

Branch: master ▼

Find file

Copy path

[ser347](#) / [2018](#) / [aula-23](#) / **analise-dados-geoespaciais.ipynb**



**gqueiroz** Update analise-dados-geoespaciais.ipynb

5694eb8 on 30 May 2018

[1 contributor](#)



Raw

Blame

History



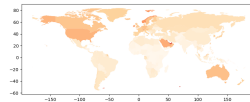
1371 lines (1371 sloc) 34.7 KB

# SER-347 - Introdução à Programação para Sensoriamento Remoto

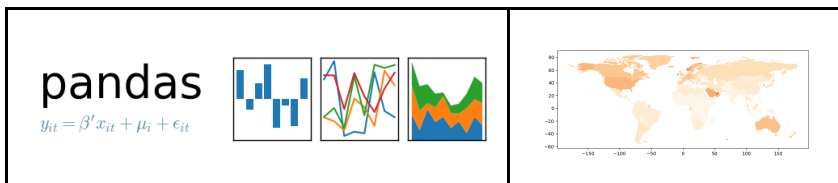
## Aula 23 - Análise de Dados Geoespaciais

(<http://geopandas.org>)

- Gilberto Ribeiro de Queiroz
- Thales Sehn Körting
- Fabiano Morelli



## 1. Pandas e GeoPandas: análise de dados em Python



### Pandas

Fornece duas estruturas de dados básicas: `Series` e `DataFrame`. Para estas estruturas, existem diversas operações de alto nível disponíveis, tais como: agregação de valores e visualização básica através da `matplotlib`.

Um objeto do tipo `Series` representa um vetor (ou array unidimensional) capaz de armazenar qualquer tipo de dado, como números inteiros, strings ou objetos como data e hora. Possui um eixo (*axis*) usado para rotular cada valor do vetor. Esses rótulos funcionam como um índice para os valores da série.

	municipio
0	Sítio Novo Do Tocantins
1	Ouro Preto
2	Mariana
3	Araxá
4	Belo Horizonte

Um objeto do tipo DataFrame representa um tabela bidimensional com os eixos rotulados (linhas e colunas).

	municipio	estado	regiao	pais	satelite	bioma	timestamp	satelite_r
0	Sítio Novo Do Tocantins	Tocantins	N	Brazil	NPP_375	Cerrado	2016/02/12 17:05:45	f
1	Sítio Novo Do Tocantins	Tocantins	N	Brazil	NPP_375	Cerrado	2016/07/17 04:00:00	f
2	Sítio Novo Do Tocantins	Tocantins	N	Brazil	AQUA_M-T	Cerrado	2016/01/15 16:40:14	t
3	Sítio Novo Do Tocantins	Tocantins	N	Brazil	NPP_375	Cerrado	2016/07/17 04:00:00	f
4	Sítio Novo Do Tocantins	Tocantins	N	Brazil	NPP_375	Cerrado	2016/02/12 17:05:45	f

## GeoPandas

Possui facilidades para tratar colunas com dados geométricos, incluindo visualização.

As duas estruturas fornecidas são:

- GeoSeries: um vetor contendo uma representação geométrica em conformidade com os tipos da *OGC Simple Feature*: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon. Essa estrutura possui as mesmas operações da classe Series do Pandas além de operações espaciais como cálculo de área, perímetro, distâncias, entre outras.


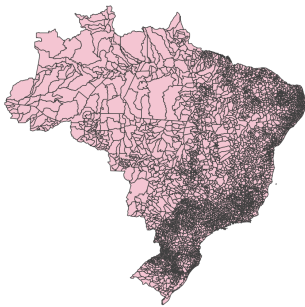
	municipio
0	POINT (-47.607 -5.673)
1	POINT (-47.606 -5.581)
2	POINT (-47.734 -5.562)
3	POINT (-47.605 -5.58)
4	POINT (-47.606 -5.677)

- GeoDataFrame: tabela com uma coluna geométrica.

	municipio	estado	regiao	pais	satelite	bioma	timestamp	satelite_r	geometry
	Sítio						2016/02/12		POINT

0	Novo Do Tocantins	Tocantins	N	Brazil	NPP_375	Cerrado	2016/02/12 17:05:45	f	(-47.607 -5.673)
1	Sítio Novo Do Tocantins	Tocantins	N	Brazil	NPP_375	Cerrado	2016/07/17 04:00:00	f	POINT (-47.606 -5.581)
2	Sítio Novo Do Tocantins	Tocantins	N	Brazil	AQUA_M-T	Cerrado	2016/01/15 16:40:14	t	POINT (-47.734 -5.562)
3	Sítio Novo Do Tocantins	Tocantins	N	Brazil	NPP_375	Cerrado	2016/07/17 04:00:00	f	POINT (-47.605 -5.58)
4	Sítio Novo Do Tocantins	Tocantins	N	Brazil	NPP_375	Cerrado	2016/02/12 17:05:45	f	POINT (-47.606 -5.677)

## 2. Dados de Entrada

	
<b>Focos de Queimada - MG e GO - 2016</b> Fonte: Programa Queimadas - INPE	<b>Malha Municipal</b> Fonte: IBGE

## 3. GeoPandas

Vamos importar as bibliotecas pandas, geopandas e matplotlib para podermos manipular os dados com focos de queimada usando um GeoDataFrame:

```
In [ ]: import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
```

### 3.1. Explorando os Focos de Queimadas

Abrir o shapefile e transformá-lo em um GeoDataFrame:

```
In [ ]: focos = gpd.read_file("/Users/gribeiro/Dados/Queimadas/focos/mg-go-2016.shp")
```

Vamos ver uma amostra dos dados:

```
In [ ]: focos.head()
```

Quantos focos foram carregados no DataFrame?

```
In [ ]: len(focos)
```

Podemos descobrir os tipos de dados das colunas do GeoDataFrame através do atributo dtypes:

```
In [ ]: focos.dtypes
```

**V1:** Visualizar os focos de queimada em um mapa.

```
In [ ]: focos.plot(marker='x', color='red', marker size=5);
```

**T1:** Alterar o formato da coluna timestamp para o tipo datetime para facilitar a manipulação dos dados desta coluna:

```
In [ ]: focos["timestamp"] = pd.to_datetime(focos["timestamp"])
```

```
In [ ]: focos.dtypes
```

```
In [ ]: focos.head()
```

**V2:** Visualizar os focos de queimada do estado de Minas Gerais:

Podemos selecionar da tabela focos apenas as linhas contendo os focos de Minas Gerais:

```
In [ ]: focos_mg = focos[focos.estado == 'Minas Gerais']
```

```
In [ ]: focos_mg.plot(marker='x', color='red', markersize=10);
```

**V3:** Visualizar os focos de queimada do estado de Minas Gerais no mês de novembro:

```
In [ ]: focos_mg_nov = focos_mg[focos_mg.timestamp
    .dt.month == 11]
```

```
In [ ]: focos_mg_nov.plot(marker='x', color='red',
    markersize=10);
```

**Q1:** Qual a distribuição dos focos ao longo dos meses do ano em 2016?

Para responder esta pergunta podemos utilizar o operador de agregação (sumarização) `groupby`, disponível em um `DataFrame`.

Neste caso, precisaremos informar:

- O critério da agregação: a parte contendo o número do mês na coluna com a data e hora da detecção do foco (coluna `timestamp`).
- Utilizar uma das colunas para realizar a contagem através do operador `count`.

```
In [ ]: focos_mes = focos.groupby(focos.timestamp.
    dt.month).estado.count()
```

O objeto `focos_mes` retornado na operação acima corresponderá um `pandas.core.series.Series`:

```
In [ ]: type(focos_mes)
```

```
In [ ]: focos_mes
```

Na saída acima podemos notar o seguinte:

- O nome da série é `estado`, por conta da coluna usada para realizar a contagem.
- Os índices da série correspondem aos índices numéricos dos meses do ano.

Podemos re-indexar a série pelo nome do mês do ano. Para isso, podemos construir uma função `lambda` e aplicá-la através do operador `map` a cada um dos valores numéricos do índice do ano. Para transformar o mês do ano em um nome, utilizaremos o módulo `calendar` da Biblioteca Padrão Python.

```
In [ ]: import calendar

    novo_indice = map(lambda v : calendar.month
        h_abbr[v], focos_mes.index)
```

```
In [ ]: l = list(novo_indice)
```

```
In [ ]: l
```

Agora podemos construir explicitamente a nova série através do construtor `pd.Series`:

```
In [ ]: focos_mes = pd.Series(data=focos_mes.values, index=1)
```

```
In [ ]: type(focos_mes)
```

```
In [ ]: focos_mes
```

Podemos também ajustar o nome da série e o rótulo do índice:

```
In [ ]: focos_mes.name= "Número Focos/Mês"
        focos_mes.index.name="mes"
```

```
In [ ]: focos_mes
```

Podemos apresentar um gráfico de barras com o total de focos por mês:

```
In [ ]: focos_mes.plot.bar(legend=True, fontsize=20);
```

Podemos melhorar nosso gráfico controlando as diversas opções de plotagem fornecidas pela Matplotlib:

```
In [ ]: ax = focos_mes.plot(kind="bar", legend=True, fontsize=20, figsize=(20,10));
        ax.set_title("Focos Mensal - 2016", fontsize=36);
        ax.set_xlabel("Mes", fontsize=24);
        ax.set_ylabel("#Focos", fontsize=24);
        ax.legend(loc=2, prop={'size': 20});
```

Podemos salvar a figura do gráfico gerado:

```
In [ ]: ax.figure.savefig("/home/gribeiro/Dados/Quemadas/focos/chart-focos-mes-2016.png",
                          dpi=100, format="png")
```

**Q2:** Qual a distribuição dos focos no ano de 2016 por bioma?

```
In [ ]: focos_bioma = focos.groupby("bioma").bioma.count()
```

Novamente teremos uma série de dados Pandas:

```
In [ ]: type(focos_bioma)
```

Podemos acertar o nome da série e do índice:

```
In [ ]: focos_bioma.index.name = "Biomass"
        focos_bioma.name = "Numero Focos por Biomassa"

        focos_bioma
```

Podemos agora desenhar um gráfico circular:

```
In [ ]: explode=[0.0, 0.1, 0.0]
        ax = focos_bioma.plot(kind="pie", explode=explode, autopct='%1.1f%%', figsize=(8,8),
                               fontsize="14");
        ax.set_title("Focos por Bioma - 2016", fontsize=20);
```