  times = 5)

## Unit: seconds
##      expr      min       lq      mean     median        uq      max neval
## data_table 3.850069 4.011904 4.159174 4.067197 4.361273 4.505427     5
##      dtplyr 2.375921 2.502947 3.472985 3.243281 4.165572 5.077204     5
##      dplyr 72.053341 72.729782 74.688258 74.686932 76.164428 77.806806     5

```

Logo, a manipulação por `dtplyr` foi a mais rápida. Além disso, é interessante também notar que utilizar a `dplyr` teve um tempo de execução dezenas de vezes maior que as estratégias por `dtplyr` e `data.table`.