Fundação Getúlio Vargas
Escola de Matemática Aplicada

Ciência de Dados e Inteligência Artificial

Relatório de Introdução à Computação

Leonardo Alexandre da Silva Ferreira
Sillas Rocha da Costa

Rio de Janeiro
Junho / 2023

Leonardo Alexandre da Silva Ferreira
Sillas Rocha da Costa

Relatório da Avaliação 2 de Introdução à Computação

Relatório da avaliação 2 da disciplina Introdução à Computação do curso Ciência de Dados e Inteligência Artificial

Professor:
Flávio Codeço Coelho

Rio de Janeiro
Junho / 2023

Sumário

1 Introdução

O relatório tem o objetivo de apresentar a trabalho realizado sobre os casos da doença de Chagas, no Brasil em 2019, utilizados para análise laboratorial. Essa pesquisa utiliza 10 questionamentos como base, todos propostos pelo professor Flávio Codeço.

No trabalho, os dados foram coletados por meio da biblioteca PySUS. Após a coleta, com o auxílio da biblioteca Pandas, os registros da doença começaram a ser estudados a partir dos questionamentos. Assim, tornou-se possível a conclusão do trabalho sobre a doença.

2 O SINAN

O Sistema de Informação de Agravos de Notificação (SINAN), desenvolvido no início da década de 90, é um sistema cujo o intuito é coletar, transmitir e disseminar casos de doenças e agravos presentes na lista nacional de doenças de notificação compulsória.

2.1 Coleta de Dados

O sistema é alimentado, principalmente, pela notificação e investigação de casos de doenças e agravos que constam da lista nacional de notificação compulsória de doenças, agravos e eventos de saúde pública.

2.2 PySUS

O PySUS oferece a possibilidade de baixar, gratuitamente, registros de casos individuais selecionados para investigação laboratorial adicional. Além disso, a biblioteca possui uma coleção de códigos auxiliares para análises de dados do Sistema Único de Saúde (SUS).

3 A Doença de Chagas

A doença de Chagas, também chamada de tripanossomíase americana, é uma infecção causada pelo protozoário Trypanosoma cruzi, o que causa sintomas, como febre, dor de cabeça e fraqueza intensa.

3.1 Diagnóstico

O diagnóstico da doença de Chagas é realizado por um médico com base na fase da doença, nos sintomas apresentados, na presença de fatores epidemiológicos compatíveis e em exames laboratoriais.

3.2 Tratamento

O tratamento da doença deve ser indicado por um médico após a confirmação da doença, o que utiliza normalmente o remédio chamado Benznidazol. Em casos raros, pode haver intolerância ao Benznidazol. Nessa situação, o Nifurtimox é disponibilizado como alternativa de tratamento.

3.3 Possíveis Complicações

Caso a doença de Chagas evolua da fase aguda para a fase crônica, diversos problemas cardíacos e digestivos podem ser desencadeados, como insuficiência cardíaca e desnutrição.

4 Atividades

```
# Dados necessários para as questões:
estados_id = {"11": "RO", "12": "AC", "13" :"AM", "14": "RR", "15": "PA",
"16": "AP", "17": "TO","21": "MA", "22": "PI", "23": "CE", "24": "RN",
"25": "PB", "26": "PE", "27": "AL", "28": "SE", "29": "BA","31": "MG",
"32": "ES", "33": "RJ", "35": "SP","41": "PR", "42": "SC", "43": "RS",
"50": "MS", "51": "MT", "52": "GO", "53": "DF"}

qnt_municipios = {"RO": 52, "AC": 22, "AM": 62, "RR": 15, "PA": 144, "AP": 16,
"TO": 139,"MA": 217,"PI": 224, "CE": 184, "RN": 167, "PB": 223, "PE": 184,
"AL": 102, "SE": 75,"BA": 417,"MG": 853,"ES": 78, "RJ": 92, "SP": 645,
"PR": 399, "SC": 295, "RS": 497,"MS": 79,"MT": 141, "GO": 246, "DF": 1}

# Funções de conversão para as questões:
def converter_id_index_estado_silgas(serie_de_ids):
    lista_estados = []
    # Aqui iremos alterar cada ID de estado pela sua sigla no dicionário
    "estados_id":
    for id in serie_de_ids.index:
        id = estados_id[str(id)]
        lista_estados.append(id)
    serie_de_ids.index = lista_estados

    # Então, retornaremos uma série pandas com as siglas convertidas:
    return serie_de_ids

def converter_datas(data_base, coluna_conversao, nome_nova_coluna):
    novas_datas = []
    # Ao converter elemento a elemento e adicioná-los a uma lista, teremos o
    seguinte resultado:
    for cada_data in data_base[coluna_conversao]:
        cada_data = str(cada_data)
        nova_data = cada_data[0:4] + "-" + cada_data[4:6] + "-" + cada_data[6:]
        novas_datas.append(nova_data)

    # Por fim, definindo essa lista como os dados da nova coluna:
```

```
    data_base[nome_nova_coluna] = pd.Series(novas_datas)
    # E converteremos essa coluna para o formato data:
    data_base[nome_nova_coluna] = pd.to_datetime(data_base[nome_nova_coluna],
    format = "%Y/%m/%d")

    return data_base

# Para a realização da nossa análise, faremos uma decodificação das idades:

from pysus.preprocessing.decoders import decodifica_idade_SINAN

chagas_2019["IDADE"] = decodifica_idade_SINAN(chagas_2019.NU_IDADE_N)
```

4.1 Exercício 1

Quantos registros existem no arquivo de dados? O resultado deve ser um número inteiro.

```
    def questao_1(datapath):
        # Aqui, há apenas uma contagem de uma coluna obrigatória a ser
        preenchida, o que retornará o número de ocorrências.
        dados = pd.read_csv(datapath, low_memory = False)
        resultado = int(dados["TP_NOT"].count())

        return resultado
```

```
    4482
```

4.2 Exercício 2

Quantos registros existem por município? a função deve retornar uma série do pandas (pd.Series)

```
    def questao_2(datapath):
```

```
    # Nesta questão, haverá uma contagem dos valores por ID de município,
    que retornará uma série do Pandas.
    dados = pd.read_csv(datapath, low_memory = False)
    resultado = dados["ID_MUNICIP"].value_counts()

    return resultado
```

```
150010    481
160030    352
150140    296
150210    263
170210    180
         ...
293015      1
290660      1
291320      1
291390      1
261220      1
Name: ID_MUNICIP, Length: 550, dtype: int64
```

4.3 Exercício 3

Qual sexo possui mais registros? Retorne uma tupla com o sexo mais numeroso e um dicionário tendo como chaves 'M' e 'F' para os sexos com a contagem de registros em cada sexo.

```
def questao_3(datapath):
    # Nesta questão, haverá a contagem de valores por gênero dos dados, e,
    então, eles serão atrelados a duas variáveis.
    dados = pd.read_csv(datapath, low_memory = False)
    generos = dados["CS_SEXO"].value_counts()
    masc = int(generos["M"])
    fem = int(generos["F"])

    # Depois haverá uma verificação de qual é o maior.
    if fem > masc:
        maior = "F"
```

```
    else:
        maior = "M"

    return (maior), {"M": masc, "F": fem}
```

```
('M', {'M': 2340, 'F': 2142})
```

4.4 Exercício 4

Qual a idade média (em anos) dos registros? retorne um float.

```
def questao_4(datapath):
    # Nesta questão, faremos a média dos valores de idades já convertidos
    anteriormente.
    dados = pd.read_csv(datapath, low_memory = False)
    resultado = float(dados["IDADE"].mean())

    return resultado
```

```
34.129613685589646
```

4.5 Exercício 5

a coluna SG_UF_NOT contém a sigla (string, por exemplo 33: 'RJ') da unidade federativa. Qual a unidade federativa com mais registros? Retorne o resultado como um dicionário tendo como chaves as siglas das unidades federativas e a quantidade de registros.

```
def questao_5(datapath):
    # Nesta questão, haverá uma contagem de ocorrência para cada estado.
    dados = pd.read_csv(datapath, low_memory = False)
    estados = dados["SG_UF_NOT"].value_counts()
    resultado = dict()
```

```
    # Após isso, haverá a troca dos IDs dos estados pelas suas siglas.
    converter_id_index_estado_silgas(estados)

    # E, por fim, tudo será convertido para um dicionário.
    for sigla in estados.index:
        resultado[str(sigla)] = int(estados[sigla])

    return resultado
```

```
{'PA': 2044, 'AP': 520, 'MG': 423, 'BA': 263, 'TO': 238, 'PE': 135, 'PB': 85, 'ES': 85, 'AC': 82, 'PI': 73, 'AM': 6
```

4.6 Exercício 6

Novamente usando a coluna SG_UF_NOT, qual a unidade federativa com mais registros de pessoas do sexo masculino? Retorne o resultado como um dicionário tendo como chaves as siglas das unidades federativas e a quantidade de registros.

```
def questao_6(datapath):
    # Nesta questão, haverá uma contagem de ocorrência para cada estado,
    mas, desta vez, com a filtragem por gênero masculino.
    # De resto, será tudo igual à questão 5.
    dados = pd.read_csv(datapath, low_memory = False)
    estados_filtro = dados["SG_UF_NOT"][dados["CS_SEXO"] == "M"].value_counts()
    resultado = dict()

    converter_id_index_estado_silgas(estados_filtro)

    for sigla in estados_filtro.index:
        resultado[str(sigla)] = int(estados_filtro[sigla])

    return resultado
```

```
{'PA': 1059, 'AP': 281, 'MG': 206, 'TO': 144, 'BA': 133, 'PE': 65, 'ES': 50, 'AC': 44, 'AM': 36, 'PB': 35, 'PI': 33
```

4.7 Exercício 7

Descubra quantos municípios existem em cada unidade federativa (UF) (busque no google). Determine, para a sua tabela de dados, que proporção dos munícípios de cada UF, tem pelo menos um registro. Retorne o resultado como um dicionário tendo como chaves as siglas das unidades federativas e a proporção de municípios com pelo menos um registro.

```
def questao_7(datapath):
    dados = pd.read_csv(datapath, low_memory = False)
    casos_nots = dict()
    # Iremos obter apenas os IDs dos municípios e colocá-los em uma lista.
    # Sendo que haverá apenas a contagem de municípios que não se repetem.
    casos_id = []
    for caso in dados["ID_MUNICIP"].unique():
        municips = str(caso)[:2]
        casos_id.append(municips)

    # Agora, converteremos esta lista para uma série do Pandas e faremos a
    contagem dos valores:
    casos_por_municip = pd.Series(casos_id).value_counts()
    # Então, trocaremos seus indexes de IDs para siglas:
    converter_id_index_estado_silgas(casos_por_municip)

    # Por fim, calcularemos a proporção para cada estado.
    # Dos municípios que registraram infecção em relação ao total de municípios
    do estado.
    for estado in casos_por_municip.index:
        proporcao = int(casos_por_municip[estado]) / int(qnt_municipios[estado])
        # Deste modo, gerando o dicionário desejado.
        casos_nots[estado] = proporcao

    return casos_nots
```

```
{'BA': 0.17266187050359713, 'MG': 0.08440797186400938, 'PA': 0.4305555555555556, 'ES': 0.3717948717948718, 'RS': 0.(
```

4.8 Exercício 8

Usando o comando pd.to_datetime do Pandas, crie uma nova coluna chamada DT_NOTIFICACAO com o tipo datetime64[ns] a partir da coluna DT_NOTIFIC, e uma coluna DT_SINTOMAS também de tipo datetime64[ns]. A partir destas duas novas colunas calcule o número de dias de atraso entre os sintomas e a notificação e salve em uma coluna ATRASO_NOT. Retorne um dataframe com as colunas DT_NOTIFICACAO, DT_SINTOMAS e ATRASO_NOT.

```
def questao_8(datapath):
    # Para esta questão, precisaremos converter inteiros para o formato aceito
    pelo ‘pd.to_datetime‘, que é: "YYYY-mm-dd".
    dados = pd.read_csv(datapath, low_memory = False)

    # Verificaremos se os dados estão no formato "YYYYmmdd" e não no formato
    "YYYY-mm-dd":
    if len(str(dados["DT_NOTIFIC"].iloc[0])) == 8:

        # Então, converteremos as datas se for necessário:
        converter_datas(dados, "DT_NOTIFIC", "DT_NOTIFICACAO")
    # Se já estiverem no formato solicitado, apenas transição para datas:
    else:
        dados["DT_NOTIFICACAO"] = pd.to_datetime(dados["DT_NOTIFIC"],
        format = "%Y/%m/%d")

    # Faremos o mesmo com a coluna "DT_SIN_PRI" que possui a data da aparição
    dos sintomas:
    if len(str(dados["DT_SIN_PRI"].iloc[0])) == 8:
        converter_datas(dados, "DT_SIN_PRI", "DT_SINTOMAS")
    else:
        dados["DT_SINTOMAS"] = pd.to_datetime(dados["DT_SIN_PRI"],
        format = "%Y/%m/%d")

    dados["ATRASO_NOT"] =  dados["DT_NOTIFICACAO"] - dados["DT_SINTOMAS"]

    # Agora, converteremos o atraso em dias e faremos o dataframe requerido.
    dados["ATRASO_NOT"] = dados["ATRASO_NOT"].dt.days
```

```
    datas_not = dados[["DT_NOTIFICACAO", "DT_SINTOMAS", "ATRASO_NOT"]]

    return datas_not
```

	DT_NOTIFICACAO	DT_SINTOMAS	ATRASO_NOT
0	2019-04-10	2019-03-01	40
1	2019-09-16	2019-08-18	29
2	2019-03-07	2019-02-28	7
3	2019-10-22	2019-09-09	43
4	2019-09-10	2019-08-28	13
...	...	...	...
4477	2019-09-05	2019-09-04	1
4478	2019-09-26	2019-08-26	31
4479	2019-01-17	2019-01-05	12
4480	2019-07-03	2019-07-03	0
4481	2019-11-08	2019-08-04	96

[4482 rows x 3 columns]

4.9 Exercício 9

Calcule a média e desvio padrão do atraso de notificação por unidade federativa. Retorne o resultado como um dicionário tendo como chaves as siglas das unidades federativas e a média e desvio padrão do atraso de notificação. Exemplo: {'RJ': (média, desvio padrão),...}.

```
def questao_9(datapath):
    # Para esta questão, iremos reutilizar os resultados da questão 8 e
    adicionar a coluna "ATRASO_NOT" à nova base de dados.
    dados = pd.read_csv(datapath, low_memory = False)
    datas_not = questao_8(datapath)
    estados_silga = []
    resultado = dict()

    # Agora, adicionaremos a coluna "SG_UF_NOT" à nossa tabela de atraso
    de notificações:
```

```
    datas_not["SG_UF_NOT"] = dados["SG_UF_NOT"]

    # Agora, iremos converter os IDs para as siglas diretamente.
    for id in datas_not["SG_UF_NOT"]:
        id = estados_id[str(id)]
        estados_silga.append(id)

    # E transferí-las para a coluna "SG_UF_NOT".
    datas_not["SG_UF_NOT"] = pd.Series(estados_silga)

    # Por fim, iremos percorrer a lista de siglas dos estados:
    for estado in list(qnt_municipios.keys()):
        # Filtrar as datas que procuramos para cada estado:
        filtro = datas_not["ATRASO_NOT"][datas_not["SG_UF_NOT"] == estado]
        # Calcular a média e desvio padrão.
        media_estado = float(filtro.mean())
        desvio_estado = float(filtro.std())
        # E por fim, adicionar tudo ao dicionário Resultado.
        resultado[estado] = (media_estado, desvio_estado)

    return resultado
```

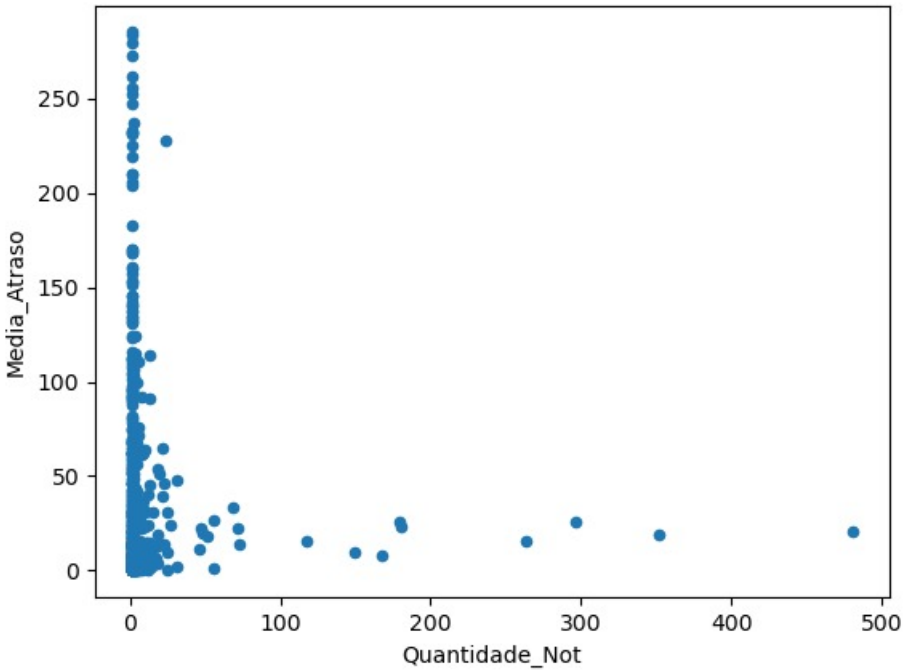
```
{'RO': (23.395833333333332, 57.050086088410374), 'AC': (32.073170731707314, 50.99522623211233), 'AM': (19.159420289
```

4.10 Exercício 10

Calcule a média do atraso de notificação por município. Plote o número de notificações (eixo x) contra o atraso médio (eixo y) em cada município como um scatter plot. Retorne o resultado como uma séries do Pandas com as médias por município.

```
def questao_10(datapath):
    dados = pd.read_csv(datapath, low_memory = False)
    datas_not = questao_8(datapath)
    lista_ids = []
    medias_id = []
```

```
quantidade_nots = []

# Passaremos os IDs dos municípios para a tabela inicial.
dados["ID_MUNICIP"] = dados["ID_MUNICIP"].astype(str)

# Agora, adicionaremos o atraso à tabela inicial.
dados["ATRASO_NOT"] = datas_not["ATRASO_NOT"]

# Para cada município, iremos obter os dados procurados:
for municipio in dados["ID_MUNICIP"].unique():
    lista_ids.append(municipio)
    # Atrasos por município:
    atrasos = dados["ATRASO_NOT"][dados["ID_MUNICIP"] == municipio]
    # Média de atraso por município:
    medias_id.append(int(atrasos.mean()))
    # Contagem de notificações filtrada por município:
    quantidade_nots.append(int(atrasos.count()))

# E, assim, criar um Data Frame com estas informações:
notifics = pd.DataFrame()
notifics["Media_Atraso"] = medias_id
notifics["Quantidade_Not"] = quantidade_nots
notifics.index = lista_ids

# Além de fazer a plotagem desejada:
notifics.plot.scatter("Quantidade_Not",
                      "Media_Atraso")

plt.show()

return notifics["Media_Atraso"]
```




```
160030    19
160060    10
150140    26
150080    26
160040     4
          ..
260320     2
261580     9
261480     6
260820     0
261220    96
Name: Media_Atraso, Length: 550, dtype: int64
```

5 Conclusão

A quantidade de registros do sexo masculino é, levemente, maior que o número de casos do sexo feminino, uma diferença aproximada de 10%. Sob outra perspectiva, há grandes discrepâncias dos totais de infectados selecionados por unidade federativa. A região norte possui mais de 50% dos afetados pela doença de Chagas, com destaque para o Pará. Por fim, vale destacar os atrasos de notificação, pois diversos números são maiores que uma centena, os quais representam retardos de meses.