Relatório - A2 - Introdução à Computação

Daniel de Miranda Almeida e Luis Felipe de Abreu Marciano

26 de junho de 2023

Sumário

1 Introdução **3**

2 Código **3**
2.1 Decodifica estados 3
2.2 Questão 1 3
2.3 Questão 2 3
2.4 Questão 3 4
2.5 Questão 4 4
2.6 Questão 5 5
2.7 Questão 6 5
2.8 Questão 7 6
2.9 Questão 8 7
2.10 Questão 9 7
2.11 Questão 10 8

1 Introdução

A base de registros de doença escolhida para esse projeto foi a de registros de casos de Raiva Humana para o ano de 2021.

2 Código

2.1 Decodifica estados

Criamos a função "decodifica_estados"no arquivo python a2.py para na questões em que é criada uma Series ou um Dataframe mudar os índices com o código do estado para a sigla do estado. A função tem um dicionário e chama a função rename do pandas para mudar os índices do dataframe/series de acordo com o dicionário.

```
def decodifica_estados(df):

    dicionario_estados = {12: 'AC', 27: 'AL', 16: 'AP', 13: 'AM',
    29: 'BA', 23: 'CE', 53: 'DF', 32: 'ES', 52: 'GO', 21: 'MA', 51:
     'MT', 50: 'MS', 31: 'MG', 15: 'PA', 25: 'PB', 41: 'PR', 26: '
    PE', 22: 'PI', 24: 'RN', 43: 'RS', 33: 'RJ', 11: 'RO', 14: 'RR'
    , 42: 'SC', 35: 'SP', 28: 'SE', 17: 'TO'}
    df.rename(index = dicionario_estados, inplace = True)

    return df
```

2.2 Questão 1

Quantos registros existem no arquivo de dados?

Como cada registro no csv é uma coluna, basta pegarmos o comprimento do índice do dataframe com esse código:

```
def questao_1(datapath):
    # recebendo database
    df = pd.read_csv(datapath)
    # retornando quantidade de linhas
    return len(df.index)
```

A função retorna um número inteiro com o número de linhas do dataframe:

*Retorno da função:*198

2.3 Questão 2

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

Para encontrar o número de registros por município usamos o método groupby na coluna ID_MUNICIP com count na coluna ID_AGRAVO para contar o número de ocorrências de ID_AGRAVO para cada municíio no dataframe (foi escolhida a coluna ID_AGRAVO por que ela sempre será não nula para todos os registros, uma vez que é o código do agravo):

```
def questao_2(datapath):
    df = pd.read_csv(datapath)
    return df.groupby(["ID_MUNICIP"])["ID_AGRAVO"].count()
```

O método groupby já retorna uma series do pandas:

Retorno da função: ID_MUNICIP
110160 1
150140 1
160010 6
171620 1
210320 1
..
522140 1
522170 1
522190 1
522205 3
530010 2
Name: ID_AGRAVO, Length: 94, dtype: int64

2.4 Questão 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 con a contagem de registros em cada sexo.

No código fizemos um agrupamento por sexo fazendo novamente a contagem de ID_AGRAVO como forma de contar todos os casos para cada sexo. Essa series foi transformada em dicionário pelo método *dict*. Para achar o sexo com maior número de casos usamos a função *max*, que recebe um dicionário e as chaves do dicionário para retornar a chave que tem o maior valor dentro do dicionário.

```
def questao_3(datapath):
    df = pd.read_csv(datapath)

    dicionario_sexo = dict(df.groupby(["CS_SEXO"])["ID_AGRAVO"].
    count())
    sexo_mais_frequente = max(dicionario_sexo, key=dicionario_sexo.
    get)

    return (sexo_mais_frequente, dicionario_sexo)
```

O retorno da função é a tupla com o índice do sexo mais frequente no dicionário e o dicionário em si:

Retorno da função:('M', 'F': 94, 'M': 104)

2.5 Questão 4

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

A função simplesmente usa o método mean do Pandas que calcula a média da soma dos valores em uma coluna do dataframe

```
def questao_4(datapath):
    df = pd.read_csv(datapath)

    return df["idade_anos"].mean()
```

O retorno da função é o retorno do método média do Pandas, um float:

Retorno da função: 34.3598899958489

2.6 Questão 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.

O código abaixo cria uma series com as contagens de casos para cada unidade federativa e usa a função *decodifica_estados* para transformar os índices da series de códigos de unidade federativa na sigla correspondente.

```
def questao_5(datapath):
    df = pd.read_csv(datapath)

    series_estados = df.groupby(["SG_UF_NOT"])["ID_AGRAVO"].count()

    series_estados = decodifica_estados(series_estados)

    return dict(series_estados)
```

O retorno da função é um dicionário feito a partir da series:

Retorno da função: {'RO': 1, 'PA': 1, 'AP': 6, 'TO': 1, 'MA': 16, 'PI': 13, 'RN': 8, 'PE': 3, 'BA': 1, 'MG': 57, 'RJ': 6, 'SP': 26, 'PR': 3, 'SC': 5, 'RS': 3, 'MS': 2, 'MT': 26, 'GO': 18, 'DF': 2}

2.7 Questão 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.

O código abaixo cria o dataframe *series_estados* que possui somente as linhas que tiverem M na coluna *CS_SEXO*. Após, agrupamos por estado e contamos o número de ocorrências do agravo. Usamos a função *decodifica_estados* para transformar os índices da series de códigos de unidade federativa na sigla correspondente. E, por fim, no retorno usamos a função *dict* para transformar a series em dicionário.

```python
def questao_6(datapath):
    df = pd.read_csv(datapath)

    series_estados = df[df.CS_SEXO == "M"]

    series_estados = series_estados.groupby(["SG_UF_NOT"])["
    ID_AGRAVO"].count()

    series_estados = decodifica_estados(series_estados)

    return dict(series_estados)
```

O retorno da função é um dicionário feito a partir da series.

Retorno da função: {'RO': 1, 'PA': 1, 'AP': 2,'MA': 6, 'PI': 6, 'RN': 2, 'PE': 3, 'BA': 1, 'MG': 31, 'RJ': 4, 'SP': 17, 'PR': 2, 'SC': 3, 'RS': 1, 'MT': 11, 'GO': 11, 'DF': 2}

2.8 Questão 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.

O código abaixo cria um dicionário com os estados como chave e o número de múnicipios como valores. Agrupamos o dataframe por estados e contamos os valores únicos na coluna *ID_MUNICIP*. Após, utilizamos a função *decodifica_estados* para tranformar os números dos estados em suas respectivas siglas. Tiramos o Distrito Federal, pois não há municípios nele. Criamos a coluna *n_municipios* a partir de um list comprehension, que, para cada sigla dos estados pegava o valor correspondente no dicionário de número de municípios. Criamos outra coluna, *proporcao_municipios*, feita a partir da operação entre duas outras colunas. E, por fim, como retorno, temos um dicionário feito a partir de um zip das siglas dos estados e da coluna de proporção.

```python
def questao_7(datapath):
    df = pd.read_csv(datapath)

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

    df_contagem_estados = pd.DataFrame(df.groupby(["SG_UF_NOT"])["
    ID_MUNICIP"].nunique())

    df_contagem_estados = decodifica_estados(df_contagem_estados)

    df_contagem_estados.drop("DF", inplace = True)

```

```
    df_contagem_estados["n_municipios"] = [
    dicionario_municipios_estados.get(index) for index in
    df_contagem_estados.index]

    df_contagem_estados["proporcao_municipios"] = round((
    df_contagem_estados["ID_MUNICIP"]/df_contagem_estados["
    n_municipios"] * 100))

    return dict(zip(df_contagem_estados.index, df_contagem_estados[
    "proporcao_municipios"]))
```

O retorno da função é um dicionário.

Retorno da função: {'RO': 2.0, 'PA': 1.0, 'AP': 6.0, 'TO': 1.0, 'MA': 1.0, 'PI': 4.0, 'RN': 3.0, 'PE': 1.0, 'BA': 0.0, 'MG': 2.0, 'RJ': 7.0, 'SP': 2.0, 'PR': 1.0, 'SC': 2.0, 'RS': 1.0, 'MS': 3.0, 'MT': 5.0, 'GO': 4.0}

2.9 Questão 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.

No código abaixo, utilizamos o método *to_datetime()* para transformar em data as colunas *DT_NOTIFIC* e *DT_SIN_PRI*. Por fim, foi criada a coluna *ATRASO_NOT*, feita a partir da operação de outras duas colunas, utilizando *np.timedelta64(1,"D")* para que o resultado seja em dias. O retorno é o novo dataframe.

```
def questao_8(datapath):
    df = pd.read_csv(datapath)

    df["data_notificacao"] = pd.to_datetime(df["DT_NOTIFIC"])
    df["data_sintomas"] = pd.to_datetime(df["DT_SIN_PRI"])
    df["ATRASO_NOT"] = (df["data_notificacao"] - df["data_sintomas"
    ]) / np.timedelta64(1, "D")

    return df
```

O retorno da função é um dataframe.

Retorno da função: O novo dataframe.

2.10 Questão 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.

No código abaixo criamos a coluna *ATRASO_NOT* da mesma forma que fizemos na questão anterior. Criamos a series *df_media* que é a média dos

atrasos de notificação, agrupando por estado, e da mesma forma criamos a series *df_desvio*, calulando o desvio padrão. Após decodificamos os codigos dos estados para suas respectivas siglas na series *df_media* e tranformamos as duas series em lista, para que possamos juntar em uma tupla com o *zip*. Por fim, criamos um dicionario com as chaves sendo as siglas dos estados e os valores sendo as tuplas com a média em seu primeiro elemento e o desvio padrão no segundo.

```
def questao_9(datapath):
    df = pd.read_csv(datapath)

    df["data_notificacao"] = pd.to_datetime(df["DT_NOTIFIC"])
    df["data_sintomas"] = pd.to_datetime(df["DT_SIN_PRI"])
    df["ATRASO_NOT"] = (df["data_notificacao"] - df["data_sintomas"
    ]) / np.timedelta64(1, "D")

    df_media = df.groupby(["SG_UF_NOT"])["ATRASO_NOT"].mean()
    df_desvio = df.groupby(["SG_UF_NOT"])["ATRASO_NOT"].std()

    decodifica_estados(df_media)

    lista_medias = list(df_media)
    lista_desvios = list(df_desvio)
    tupla_medias_desvios = tuple(zip(lista_medias, lista_desvios))

    dicionario_medias_desvios = dict(zip(df_media.index,
    tupla_medias_desvios))

    return dicionario_medias_desvios
```

O retorno da função é um dicionário.

Retorno da função: {'RO': (0.0, nan), 'PA': (0.0, nan), 'AP': (0.0, 0.0), 'TO': (0.0, nan), 'MA': (0.75, 1.61245154965971), 'PI': (2.3076923076923075, 4.289223227746009), 'RN': (7.875, 16.374522893812816), 'PE': (2.6666666666666665, 1.5275252316519468), 'BA': (0.0, nan), 'MG': (1.1228070175438596, 4.602290483469681), 'RJ': (3.3333333333333335, 6.055300708194983 5), 'SP': (29.576923076923077, 150.81330792126352), 'PR': (15.333333333333334, 25.69695182961071), 'SC': (1.0, 2.2360679774997894), 'RS': (0.0, 0.0), 'MS': (0.0, 0.0), 'MT': (0.19230769230769232, 0.4914656259988704), 'GO': (2.0555555555555554, 6.557189338896948), 'DF': (1.0, 1.4142135623730951)}

2.11 Questão 10

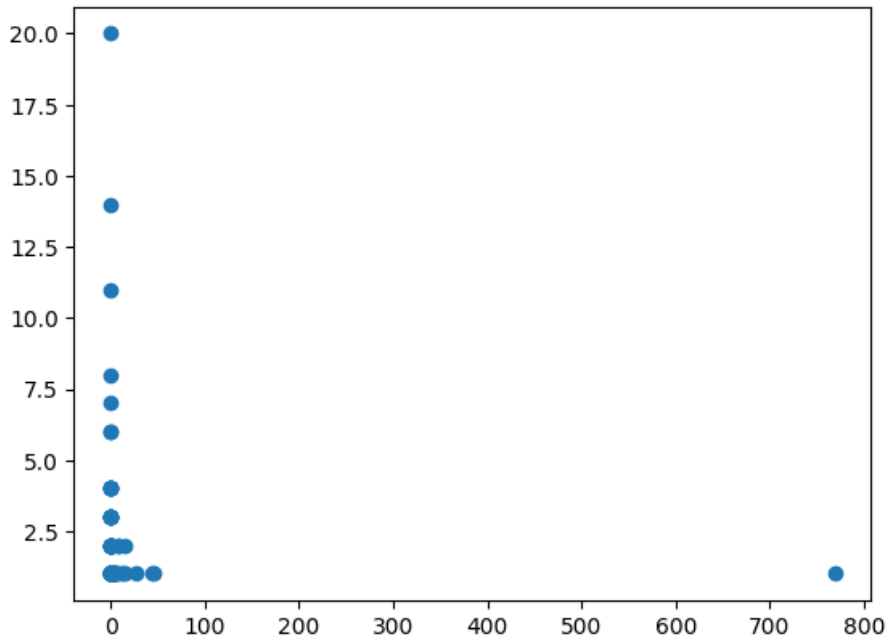
Calcule a média do atraso de notificação por município. Plote o número de notificações contra o atraso médio em cada município como um scatter plot. Retorne o resultado como uma séries do Pandas com as médias por município.

No código abaixo criamos a coluna *ATRASO_NOT* (explicação na questão 8). Criamos uma series agrupando por município e calculando a média de atraso de noticação e transformamos no data frame *df_atraso_municipio*. Criamos a

coluna *n_noticacoes*, que é a contagem de casos doa agravo agrupando por município. A função plota um scatter plot

```
def questao_10(datapath):
    # recebendo database
    df = pd.read_csv(datapath)

    df["data_notificacao"] = pd.to_datetime(df["DT_NOTIFIC"])
    df["data_sintomas"] = pd.to_datetime(df["DT_SIN_PRI"])
    df["ATRASO_NOT"] = (df["data_notificacao"] - df["data_sintomas"
    ]) / np.timedelta64(1, "D")

    # criando dataframe com municipios e a media para cada um
    df_atraso_municipio = pd.DataFrame(df.groupby(["ID_MUNICIP"])["
    ATRASO_NOT"].mean())

    # adicionando coluna com o numero de notificacoes no dataframe
    df_atraso_municipio["n_notificacoes"] = df.groupby(["ID_MUNICIP
    "])["ID_AGRAVO"].count()

    # fazendo gr fico
    fig, plot_municipios = plt.subplots()
    # plotando valores
    plot_municipios.scatter(df_atraso_municipio["ATRASO_NOT"],
    df_atraso_municipio["n_notificacoes"])

    return df_atraso_municipio["ATRASO_NOT"]
```

O retorno da função é uma series do Pandas. E ela faz o plot do seguinte scatter plot:



Retorno da função:
ID_MUNICIP
110160 0.000000
150140 0.000000
160010 0.000000
171620 0.000000
210320 3.000000
...
522140 3.000000
522170 28.000000
522190 1.000000
522205 0.666667
530010 1.000000
Name: ATRASO_NOT, Length: 94, dtype: float64