Capítulo 1

Análise Exploratória de Dados Espaciais com Python

Gesiel Rios Lopes, Karina Jorge Pelarigo, Alexandre C. B. Delbem, Joélcio Braga de Sousa

Abstract

The sharp development of technologies for data analysis in the geographic space that occurred in recent years has offered innovative possibilities to understand the influence of spatial effects in explaining various phenomena and constitutes an excellent challenge for several areas of scientific knowledge. Exploratory Spatial Data Analysis (ESDA) is the extension of Exploratory Data Analysis (EDA) to the problem of detecting spatial properties of data sets where there is locational data for each attribute value. This location data references the point or area to which the attribute refers. ESDA is a set of techniques that aim to describe and visualize spatial distributions, identify atypical or spatial outliers, discover spatial association patterns and clusters, and suggest spatial regimes or other forms of spatial heterogeneity. This chapter aims to align theory and practice, presenting some concepts and techniques associated with ESDA with python, focusing on tabular vector data.

Resumo

O desenvolvimento acentuado de tecnologias para análise de dados no espaço geográfico ocorrido nos últimos anos tem oferecido possibilidades inovadoras ao entendimento da influência dos efeitos espaciais na explicação de vários fenômenos e constitui um grande desafio para diversas áreas do conhecimento científico. A análise exploratória de dados espaciais (AEDE) é a extensão da análise exploratória de dados (AED) para o problema de detecção de propriedades espaciais de conjuntos de dados onde, para cada valor de atributo, existe um dado locacional. Este dado de localização referencia o ponto ou a área à qual o atributo se refere. AEDE é um conjunto de técnicas que visa descrever e visualizar distribuições espaciais, identificar locais atípicos ou discrepantes espaciais, descobrir padrões de associação espacial, clusters e sugerir regimes espaciais ou outras formas de heterogeneidade espacial. O objetivo deste minicurso é alinhar a teoria e a prática, apresentando alguns dos conceitos e técnicas associadas à AEDE com python, com foco em dados vetoriais tabulares.

1.1. Introdução

A compreensão da influência dos efeitos espaciais na explicação de vários fenômenos constitui um grande desafio para diversas áreas do conhecimento, seja em saúde, em geologia, em agronomia, computação ou entre tantas outras. Tais estudos vêm se tornando cada vez mais comum, devido o aumento significativo na disponibilidade de dados com informações geoespaciais (tempo e espaço) e o rápido desenvolvimento de tecnologias para análise desses dados [Monteiro et al. 2004, Almeida 2012, Andrade et al. 2007, Lopes et al. 2021, Domingues et al. 2020].

A Análise Exploratória de Dados Espaciais (AEDE) é a coleção de técnicas para descrever e visualizar distribuições espaciais, identificar localidades atípicas (*outliers* espaciais), descobrir padrões de associação espacial (*clusters* espaciais) e sugerir diferentes regimes espaciais e outras formas de instabilidade espacial [Almeida 2012, Anselin 2005].

AEDE permite considerar o contexto espacial em que ocorrem a maioria dos eventos geograficamente referenciados. Esses eventos não podem ser vistos como desconexos e espacialmente independentes, caso contrário, as conclusões sobre os dados e, mais importante, as considerações teóricas sobre os processos sociais que ocorrem no espaço seriam negligenciadas [Haining et al. 1998].

As técnicas de AEDE são geralmente divididas em dois grupos principais: ferramentas para analisar autocorrelação espacial global e local. O primeiro considera a tendência geral que a localização dos valores segue e possibilita afirmações sobre o grau de agrupamento no conjunto de dados. Os valores geralmente seguem um padrão particular em sua distribuição geográfica? Os valores semelhantes estão mais próximos de outros valores semelhantes do que esperaríamos por puro acaso? Estas são algumas das questões que as ferramentas de autocorrelação espacial global permitem responder.

As ferramentas para autocorrelação espacial local, focam na instabilidade espacial: a partida de partes de um mapa da tendência geral. A ideia aqui é que, embora haja uma determinada tendência para os dados em termos da natureza e força da associação espacial, algumas áreas particulares podem divergir substancialmente do padrão geral. Independentemente do grau geral de concentração nos valores, podemos observar bolsões de valores anormalmente altos ou baixos próximos a outros valores altos ou baixos, no que chamaremos de pontos quentes ou frios. Além disso, também é possível observar alguns valores altos ou baixos cercados por valores baixos, ou altos, e chamaremos esses de "*outliers* espaciais". A principal técnica que revisaremos nesta sessão para explorar a autocorrelação espacial local são os Indicadores Locais de Associação Espacial (do inglês, *Local Indicators of Spatial Association* - LISA).

1.1.1. Por que usar python ?

Python é uma linguagem de programação interpretada de uso geral, ao contrário de R ou Matlab, centrada na legibilidade do código, o que a torna uma linguagem fácil de aprender. Python é uma linguagem dinâmica sendo adequada para desenvolvimento interativo e prototipagem rápida com o poder de suportar o desenvolvimento de grandes aplicativos, também é uma linguagem amplamente usada para aprendizado de máquina e ciência de dados devido ao excelente suporte à biblioteca. Ela rapidamente se tornou uma das

plataformas dominantes para praticantes de aprendizado de máquina e ciência de dados [Lopes et al. 2019]

Para o desenvolvimento deste minicurso, será utilizado o ambiente do *Jupyter Notebook*, uma interface de programação literária interativa[1] muito utilizada para prototipar e compartilhar *scripts* de soluções em ciência de dados [Kluyver et al. 2016]. Em [Pimentel et al. 2021] e [Lopes et al. 2019] pode ser encontrado um vasto material de como utilizar o *Jupyter Notebook*.

1.1.2. Biblioteca Python para Análise Exploratória de Dados Espaciais

Para este minicurso, usaremos a biblioteca de análise espacial Python *PySAL*, uma biblioteca multiplataforma de código aberto para ciência de dados geoespaciais com ênfase em dados vetoriais escrita em Python [Rey and Anselin 2007]. *Pysal* suporta o desenvolvimento de aplicações de alto nível para análise espacial, como:

- detecção de *clusters* espaciais, *hot-spots* e *outliers*;
- construção de gráficos a partir de dados espaciai;
- análise exploratória de dados espaço-temporais, dentre outras funcionalidades.

PySAL é uma família de pacotes para ciência de dados espaciais, dividido em quatro componentes principais:

- **Lib:** Estruturas de dados espaciais, IO de arquivo. Construção e edição interativa de matrizes e gráficos de pesos espaciais. Formas alfa, índices espaciais e relações espaço-topológicas.
- **Explore:** Módulos para realizar análises exploratórias de dados espaciais e espaço-temporais.
- **Model:** Estimativa de relações espaciais em dados com uma variedade de modelos lineares, lineares generalizados, aditivos generalizados e não lineares
- **Viz:** Visualize padrões em dados espaciais para detectar *clusters*, *outliers* e pontos de acesso.

Uma visão geral dos principais componentes do *PySAL* é apresentada na Figura 1.1. Ele é organizado em seis categorias principais de funcionalidade que lidam com operações básicas de dados, como a construção e manipulação de pesos espaciais e funções essenciais de geometria computacional, exploração de dados como métodos de agrupamento e análise exploratória de dados espaciais, e modelagem espacial, como dinâmica espacial e econometria espacial [Rey and Anselin 2007].

[1] O paradigma de programação literária visa ajudar na comunicação de programas através da alternância de texto em linguagem natural formatada, pedaços de código executáveis, e resultados de computações [Pimentel et al. 2021]

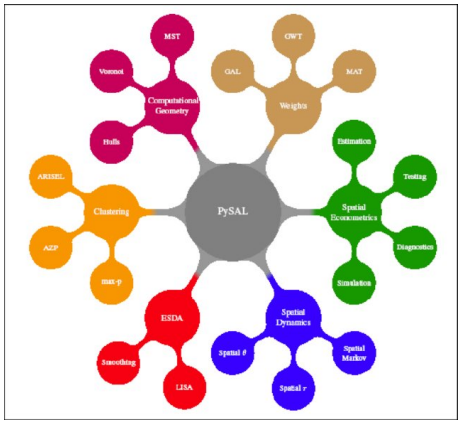


Figura 1.1. Componentes do *PySal* [Rey and Anselin 2007]

1.2. Dados Espaciais

Dados espaciais, dados geográficos ou dados geoespaciais têm duas características muito úteis. Primeiro, os dados geográficos são onipresentes. Tudo tem uma localização no espaço-tempo e essa localização pode ser usada diretamente para fazer melhores previsões ou inferências. Mas, assim como o "tempo" é mais que a posição do relógio, a geografia é mais que a posição da Terra: a localização permite entender as relações entre as observações. Muitas vezes são as relações úteis na ciência de dados porque nos permitem contextualizar nossos dados. Como argumenta o geógrafo Waldo Tobler em [Tobler 1970], *coisas próximas tendem a ser mais relacionadas do que coisas distantes, tanto no espaço quanto no tempo*, também conhecida como Primeira lei da geografia [Almeida 2012]. Portanto, se aprendermos adequadamente com essas informações contextuais, poderemos construir modelos melhores.

Assim como um modelo estatístico, um mapa é uma representação do processo geográfico subjacente, mas não é o processo. Apesar do fato de que essas representações não são exatamente corretas em algum sentido, elas são úteis para entender o que é importante sobre um processo geográfico [Almeida 2012, Andrade et al. 2007, Câmara et al. 2004].

Dados espaciais podem ser definidos como dados ou informações associados às coordenadas geográficas que representam a superfície terrestre. Os tipos mais comuns de dados espaciais são dados *raster* ou matriciais e dados vetoriais.

- Dados *raster* ou matriciais: essa estrutura de dados armazena informações geográficas por meio de grades ou matrizes. Um exemplo clássico de dados *raster* são as imagens de satélite obtidas por meio de sensoriamento remoto, nas quais, cada pixel da imagem estará sempre associado a coordenadas geográficas que representam determinada região da superfície terrestre.

- Dados vetoriais: essa estrutura de dados é representada por informações geográficas associadas a pontos, linhas e polígonos. Os limites de uma propriedade, a localização de um empreendimento e delimitações de estradas, biomas e bacias hidrográficas são exemplos de dados de natureza vetorial.

Neste minicurso, por simplicidade, o foco será em dados vetoriais através das tabelas geográficas do tipo `GeoDataFrame`, uma subclasse do `pandas.DataFrame` capaz de armazenar colunas geométricas e realizar operações espaciais [Lopes et al. 2021]. O primeiro passo, antes de ler alguns dados geoespaciais, é declarar o uso da biblioteca `GeoPandas` (Figura 1.2). Com a declaração do `GeoPandas`, usaremos `%matplotlib inline`, linha 1, uma configuração do `matplotlib` para permitir que os mapas apareçam diretamente no nosso *notebook*, ao invés de serem exibidos em uma janela diferente.

```
1 %matplotlib inline
2 import geopandas as gpd
```

Figura 1.2. Declaração do GeoPandas.

Precisamos agora ter acesso a um conjunto de dados geográficos e existem diversas plataformas governamentais que disponibilizam esses dados gratuitamente, como, por exemplo:

1. INDE (Infraestrutura Nacional de Dados Espaciais),
2. IBGE (Instituto Brasileiro de Geografia e Estatística),
3. ANA (Agência Nacional das Águas), etc.

Assumindo que temos um arquivo contendo dados espaciais, podemos lê-lo facilmente usando a função $gpd.read\_file$, que detecta automaticamente o tipo de arquivo e cria um `GeoDataFrame`. Para criar nosso primeiro mapa, utilizaremos a malha de setores censitários do estado de São Paulo utilizados no CENSO de 2010 do IBGE que pode ser baixado no seguinte endereço: `https://shorturl.at/klAF7`.

A Figura 1.3 apresenta os dois registros do `GeoDataFrame` da malha de setores censitários do estado de São Paulo.

```
1 setores_censitarios_sp = gpd.read_file(
2     'sp_setores_censitarios.zip'
3 )
4 setores_censitarios_sp.head(2)
```

	ID	CD_GEOCODI	TIPO	CD_GEOCODS	NM_SUBDIST	CD_GEOCODD	NM_DISTRIT	CD_GEOCODM	NM_MUNICIP	NM_MICRO	NM_ME
0	98237.0	354100005000009	URBANO	35410000500	None	354100005	PRAIA GRANDE	3541000	PRAIA GRANDE	SANTOS	METROPOLITA DE SÃO PAU
1	98232.0	354100005000004	URBANO	35410000500	None	354100005	PRAIA GRANDE	3541000	PRAIA GRANDE	SANTOS	METROPOLITA DE SÃO PAU

Figura 1.3. Malha de setores censitários do estado de São Paulo.

Cada linha desta tabela é um único setor censitário do estado de São Paulo. Cada setor possui um conjunto de informações como código do setor (CD_GEOCODI), nome do município (NM_MUNICIP), dentre outras. A geometria do limite dos setores censitários é armazenada na coluna de `geometry`. Assim como em outras estruturas de

dados baseadas em tabela em Python, cada linha e coluna tem um índice que as identifica exclusivamente e é renderizada em negrito no lado esquerdo da tabela.

No `geopandas`, assim como em outros pacotes que representam dados geográficos, a coluna de `geometry` possui características especiais que uma coluna "normal" não possui. Por exemplo, quando plotamos o `dataframe`, a coluna `geometry` é usada como a forma principal para plotar, conforme mostrado na Figura 1.4.

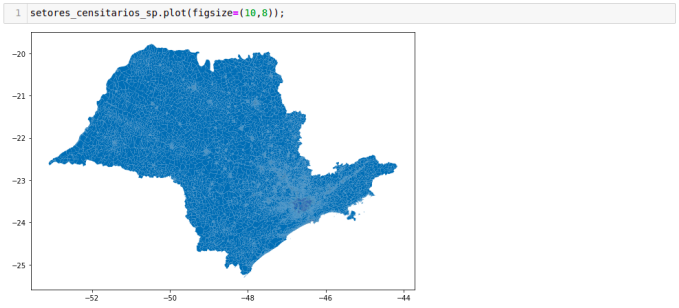


Figura 1.4. Mapa com a malha de setores censitários do estado de São Paulo.

Por ser um objeto derivado do `DataFrame`, é possível manipular o `GeoDataFrame` da mesma forma que manipulamos um `DataFrame`, por exemplo, é possível selecionar apenas a malha de setores censitários da cidade de São Paulo através da função `query`, linha 1 da Figura 1.5.

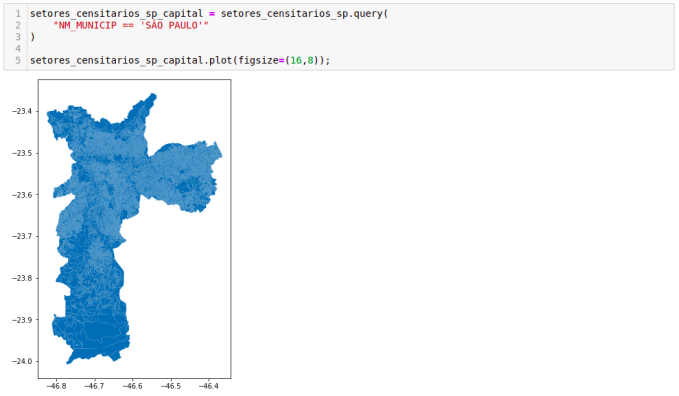


Figura 1.5. Mapa com a malha de setores censitários da cidade de São Paulo - SP.

Em muitos casos, as tabelas geográficas terão geometrias de um único tipo; todos os registros serão `Point` ou `LineString`, por exemplo. No entanto, não há nenhum requisito formal de que uma tabela geográfica tenha geometrias que sejam todas do mesmo tipo.

Ao longo do capítulo usaremos tabelas geográficas extensivamente, armazenando polígonos, mas também pontos e linhas. Maiores detalhes sobre como manipular essas outras formas geométricas com `geopandas` podem ser encontrado em [Lopes et al. 2021].

1.3. Matrizes de Ponderação Espacial

Matrizes de Ponderação Espacial ou "pesos espaciais" são uma maneira de representar gráficos em ciência de dados geográficos e estatísticas espaciais. São construções amplamente utilizadas que representam relações geográficas entre as unidades observacionais em um conjunto de dados referenciado espacialmente. Implicitamente, pesos espaciais conectam objetos em uma tabela geográfica mutualmente usando as relações espaciais entre eles. Ao expressar a noção de proximidade geográfica ou conectividade, os pesos espaciais são o principal mecanismo pelo qual as relações espaciais nos dados geográficos são levadas a efeito na análise subsequente [Almeida 2012, Andrade et al. 2007].

1.3.1. Pesos de Contiguidade

A matriz de pesos espaciais pode ser construída em consonância com a ideia de vizinha baseada na contiguidade, em que duas regiões são vizinhas, caso elas partilhem de uma fronteira física comum [Almeida 2012]. A ideia é que duas regiões contíguas possuem uma maior interação espacial, dessa forma, podemos definir uma matriz de pesos da seguinte forma:

$$w_{ij} = \begin{cases} 1 & \text{se } i \text{ e } j \quad \text{são contíguos} \\ 0 & \text{se } i \text{ e } j \quad \text{não são contíguos} \end{cases} \tag{1}$$

Convencionalmente, é presumido que $w_{ij} = 0$, ou seja, a região não é vizinha dela mesma, implicando que a matriz de contiguidade possua a sua diagonal principal composta por valores nulos. À primeira vista isso parece simples, no entanto, na prática, isso acaba sendo mais complicado. A primeira complicação é que existem diferentes maneiras pelas quais os objetos podem "compartilhar uma borda comum". Começaremos com o exemplo de uma grade de três por três (Figura 1.6), que resulta na grade mostrada na Figura 1.7.

```python
import numpy as np
import matplotlib.pyplot as plt
from shapely.geometry import Polygon

# Get points in a grid
l = np.arange(3)
xs, ys = np.meshgrid(l, l)
# Set up store
polys = []
# Generate polygons
for x, y in zip(xs.flatten(), ys.flatten()):
    poly = Polygon([(x, y), (x+1, y), (x+1, y+1), (x, y+1)])
    polys.append(poly)
# Convert to GeoSeries
polys = gpd.GeoSeries(polys)
gdf = gpd.GeoDataFrame(
    {
        'geometry': polys,
        'id': ['P-%s'%str(i).zfill(2) for i in range(len(polys))]
    }
)
```

Figura 1.6. Código para construção de uma grade 3x3.

Uma forma comum de expressar as relações de contiguidade/adjacência surge de uma analogia com os movimentos legais que diferentes peças de xadrez podem fazer. A contiguidade é considerada como *torre* (*rook*) caso apenas as fronteiras físicas com a extensão diferente de zero entre as regiões sejam consideradas. Se além das fronteiras com extensão diferente de zero puderem ser considerados os vértices como contíguos, a contiguidade é considerada como *rainha* (*queen*) se apenas os vértices forem considerados para definir a contiguidade, a convenção é demoninada *bisbo* (*bishop*) [Almeida 2012].

```
# Plot grid geotable
ax = gdf.plot(facecolor='w', edgecolor='k')

# Loop over each cell and add the text
for x, y, t in zip(
    [p.centroid.x-.25 for p in polys],
    [p.centroid.y-.25 for p in polys],
    [i for i in gdf['id']]
):
    plt.text(
        x, y, t, verticalalignment='center', horizontalalignment='center'
    )

# Remove axes
ax.set_axis_off()
plt.show()
```

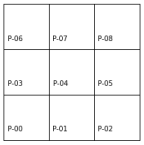
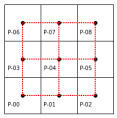


Figura 1.7. Plot da grade 3x3.

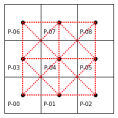
Na Figura 1.8(a) e 1.8(b) temos exemplos das convenções mais utilizadas na literatura, respectivamente, *Rook* e *Queen*.

```
# Build a regular 3x3 lattice and draw it here
wr = weights.contiguity.Rook.from_dataframe(gdf)

# Set up figure
f,ax = plt.subplots(1,1, subplot_kw=dict(aspect='equal'))
# Plot grid
gdf.plot(facecolor='w', edgecolor='k', ax=ax)
# Loop over each cell and add the text
for x, y, t in zip(
    [p.centroid.x-.25 for p in polys],
    [p.centroid.y-.25 for p in polys],
    [i for i in gdf['id']]
):
    plt.text(
        x, y, t, verticalalignment='center', horizontalalignment='center'
    )
# Plot weights connectivity
wr.plot(gdf, edge_kws=dict(color='r', linestyle=':'), ax =ax)
# Remove axes
ax.set_axis_off()
```



(a)

```
# Build a regular 3x3 lattice and draw it here
wr = weights.contiguity.Queen.from_dataframe(gdf)

# Set up figure
f,ax = plt.subplots(1,1, subplot_kw=dict(aspect='equal'))
# Plot grid
gdf.plot(facecolor='w', edgecolor='k', ax=ax)
# Loop over each cell and add the text
for x, y, t in zip(
    [p.centroid.x-.25 for p in polys],
    [p.centroid.y-.25 for p in polys],
    [i for i in gdf['id']]
):
    plt.text(
        x, y, t, verticalalignment='center', horizontalalignment='center'
    )
# Plot weights connectivity
wr.plot(gdf, edge_kws=dict(color='r', linestyle=':'), ax =ax)
# Remove axes
ax.set_axis_off()
```



(b)

Figura 1.8. Contiguidade para uma grade 3x3: (a) *rook* e (b) *queen*.

Outra forma de critério de proximidade na definição dos pesos espaciais é a distância geográfica. A ideia por trás é que duas regiões mais próximas geograficamente têm uma maior interação espacial. Uma matriz w muito adotada na literatura é a matriz dos k vizinhos mais próximos (*KNN*). Na Figura 1.9 temos um exemplo com três vizinhos para cada região (parâmetro $k = 3$ na linha 2).

Vamos agora definir matrizes de vizinhança utilizando os critérios *queen*, *rook* e *knn* para os setores censitários urbanos da cidade de São Bernardo do Campo, no interior de São Paulo. Primeiro devemos selecionar os respectivos setores censitários urbanos da cidade (linhas de 1 a 3 da Figura 1.10) e em seguida criar as matrizes de pesos para os respectivos critérios (linhas de 6 a 8 da Figura 1.10). De posse dos setores e das matrizes podemos plotar os mapas. A Figura 1.11 temos o código para gerar os respectivos mapas e na Figura 1.12) temos os mapas dos setores censitários de São Bernado do Campo com os critérios de vizinhança *queen*, *rook* e *knn*, respectivamente.

1.4. Mapas coropléticos

Agora que entendemos os processos geográficos e os dados que os mensuram, apresentaremos a AEDE. A AEDE aumenta a análise exploratória de dados de Tukey [Tukey 1977]

```
# Build a regular 3x3 lattice and draw it here
wr = weights.KNN.from_dataframe(gdf, k=3)

# Set up figure
f,ax = plt.subplots(1,1, subplot_kw=dict(aspect='equal'))
# Plot grid
gdf.plot(facecolor='w', edgecolor='k', ax=ax)
# Loop over each cell and add the text
for x, y, t in zip(
    [p.centroid.x-.25 for p in polys],
    [p.centroid.y-.25 for p in polys],
    [i for i in gdf['id']]
):
    plt.text(
        x, y, t, verticalalignment='center', horizontalalignment='center'
    )
# Plot weights connectivity
wr.plot(gdf, edge_kws=dict(color='r', linestyle=':'), ax =ax)
# Remove axes
ax.set_axis_off()
```

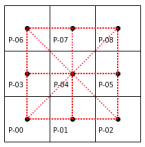


Figura 1.9. Contiguidade *knn* com três vizinhos para uma grade 3x3.

```
sao_bernardo_campo = setores_censitarios_sp.query(
    "NM_MUNICIP == 'SÃO BERNARDO DO CAMPO' and TIPO =='URBANO'"
)
sao_bernardo_campo.reset_index(drop=True, inplace=True)

w_queen = weights.contiguity.Queen.from_dataframe(sao_bernardo_campo)
w_rook = weights.contiguity.Queen.from_dataframe(sao_bernardo_campo)
w_knn = weights.distance.KNN.from_dataframe(sao_bernardo_campo, k=4)
```

Figura 1.10. Código para seleção dos setores censitários urbanos da cidade de São Bernardo do Campo com a definição da matriz de vizinhança *queen*, *rook* e *knn*

e envolve uma grande coleção de técnicas usadas para "orientar-se" (encontrar estrutura) dentro de seu conjunto de dados. Para problemas geográficos, isso geralmente envolve entender se nossos dados exibem um padrão geográfico.

Mapas coropléticos são mapas geográficos que exibem informações estatísticas codificadas em uma paleta de cores. Os mapas coropléticos desempenham um papel proeminente na ciência de dados geográficos, pois nos permitem exibir atributos ou variáveis não geográficas em um mapa geográfico e é a forma usual de apresentação de dados agregados por áreas [Câmara et al. 2004].

A eficácia de um mapa coroplético depende na maioria do propósito do mapa. Qual mensagem você deseja comunicar moldará quais opções são preferíveis em relação a outras. Podemos considerar três dimensões sobre as quais colocar o pensamento intencional valerá a pena. Os mapas coropléticos gira em torno de: primeiro, selecionar um número de grupos menor que n em que todos os valores em nosso conjunto de dados serão mapeados; segundo, identificar um algoritmo de classificação que execute tal mapeamento, seguindo algum princípio alinhado ao nosso interesse; e terceiro, uma vez que sabemos em quantos grupos reduziremos todos os valores em nossos dados, qual cor é atribuída a cada grupo para garantir que ele codifique as informações que queremos refletir. Em termos gerais, o esquema de classificação define o número de classes, bem como as regras de atribuição; enquanto uma boa simbolização transmite informações sobre a diferenciação de valor entre as classes.

```
# Set up figure and axis
f, axs = plt.subplots(1, 3, figsize=(16, 10))

# Queen
ax = axs[0]
sao_bernardo_campo.plot(ax=ax)
w_queen.plot(
    sao_bernardo_campo,
    edge_kws=dict(linewidth=1, color='orangered'),
    node_kws=dict(marker='*'),
    ax=ax
)
ax.set_axis_off()
ax.set_title('Queen')

# Rook
ax = axs[1]
sao_bernardo_campo.plot(ax=ax)
w_rook.plot(
    sao_bernardo_campo,
    edge_kws=dict(linewidth=1, color='orangered'),
    node_kws=dict(marker='*'),
    ax=ax
)
ax.set_axis_off()
ax.set_title('Rook')

# KNN
ax = axs[2]
sao_bernardo_campo.plot(ax=ax)
w_knn.plot(
    sao_bernardo_campo,
    edge_kws=dict(linewidth=1, color='orangered'),
    node_kws=dict(marker='*'),
    ax=ax
)
ax.set_axis_off()
ax.set_title('KNN 4');
```

Figura 1.11. Código para a geração dos mapas com os setores censitários urbanos da cidade de São Bernardo do Campo com os critérios de vizinhança *queen*, *rook* e *knn*

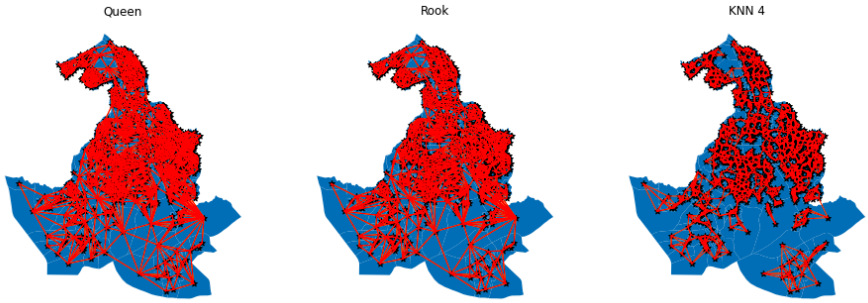


Figura 1.12. Mapas com os setores censitários urbanos da cidade de São Bernardo do Campo com critérios de vizinhança *queen*, *rook* e *knn*

1.4.1. Classificação de dados quantitativos

Selecionar o número de grupos aos quais queremos atribuir os valores em nossos dados e como cada valor é atribuído a um grupo seja um problema de classificação. A classificação dos dados considera o problema de particionar os valores dos atributos em grupos mutuamente exclusivos e exaustivos. A maneira precisa como isso é feito será função da escala de medição do atributo em questão. Para atributos quantitativos (escalas ordinais, intervalares, de razão) as classes terão uma ordenação explícita. Mais formalmente, o problema de classificação é definir limites de classe de tal forma que:

$$c_j < u_i \leq c_{i+1} \forall y_i \in C_j \tag{2}$$

onde y_i é o valor do atributo para localização espacial i, j é um índice de classe, e c_j representa o limite inferior do intervalo j. Diferentes esquemas de classificação são obtidos a partir de sua definição dos limites de classe. A escolha do esquema de classificação

deve levar em consideração a distribuição estatística dos valores dos atributos, bem como o objetivo do nosso mapa (por exemplo, destacar *outliers* versus retratar com precisão a distribuição dos valores).

Existem diversos algoritmos de classificação disponíveis para classificação que seguem critérios que podem ser de interesse em diferentes contextos, pois focam em diferentes prioridades. Abaixo, vamos nos concentrar em alguns deles. Para calcular a classificação, vamos basear no `mapclassify` da família *Pysal*.

Para testar os diferentes algoritmos de classificação disponibilizados no `mapclassify` vamos criar um `GeoDataFrame` com as malhas dos municípios brasileiros disponibilizadas pelos IBGE, com os casos confirmados de covid-19 disponibilizados em `https://brasil.io/dataset/covid19/caso_full/`. O código com a criação desse `GeoDataFrame` pode ser observado na Figura 1.13.

```
municipios_br = gpd.read_file('BR_Municipios_2021.zip')
municipios_br['CD_MUN'] = municipios_br['CD_MUN'].astype(int)

casos_confirmados_covid_br = pd.merge(
    casos_confirmados_covid,
    municipios_br,
    left_on='city_ibge_code',
    right_on='CD_MUN',
)

casos_confirmados_covid_br = gpd.GeoDataFrame(
    casos_confirmados_covid_br,
    crs = municipios_br.crs
)

casos_confirmados_covid_br.head()
```

	city_ibge_code	new_confirmed	CD_MUN	NM_MUN	SIGLA	AREA_KM2	geometry
0	1100015	4089	1100015	Alta Floresta D'Oeste	RO	7067.127	POLYGON ((-62.19465 -11.82746, -62.19332 -11.8...
1	1100023	22657	1100023	Ariquemes	RO	4426.571	POLYGON ((-62.53648 -9.73222, -62.52765 -9.736...
2	1100031	918	1100031	Cabixi	RO	1314.352	POLYGON ((-60.37119 -13.36655, -60.37134 -13.3...
3	1100049	14904	1100049	Cacoal	RO	3793.000	POLYGON ((-61.00080 -11.29737, -61.00103 -11.3...
4	1100056	2570	1100056	Cerejeiras	RO	2783.300	POLYGON ((-61.49976 -13.00525, -61.49809 -13.0...

Figura 1.13. Criação do GeoDataFrame com os casos de covid-19 agregados pela malha dos municípios brasileiros.

1.4.1.1. Intervalos iguais

A abordagem de Freedman-Diaconis [Freedman and Diaconis 1981] fornece uma regra para determinar a largura e, por sua vez, o número de caixas para a classificação. Este é um caso especial de um classificador mais geral conhecido como "intervalos iguais", onde cada uma das caixas tem a mesma largura no espaço de valor. Para um determinado valor de k, a classificação de intervalos iguais divide o intervalo do espaço de atributo em k intervalos de comprimento igual, com cada intervalo tendo uma largura $w = \frac{x_0 - x_{n-1}}{k}$. Assim, a classe máxima é $(x_{n-1} - w, x_{n-1}]$ e a primeira classe é $(-\infty, x_{n-1} - (k-1)w]$.

Intervalos iguais têm as vantagens duplas de simplicidade e facilidade de interpretação. No entanto, essa regra considera apenas os valores extremos da distribuição e, em alguns casos, isso pode resultar em uma ou mais classes esparsas.

Este é claramente o caso em nosso conjunto de dados de covid-19, criado na Figura 1.13, pois a maioria dos valores é colocada na primeira classe, deixando as demais classes bastante esparsas (Figura 1.14).

```
ei5 = mapclassify.EqualInterval(casos_confirmados_covid_br['new_confirmed'], k=5)
ei5
```

```
EqualInterval

       Interval          Count
------------------------------
[    10.00, 195250.80] |  5560
(195250.80, 390491.60] |     7
(390491.60, 585732.40] |     1
(585732.40, 780973.20] |     1
(780973.20, 976214.00] |     1
```

Figura 1.14. Distribuição dos casos de covid-19 pela abordagem de intervalos iguais.

Observe na Figura 1.14 que cada um dos intervalos tem igual largura de $w = 195240,8$. Deve-se notar também que a primeira classe é fechada no limite inferior, em contraste com a abordagem geral definida acima.

1.4.1.2. Quantis

Para evitar o problema potencial de classes esparsas, os *quantis* da distribuição podem ser usados para identificar os limites das classes. De fato, cada classe terá aproximadamente $\left\lfloor \frac{n}{k} \right\rfloor$ observações usando o classificador de quantil. Se $k = 5$ os quintis da amostra são usados para definir os limites superiores de cada classe, como pode ser observado na Figura 1.15.

```
q5 = mapclassify.Quantiles(casos_confirmados_covid_br['new_confirmed'], k=5)
q5
```

```
Quantiles

       Interval          Count
------------------------------
[    10.00,    440.00] |  1118
(   440.00,    809.00] |  1113
(   809.00,   1487.80] |  1111
(  1487.80,   3379.00] |  1115
(  3379.00, 976214.00] |  1113
```

Figura 1.15. Distribuição dos casos de covid-19 pela abordagem de *quantis*.

Observe que na Figura 1.15, embora os números de valores em cada classe sejam aproximadamente iguais, as larguras dos quatro primeiros intervalos são bastante diferentes. Embora os *quantis* evitem a armadilha das classes esparsas, essa classificação não está livre de problemas. As larguras variáveis dos intervalos podem ser marcadamente diferentes, o que pode levar a problemas de interpretação. Um segundo desafio enfrentado pelos *quantis* surge quando há um grande número de valores duplicados na distribuição, de modo que os limites para uma ou mais classes se tornam ambíguos. Por exemplo, se alguém tivesse uma variável com $n = 20$ mas 10 das observações assumiram o mesmo valor que foi o mínimo observado, então para valores de $k > 2$, os limites de classe tornam-se mal definidos, pois uma simples regra de divisão n/k no valor observado classificado dependeria de como os empates são tratados na classificação.

Geraremos uma variável sintética com essas características (Figura 1.16).

E agora executaremos a classificação com a abordagem de *quantil* (Figura 1.15).

Nesse caso, é formado uma classificação com um número menor de classes usando pseudo quantis, ou quantis definidos nos valores únicos da amostra.

```
# Set seed for reproducibility
np.random.seed(12345)
# Generate a variable of 20 values randomly
# selected from 0 to 10
x = np.random.randint(0,10,20)
# Manually ensure the first ten values are 0 (the
# minimum value)
x[0:10] = x.min()
x
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 7, 6, 0, 2, 9, 1, 2, 6, 7])
```

Figura 1.16. Código com a geração de um conjunto com valores repetidos.

```
ties = mapclassify.Quantiles(x, k=5)
ties
```

```
Quantiles

  Interval     Count
--------------------
[0.00, 0.00] |    11
(0.00, 1.40] |     1
(1.40, 6.20] |     4
(6.20, 9.00] |     4
```

Figura 1.17. Distribuição pela abordagem de *quantis* para o conjunto com valores repetidos.

1.4.1.3. Jenks Caspall

Originalmente proposto por [Jenks and Caspall 1971], aborda o desafio da classificação de uma perspectiva heurística, e não determinística, visando minimizar a soma dos desvios absolutos em torno das médias das classes. A abordagem começa com um número pré-especificado de classes e um conjunto inicial arbitrário de quebras de classe - por exemplo, usando *quintis*. O algoritmo tenta melhorar a função objetivo considerando o movimento de observações entre classes adjacentes. Por exemplo, o maior valor no *quintil* mais baixo seria considerado para o movimento para o segundo quintil, enquanto o valor mais baixo no segundo *quintil* seria considerado para um possível movimento para o primeiro *quintil*. O movimento candidato que resultar na maior redução na função objetivo seria feito, e o processo continua até que nenhum outro movimento de melhoria seja possível. O algoritmo Jenks Caspall é o caso unidimensional do algoritmo *K-Means* amplamente utilizado para agrupamento. Na Figura 1.18 temos a aplicação dessa abordagem para o conjunto de dados de casos de covid-19.

```
jc5 = mapclassify.JenksCaspall(casos_confirmados_covid_br['new_confirmed'], k=5)
jc5
```

```
JenksCaspall

       Interval          Count
------------------------------
[    10.00,   1675.00] |  3517
(  1675.00,   6209.00] |  1417
(  6209.00,  22031.00] |   470
( 22031.00,  80865.00] |   135
( 80865.00, 976214.00] |    31
```

Figura 1.18. Distribuição dos casos de covid-19 pela abordagem Jenks Caspall.

1.4.1.4. Fisher Jenks

O segundo algoritmo ótimo adota uma abordagem de programação dinâmica para minimizar a soma dos desvios absolutos em torno das medianas de classe. Em contraste com

a abordagem Jenks-Caspall, o Fisher-Jenks garante a produção de uma classificação ideal para um número pré-especificado de classes. Na Figura 1.19 temos a aplicação dessa abordagem para o conjunto de dados de casos de covid-19.

```
fj5 = mapclassify.FisherJenks(casos_confirmados_covid_br['new_confirmed'], k=5)
fj5
```

```
FisherJenks

      Interval          Count
------------------------------
[    10.00,  21053.00] |  5401
( 21053.00,  98093.00] |   145
( 98093.00, 278370.00] |    21
(278370.00, 597871.00] |     2
(597871.00, 976214.00] |     1
```

Figura 1.19. Distribuição dos casos de covid-19 pela abordagem Fisher Jenks.

1.4.1.5. Comparando esquemas de classificação

Como um caso especial de agrupamento, a definição do número de classes e os limites de classe representam um problema para o projetista do mapa. Para a classificação de mapas, um critério de otimização comum é uma medida de ajuste. No `mapclassify`, o "desvio absoluto em torno das medianas de classe" (do inglês *absolute deviation around class medians* - ADCM) é calculado e fornece uma medida de ajuste que permite a comparação de classificadores alternativos para o mesmo valor de k. O ADCM nos dará uma noção de quão "compacto" é cada grupo. Para ver isso, podemos comparar diferentes classificadores para k sobre os dados de casos de covid-19 (Figura 1.20.

```
# Bunch classifier objects
class5 = ei5, q5, msd, bp, jc5, fj5
# Collect ADCM for each classifier
fits = np.array([ c.adcm for c in class5])
# Convert ADCM scores to a DataFrame
adcms = pd.DataFrame(fits)
# Add classifier names
adcms['classifier'] = [c.name for c in class5]
# Add column names to the ADCM
adcms.columns = ['ADCM', 'Classifier']
ax = sns.barplot(
    y='Classifier', x='ADCM', data=adcms, palette='Pastel1'
)
```

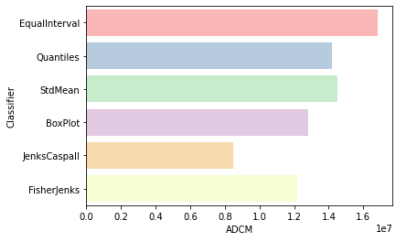


Figura 1.20. ADC dos classificadores usados na base de dados de casos de covid-19.

É possível observar na Figura 1.20 que o classificador Jenks-Caspall domina todos os outros classificadores para $k = 5$ com um ADCM de $0.848 \times e^7$ (lembre-se, quanto menor, melhor).

1.5. Autocorrelação Espacial Global

A noção de autocorrelação espacial refere-se à existência de uma "relação funcional entre o que acontece em um ponto do espaço e o que acontece em outro" [Anselin 1988].

A autocorrelação espacial, portanto, tem a ver com o grau onde a similaridade de valores entre observações em um conjunto de dados está relacionada à similaridade nas localizações de tais observações. Isso é semelhante à ideia tradicional de correlação entre duas variáveis, que nos informa sobre como os valores de uma variável mudam em função dos da outra, embora com algumas diferenças. De maneira semelhante, a autocorrelação espacial também está relacionada (mas distinta) à contraparte temporal, a autocorrelação temporal, que relaciona o valor de uma variável em um determinado momento com os de períodos anteriores. Em contraste com essas outras ideias de correlação, a autocorrelação espacial relaciona o valor da variável de interesse em um determinado local, com valores da mesma variável em outros locais. Uma forma alternativa de entender o conceito é como o grau de informação contido no valor de uma variável em um determinado local sobre o valor dessa mesma variável em outros locais [Almeida 2012, Andrade et al. 2007, Anselin et al. 2013, Anselin 2005].

Para entender melhor a noção de autocorrelação espacial, é útil começar considerando como é o mundo na sua ausência. Uma ideia chave neste contexto é a da aleatoriedade espacial: uma situação onde a localização de uma observação não fornece nenhuma informação sobre o seu valor. Em outras palavras, uma variável é espacialmente aleatória se sua distribuição não segue um padrão espacial discernível. A autocorrelação espacial pode assim ser definida como a "ausência de aleatoriedade espacial" [Almeida 2012, Andrade et al. 2007, Anselin et al. 2013, Anselin 2005].

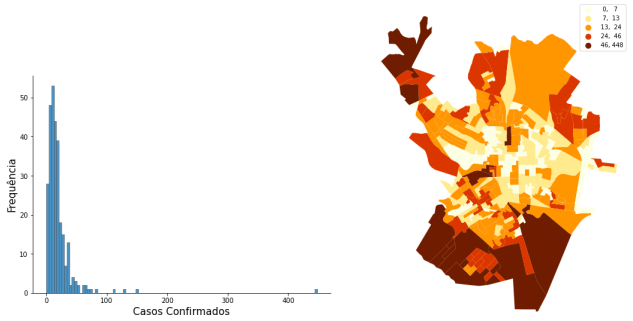
Para ilustrar a noção de autocorrelação espacial e suas diferentes variantes, utilizaremos as notificações confirmadas de casos de dengue do município de São Carlos, localizado na região central do estado de São Paulo, no ano de 2019, disponibilizados pela vigilância epidemiológica do município. Antes, importaremos todas as bibliotecas relevantes que usaremos ao longo desta seção (Figura 1.21).

```
# Graphics
import matplotlib.pyplot as plt
import seaborn
from pysal.viz import splot
from splot.esda import plot_moran
import contextily
# Analysis
import geopandas
import pandas
from pysal.explore import esda
from pysal.lib import weights
from numpy.random import seed
```

Figura 1.21. Importação das bibliotecas necessárias para análise da correlação espacial.

Para gerar essa base de dados de casos de dengue, foram considerados todos os casos confirmados de dengue notificados no Sistema de Informação de Agravos de Notificação - Sinan Dengue/Chikungunya dos residentes da área urbana do município de São Carlos-SP no período de 1 de janeiro à 31 de dezembro do ano de 2019. Essas notificações foram georreferenciadas e agregadas aos setores censitários por meio de uma função de *join spatial*. Na Figura 1.22(a) temos o histograma com a frequência de casos confirmados de dengue nos setores censitários e na Figura 1.22(b) é apresentado distribuição espacial dos casos confirmados de dengue, classificados segundo o algoritmo de *Jenks Caspall*.

Antes de nos aprofundarmos na autocorrelação, precisamos gerar a matriz de pesos espaciais. Usaremos oito vizinhos mais próximo, vale ressaltar que outros critérios



(a) (b)

Figura 1.22. (a) Histograma com a frequência de casos confirmados e (b) Representação coroplética da distribuição espacial dos casos confirmados de dengue.

também podem ser utilizados. Na Figura 1.23 temos o código necessário para gerar a nossa matriz de pesos espacial.

```
# Generate W from the GeoDataFrame
w = weights.KNN.from_dataframe(dengue_sc, k=8)
# Row-standardization
w.transform = 'R'
```

Figura 1.23. Código necessário para gerar a nossa matriz de pesos espacial.

Ao analisar um mapa como o da Figura 1.22(b), estamos buscando padrões do ponto de vista espacial e um dos aspectos necessários que temos que responder, como já mencionado, é verificar se o evento em estudo e os fatores relacionados a ele possuem distribuição espacialmente condicionada (dependência espacial), ou seja, se os valores do atributo em estudo em uma determinada região depende ou não dos valores deste atributo nas regiões vizinhas, tarefa extremamente difícil de se realizar visualmente [Almeida 2012].

1.5.1. Estatística *I* de Moran

[Moran 1948] propôs a elaboração de um coeficiente de autocorrelação espacial, usando a média de autocovariância na forma de produto cruzado. Assim surgia o primeiro coeficiente de autocorrelação espacial, denominado de *I* de Moran, definido pela Equação 3.

$$I = \frac{n}{\sum_i \sum_j w_{ij}} \frac{\sum_i \sum_j w_{ij} z_i z_j}{\sum_i z_i^2} \tag{3}$$

em que n é o número de observações, ${}_z i$ é o valor padronizado da variável de interesse na região i, e ${}_{wij}$ é a célula correspondente à i-ésima linha da j-ésima coluna da matriz de pesos espaciais.

A estatística *I* de Moran é a estatística mais difundida e constitui em uma espécie de coeficiente de autocorrelação, ou seja, é a relação da autocovariância do tipo produto cruzado pela variância dos dados [Andrade et al. 2007, Almeida 2012].

Uma abordagem alternativa para entender a intuição por trás de sua matemática, é através de uma interpretação gráfica e essa interpretação gráfica pode ser efetu-

ada através do diagrama de dispersão de Moran, também conhecido como Moran Plot. O Moran Plot é uma maneira de visualizar um conjunto de dados espaciais para explorar a natureza e a força da autocorrelação espacial. É essencialmente um gráfico de dispersão tradicional em que a variável de interesse é exibida em relação à defasagem espacial da variável de interesse. Para poder interpretar valores como acima ou abaixo da média, a variável de interesse geralmente é padronizada subtraindo sua média [Andrade et al. 2007, Almeida 2012].

A Figura 1.24 mostra a relação entre a porcentagem padronizada casos confirmados de dengue e sua defasagem espacial que, devido à padronização por linha de w, pode ser interpretado como a densidade padronizada média dos casos confirmados de dengue na vizinhança de cada observação. A fim de orientar a interpretação do gráfico, um ajuste linear também é incluído. Essa linha representa o melhor ajuste linear ao gráfico de dispersão ou, em outras palavras, qual é a melhor forma de representar a relação entre às duas variáveis como uma linha reta.

```
# spacial lag
dengue_sc['dengue-2019_lag'] = weights.spatial_lag.lag_spatial(
    w, dengue_sc['dengue-2019']
)

dengue_sc['dengue-2019_std'] = ( dengue_sc['dengue-2019'] - dengue_sc['dengue-2019'].mean() )
dengue_sc['dengue-2019_lag_std'] = ( dengue_sc['dengue-2019_lag'] - dengue_sc['dengue-2019_lag'].mean() )

# Moran plot
f, ax = plt.subplots(1, figsize=(6, 6))
sns.regplot(
    x='dengue-2019_std',
    y='dengue-2019_lag_std',
    ci=None,
    data=dengue_sc,
    line_kws={'color':'r'}
)
ax.axvline(0, c='k', alpha=0.5)
ax.axhline(0, c='k', alpha=0.5)
ax.set_title('Moran Plot - % Leave')
plt.show()
```

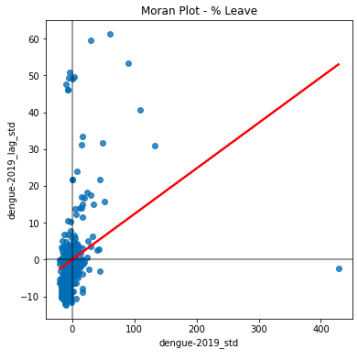


Figura 1.24. Gráfico de dispersão de Moran.

O gráfico da Figura 1.24 mostra uma relação positiva entre ambas as variáveis. Isso indica a presença de autocorrelação espacial positiva: valores semelhantes tendem a se localizar próximos uns dos outros. Isso significa que a tendência geral é que os valores altos estejam próximos de outros valores altos e que os valores baixos sejam cercados por outros valores baixos. Isso, no entanto, não significa que este seja o único caso no conjunto de dados: é claro que pode haver situações particulares em que valores altos são cercados por valores baixos e vice-versa. Mas isso significa que, se tivéssemos que resumir o padrão principal dos dados em termos de quão agrupados são os valores semelhantes, a melhor maneira seria dizer que eles são positivamente correlacionados, portanto, agrupados no espaço.

Para calcular a estatística *I* de Moran em nosso conjunto de dados, podemos chamar uma função específica do pacote `esda` diretamente. Na Figura 1.25 temos o cálculo da estatística *I* de Moran através do pacote `esda` com seu respectivo valor.

```
1 moran_dengue_2019 = esda.moran.Moran(dengue_sc['dengue-2019'], w)
2 moran_dengue_2019.I
```

```
0.12375745247599824
```

Figura 1.25. Cálculo da estatística *I* de Moran e seu respectivo valor.

A outra informação que pode ser extraída da estatística *I* de Moran está relacionada à inferência estatística, ou seja, qual a probabilidade do padrão que observamos no mapa e que a estatística *I* de Moran captura em seu valor ser gerado por um processo inteiramente aleatório? Se considerássemos a mesma variável, mas embaralhemos suas localizações aleatoriamente, obteríamos um mapa com características semelhantes? Para obter *insights* sobre essas questões, o pacote `esda` realiza uma simulação e retorna uma medida de certeza sobre a probabilidade de obter um padrão como o que observamos em um processo espacialmente aleatório. Isso está resumido no atributo `p_sim` (Figura 1.26).

```
1 moran_dengue_2019.p_sim
```

```
0.001
```

Figura 1.26. Valor do *p_sim* para os casos confirmados de dengue.

O valor é calculado como um $p-valor$ empírico que representa a proporção de realizações na simulação sob aleatoriedade espacial que são mais extremas do que o valor observado. Um valor de p suficientemente pequeno associado à estatística *I* de Moran de um mapa permite rejeitar a hipótese de que o mapa é aleatório. Em outras palavras, podemos concluir que o mapa apresenta mais padrão espacial do que esperávamos se os valores tivessem sido alocados aleatoriamente para um local.

O `pysal` disponibiliza um módulo de visualização que combina o Moran Plot (direita) com um gráfico do teste empírico que realizamos para obter `p_sim` (esquerda), chamado de `plot_moran` e sua utilização pode ser observado na Figura 1.27.

1.5.2. Outros índices globais

A estatística *I* de Moran é provavelmente a estatística mais utilizada para autocorrelação espacial global, porém não é a única. Apresentamos duas medidas adicionais comuns no trabalho aplicado. Embora todos considerem a autocorrelação espacial, diferem na forma como o conceito é abordado na especificação de cada teste.

Índice *C* de Geary

A razão de contiguidade, proposto por [Geary 1954], é dado pela Equação 4.

$$C = \frac{(n-1)}{2\sum_i \sum_j w_{ij}} \frac{\sum_i \sum_j w_{ij}(y_i - y_j)^2}{\sum_i (y_i - \bar{y})^2} \tag{4}$$

```
plot_moran(
    moran_dengue_2019,
    zstandard=True,
    figsize=(10,5)
);
```

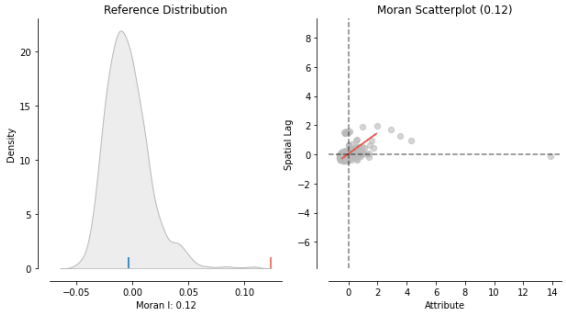


Figura 1.27. *plot_moran* módulo de visualização disponibilizado pelo *pysal*.

onde n é o número de observações, w_{ij} é a célula em uma matriz binária W expressando se i e j são vizinhos ($w_{ij} = 1$) ou não ($w_{ij} = 0$), y_i é a i-ésima observação da variável de interesse, e $\bar{y}$ é sua média amostral. Quando comparado à estatística I de Moran, fica evidente que ambas as medidas comparam a relação de Y dentro da vizinhança local de cada observação para aquela em toda a amostra. No entanto, também existem diferenças sutis. Enquanto a estatística I de Moran usa produtos cruzados nos valores padronizados, o índice C de Geary usa diferenças nos valores sem qualquer padronização.

Computacionalmente, o índice C de Geary é mais exigente, mas pode ser facilmente calculado usando o pacote `esda` do `pysal`, a Figura 1.28 mostra a sua utilização.

```
geary = esda.geary.Geary(dengue_sc['dengue-2019'], w)
geary.C
```

```
0.7905555768777744
```

Figura 1.28. Determinação do índice *C* de Geary através do pacote *esda* do *pysal*.

A inferência é realizada semelhantemente estatística I de Moran. Podemos realizar uma simulação que nos permite traçar uma distribuição empírica da estatística sob a nulidade de aleatoriedade espacial e depois compará-la com a estatística obtida ao usar a distribuição geográfica observada de os dados. Para acessar o pseudo valor-p, calculado como no caso de Moran, podemos chamar `p_sim` (Figura 1.29).

```
geary.p_sim
```

```
0.035
```

Figura 1.29. Valor do *p_sim* do índice *C* de Geary para os casos confirmados de dengue.

Nesse caso, o índice C de Geary aponta na mesma direção que a estatística I de Moran, ou seja, há uma clara indicação de que a estatística que calculamos no conjunto de dados observado difere que seria esperado em uma situação de pura aleatoriedade espacial. Assim, a partir desta análise, podemos concluir também que a autocorrelação espacial está presente.

Índice *G* de Getis e Ord

Originalmente proposto por [Getis and Ord 2010], o índice G é uma medida de autocorrelação espacial de natureza global. Na primeira versão dessa estatística, ela é computada para apenas valores positivos de uma variável por definir um conjunto de vizinhos para cada região como aquelas observações que fiquem dentre de uma distância de corte (*cutt-off*) fiza (d) da região [Almeida 2012]. O índice G de Getis e Ord é definida pela Equação 5.

$$G(d) = \frac{\sum_i \sum_j w_{ij}(d) y_i y_j}{\sum_i \sum_j y_i y_j} \tag{5}$$

onde w_{ij} é o peso binário atribuído na relação entre as observações i e j seguindo um critério de distância.

Para ilustrar seu cálculo, vamos calcular uma matriz de distância binária W. Para garantir que cada observação tenha pelo menos um vizinho, usaremos o método *min_threshold_distance* e projetaremos o conjunto de dados no *Ordnance Survey* CRS (código EPSG 27700), expresso em metros, esse procedimento pode ser observado na Figura 1.30.

```
1 dengue_sc_osgb = dengue_sc.to_crs(epsg=27700)
2 pts = dengue_sc_osgb.centroid
3 xys = pd.DataFrame({'X': pts.x, 'Y': pts.y})
4 min_thr = weights.util.min_threshold_distance(xys)
5 min_thr
```

```
2143.9486593791926
```

Figura 1.30. Determinação da matriz distância binária W.

Para que cada setor censitário tenha um vizinho, a faixa de distância deve ser de, pelo menos, cerca de 2143 metros. Essas informações podem ser passadas para no construtor `DistanceBand` (linha 1 da Figura 1.31).

```
1 w_dengue_sc = weights.DistanceBand.from_dataframe(dengue_sc_osgb, min_thr)
2 gao = esda.getisord.G(dengue_sc['dengue-2019'], w_dengue_sc)
3 print("Getis & Ord G: %.3f | Pseudo P-value: %.3f"%(gao.G, gao.p_sim))
```

```
Getis & Ord G: 0.123 | Pseudo P-value: 0.088
```

Figura 1.31. Determinando a distância miníma para os setores censitários e o índice *G* de Getis e Ord.

O acesso à estatística (`gao.G`) e atributos adicionais podem ser obtidos da mesma forma que com as estatísticas anteriores.

Da mesma forma, a inferência também pode ser realizada por meio de simulações computacionais que replicam várias instâncias de aleatoriedade espacial usando os valores da variável de interesse, mas embaralhando suas localizações. Neste caso, o pseudo valor p calculado sugere um claro afastamento da hipótese de não concentração.

1.6. Autocorrelação Espacial Local

As estatísticas globais de autocorrelação espaciais, vista na Seção anterior, fornecem padrões de associação linear espacial, ou seja, o grau em que o conjunto dos dados está agrupado, disperso ou distribuído aleatoriamente [Almeida 2012, Fotheringham et al. 2000].

A presença de autocorrelação espacial tem implicações importantes para análises estatísticas subsequentes. De uma perspectiva substantiva, a autocorrelação espacial poderia refletir a operação de processos que geram associação entre os valores em localidades próximas. Isso pode representar transbordamentos, onde os resultados em um local influenciam outros locais ou pode indicar contágio, onde os resultados em um local influenciam causalmente outros locais.

Apesar de sua importância, as medidas globais de autocorrelação espacial são estatísticas de "mapa inteiro". Eles fornecem um único resumo para um conjunto de dados inteiro, ou seja, as medidas pode nos dizer se os valores em nosso mapa se agrupam (ou se dispersam) em geral, mas não nos informará sobre onde estão os agrupamentos específicos (ou valores discrepantes) [Almeida 2012, Fotheringham et al. 2000].

1.6.1. *I* de Moran local

Proposto por [Anselin 1995], os Indicadores Locais de Associação Espacial (do inglês, *Local Indicators of Spatial Association* - LISA), visa identificar casos em que o valor de uma observação e a média de seus arredores são mais semelhantes (alto-alto, do inglês *high-high* - HH ou baixo-baixo, do inglês *low-low* - LL no gráfico de dispersão de Moran) ou diferentes (alto-baixo, do inglês *high-low* - HL ou baixo-alto, do inglês *low-high* - LH) do que esperaríamos do puro acaso. O mecanismo para fazer isso é semelhante ao do índice *I* de Moran global, mas aplicado neste caso a cada observação. Isso resulta em tantas estatísticas quanto as observações originais. A representação formal da estatística pode ser escrita pela Equação 6.

$$I_i = \frac{z_i}{m_2} \sum_j w_{ij} z_j ; m_2 = \frac{\sum_i z_i^2}{n} \tag{6}$$

onde m_2 é o segundo momento (variância) da distribuição de valores nos dados, $z_i = u_i - \bar{y}, w_{ij}$, é o peso espacial para o par de observações i e j, e n é o número de observações.

LISA é amplamente utilizada em muitos campos para identificar agrupamentos geográficos de valores ou encontrar discrepâncias geográficas. É uma ferramenta útil que pode retornar rapidamente as áreas em que os valores estão concentrados e fornecer evidências sugestivas sobre os processos que podem estar em ação. Por esses motivos, eles têm um lugar privilegiado na caixa de ferramentas da ciência de dados geográficos. Entre muitas outras aplicações, LISA têm sido usadas para identificar aglomerados geográficos de pobreza [Dawson et al. 2018], delinear áreas de atividade econômica particularmente alta/baixa [Torres-Preciado et al. 2014] ou identificar aglomerados de doenças contagiosas [Zhang et al. 2020]. É possível determinar os valores de LISA através da chamada da função `Moran_Local` do `esda`, conforme mostra na linha 1 da Figura 1.32. Precisamos passar a variável de interesse - casos confirmados de dengue da cidade de São Carlos-SP e os pesos espaciais que descrevem as relações de vizinhança entre as diferentes áreas que compõem o conjunto de dados. Isso cria um objeto LISA (lisa) que possui vários atributos de interesse. Os próprios indicadores locais estão no atributo `Is`, aonde se pode ter uma noção de sua distribuição.

O objeto LISA (linha 1 da Figura 1.32) contém as estatísticas *I* de Moran para cada setor censitário, com seus respectivos níveis de significância e essa copiosa quantidade

```
lisa = esda.moran.Moran_Local(dengue_sc['dengue-2019'], w)

# Draw KDE line
ax = sns.kdeplot(lisa.Is)
# Add one small bar (rug) for each observation
# along horizontal axis
sns.rugplot(lisa.Is, ax=ax);
```

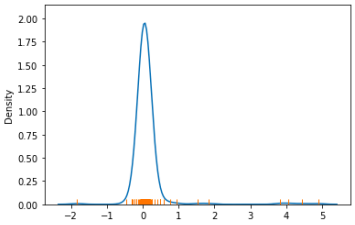


Figura 1.32. Determinação dos índices LISA para os casos confirmados de dengue da cidade de São Carlos-SP.

de informações pode confundir o pesquisador, se colocada em tabelas [Almeida 2012]. Uma forma mais eficiente de apresentar este conjunto de estatísticas é mapeá-las. Na Figura 1.33, o mapa de significância LISA exibe as regiões com estatística I local de Moran significativos para os setores censitários de casos confirmados de dengue para a cidade de São Carlos-SP.

```
dengue_sc['LISA'] = 'não significante'
dengue_sc.loc[lisa.p_sim < 0.05, 'LISA'] = 'p = 0.05'
dengue_sc.loc[lisa.p_sim < 0.01, 'LISA'] = 'p = 0.01'

ax = dengue_sc.plot(
    figsize=(8,8),
    column='LISA',
    legend=True,
    edgecolor='black',
    cmap='gray_r',
)
ax.set_axis_off();
```

Figura 1.33. Mapa de significância LISA para os setores censitários com casos confirmados de dengue da cidade de São Carlos-SP.

O mapa de *clusters* LISA combina a informação do diagrama de dispersão de Moran e a informação do mapa de significância das médias de associação local I_i. O *Pysal* disponibiliza a função `lisa_cluster` do módulo `splot.esda` com essa finalidade, agrupando os *clusters* em HH, HL, LH, LL e ns (não significante). A Figura 1.34 apresenta os *clusters* que passaram no teste de significância estatística ndo I de Moran local.

```
lisa_cluster(
    lisa,
    dengue_sc,
    p=0.05,
    figsize=(6,6),
);
```

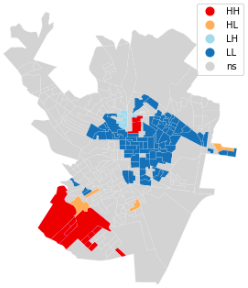


Figura 1.34. Mapa de *clusters* LISA para os setores censitários com casos confirmados de dengue da cidade de São Carlos-SP.

1.6.2. Estatística local de Getis e Ord

Semelhante ao caso global, há mais indicadores locais de correlação espacial do que o índice I de Moran local. O *pysal* inclui estatística local de Getis e Ord do tipo G_i. Estes são um tipo diferente de estatística local que são comumente usados em duas formas: a estatística G_i, que omite o valor em um site em seu resumo local, e a estatística G_i^*, que inclui o próprio valor do site no resumo local. A forma de calculá-los também segue padrões semelhantes às estatísticas do Moran Local.

1.7. Considerações finais

Este capítulo mostrou que as técnicas de análise exploratória de dados espaciais podem ampliar consideravelmente a capacidade de compreender os padrões espaciais associados a dados de área que apresentam autocorrelação espacial global e local. Técnicas exploratórias como os indicadores de Moran e os gráficos de dispersão de Moran são muito úteis para mostrar as agregações espaciais e indicar áreas prioritárias do fenômeno em estudo.

A Seção 1.2, foi apresentado alguns conceitos e características sobre dados espaciais e como manipulá-los através da linguagem Python. Já a Seção 1.3 apresentou os conceitos-chave sobre uma matriz de ponderação espacial, uma matriz quadrada de dimensão n por n que representa o grau de conexão entre as regiões segundo algum critério de proximidade, destacando a influência da região j sobre a região i. A Seção 1.4 apresentou alguns aspectos relacionados aos mapas coropléticos, a forma usual de apresentação de dados agregados por áreas. Na Seção 1.5 apresentamos as estatísticas globais de auto correlação espacial, técnicas utilizadas na tarefa de descobrir se os dados são aleatoriamente distribuídos através do espaço, isto é, se os dados estão correlacionados espacialmente. Por fim, a Seção 1.6 as estatísticas utilizadas para a identificação de padrões locais de autocorrelação espacial.

Disponibilidade de dados e código

O código-fonte utilizado no texto do capítulo, bem como o conjunto de dados estão disponíveis no repositório `shorturl.at/kY129`.

Agradecimentos

Este trabalho foi realizado com apoio financeiro da CAPES (PROEX), CNPQ (processo 312605/2018-8), C4AI, FAPESP (processo 2020/16578-5). Agradecemos também ao ICMC-USP e ao LCR por oferecerem a infraestrutura necessária para o desenvolvimento deste trabalho.

Referências

[Almeida 2012] Almeida, E. (2012). Econometria espacial. *Campinas–SP. Alínea*.

[Andrade et al. 2007] Andrade, A. L., Monteiro, A. M. V., Barcellos, C., Lisboa, E., Acosta, L. M. W., Almeida, M. C. d. M., Brito, M. R. V., Carvalho, M. S., Santos, M. A. d., Cruz, O., et al. (2007). Introdução à estatística espacial para a saúde pública.

[Anselin 1988] Anselin, L. (1988). *Spatial econometrics: methods and models*, volume 4. Springer Science & Business Media.

[Anselin 1995] Anselin, L. (1995). Local indicators of spatial association—lisa. *Geographical analysis*, 27(2):93–115.

[Anselin 2005] Anselin, L. (2005). Exploring spatial data with geodatm: a workbook. *Center for spatially integrated social science*, pages 165–223.

[Anselin et al. 2013] Anselin, L., Florax, R., and Rey, S. J. (2013). *Advances in spatial econometrics: methodology, tools and applications*. Springer Science & Business Media.

[Câmara et al. 2004] Câmara, G., Monteiro, A. M., Fucks, S. D., and Carvalho, M. S. (2004). Análise espacial e geoprocessamento. *Análise espacial de dados geográficos. Brasília: EMBRAPA*, pages 21–54.

[Dawson et al. 2018] Dawson, T., Sandoval, J. O., Sagan, V., and Crawford, T. (2018). A spatial analysis of the relationship between vegetation and poverty. *ISPRS International Journal of Geo-Information*, 7(3):83.

[Domingues et al. 2020] Domingues, A., Silva, F., Santos, L., Souza, R., Coimbra, G., and Loureiro, A. A. F. (2020). Dados geoespaciais: Conceitos e técnicas para coleta, armazenamento, tratamento e visualização. *Sociedade Brasileira de Computação*.

[Fotheringham et al. 2000] Fotheringham, A. S., Brunsdon, C., and Charlton, M. (2000). *Quantitative geography: perspectives on spatial data analysis*. Sage.

[Freedman and Diaconis 1981] Freedman, D. and Diaconis, P. (1981). On the histogram as a density estimator: L 2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 57(4):453–476.

[Geary 1954] Geary, R. C. (1954). The contiguity ratio and statistical mapping. *The incorporated statistician*, 5(3):115–146.

[Getis and Ord 2010] Getis, A. and Ord, J. K. (2010). The analysis of spatial association by use of distance statistics. In *Perspectives on spatial data analysis*, pages 127–145. Springer.

[Haining et al. 1998] Haining, R., Wise, S., and Ma, J. (1998). Exploratory spatial data analysis. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 47(3):457–469.

[Jenks and Caspall 1971] Jenks, G. F. and Caspall, F. C. (1971). Error on choroplethic maps: definition, measurement, reduction. *Annals of the Association of American Geographers*, 61(2):217–244.

[Kluyver et al. 2016] Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J. B., Grout, J., Corlay, S., et al. (2016). *Jupyter Notebooks-a publishing format for reproducible computational workflows.*, volume 2016.

[Lopes et al. 2019] Lopes, G. R., Almeida, A. W. S., Delbem, A., and Toledo, C. F. M. (2019). Introdução à análise exploratória de dados com python. *Minicursos ERCAS ENUCMPI*, 2019:160–176.

[Lopes et al. 2021] Lopes, G. R., Delbem, A. C., and de Sousa, J. B. (2021). Introdução à análise de dados geoespaciais com python. *Sociedade Brasileira de Computação*.

[Monteiro et al. 2004] Monteiro, A. M. V., Câmara, G., Carvalho, M., and Druck, S. (2004). Análise espacial de dados geográficos. *Brasília: Embrapa*.

[Moran 1948] Moran, P. A. (1948). The interpretation of statistical maps. *Journal of the Royal Statistical Society. Series B (Methodological)*, 10(2):243–251.

[Pimentel et al. 2021] Pimentel, J. F., Oliveira, G. P., Silva, M. O., Seufitelli, D. B., and Moro, M. M. (2021). Ciência de dados com reprodutibilidade usando jupyter. *Sociedade Brasileira de Computação*.

[Rey and Anselin 2007] Rey, S. J. and Anselin, L. (2007). PySAL: A Python Library of Spatial Analytical Methods. *The Review of Regional Studies*, 37(1):5–27.

[Tobler 1970] Tobler, W. R. (1970). A computer movie simulating urban growth in the detroit region. *Economic geography*, 46(sup1):234–240.

[Torres-Preciado et al. 2014] Torres-Preciado, V. H., Polanco-Gaytan, M., and Tinoco-Zermeño, M. Á. (2014). Technological innovation and regional economic growth in mexico: a spatial perspective. *The Annals of Regional Science*, 52(1):183–200.

[Tukey 1977] Tukey, J. W. (1977). *Exploratory Data Analysis*. Addison-Wesley, 1st edition.

[Zhang et al. 2020] Zhang, X., Rao, H., Wu, Y., Huang, Y., and Dai, H. (2020). Comparison of spatiotemporal characteristics of the covid-19 and sars outbreaks in mainland china. *BMC infectious diseases*, 20(1):1–7.