

# IESB EAD - Probabilidade e estatística

## Caro aluno,

1. Item da lista
2. Item da lista

Pontuação (0,25 Pontos)

- Quantas reclamações foram registradas em 2016, contidas nesta base?
- Qual região houve mais reclamação? E qual houve menos?
- Qual a quantidade de reclamações realizadas por homens? E por mulheres?

Pontuação (0,25 Pontos)

- Em qual região os homens fizeram mais reclamações que as mulheres?
- Qual a faixa etária que realizou mais reclamações?
- Quais assuntos apresentaram mais reclamações em 2016?
- Analisando o nome fantasia, determine qual empresa recebeu mais reclamações em 2016.
- Qual empresa possui um maior percentual de resolução de reclamações?

Pontuação (1 Ponto)

- Monte um gráfico mostrando a quantidade de reclamações por região, separando as mesmas por sexo.
- Elabore um gráfico de linhas com a quantidade de reclamações por mês. Este gráfico lembra alguma distribuição estatística?
- Elabore um gráfico boxplot mostrando a duração das reclamações por região.

Pontuação (0,5 Pontos)

- É possível afirmar que existe correlação entre o número de reclamações E o número de habitantes por Estado? Elabore um gráfico de dispersão e calcule o índice de correlação destes dois fatores.

Todo o trabalho deverá ser feito utilizando notebook Python. Pode ser utilizado a plataforma Google Colaboratory e todos os itens acima deverão ser demonstrados. NÃO SERÃO ACEITAS APENAS AS RESPOSTAS AS PERGUNTAS. O aluno deverá apresentar todos os comandos utilizados para obter as informações solicitadas.

O aluno deverá imprimir o notebook em formato PDF e submeter no campo atividade ativa, juntamente com o Notebook do projeto.

Não serão corrigidos trabalhos com formato diferente de PDF como .ZIP,tar, ou imagens.

In [88]:

```
!|pip install nbconvert
```

```
Requirement already satisfied: nbconvert in /usr/local/lib/python3.7/dist-packages (5.6.1)
Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.7/dist-packages (from nbconvert) (0.8.4)
Requirement already satisfied: jinja2>=2.4 in /usr/local/lib/python3.7/dist-packages (from nbconvert) (2.11.3)
Requirement already satisfied: nbformat>=4.4 in /usr/local/lib/python3.7/dist-packages (from nbconvert) (5.2.0)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.7/dist-packages (from nbconvert) (0.7.1)
Requirement already satisfied: bleach in /usr/local/lib/python3.7/dist-packages (from nbconvert) (4.1.0)
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.7/dist-packages (from nbconvert) (5.1.1)
Requirement already satisfied: jupyter-core in /usr/local/lib/python3.7/dist-packages (from nbconvert) (4.9.2)
Requirement already satisfied: testpath in /usr/local/lib/python3.7/dist-packages (from nbconvert) (0.6.0)
Requirement already satisfied: pygments in /usr/local/lib/python3.7/dist-packages (from nbconvert) (2.6.1)
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.7/dist-packages (from nbconvert) (1.5.0)
Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3.7/dist-packages (from nbconvert) (0.4)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-packages (from jinja2>=2.4->nbconvert) (2.0.1)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in /usr/local/lib/python3.7/dist-packages (from nbformat>=4.4->nbconvert) (4.3.3)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (4.11.3)
Requirement already satisfied: pyrsistent!=0.17.0,!>0.17.1,!>0.17.2,>=0.1.4.0 in /usr/local/lib/python3.7/dist-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (0.18.1)
Requirement already satisfied: importlib-resources>=1.4.0 in /usr/local/lib/python3.7/dist-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (5.4.0)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (3.10.0.2)
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.7/dist-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (21.4.0)
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.7/dist-packages (from importlib-resources>=1.4.0->jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (3.7.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.7/dist-packages (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist-packages (from bleach->nbconvert) (1.15.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packages (from bleach->nbconvert) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging->bleach->nbconvert) (3.0.7)
```

In [89]:

```
!jupyter nbconvert --to html AtividadeAtiva.ipynb
```

[NbConvertApp] WARNING | pattern 'AtividadeAtiva.ipynb' matched no files  
This application is used to convert notebook files (\*.ipynb)  
to various other formats.

WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

## Options

=====

The options below are convenience aliases to configurable class-options,  
as listed in the "Equivalent to" description-line of the aliases.

To see all configurable class-options for some <cmd>, use:

<cmd> --help-all

### --debug

set log level to logging.DEBUG (maximize logging output)  
Equivalent to: [--Application.log\_level=10]

### --show-config

Show the application's configuration (human-readable format)  
Equivalent to: [--Application.show\_config=True]

### --show-config-json

Show the application's configuration (json format)  
Equivalent to: [--Application.show\_config\_json=True]

### --generate-config

generate default config file  
Equivalent to: [--JupyterApp.generate\_config=True]

### -y

Answer yes to any questions instead of prompting.  
Equivalent to: [--JupyterApp.answer\_yes=True]

### --execute

Execute the notebook prior to export.  
Equivalent to: [--ExecutePreprocessor.enabled=True]

### --allow-errors

Continue notebook execution even if one of the cells throws an error and include the error message in the cell output (the default behaviour is to abort conversion). This flag is only relevant if '--execute' was specified, too.

Equivalent to: [--ExecutePreprocessor.allow\_errors=True]

### --stdin

read a single notebook file from stdin. Write the resulting notebook with default basename 'notebook.\*'

Equivalent to: [--NbConvertApp.from\_stdin=True]

### --stdout

Write notebook output to stdout instead of files.

Equivalent to: [--NbConvertApp.writer\_class=StdoutWriter]

### --inplace

Run nbconvert in place, overwriting the existing notebook (only relevant when converting to notebook format)

Equivalent to: [--NbConvertApp.use\_output\_suffix=False --NbConvertApp.export\_format=notebook --FilesWriter.build\_directory=]

### --clear-output

Clear output of current file and save in place,  
overwriting the existing notebook.

Equivalent to: [--NbConvertApp.use\_output\_suffix=False --NbConvertApp.export\_format=notebook --FilesWriter.build\_directory= --ClearOutputPreprocessor.enabled=True]

### --no-prompt

Exclude input and output prompts from converted document.

Equivalent to: [--TemplateExporter.exclude\_input\_prompt=True --TemplateExporter.exclude\_output\_prompt=True]

### --no-input

Exclude input cells and output prompts from converted document.

This mode is ideal for generating code-free reports.

Equivalent to: [--TemplateExporter.exclude\_output\_prompt=True --TemplateExporter.exclude\_input=True]

--log-level=<Enum>  
Set the log level by value or name.  
Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR', 'CRITICAL']  
Default: 30  
Equivalent to: [--Application.log\_level]

--config=<Unicode>  
Full path of a config file.  
Default: ''  
Equivalent to: [--JupyterApp.config\_file]

--to=<Unicode>  
The export format to be used, either one of the built-in formats  
['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf', 'python', 'rst', 'script', 'slides']  
or a dotted object name that represents the import path for an `Exporter` class  
Default: 'html'  
Equivalent to: [--NbConvertApp.export\_format]

--template=<Unicode>  
Name of the template file to use  
Default: ''  
Equivalent to: [--TemplateExporter.template\_file]

--writer=<DottedObjectName>  
Writer class used to write the results of the conversion  
Default: 'FileWriter'  
Equivalent to: [--NbConvertApp.writer\_class]

--post=<DottedOrNone>  
PostProcessor class used to write the results of the conversion  
Default: ''  
Equivalent to: [--NbConvertApp.postprocessor\_class]

--output=<Unicode>  
overwrite base name use for output files.  
can only be used when converting one notebook at a time.  
Default: ''  
Equivalent to: [--NbConvertApp.output\_base]

--output-dir=<Unicode>  
Directory to write output(s) to. Defaults to output to the directory of each notebook. To recover previous default behaviour (outputting to the current working directory) use . as the flag value.  
Default: ''  
Equivalent to: [--FileWriter.build\_directory]

--reveal-prefix=<Unicode>  
The URL prefix for reveal.js (version 3.x).  
This defaults to the reveal CDN, but can be any url pointing to a copy of reveal.js.  
For speaker notes to work, this must be a relative path to a local copy of reveal.js: e.g., "reveal.js".  
If a relative path is given, it must be a subdirectory of the current directory (from which the server is run).  
See the usage documentation

```
(https://nbconvert.readthedocs.io/en/latest/usage.html#revealjs-html-slideshow)
for more details.
Default: ''
Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
The nbformat version to write.
Use this to downgrade notebooks.
Choices: any of [1, 2, 3, 4]
Default: 4
Equivalent to: [--NotebookExporter.nbformat_version]
```

## Examples

-----

The simplest way to use nbconvert is

```
> jupyter nbconvert mynotebook.ipynb
```

which will convert mynotebook.ipynb to the default format (probably HTML).

You can specify the export format with `--to`.

Options include ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf', 'python', 'rst', 'script', 'slides'].

```
> jupyter nbconvert --to latex mynotebook.ipynb
```

Both HTML and LaTeX support multiple output templates. LaTeX includes 'base', 'article' and 'report'. HTML includes 'basic' and 'full'. You can specify the flavor of the format used.

```
> jupyter nbconvert --to html --template basic mynotebook.ipynb
```

b

You can also pipe the output to stdout, rather than a file

```
> jupyter nbconvert mynotebook.ipynb --stdout
```

PDF is generated via latex

```
> jupyter nbconvert mynotebook.ipynb --to pdf
```

You can get (and serve) a Reveal.js-powered slideshow

```
> jupyter nbconvert myslides.ipynb --to slides --post serve
```

Multiple notebooks can be given at the command line in a couple of different ways:

```
> jupyter nbconvert notebook*.ipynb
> jupyter nbconvert notebook1.ipynb notebook2.ipynb
```

or you can specify the notebooks list in a config file, containing::

```
c.NbConvertApp.notebooks = ["my_notebook.ipynb"]
```

```
> jupyter nbconvert --config mycfg.py
```

To see all available configurables, use `--help-all`.

In [44]:

```
#Iniciando a biblioteca pandas
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [46]:

```
#Os dados podem ser importados diretamente da internet, sem a necessidade de ser armazenados no computador Local utilizando os comandos listados abaixo
#Fazendo o upload via google Drive. Para isso será necessário o uso de uma função que foi desenvolvida abaixo
```

```
import requests
from io import StringIO

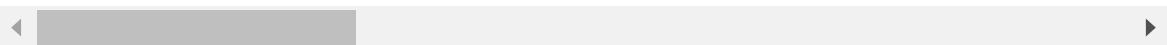
orig_url='https://drive.google.com/file/d/1pkOP40FvztNt0pnWgKMG5iTzRt4TW_N/view?usp=sharing'

file_id = orig_url.split('/')[-2]
dwn_url='https://drive.google.com/uc?export=download&id=' + file_id
url = requests.get(dwn_url).text
csv_raw = StringIO(url)
dfs = pd.read_csv(csv_raw)
(dfs.head())
```

Out[46]:

	AnoCalendario	DataArquivamento	DataAbertura	CodigoRegiao	Regiao	UF	strRazao
0	2016	2016-02-17 13:43:08.000	2015-10-29 10:59:59.000	2	Nordeste	PE	CVC OPERA AGEI VIAGEI
1	2016	2016-12-15 18:11:35.000	2016-11-18 14:52:22.000	2	Nordeste	PE	ARMANI CO
2	2016	2016-09-16 09:01:46.000	2016-06-29 13:58:35.000	2	Nordeste	PE	COM ENEF PERNA
3	2016	2016-02-18 13:30:30.000	2016-01-21 12:23:51.000	2	Nordeste	PE	I
4	2016	2016-11-09 14:20:36.000	2016-10-07 16:39:59.000	2	Nordeste	PE	ASSOCIA UN PAUL ENSINO

5 rows × 23 columns



In [47]:

dfs.info()

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 203486 entries, 0 to 203485
Data columns (total 23 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   i»¿AnoCalendario    203486 non-null   int64  
 1   DataArquivamento   203486 non-null   object  
 2   DataAbertura       203486 non-null   object  
 3  CodigoRegiao        203486 non-null   int64  
 4   Regiao             203486 non-null   object  
 5   UF                203486 non-null   object  
 6   strRazaoSocial     203483 non-null   object  
 7   strNomeFantasia   173722 non-null   object  
 8   Tipo               203486 non-null   int64  
 9   NumeroCNPJ         195292 non-null   float64 
 10  RadicalCNPJ        195292 non-null   float64 
 11  RazaoSocialRFB     187287 non-null   object  
 12  NomeFantasiaRFB   92561  non-null   object  
 13  CNAEPrincipal      187287 non-null   float64 
 14  DescCNAEPrincipal   185784 non-null   object  
 15  Atendida           203486 non-null   object  
 16  CódigoAssunto       203486 non-null   int64  
 17  DescricaoAssunto   203486 non-null   object  
 18  CódigoProblema     203476 non-null   float64 
 19  DescricaoProblema  203476 non-null   object  
 20  SexoConsumidor      203484 non-null   object  
 21  FaixaEtariaConsumidor 203486 non-null   object  
 22  CEPConsumidor       189400 non-null   float64 

dtypes: float64(5), int64(4), object(14)
memory usage: 35.7+ MB

```

## 1. Quantas reclamações foram registradas em 2016, contidas nesta base?

Resposta: 203.485 reclamações, levando em consideração que a primeira linha é descritiva.

\*Explicando o código: len (para verificar o tamanho do dataframe)+dfs(variável)

In [48]:

len(dfs)

Out[48]:

203486

## 2.Qual região houve mais reclamação? E qual houve menos?

Resposta: Região com mais reclamações (Nordeste-66.411)/Região com menos reclamações (Norte-19.202).

\*Explicando o código: dfs (variavel, na qual se encontra a planilha do Excel)+['Regiao'] que é a coluna que eu quero analisar +.value\_counts() comando usado para obter os valores da coluna.

In [49]:

```
dfs ['Regiao'].value_counts()
```

Out[49]:

Nordeste	66411
Sudeste	62974
Centro-oeste	28786
Sul	26113
Norte	19202
Name: Regiao, dtype: int64	

## 3.Qual a quantidade de reclamações realizadas por homens? E por mulheres?

Resposta: Homens: 94446; Mulheres: 107229.

\*Explicando o código: dfs (variavel, na qual se encontra a planilha do Excel)+['SexoConsumidor] coluna que eu quero usar + .value\_counts() obter o valor da coluna.

In [50]:

```
dfs['SexoConsumidor'].value_counts()
```

Out[50]:

F	107229
M	94446
N	1809
Name: SexoConsumidor, dtype: int64	

## 4.Em qual região os homens fizeram mais reclamações que as mulheres?

Resposta: Sul.

\*Explicando o código:Fui tentar criar uma variável como tava na "Introdução à pandas", mas não deu certo, já tinha usado o value\_counts para saber as reclamações e deu Sul. dfs (variavel, na qual se encontra a planilha do Excel)+dentro de []eu coloquei novamente a dfs, a coluna que eu queria analisar, junto com a informação que eu quero. Tirei isso da "Introdução à pandas". ['Regiao'] que a pergunta é a regiao da reclamação, e .max() pra mostrar o valor máximo das reclamações.

In [51]:

```
dfs[dfs['SexoConsumidor']=='M']['Regiao'].max()
```

Out[51]:

```
'Sul'
```

## 5.Qual a faixa etária que realizou mais reclamações?

Resposta: entre 31 a 40 anos.

\*Explicando o código: dfs (variável, na qual se encontra a planilha do Excel)+['FaixaEtariaConsumidor'] coluna que eu quero analisar + .describe() que mostra a descrição da coluna.

In [52]:

```
dfs['FaixaEtariaConsumidor'].describe()
```

Out[52]:

```
count          203486
unique           8
top      entre 31 a 40 anos
freq          45422
Name: FaixaEtariaConsumidor, dtype: object
```

## 6.Quais assuntos apresentaram mais reclamações em 2016?

Resposta: Telefone Celular, Cartões de Crédito e telefonia fixa.

\*Explicando o código: dfs (variável) + value\_counts, conseguindo o valor que mais aparecia no dataframe.

In [53]:

```
dfs ['DescricaoAssunto'].value_counts()
```

Out[53]:

```
Telefone ( Convencional, Celular, Interfone, Etc. )
21272
Telefonia Celular
15102
Cartão de Crédito
11774
Telefonia Fixa ( Plano de Expansão / Compra e Venda / Locação )
11056
Banco comercial
9274

...
Pã's para preparo (refresco, gelatina, pudim, bolos, pão de queijo, pizza
e sorvete)      1
Profissional Liberal - Veterinário
1
Venda através de reembolso postal a domicilio
1
Ã gua
1
Ovos
1
Name: DescricaoAssunto, Length: 212, dtype: int64
```

## 7. Analisando o nome fantasia, determine qual empresa recebeu mais reclamações em 2016.

Resposta: A empresa que mais recebeu reclamações em 2016, foi a OI.

\*Explicado o código: Usando o describe pra imprimir o total das linhas e a empresa.

In [54]:

```
dfs[['NomeFantasiaRFB', 'strNomeFantasia']].describe()
```

Out[54]:

	NomeFantasiaRFB	strNomeFantasia
count	92561	173722
unique	12865	26719
top	OI	OI
freq	11216	2732

## 8.Qual empresa possui um maior percentual de resolução de reclamações?

Resposta: TELEMAR NORTE LESTE S/A, foi a empresa com o maior percentual de resolução, com 79%.

In [55]:

```
dfs[['Atendida', 'RazaoSocialRFB', 'strRazaoSocial']].value_counts()
#Vou dividir essa respostas em algumas linhas de código. Observando esses números; Tele
mar Norte Leste S/A apareceu com mais reclamações atendidas.
```

Out[55]:

Atendida	RazaoSocialRFB	strRazaoSocial
S	TELEMAR NORTE LESTE S/A	TELEMAR NORTE LE
STE S/A	2350	
LETRONICO S/A	BRUXELAS EMPREENDIMENTOS E PARTICIPACOES S/A 1761	CNOVA COMERCIO E
	CLARO S.A.	CLARO S/A
1491		CLARO S.A.
1296	SAMSUNG ELETRONICA DA AMAZONIA LTDA	SAMSUNG ELETRONI
	CA DA AMAZONIA LTDA 1244	
...		
N	RUBENS DA SILVA MEIRELES 22155436807	RUBENS DA SILVA
MEIRELES	22155436807 1	
CORACOES	RUBENS CARUSO DECORACOES - ME 1	RUBENS CARUSO DE
GAMES LTDA	RUBENS & THIAGO GAMES LTDA - ME 1	RUBENS E THIAGO
SERVICOS LTDA - ME	RTX COMERCIAL E SERVICOS LTDA - ME 1	RTX COMERCIAL E
S	ZZAB COMERCIO DE CALCADOS LTDA.	ZZAB COMERCIO DE
CALCALDOS LTDA	1	
	Length: 32528, dtype: int64	

In [56]:

```
#Continuação...
#Sabendo que a empresa Telemar Norte Leste, teve 3.759 reclamações atendidas. Em um tot
al de 4749, no qual usei o (.shape), pra saber o total de linhas que apareceu a empresa
dfs[dfs['RazaoSocialRFB']=='TELEMAR NORTE LESTE S/A'].shape
```

Out[56]:

(4749, 23)

In [57]:

```
#Continuação...
#Fazendo o uso da probabilidade como o professor fez no colab dele...
prob = (3759/4749)*100
print(prob)
```

79.15350600126342

## 9. Monte um gráfico mostrando a quantidade de reclamações por região, separando as mesmas por sexo.

In [58]:

```
#Vou montar um dataset temporário das colunas que eu quero, para responder a questão abaixo.
```

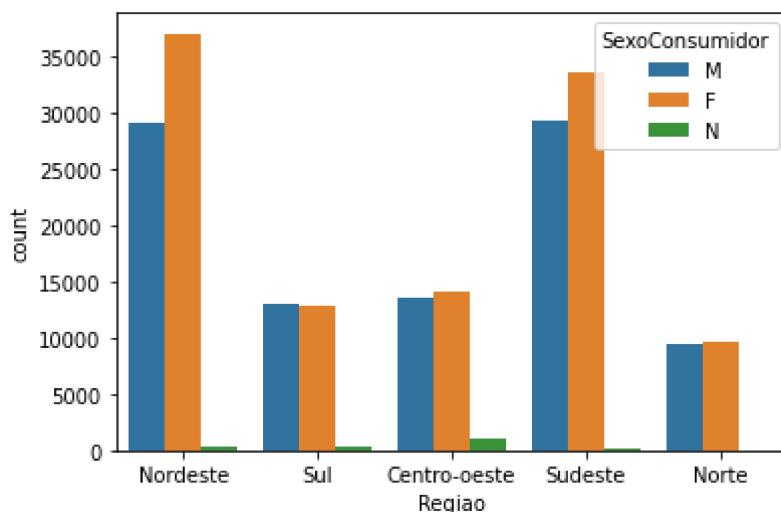
```
dfs_temp = dfs[['Regiao', 'SexoConsumidor']]
```

In [59]:

```
sns.countplot(x='Regiao', hue='SexoConsumidor', data=dfs_temp)
```

Out[59]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f756949a490>
```



## 10. Elabore um gráfico de linhas com a quantidade de reclamações por mês. Este gráfico lembra alguma distribuição estatística?

Resposta: Lembra Distribuição de Poisson, pois tem um momento no gráfico aonde ele é mais elevado que os outros. As reclamações ocorrem em intervalos, podemos ver que em determinados meses a elevação é pouca. Os mês com o maior número de reclamação, observando o gráfico é o mês de Março, entre os números 2 e 4.

Explicando o código: fiz uma variável para mudar a coluna de object para DateTime64, o que possibilitaria eu tirar uma informação (mês). Criei mais uma variável, tirando o que eu queria (mês). Usei o value\_counts() que me daria o valor de reclamação por mês. Por fim, usei o sns.lineplot para plotar o gráfico de linhas, arrumando as informações que queria no gráfico.

In [60]:

```
dfs ['DataAbertura'] = dfs ['DataAbertura'].astype('datetime64')
```

In [61]:

```
dfs_month = dfs ['DataAbertura'].dt.month
```

In [62]:

```
dfs_monthDados = dfs_month.value_counts()
```

In [63]:

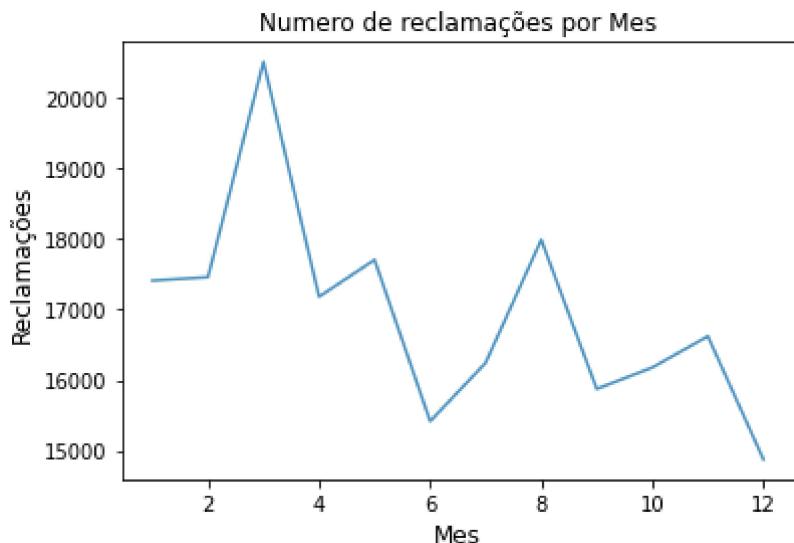
```
sns.lineplot(dfs_monthDados.index, dfs_monthDados.values, alpha=0.8)
plt.title('Número de reclamações por Mes')
plt.ylabel('Reclamações', fontsize=12)
plt.xlabel('Mes', fontsize=12)
```

/usr/local/lib/python3.7/dist-packages/seaborn/\_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

FutureWarning

Out[63]:

```
Text(0.5, 0, 'Mes')
```

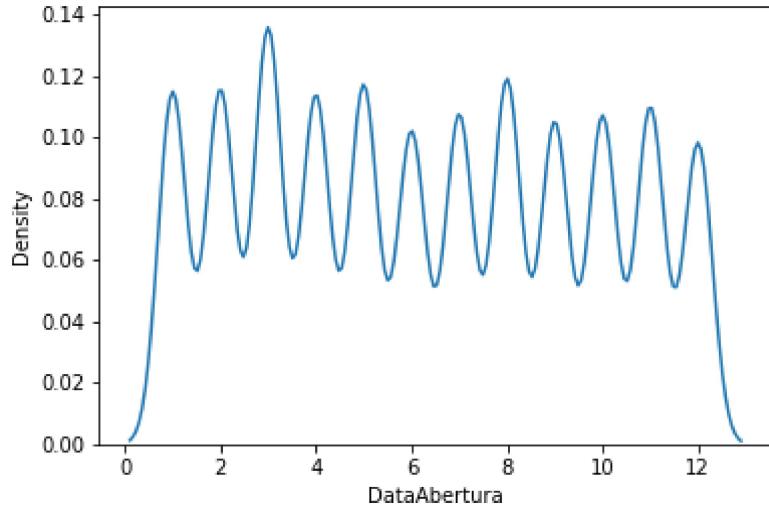


In [64]:

```
#Gráfico pode ser feito dessa forma também  
sns.kdeplot(dfs_month)
```

Out[64]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f756938b090>
```



**11. Elabore um gráfico boxplot mostrando a duração das reclamações por região.**

In [65]:

dfs.info()

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 203486 entries, 0 to 203485
Data columns (total 23 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   i»;AnoCalendario    203486 non-null   int64  
 1   DataArquivamento    203486 non-null   object  
 2   DataAbertura        203486 non-null   datetime64[ns] 
 3  CodigoRegiao         203486 non-null   int64  
 4   Regiao              203486 non-null   object  
 5   UF                  203486 non-null   object  
 6   strRazaoSocial       203483 non-null   object  
 7   strNomeFantasia     173722 non-null   object  
 8   Tipo                203486 non-null   int64  
 9   NumeroCNPJ          195292 non-null   float64 
 10  RadicalCNPJ         195292 non-null   float64 
 11  RazaoSocialRFB      187287 non-null   object  
 12  NomeFantasiaRFB    92561  non-null    object  
 13  CNAEPrincipal       187287 non-null   float64 
 14  DescCNAEPrincipal   185784 non-null   object  
 15  Atendida            203486 non-null   object  
 16  CódigoAssunto       203486 non-null   int64  
 17  DescricaoAssunto    203486 non-null   object  
 18  CódigoProblema      203476 non-null   float64 
 19  DescricaoProblema   203476 non-null   object  
 20  SexoConsumidor       203484 non-null   object  
 21  FaixaEtariaConsumidor 203486 non-null   object  
 22  CEPConsumidor        189400 non-null   float64 

dtypes: datetime64[ns](1), float64(5), int64(4), object(13)
memory usage: 35.7+ MB

```

In [66]:

dfs['DataArquivamento'] = dfs ['DataArquivamento'].astype('datetime64')

In [67]:

dfs['DataArquivamento'].dt.day

Out[67]:

```

0      17
1      15
2      16
3      18
4       9
..
203481  24
203482  27
203483  21
203484  16
203485  21
Name: DataArquivamento, Length: 203486, dtype: int64

```

In [68]:

```
dfs['DataAbertura'].dt.day
```

Out[68]:

```
0      29  
1      18  
2      29  
3      21  
4       7  
..  
203481     6  
203482     2  
203483    29  
203484    19  
203485    19  
Name: DataAbertura, Length: 203486, dtype: int64
```

In [69]:

```
dfs['Diferença']=dfs['DataArquivamento']-dfs['DataAbertura']
```

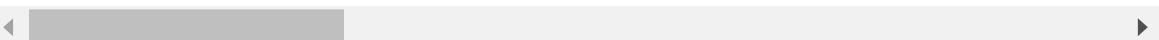
In [70]:

```
dfs[dfs.Diferenca.duplicated()]
```

Out[70]:

	AnoCalendario	DataArquivamento	DataAbertura	CodigoRegiao	Regiao	UF
128	2016	2016-07-08 11:17:04	2014-05-09 08:45:28	2	Nordeste	PI IN
130	2016	2016-06-17 10:39:03	2014-05-20 09:08:31	2	Nordeste	PI IN
134	2016	2016-07-05 10:15:15	2014-02-11 11:46:14	2	Nordeste	PI IN
139	2016	2016-11-09 12:12:58	2016-10-11 11:12:53	4	Sul	SC
142	2016	2016-11-25 16:12:06	2016-10-31 15:19:54	4	Sul	SC
...	...	...	...	...	...	...
203418	2016	2016-07-12 08:54:32	2016-06-20 11:38:30	2	Nordeste	PE E
203419	2016	2016-07-12 08:54:32	2016-06-20 11:38:30	2	Nordeste	PE
203438	2016	2016-11-09 13:23:55	2016-08-01 08:46:29	3	Sudeste	SP EC
203452	2016	2016-05-31 10:28:22	2016-04-25 10:40:31	3	Sudeste	SP I DI
203463	2016	2016-08-25 15:10:44	2016-07-12 13:43:29	3	Sudeste	SP AD

35870 rows × 24 columns



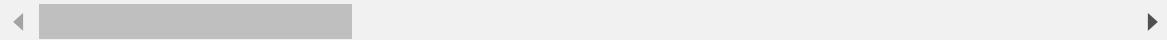
In [71]:

dfs[~dfs.Diferenca.duplicated()]

Out[71]:

	AnoCalendario	DataArquivamento	DataAbertura	CodigoRegiao	Regiao	UF
0	2016	2016-02-17 13:43:08	2015-10-29 10:59:59	2	Nordeste	PE
1	2016	2016-12-15 18:11:35	2016-11-18 14:52:22	2	Nordeste	PE
2	2016	2016-09-16 09:01:46	2016-06-29 13:58:35	2	Nordeste	PE
3	2016	2016-02-18 13:30:30	2016-01-21 12:23:51	2	Nordeste	PE
4	2016	2016-11-09 14:20:36	2016-10-07 16:39:59	2	Nordeste	PE
...	...	...	...	...	...	...
203481	2016	2016-08-24 10:19:29	2016-05-06 11:21:45	3	Sudeste	SP
203482	2016	2016-10-27 09:19:09	2016-05-02 15:28:57	3	Sudeste	SP
203483	2016	2016-09-21 09:11:21	2016-04-29 16:20:30	3	Sudeste	SP
203484	2016	2016-08-16 10:58:52	2016-04-19 07:58:16	3	Sudeste	SP
203485	2016	2016-09-21 09:44:57	2016-04-19 08:42:27	3	Sudeste	SP

167616 rows × 24 columns



In [72]:

```
dfs[~dfs.Diferença.duplicated()]['Diferença'].value_counts()
```

Out[72]:

```
111 days 02:43:09      1
90 days 22:49:33       1
150 days 16:49:40      1
197 days 01:22:37      1
327 days 22:06:40      1
..
60 days 17:05:42       1
175 days 01:48:03      1
57 days 06:50:09       1
87 days 21:59:36       1
155 days 01:02:30      1
Name: Diferença, Length: 167616, dtype: int64
```

In [73]:

```
dfs[~dfs.Diferença.duplicated()].describe()
```

Out[73]:

	AnoCalendario	CodigoRegiao	Tipo	NumeroCNPJ	RadicalCNPJ	CNAEP
<b>count</b>	167616.0	167616.000000	167616.000000	1.618130e+05	1.618130e+05	1.5501
<b>mean</b>	2016.0	2.905164	0.996880	2.315880e+13	1.166777e+08	5.6106
<b>std</b>	0.0	1.154209	0.055772	2.505512e+13	2.418607e+09	1.5754
<b>min</b>	2016.0	1.000000	0.000000	1.910000e+02	0.000000e+00	1.1560
<b>25%</b>	2016.0	2.000000	1.000000	4.206050e+12	4.206050e+06	4.7130
<b>50%</b>	2016.0	3.000000	1.000000	1.077276e+13	1.083593e+07	6.1205
<b>75%</b>	2016.0	4.000000	1.000000	3.873365e+13	4.043254e+07	6.4239
<b>max</b>	2016.0	5.000000	1.000000	9.840807e+13	9.965391e+10	9.7005

In [74]:

```
dfs_unique = dfs[~dfs.Diferença.duplicated()]
```

In [75]:

```
dfs['Diferença'].nunique()
```

Out[75]:

167616

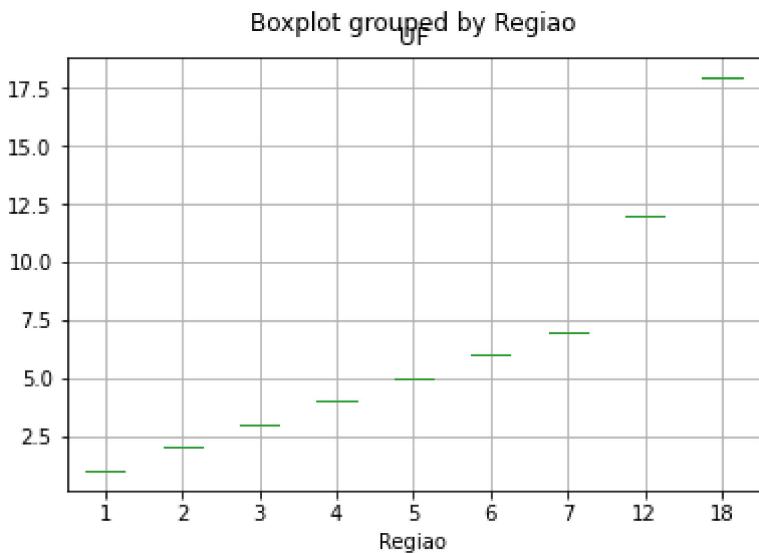
In [76]:

```
dfs_groupby=dfs.groupby(['DataArquivamento','DataAbertura','Diferença']).count()
dfs_groupby.boxplot(by='Regiao',column='UF')
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/cbook/__init__.py:1376:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequence
s (which is a list-or-tuple of lists-or-tuples-or ndarrays with different
lengths or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray.
    X = np.atleast_1d(X.T if isinstance(X, np.ndarray) else np.asarray(X))
```

Out[76]:

&lt;matplotlib.axes.\_subplots.AxesSubplot at 0x7f75692a51d0&gt;



## Elabore um gráfico boxplot mostrando a quantidade de reclamações por Região.

Explicando o código: Usei o groupby, juntando a região e uf, pra verificar se conseguia os números para plotar o boxplot. Conseguí os valores de máx,med,min. (.count()) para contabilizar isso. (.reset\_index()) para ajustar as informações que estavam em nível diferente (Regiao e UF). E com isso tudo, fiz a minha variável. Usei a variável, coloquei (.boxplot) e usei a DataAbertura como coluna, já que agrupei o box com a Região. Figszie para ajustas o tamanho do box.

In [77]:

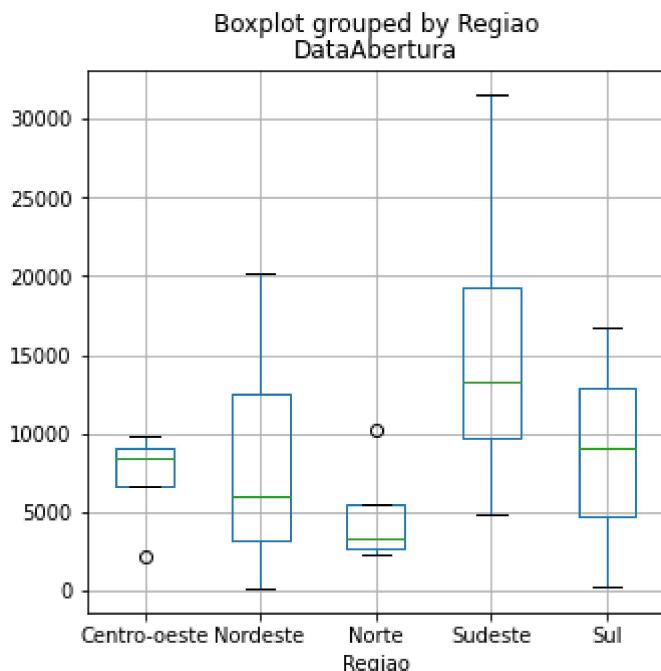
```
dfs_agrupado = dfs.groupby(["Regiao", "UF"]).count().reset_index()
dfs_agrupado.boxplot(by="Regiao", column = "DataAbertura", figsize = (5,5))
```

/usr/local/lib/python3.7/dist-packages/matplotlib/cbook/\_\_init\_\_.py:1376:  
 VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences  
 (which is a list-or-tuple of lists-or-tuples-or ndarrays with different  
 lengths or shapes) is deprecated. If you meant to do this, you must specify  
 'dtype=object' when creating the ndarray.

```
X = np.atleast_1d(X.T if isinstance(X, np.ndarray) else np.asarray(X))
```

Out[77]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7569183890>
```



**12. É possível afirmar que existe correlação entre o número de reclamações E o número de habitantes por Estado? Elabore um gráfico de dispersão e calcule o índice de correlação destes dois fatores.**

Resposta: Não é possível afirmar, pois tem algumas regiões na qual não temos o valor de reclamação por UF. Gráfico não ficou linear também.

Explicando o código: não consegui importar duas tabelas de excel, então fiz um dataframe novo. Colocando o numero de habitantes por regiao (separando por UF), assim como reclamação, juntando por UF. Tirei os dados de Habitantes de uma planilha, Habitantes 2016. Hnorte: Habitantes Norte. Rnorte: Reclamações Norte. E assim por diante. Tive que colocar os valores por UF, e já que cada coluna tem que ter o mesmo tanto de linhas, nos valores que eu não tinha, coloquei 0. Exemplo: Regiao Norte tem 9UFs, já na Sul tem 4, então usei 5 zeros para completar as colunas mesmo. Por fim, criei um gráfico de dispersão para cada Regiao, juntando habitantes e reclamações, depois coloquei todas essas linhas em uma só linha de código. Juntando então um gráfico de dispersão com todas as regiões, separei por cores, pra ficar diferente. Fiz um heatmap também, para mostrar o método de pearson.

In [78]:

```

data = {'HNorte': [1787279, 816687, 4001667, 514229, 8272724, 782295, 1532902, 0, 0],  

        'HNordeste': [6954036, 3212180, 8963663, 3474998, 3999415, 9410336, 3358963, 2265779, 1  

5276566],  

        'HSudeste': [20997560, 3973697, 16635996, 44749699, 0, 0, 0, 0, 0],  

        'HSul': [11242720, 6910553, 11286500, 0, 0, 0, 0, 0, 0],  

        'HCentro': [2682386, 3305531, 6695855, 2977216, 0, 0, 0, 0, 0],  

        'RNorte': [3922, 0, 2318, 0, 2730, 0, 10232, 0, 0],  

        'RNordeste': [3597, 2189, 10833, 4443, 7464, 20175, 191, 0, 17519],  

        'RSudeste': [15159, 11383, 4884, 31548, 0, 0, 0, 0, 0],  

        'RSul': [9132, 16718, 263, 0, 0, 0, 0, 0, 0],  

        'RCentro': [9789, 8770, 8060, 2167, 0, 0, 0, 0, 0],  

    }  
  

df = pd.DataFrame(data,columns=['HNorte', 'HNordeste', 'HSudeste', 'HSul', 'HCentro',  

                                 'RNorte', 'RNordeste', 'RSudeste', 'RSul', 'RCentro'])  

print (df)

```

	HNorte	HNordeste	HSudeste	HSul	HCentro	RNorte	RNordeste	RSudeste	RSul	RCentro
0	1787279	6954036	20997560	11242720	2682386	3922	3597			
1	816687	3212180	3973697	6910553	3305531	0	2189			
2	4001667	8963663	16635996	11286500	6695855	2318	10833			
3	514229	3474998	44749699	0	2977216	0	4443			
4	8272724	3999415	0	0	0	2730	7464			
5	782295	9410336	0	0	0	0	20175			
6	1532902	3358963	0	0	0	10232	191			
7	0	2265779	0	0	0	0	0			
8	0	15276566	0	0	0	0	17519			

In [79]:

```
df.corr(method='pearson')
```

Out[79]:

	<b>HNorte</b>	<b>HNordeste</b>	<b>HSudeste</b>	<b>HSul</b>	<b>HCentro</b>	<b>RNorte</b>	<b>RNordeste</b>	<b>RSul</b>
<b>HNorte</b>	1.000000	-0.150598	-0.120275	0.120315	0.099581	0.232338	0.006428	-0.2
<b>HNordeste</b>	-0.150598	1.000000	-0.157010	0.088952	0.007409	-0.242912	0.845493	-0.2
<b>HSudeste</b>	-0.120275	-0.157010	1.000000	0.269807	0.561837	-0.151243	-0.191097	0.9
<b>HSul</b>	0.120315	0.088952	0.269807	1.000000	0.808700	0.054197	-0.131950	0.2
<b>HCentro</b>	0.099581	0.007409	0.561837	0.808700	1.000000	-0.115174	-0.092805	0.4
<b>RNorte</b>	0.232338	-0.242912	-0.151243	0.054197	-0.115174	1.000000	-0.401000	-0.2
<b>RNordeste</b>	0.006428	0.845493	-0.191097	-0.131950	-0.092805	-0.401000	1.000000	-0.3
<b>RSudeste</b>	-0.224402	-0.278443	0.936314	0.223875	0.466307	-0.216541	-0.301040	1.0
<b>RSul</b>	-0.159483	-0.222952	0.016210	0.559145	0.318902	-0.118706	-0.340836	0.2
<b>RCentro</b>	0.015039	-0.040712	0.364624	0.953420	0.801161	-0.036926	-0.238929	0.4

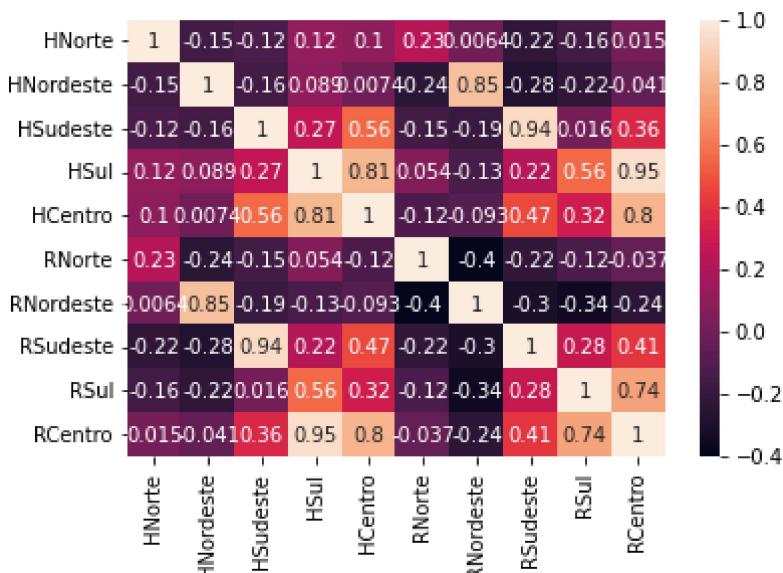


In [80]:

```
sns.heatmap(df.corr(), annot= True)
```

Out[80]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7569182750>
```

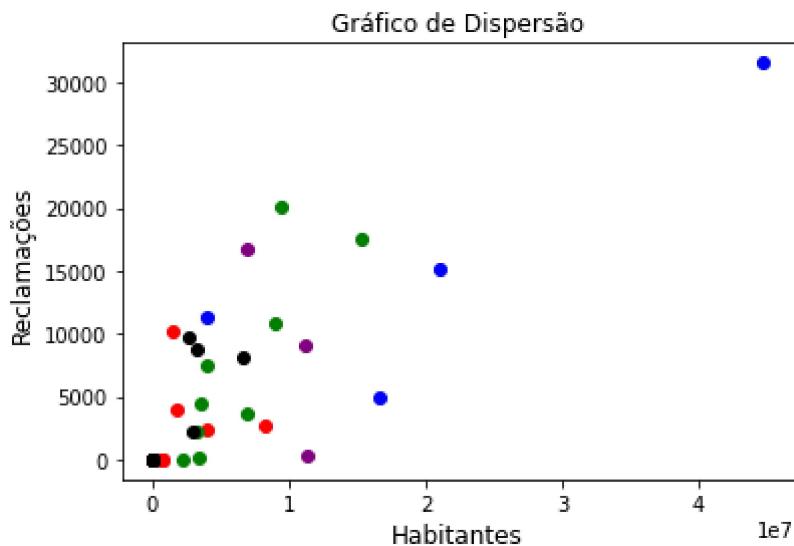


In [87]:

```
#Descrição: Habitantes e Reclamações NORTE (Vermelho); Nordeste (Verde); Sudeste(Azul);
Sul (Roxo); Centro(Preto)
plt.scatter(df['HNorte'],df['RNorte'], color='red')
plt.scatter(df['HNordeste'],df['RNordeste'], color='green')
plt.scatter(df['HSudeste'],df['RSudeste'], color='blue')
plt.scatter(df['HSul'],df['RSul'], color='purple')
plt.scatter(df['HCentro'],df['RCentro'], color='black')
plt.title('Gráfico de Dispersão')
plt.ylabel('Reclamações', fontsize=12)
plt.xlabel('Habitantes', fontsize=12)
```

Out[87]:

Text(0.5, 0, 'Habitantes')

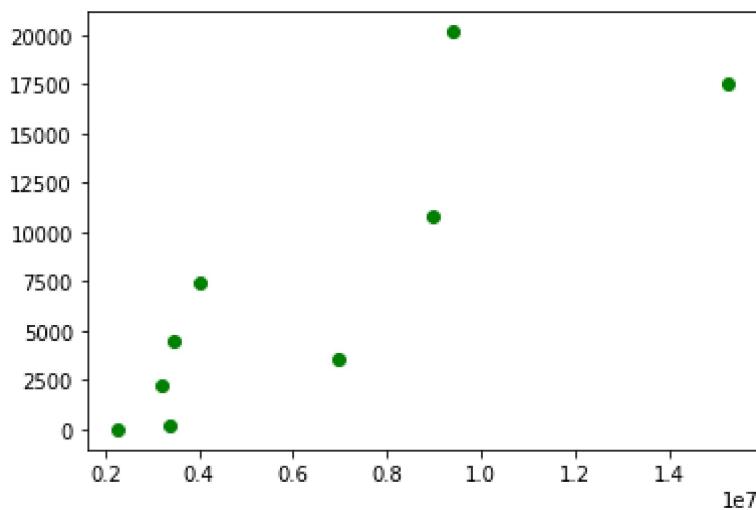


In [83]:

```
plt.scatter(df['HNordeste'], df['RNordeste'], color='green')
```

Out[83]:

```
<matplotlib.collections.PathCollection at 0x7f756d9f4ad0>
```

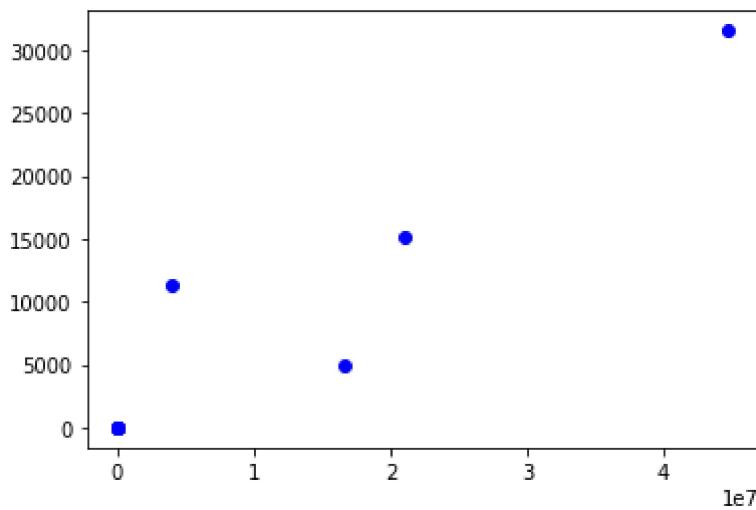


In [84]:

```
plt.scatter(df['HSudeste'], df['RSudeste'], color='blue')
```

Out[84]:

```
<matplotlib.collections.PathCollection at 0x7f756916c8d0>
```

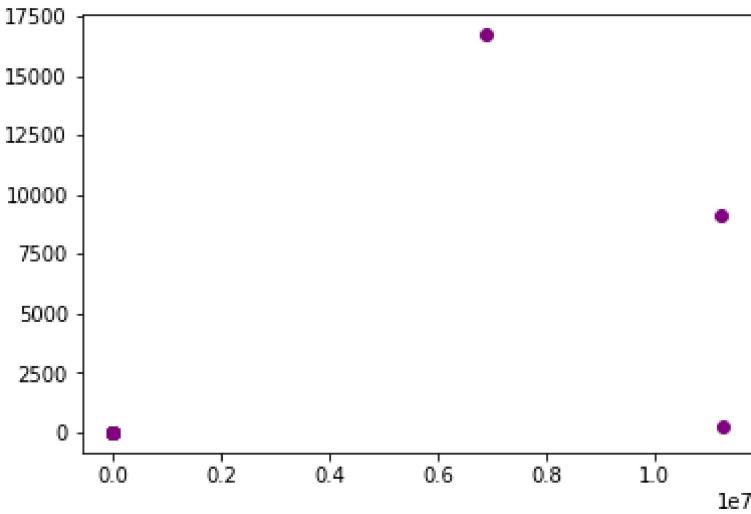


In [85]:

```
plt.scatter(df['HSul'], df['RSul'], color='purple')
```

Out[85]:

```
<matplotlib.collections.PathCollection at 0x7f7568d79210>
```



In [86]:

```
plt.scatter(df['HCentro'], df['RCentro'], color='black')
```

Out[86]:

```
<matplotlib.collections.PathCollection at 0x7f7568cde710>
```

