aula04

ME315-2S2025

brotto

2025-08-20

Sumário

§ Natureza

O presente documento configura uma proposta de solução aos problemas:

- Laboratório n.° 2;
- Desafio n.° 2.

Apresentados durante a aula de ME315-2S2025 na Quinta-feira de, 14, de Agosto.

§ Resumo

São apresentadas a seguir exemplos de soluções aos problemas citados anteriormente. O primeiro é um problema elaborado por professores que já ministraram essa disciplina em semestres passados; o segundo é um desafio e consiste em replicar a atividade em linguagem Python.

A ferramenta RStudio junto de pacotes desenvolvidos pela comunidade R permitem que essas atividades sejam integradas num só documento Rmd. Este pdf é compilado a partir de um código fonte Rmd e propõe-se a integrar a elaboração das duas atividades.

§ Introdução

Inicialmente a proposta da atividade elaborada pelos Professores Benilton e Guilherme visa desenvolver um conjunto de habilidades muito importantes a todo estatísitico – manipular conjuntos de dados (incluindo os volumosos) – mas visto que este documento consiste de propostas de soluções a duas atividades, o escopo das habilidades a serem desenvolvidas se tornou maior.

Dado essa fato, detalhes sobre as soluções não são constantemente apresentados; para mais detalhes e dicas, recomenda-se a leitura do código-fonte responsável pela compilação deste .Rmd.

Mais ainda, algo muito importante sobre a compilação desse mesmo código fonte: Este documento possui várias dependências e a instalação dos pacotes nescessários está inclusa no documento, na forma dos blocos de código desativados. Quando, e se, desejar-se compilar este documento, certifique-se de que o pacotes inclusos nos blocos desativados estão instalados. (Sujestão: Substitua a propriedade FALSE, por TRUE, nas opções de inicialização dos mesmos.)

Tabela 1: Os dois primeiros registros do CD.

YEAR	MONTH	DAY	DAY_OF_WEEK	AIRLINE	FLIGHT_NUMBER
2015	1	1	4	AS	98
2015	1	1	4	AA	2336

TAIL_NUMBER	ORIGIN_AIRPORT	DESTINATION_AIRPORT
N407AS	ANC	SEA
N3KUAA	LAX	PBI

SCHEDULED_DEPARTURE	DEPARTURE_TIME	DEPARTURE_DELAY	TAXI_OUT
0005	2354	-11	21
0010	0002	-8	12

WHEELS_OFF	SCHEDULED_TIME	ELAPSED_TIME	AIR_TIME	DISTANCE
0015	205	194	169	1448
0014	280	279	263	2330

WHEELS_ON	TAXI_IN	SCHEDULED_ARRIVAL	ARRIVAL_TIME	ARRIVAL_DELAY
0404	4	0430	0408	-22
0737	4	0750	0741	-9

DIVERTED	CANCELLED	CANCELLATION_REASON	AIR_SYSTEM_DELAY
0	0	NA	NA
0	0	NA	NA

SECURITY_DELAY	AIRLINE_DELAY	LATE_AIRCRAFT_DELAY	WEATHER_DELAY
NA	NA	NA	NA
NA	NA	NA	NA

§§ Conjunto de dados (CD): `flights.csv.zip`

```
#! CHUNK DESATIVADA
## Instala dependências
   install.packages("readr")
   devtools::install("./temp.package")
```

```
## Inicializa dependências
   library(readr)
   library(temp.package)
```

```
## Amostra do CD os 2 primeiros registros
   a = read_csv("flights.csv.zip", n_max = 2)
   lista_partes <- split.default(a, cut(1:ncol(a), breaks = seq(0, ncol(a), by = 1)))
```

```
## Trata Tabela 1
   u = (lista_partes %>% my_kbl())[1]
   v = substring(u, 14, nchar(u))
   ss2 = "\\caption{Os dois primeiros registros do CD.}"; ss1 = "\\begin{table}"
   paste0(ss1,ss2,v) %>% cat()
```

Laboratório 2: Processamento de Bases de Dados em Lote

Benilton Carvalho & Guilherme Ludwig

§ Instruções

1. Quais são as estatísticas suficientes para a determinação do percentual de vôos atrasados na chegada (`ARRIVAL\_DELAY` > 10)?

Nº total de vôos e Nº de vôos atrasadados.

---

2. Crie uma função chamada `getStats` que, para um conjunto de qualquer tamanho de dados provenientes de `flights.csv.zip`, execute as seguintes tarefas (usando apenas verbos do `dplyr`):

a) Filtre o conjunto de dados de forma que contenha apenas observações das seguintes Cias. Aéreas: AA, DL, UA e US;

b) Remova observações que tenham valores faltantes em campos de interesse;

c) Agrupe o conjunto de dados resultante de acordo com: dia, mês e cia. aérea;

d) Para cada grupo em b), determine as estatísticas suficientes apontadas no item 1. e os retorne como um objeto da classe `tibble`;

e) A função deve receber apenas dois argumentos:

   i. `input`: o conjunto de dados (referente ao lote em questão);

   ii. `pos`: argumento de posicionamento de ponteiro dentro da base de dados. Apesar de existir na função, este argumento não será empregado internamente. É importante observar que, sem a definição deste argumento, será impossível fazer a leitura por partes.

```
getStats = function(input, pos){
  input %>%
    filter(AIRLINE == "AA" | AIRLINE == "DL" | AIRLINE == "UA" | AIRLINE == "US",
     ↪  !is.na(ARRIVAL_DELAY)) %>%
    group_by(DAY, MONTH, AIRLINE) %>%
    mutate(late = (ARRIVAL_DELAY > 10))
}
```

---

3. Utilize alguma função `readr::read\_***\_chunked` para importar o arquivo `flights.csv.zip`.

a) Configure o tamanho do lote (chunk) para 100 mil registros;

b) Configure a função de `callback` para instanciar DataFrames aplicando a função `getStats` criada acima;

c) Configure o argumento `col\_types` de forma que ele leia, diretamente do arquivo, apenas as colunas de interesse (veja nota de aula para identificar como realizar esta tarefa);

```
mycols = cols_only(DAY='i', MONTH='i', YEAR='i', AIRLINE='c', ARRIVAL_DELAY='i')
t0 = Sys.time()
in2 = read_csv_chunked("flights.csv.zip",
                       callback = DataFrameCallback$new(getStats),
                       chunk_size = 1e5,
                       col_types = mycols)
t1 = Sys.time()
```

$$\Delta t = 23.5451850891113\ s$$

---

4. Crie uma função chamada `computeStats` que:

a) Combine as estatísticas suficientes para compor a métrica final de interesse (percentual de atraso por dia/mês/cia aérea);

b) Retorne as informações em um `tibble` contendo apenas as seguintes colunas:

   i. `Cia`: sigla da companhia aérea;

   ii. `Data`: data, no formato AAAA-MM-DD (dica: utilize o comando `as.Date`);

   iii. `Perc`: percentual de atraso para aquela cia. aérea e data, apresentado como um número real no intervalo $[0, 1]$.

```
computeStats = function(input){
  input %>%
    mutate(DATE=as.Date(paste(YEAR,MONTH,DAY,sep="-"))) %>%
    group_by(AIRLINE,DATE) %>%
    reframe(Perc = sum( 1*(late))/n())
}
```

---

5. Produza um mapa de calor em formato de calendário para cada Cia. Aérea.

a) Instale e carregue os pacotes `ggcal` e `ggplot2`.

b) Defina uma paleta de cores em modo gradiente. Utilize o comando `scale\_fill\_gradient.` A cor inicial da paleta deve ser `\#4575b4` e a cor final, `\#d73027`. A paleta deve ser armazenada no objeto `pal`.

c) Crie uma função chamada `baseCalendario` que recebe 2 argumentos a seguir: `stats` (`tibble` com resultados calculados na questão 4) e `cia` (sigla da Cia. Aérea de interesse). A função deverá:

   i. Criar um subconjunto de `stats` de forma a conter informações de atraso e data apenas da Cia. Aérea dada por cia.

   ii. Para o subconjunto acima, montar a base do calendário, utilizando `ggcal(x,y)`. Nesta notação, `x` representa as datas de interesse e `y`, os percentuais de atraso para as datas descritas em `x`.

   iii. Retornar para o usuário a base do calendário criada acima.

d) Executar a função `baseCalendario` para cada uma das Cias. Aéreas e armazenar os resultados, respectivamente, nas variáveis: `cAA`, `cDL`, `cUA` e `cUS`.

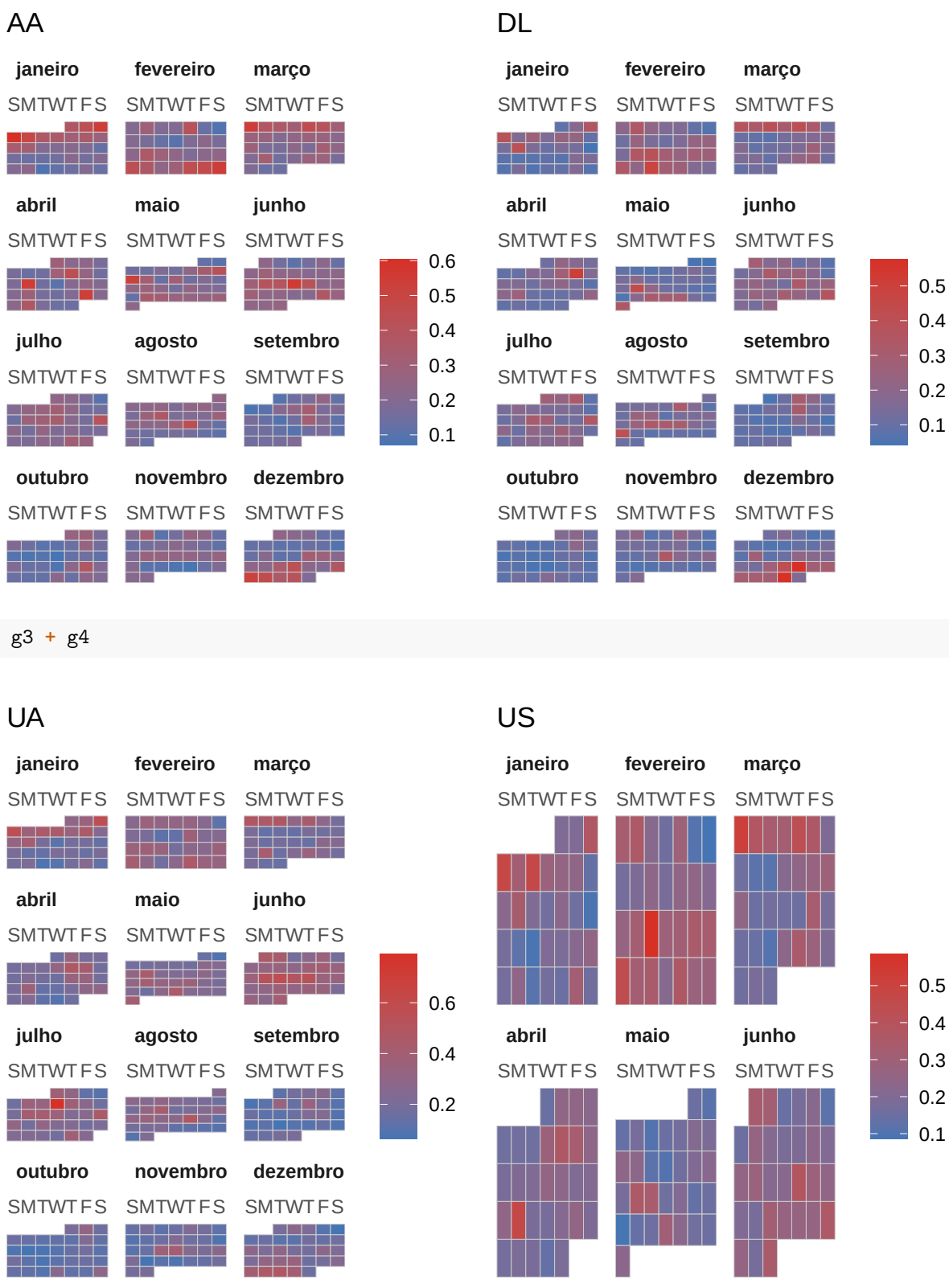
e) Para cada uma das Cias. Aéreas, apresente o mapa de calor respectivo utilizando a combinação de camadas do `ggplot2`. Lembre-se de adicionar um título utilizando o comando `ggtitle`. Por exemplo, `cXX + pal + ggtitle(`Titulo')`.

```
#! CHUNK DESATIVADA
## Instala dependências
   install.packages("vroom")
   install.packages("ggplot2")
   devtools::install_github("jayjacobs/ggcal")
   install.packages("patchwork")
```

```
## Inicializa dependências
   library(vroom)
   library(ggplot2)
   library(ggcal)
   library(patchwork)
```

```
## Configura calendário
   pal = scale_fill_gradient(low="#4575b4", high="#d73027")
   baseCalendario = function(stats, cia){
     input = stats %>% filter(AIRLINE==cia)
     ggcal(input$DATE, input$Perc)
   }

   g1 = baseCalendario(in4, "AA") + pal + ggtitle("AA")
   g2 = baseCalendario(in4, "DL") + pal + ggtitle("DL")
   g3 = baseCalendario(in4, "UA") + pal + ggtitle("UA")
   g4 = baseCalendario(in4, "US") + pal + ggtitle("US")
```

```
## Imprime calendário
   g1 + g2
```




```
g3 + g4
```

Figura 1: Mapa de calor dos vôos atrasados das Cias. aéereas AA, DL, UA e US, respectivamente

Resultados Extras:

Tabela 2: Média de vôos atrasados por Cia estudada.

```
in3 %>% my_kbl()
```

AIRLINE	Perc
AA	0.2161684
DL	0.1630393
UA	0.2384739
US	0.2281244

Tabela 3: Média de vôos atrasados por Cia estudada e por dia.

```
in4 %>%
  head(10) %>%
  my_kbl() %>%
  footnote(general_title="1-10 de 1276
  ↪  linhas", general="")
```

AIRLINE	DATE	Perc
AA	2015-01-01	0.3727339
AA	2015-01-02	0.4303714
AA	2015-01-03	0.5382456
AA	2015-01-04	0.6021433
AA	2015-01-05	0.4912869
AA	2015-01-06	0.3656934
AA	2015-01-07	0.3549075
AA	2015-01-08	0.3074753
AA	2015-01-09	0.3422724
AA	2015-01-10	0.2797147

1-10 de 1276 linhas

§ Python

```
## Instala dependências
   install.packages("reticulate")
```

```
## Inicializa dependências
   library(reticulate)
```

```
## Configura Python
   venv_python = "temp.venv"

   virtualenv_create(venv_python)
```

```
## Using Python: C:/Users/ra260181/AppData/Local/Programs/Python/Python312/python.exe
## Creating virtual environment "temp.venv" ...
## Done!
## Installing packages: pip, wheel, setuptools
## Installing packages: numpy
## Virtual environment 'temp.venv' successfully created.
```

```
   use_virtualenv(venv_python)
   py_install("pandas", envname = venv_python)
```

```
## Using virtual environment "temp.venv" ...
```

```
   py_install("calplot", envname = venv_python)
```

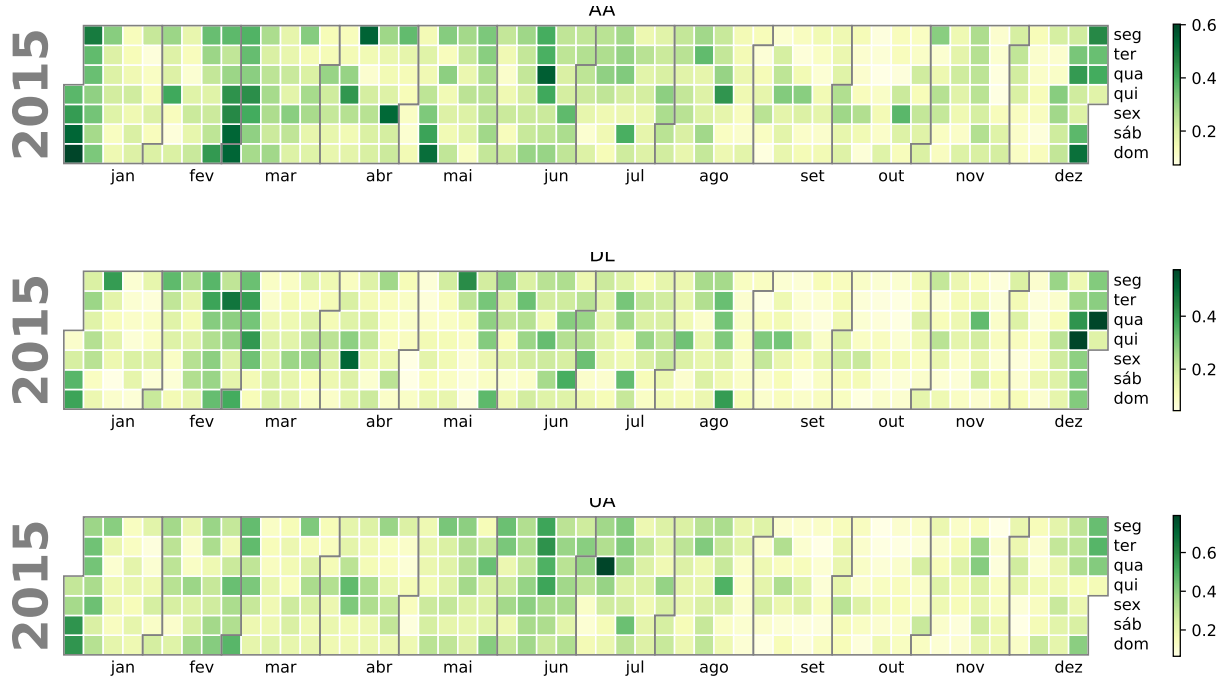
```
## Using virtual environment "temp.venv" ...
```

```
def getStats(cd,lista_chunck):
    cias = ['AA','DL','UA','US']
    cd_filtro = cd[['DAY','MONTH','YEAR','AIRLINE','ARRIVAL_DELAY']]
    cd_filtro = cd_filtro[cd_filtro['AIRLINE'].isin(cias)]
    cd_filtro = cd_filtro.dropna(subset=['ARRIVAL_DELAY'])
    cd_filtro['late'] = cd_filtro['ARRIVAL_DELAY'] > 10
    lista_chunck.append(cd_filtro)
```

```
def computeStats(cd):
    cd_filtro = cd
    cd_filtro['date'] =
↪   pd.to_datetime(cd_filtro[['YEAR','MONTH','DAY']],format="%Y-%m-%d")
    cd_filtro = cd_filtro[['date','AIRLINE','ARRIVAL_DELAY','late']]
    x = cd_filtro.groupby(['AIRLINE','date']).agg(
      late=('late','sum'),
      N=('AIRLINE','count')
    ).reset_index()
    x['Perc'] = x['late']/x['N']
    x = x[['AIRLINE','date','Perc']]
    return x
```

```
## A única grande diferença está na forma de lidar com a lógica de pertencimento ao
↪   subconjunto de Cias. "['AA','DL','UA','US']", que é mais simples do que em R, onde o
↪   operador "|" (ou) é usado repetidas vezes.
```

```python
# Bibliotecas
import matplotlib.pyplot as plt
import calplot
import pandas as pd

# Atribui o endereço de máquina do CD a uma variável
arquivo_csv = 'flights.csv.zip'

# Define o tamanho de cada "chunck" a ser lido
tamanho_chunck = 100000

# Lista para armazenar os subconjuntos processados
subconjuntos_processados = []

# Cria um objeto leitor que vai iterar sobre o CD
leitor_csv = pd.read_csv(arquivo_csv, chunksize=tamanho_chunck)

# Itera sobre cada chunck do CD
for chunck in leitor_csv:
    getStats(chunck,subconjuntos_processados)

# Concatena todos os subconjuntos em um único DataFrame
df_final = pd.concat(subconjuntos_processados, ignore_index=True)

# Computa estatísticas
stats = computeStats(df_final)
```



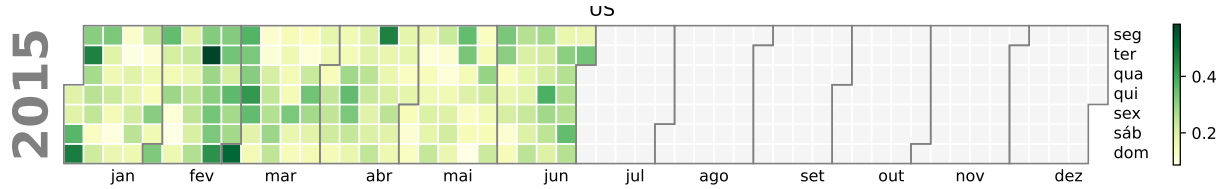


Figura 2: Mapa de calor dos vôos atrasados das Cias. aéreas AA, DL, UA e US, respectivamente. (Obtida através do Python)

§ Resultados e comparações

Tabela 4: Média de vôos atrasados por Cia estudada. (Obtida através do Python)

```
in5 = py$stats
in5 %>%
  head(10) %>%
  my_kbl() %>%
  footnote(general_title="1-10 de 1276
  ↪  linhas", general="")
```

AIRLINE	date	Perc
AA	2014-12-31 22:00:00	0.3727339
AA	2015-01-01 22:00:00	0.4303714
AA	2015-01-02 22:00:00	0.5382456
AA	2015-01-03 22:00:00	0.6021433
AA	2015-01-04 22:00:00	0.4912869
AA	2015-01-05 22:00:00	0.3656934
AA	2015-01-06 22:00:00	0.3549075
AA	2015-01-07 22:00:00	0.3074753
AA	2015-01-08 22:00:00	0.3422724
AA	2015-01-09 22:00:00	0.2797147

1-10 de 1276 linhas

Tabela 3: Média de vôos atrasados por Cia estudada e por dia.

```
in4 %>%
  head(10) %>%
  my_kbl() %>%
  footnote(general_title="1-10 de 1276
  ↪  linhas", general="")
```

AIRLINE	DATE	Perc
AA	2015-01-01	0.3727339
AA	2015-01-02	0.4303714
AA	2015-01-03	0.5382456
AA	2015-01-04	0.6021433
AA	2015-01-05	0.4912869
AA	2015-01-06	0.3656934
AA	2015-01-07	0.3549075
AA	2015-01-08	0.3074753
AA	2015-01-09	0.3422724
AA	2015-01-10	0.2797147

1-10 de 1276 linhas

```
  virtualenv_remove(venv_python)
```

```
## Virtual environment "temp.venv" removed.
```