## 2.7.3 空间自相关分析

### 2.7.3.1 数据准备——公共健康数据

对数据考虑使用空间相关分析前，通常需要初步判断测量值是否可能具有空间相关性，这不仅在于数据内容，也在于测量点或区域的分布尺度，例如污染气体浓度通常在局部区域（扩散范围）内起作用，此时扩散范围内的多个测量点可能是空间相关的，但是超出了这个扩散范围的多个测量点之间可能就不具有空间相关性，例如，对于AoT测量数据，因为测量节点位于芝加哥全城，点间密度稀疏，因此计算获得的全局和局部空间自相关指数均表明已有测量节点之间在气体浓度这一测量内容上空间相关性不明显。在相关性分析和建立回归模型时均使用到[芝加哥公共健康数据](https://data.cityofchicago.org/Health-Human-Services/Public-Health-Statistics-Selected-public-health-in/iqnk-2tcu)①，初步判断测量内容可能存在空间相关性，因此重新下载最新数据（更新日期为：2022-02-04），用于空间自相关分析。

公共健康数据不包含地理空间信息，但是含有Community Area和Community Area Name关键字段。从[Chicago Data Portal](https://data.cityofchicago.org/)②下载的“Boundaries - Community Areas (current) ”社区范围数据含有地理空间信息（GeoJSON的geometry字段或SHP/Shapefile，KML等格式数据），并含有area\_numbe关键字段，对应公共健康数据的Community Area字段，community对应Community Area Name，只是名称字段需要注意大小写，因此合并数据时，使用area\_numbe和对应的Community Area关键字段。为方便包含有属性的DataFrame格式数据合并包含有地理空间信息的GeoDataFrame格式数据，定义df\_linking\_geometry()函数。

参数定义和参数值的存储使用AttrDict类方法（查看“Cityscapes数据集——参数管理”一节）。定义的子属性包括db用于存储数据库相关参数值；gi用于存储epsg地理空间投影信息；data用于存储待读取的数据文件路径。

* 定义参数和参数值

from database import postSQL2gpd,gpd2postSQL  
from util\_misc import AttrDict  
\_\_C=AttrDict()   
args=\_\_C  
  
\_\_C.db=AttrDict()   
\_\_C.db.UN='postgres'  
\_\_C.db.PW='123456'  
\_\_C.db.DB='AoT\_20220831'  
\_\_C.db.GC='geometry'   
\_\_C.db.db\_info=dict(geom\_col=args.db.GC,myusername=args.db.UN,mypassword=args.db.PW,mydatabase=args.db.DB)  
  
\_\_C.gi=AttrDict()  
\_\_C.gi.Chicago\_epsg=32616  
\_\_C.gi.epsg\_wgs84=4326  
  
\_\_C.data=AttrDict()  
\_\_C.data.public\_health\_statistics="./data/Public\_Health\_Statistics\_Selected\_public\_health\_indicators\_by\_Chicago\_community\_area\_Historical\_202202.csv"  
\_\_C.data.community\_areas="./data/ChicagoCommunityAreas/ChicagoCommunityAreas.shp"

* 读取属性数据——公共健康数据

import pandas as pd  
import geopandas as gpd  
  
pubicHealth\_Statistic\_df=pd.read\_csv(args.data.public\_health\_statistics)  
pubicHealth\_Statistic\_df

|  | **Community Area** | **Community Area Name** | **Birth Rate** | **General Fertility Rate** | **Low Birth Weight** | **Prenatal Care Beginning in First Trimester** | **Preterm Births** | **Teen Birth Rate** | **Assault (Homicide)** | **Breast cancer in females** | **...** | **Childhood Lead Poisoning** | **Gonorrhea in Females** | **Gonorrhea in Males** | **Tuberculosis** | **Below Poverty Level** | **Crowded Housing** | **Dependency** | **No High School Diploma** | **Per Capita Income** | **Unemployment** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | Rogers Park | 16.4 | 62.0 | 11.0 | 73.0 | 11.2 | 40.8 | 7.7 | 23.3 | ... | 0.5 | 322.5 | 423.3 | 11.4 | 22.7 | 7.9 | 28.8 | 18.1 | 23714 | 7.5 |
| **1** | 2 | West Ridge | 17.3 | 83.3 | 8.1 | 71.1 | 8.3 | 29.9 | 5.8 | 20.2 | ... | 1.0 | 141.0 | 205.7 | 8.9 | 15.1 | 7.0 | 38.3 | 19.6 | 21375 | 7.9 |
| **2** | 3 | Uptown | 13.1 | 50.5 | 8.3 | 77.7 | 10.3 | 35.1 | 5.4 | 21.3 | ... | 0.5 | 170.8 | 468.7 | 13.6 | 22.7 | 4.6 | 22.2 | 13.6 | 32355 | 7.7 |
| **3** | 4 | Lincoln Square | 17.1 | 61.0 | 8.1 | 80.5 | 9.7 | 38.4 | 5.0 | 21.7 | ... | 0.4 | 98.8 | 195.5 | 8.5 | 9.5 | 3.1 | 25.6 | 12.5 | 35503 | 6.8 |
| **4** | 5 | North Center | 22.4 | 76.2 | 9.1 | 80.4 | 9.8 | 8.4 | 1.0 | 16.6 | ... | 0.9 | 85.4 | 188.6 | 1.9 | 7.1 | 0.2 | 25.5 | 5.4 | 51615 | 4.5 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **72** | 73 | Washington Heights | 12.0 | 61.0 | 19.6 | 75.4 | 16.2 | 65.0 | 38.0 | 47.9 | ... | 1.5 | 1298.2 | 1274.2 | 3.0 | 15.7 | 1.1 | 42.4 | 15.6 | 19709 | 18.3 |
| **73** | 74 | Mount Greenwood | 12.5 | 59.0 | 8.4 | 94.5 | 15.1 | 7.7 | 2.2 | 34.6 | ... | 0.0 | NaN | . | 0.0 | 3.1 | 1.1 | 37.0 | 4.5 | 34221 | 6.9 |
| **74** | 75 | Morgan Park | 13.2 | 67.5 | 10.6 | 74.5 | 12.3 | 46.7 | 19.9 | 32.4 | ... | 1.3 | 800.5 | 741.1 | 2.6 | 13.7 | 0.8 | 39.4 | 10.9 | 26185 | 14.9 |
| **75** | 76 | O'Hare | 15.8 | 70.0 | 3.5 | 82.0 | 5.0 | 15.9 | 5.6 | 20.5 | ... | 0.5 | NaN | . | 6.3 | 9.5 | 1.9 | 26.5 | 11.0 | 29402 | 4.7 |
| **76** | 77 | Edgewater | 12.1 | 48.1 | 7.5 | 76.1 | 7.4 | 15.1 | 5.8 | 18.5 | ... | 0.9 | 120.1 | 427.5 | 10.5 | 16.6 | 3.9 | 23.4 | 9.0 | 33364 | 9.0 |

77 rows × 29 columns

* 读取含有地理空间信息的数数据————社区范围数据

community\_area\_gdf=gpd.read\_file(args.data.community\_areas)  
community\_area\_gdf.area\_numbe=community\_area\_gdf.area\_numbe.astype("int64")  
community\_area\_gdf.sort\_values(by=["area\_numbe"],inplace=True)  
community\_area\_gdf

|  | **area** | **area\_num\_1** | **area\_numbe** | **comarea** | **comarea\_id** | **community** | **perimeter** | **shape\_area** | **shape\_len** | **geometry** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **9** | 0.0 | 1 | 1 | 0.0 | 0.0 | ROGERS PARK | 0.0 | 5.125990e+07 | 34052.397576 | POLYGON ((-87.65456 41.99817, -87.65574 41.998... |
| **19** | 0.0 | 2 | 2 | 0.0 | 0.0 | WEST RIDGE | 0.0 | 9.842909e+07 | 43020.689458 | POLYGON ((-87.68465 42.01948, -87.68464 42.019... |
| **30** | 0.0 | 3 | 3 | 0.0 | 0.0 | UPTOWN | 0.0 | 6.509564e+07 | 46972.794555 | POLYGON ((-87.64102 41.95480, -87.64400 41.954... |
| **5** | 0.0 | 4 | 4 | 0.0 | 0.0 | LINCOLN SQUARE | 0.0 | 7.135233e+07 | 36624.603085 | POLYGON ((-87.67441 41.97610, -87.67440 41.976... |
| **47** | 0.0 | 5 | 5 | 0.0 | 0.0 | NORTH CENTER | 0.0 | 5.705417e+07 | 31391.669754 | POLYGON ((-87.67336 41.93234, -87.67342 41.932... |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **71** | 0.0 | 73 | 73 | 0.0 | 0.0 | WASHINGTON HEIGHTS | 0.0 | 7.963575e+07 | 42222.598163 | POLYGON ((-87.63373 41.72885, -87.63369 41.726... |
| **72** | 0.0 | 74 | 74 | 0.0 | 0.0 | MOUNT GREENWOOD | 0.0 | 7.558429e+07 | 48665.130539 | POLYGON ((-87.69646 41.70714, -87.69644 41.706... |
| **73** | 0.0 | 75 | 75 | 0.0 | 0.0 | MORGAN PARK | 0.0 | 9.187734e+07 | 46396.419362 | POLYGON ((-87.64215 41.68508, -87.64249 41.685... |
| **74** | 0.0 | 76 | 76 | 0.0 | 0.0 | OHARE | 0.0 | 3.718356e+08 | 173625.984660 | MULTIPOLYGON (((-87.83658 41.98640, -87.83658 ... |
| **75** | 0.0 | 77 | 77 | 0.0 | 0.0 | EDGEWATER | 0.0 | 4.844999e+07 | 31004.830946 | POLYGON ((-87.65456 41.99817, -87.65456 41.998... |

77 rows × 10 columns

* 融合地理空间数据

def df\_linking\_geometry(df,gdf,key\_df,key\_gdf,how="inner"):  
 '''  
 包含有属性的DataFrame格式数据合并包含有地理空间信息的GeoDataFrame格式数据，仅合并“geometry”字段和用于链接的关键字段列  
  
 Parameters  
 ----------  
 df : DataFrame  
 属性数据，含有用于链接的关键字段.  
 gdf : GeoDataFrame  
 还有地理空间信息geometry字段和用于链接的关键字段.  
 key\_df : string  
 属性数据用于链接的关键字段名.  
 key\_gdf : string  
 含有地理空间信息数据用于链接的关键字段名.  
 how : string, optional  
 {‘left’, ‘right’, ‘outer’, ‘inner’, ‘cross’}，具体解释参考pandas.DataFrame.merg. The default is "inner".  
  
 Returns  
 -------  
 gdf\_linked : GeoDataFrame  
 属性数据合并地理空间信息后的数据.  
  
 '''   
 gdf\_linked=gdf[[key\_gdf,"geometry"]].merge(df,left\_on=key\_gdf,right\_on=key\_df,how=how,)  
   
 return gdf\_linked  
   
pubicHealth\_Statistic\_gdf=df\_linking\_geometry(pubicHealth\_Statistic\_df,community\_area\_gdf,key\_df="Community Area",key\_gdf="area\_numbe")   
pubicHealth\_Statistic\_gdf=pubicHealth\_Statistic\_gdf.to\_crs(args.gi.Chicago\_epsg)  
print(f"{pubicHealth\_Statistic\_gdf.crs}")  
pubicHealth\_Statistic\_gdf[:3]

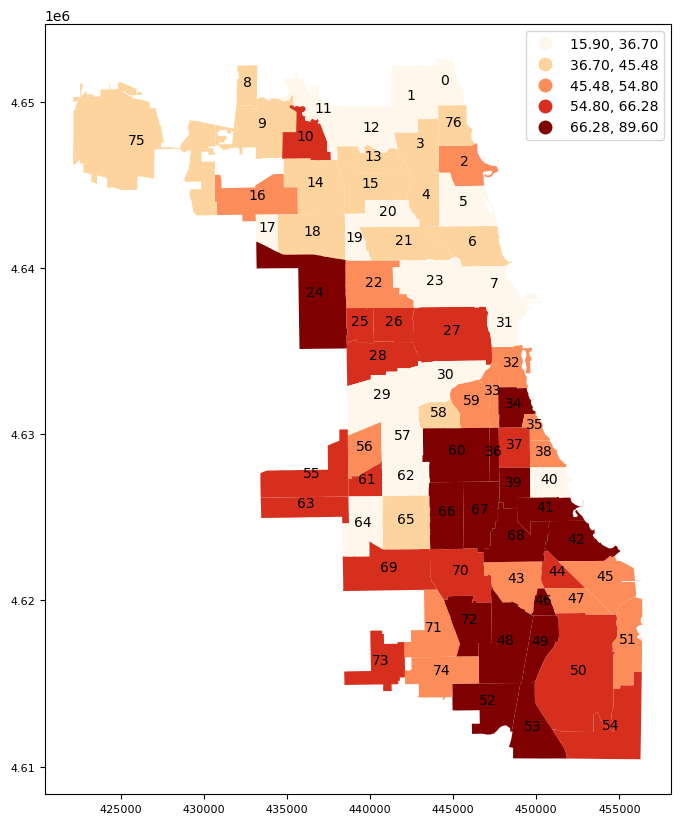
epsg:32616

|  | **area\_numbe** | **geometry** | **Community Area** | **Community Area Name** | **Birth Rate** | **General Fertility Rate** | **Low Birth Weight** | **Prenatal Care Beginning in First Trimester** | **Preterm Births** | **Teen Birth Rate** | **...** | **Childhood Lead Poisoning** | **Gonorrhea in Females** | **Gonorrhea in Males** | **Tuberculosis** | **Below Poverty Level** | **Crowded Housing** | **Dependency** | **No High School Diploma** | **Per Capita Income** | **Unemployment** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | POLYGON ((445789.555 4649779.811, 445691.750 4... | 1 | Rogers Park | 16.4 | 62.0 | 11.0 | 73.0 | 11.2 | 40.8 | ... | 0.5 | 322.5 | 423.3 | 11.4 | 22.7 | 7.9 | 28.8 | 18.1 | 23714 | 7.5 |
| **1** | 2 | POLYGON ((443315.818 4652166.308, 443316.880 4... | 2 | West Ridge | 17.3 | 83.3 | 8.1 | 71.1 | 8.3 | 29.9 | ... | 1.0 | 141.0 | 205.7 | 8.9 | 15.1 | 7.0 | 38.3 | 19.6 | 21375 | 7.9 |
| **2** | 3 | POLYGON ((446874.222 4644956.738, 446627.798 4... | 3 | Uptown | 13.1 | 50.5 | 8.3 | 77.7 | 10.3 | 35.1 | ... | 0.5 | 170.8 | 468.7 | 13.6 | 22.7 | 4.6 | 22.2 | 13.6 | 32355 | 7.7 |

3 rows × 31 columns

经常需要打印地理空间信息数据，显示指定字段的属性值查看分布，同时也会增加标注，例如定位位置信息或相关分类信息等内容，因此定义gdf\_plot\_annotate()函数，方便调用。

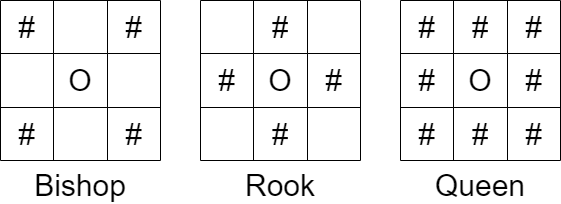
def gdf\_plot\_annotate(gdf\_,value\_column,annotate\_column,\*\*setting):  
 '''  
 打印GeoDataFrame格式地理空间信息数据  
  
 Parameters  
 ----------  
 gdf\_ : GeoDataFrame  
 待打印的数据.  
 value\_column : string  
 数值显示字段名.  
 annotate\_column : string  
 标注显示字段名.  
 \*\*setting : key args  
 用于配置图表的参数，键和默认值如下  
 setting\_dict=dict(annotate\_fontsize=8,  
 figsize=(10,10),   
 legend\_position="right",  
 legend\_size="5%",  
 legend\_pad=0.1,  
 legend\_bbox\_to\_anchor=(1, 1),  
 cmap='OrRd',  
 labelsize=8,  
 scheme=None, # 等值分类图，例如 ‘BoxPlot’, ‘EqualInterval’, ‘FisherJenks’,‘FisherJenksSampled’, ‘HeadTailBreaks’, ‘JenksCaspall’,   
 ‘JenksCaspallForced’, ‘JenksCaspallSampled’, ‘MaxP’, ‘MaximumBreaks’, ‘NaturalBreaks’, ‘Quantiles’,   
 ‘Percentiles’, ‘StdMean’, ‘UserDefined’等  
 k=5, # 分类数量， 对应scheme参数，如果scheme参数为None，则k参数忽略  
 categorical=False # 为True时为分类数据，为False时为数值数据  
 ).  
  
 Returns  
 -------  
 ax : TYPE  
 DESCRIPTION.  
  
 '''   
 import matplotlib.pyplot as plt  
 from mpl\_toolkits.axes\_grid1 import make\_axes\_locatable  
   
 gdf=gdf\_.copy(deep=True)  
 setting\_dict=dict(annotate\_fontsize=8,  
 figsize=(10,10),   
 legend\_position="right",  
 legend\_size="5%",  
 legend\_pad=0.1,  
 legend\_bbox\_to\_anchor=(1, 1),  
 cmap='OrRd',  
 labelsize=8,  
 scheme=None,  
 k=5,  
 categorical=False  
 )  
 setting\_dict.update(setting)  
 gdf["index"]=gdf.index  
   
 fig, ax=plt.subplots(figsize=setting\_dict["figsize"])  
 divider=make\_axes\_locatable(ax)   
 if setting\_dict["scheme"]:  
 gdf.plot(column=value\_column,scheme=setting\_dict["scheme"], k= setting\_dict["k"],ax=ax,legend=True,cmap=setting\_dict["cmap"],legend\_kwds={'bbox\_to\_anchor':setting\_dict["legend\_bbox\_to\_anchor"]})   
 elif setting\_dict["categorical"]:  
 gdf.plot(column=value\_column,categorical=True,ax=ax,legend=True,cmap=setting\_dict["cmap"],edgecolor='white',legend\_kwds={'bbox\_to\_anchor':setting\_dict["legend\_bbox\_to\_anchor"]})   
 else:   
 cax=divider.append\_axes(setting\_dict["legend\_position"], size=setting\_dict["legend\_size"], pad=setting\_dict["legend\_pad"]) # 配置图例参数  
 gdf.plot(column=value\_column,scheme=setting\_dict["scheme"], k= setting\_dict["k"],ax=ax,cax=cax,legend=True,cmap=setting\_dict["cmap"])   
 gdf.apply(lambda x: ax.annotate(text=x[annotate\_column], xy=x.geometry.centroid.coords[0], ha='center',fontsize=setting\_dict["annotate\_fontsize"]),axis=1) # 增加标注  
 ax.tick\_params(axis='both', labelsize=setting\_dict["labelsize"])  
  
 plt.show()  
  
gdf\_plot\_annotate(pubicHealth\_Statistic\_gdf,"Lung Cancer","index",annotate\_fontsize=10,scheme="Quantiles",k=5)



### 2.7.3.2 空间权重 （Spatial Weights）[1][2]

#### 1） 空间权重的基本解释

空间权重是地理数据科学和空间统计中用来表示空间关系的数学（拓扑）结构，可被理解为一个图的邻接矩阵（graph adjacency maxtrix），代表了地理信息数据集中观察单位之间的空间地理关系，表达地理位置上的接近或关联性，如空间权重表达了地点和之间地理关系，通常包括毗邻性（contiguity）、地理空间距离（geospatial distance）和一般距离（general distances）。毗邻性包括3种表现形式，为共领边连接的Rook（车）连通，共顶点连接的Queen（后）连通，和共顶点连接又共领边连接的Bishop（象）连通。一般距离则是根据研究内容确定的某种成本距离，例如交通分析下的时间成本距离。许多空间分析方法，例如空间自相关和区域化算法都依赖于空间权重。



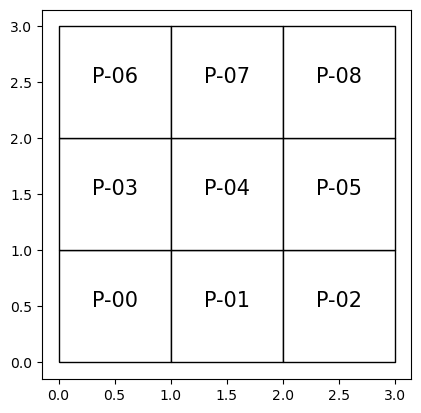
在计算公共健康数据空间权重前，通过定义函数building\_meshgrid\_gdf()构建简单的网格状地理几何数据，用[PySAL](https://pysal.org/)③</sup库提供的方法计算空间权重，并定义weights\_plot()函数显示空间权重，方便观察和理解空间权重的拓扑结构和相关统计值。

def building\_meshgrid\_gdf(x\_num=3,y\_num=3,distance=1):  
 '''  
 给定高宽数量和间隔距离，生成polygon的网格状GeoDataFrame格式的地理几何数据，含字段id，标识polygon单元  
  
 Parameters  
 ----------  
 x\_num : int, optional  
 高数. The default is 3.  
 y\_num : int, optional  
 宽数. The default is 3.   
 distance : float, optional  
 间隔距离. The default is 1.  
  
 Returns  
 -------  
 p\_gdf : GeoDataFrame  
 网格状polygon地理几何数据.  
  
 '''   
   
 import numpy as np  
 from shapely.geometry import Polygon  
 import geopandas as gpd  
  
 xn=np.arange(x\_num)  
 yn=np.arange(y\_num)  
 xs,ys=np.meshgrid(xn,yn)  
 polygons=[]  
 for x,y in zip(xs.flatten(),ys.flatten()):  
 polygon=Polygon([(x, y), (x + distance, y), (x + distance, y + distance), (x, y + distance)])  
 polygons.append(polygon)  
  
 p\_gs=gpd.GeoSeries(polygons)  
 p\_gdf=gpd.GeoDataFrame({  
 "geometry":p\_gs,  
 "id":["P-%s" % str(i).zfill(2) for i in range(len(polygons))],  
 })  
 return p\_gdf  
   
p\_gdf=building\_meshgrid\_gdf(3,3)  
p\_gdf

|  | **geometry** | **id** |
| --- | --- | --- |
| **0** | POLYGON ((0.00000 0.00000, 1.00000 0.00000, 1.... | P-00 |
| **1** | POLYGON ((1.00000 0.00000, 2.00000 0.00000, 2.... | P-01 |
| **2** | POLYGON ((2.00000 0.00000, 3.00000 0.00000, 3.... | P-02 |
| **3** | POLYGON ((0.00000 1.00000, 1.00000 1.00000, 1.... | P-03 |
| **4** | POLYGON ((1.00000 1.00000, 2.00000 1.00000, 2.... | P-04 |
| **5** | POLYGON ((2.00000 1.00000, 3.00000 1.00000, 3.... | P-05 |
| **6** | POLYGON ((0.00000 2.00000, 1.00000 2.00000, 1.... | P-06 |
| **7** | POLYGON ((1.00000 2.00000, 2.00000 2.00000, 2.... | P-07 |
| **8** | POLYGON ((2.00000 2.00000, 3.00000 2.00000, 3.... | P-08 |

打印显示生成的网格状地理几何数据，总共9个单元，每个单元高宽为1，并标注字段id信息。

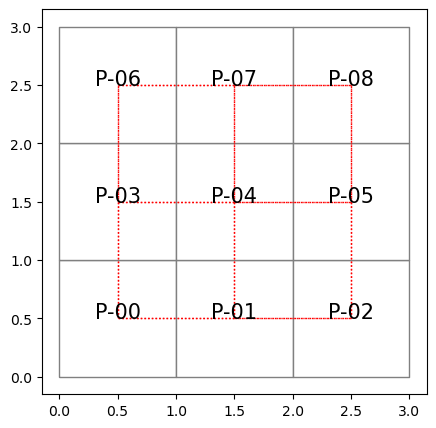
ax=p\_gdf.plot(facecolor="w", edgecolor="k")  
\_=p\_gdf.apply(lambda x: ax.annotate(text=x["id"], xy=x.geometry.centroid.coords[0], ha='center',fontsize=15),axis=1)



import warnings  
warnings.simplefilter(action='ignore', category=FutureWarning)  
  
def weights\_plot(gdf,weights,annotate\_column=None,\*\*setting):   
 '''  
 打印显示空间权重  
  
 Parameters  
 ----------  
 gdf : GeoDataFrame  
 地理空间数据.  
 weights : libpysal.weights  
 有PySAL库计算的空间权重.  
 annotate\_column : string, optional  
 用于标注的列名. The default is None.  
 \*\*setting : key args  
 打印样式参数配置，包括：  
 setting\_dict=dict(figsize=(10,10),  
 annotate\_fontsize=8,  
 ax=None,  
 ).  
  
 Returns  
 -------  
 ax : AxesSubplot  
 子图.  
  
 '''   
 setting\_dict=dict(figsize=(10,10),  
 annotate\_fontsize=8,  
 ax=None,  
 )  
 setting\_dict.update(setting)  
  
 if setting\_dict["ax"]:  
 ax=setting\_dict["ax"]  
 gdf.plot(edgecolor="grey",facecolor="w",figsize=setting\_dict["figsize"],ax=ax)  
 else:  
 ax=gdf.plot(edgecolor="grey",facecolor="w",figsize=setting\_dict["figsize"])  
 f,ax=weights.plot(gdf,  
 ax=ax,  
 edge\_kws=dict(color='r', linestyle=':', linewidth=1),  
 node\_kws=dict(marker='')  
 )  
 if annotate\_column:  
 gdf["index"]=gdf.index  
 gdf.apply(lambda x: ax.annotate(text=x[annotate\_column], xy=x.geometry.centroid.coords[0], ha='center',fontsize=setting\_dict["annotate\_fontsize"]),axis=1) # 增加标注

Rook连通/邻接（contiguity）如下图，单元间共享一条边，红色虚线为邻接连通，例如单元0的Rook邻接有1和3，单元1的Rook邻接有0、2和4，单元4的Rook邻接有1、3、7和5。

import libpysal.weights as LW  
  
wr=LW.contiguity.Rook.from\_dataframe(p\_gdf)  
weights\_plot(p\_gdf,wr,annotate\_column="id",figsize=(5,5),annotate\_fontsize=15)



PySAL计算返回的权重值包含多个属性，因为空间矩阵往往十分稀疏，因此用字典存储邻接矩阵的w.neighbors方法，和存储权重矩阵的w.weights方法，节省存储空间。可以通过w.nonzeros方法查看非零值的数量为24，表明有个零值，并通过值为0表示非连通，和值为1表示连通的矩阵表述。该矩阵量化了包含空值的数量，且大部分包含空值零，为稀疏矩阵。对于稀疏矩阵，尤其规模大的稀疏矩阵，通常只存储非零值而不是包含空值的所有值，以减少存储规模并方便计算。

import pandas as pd  
  
print(f"neighbors: {wr.neighbors};\nnonzeros: {wr.nonzero}")  
wr\_neighbors\_df=pd.DataFrame(\*wr.full()).astype(int)  
wr\_neighbors\_df

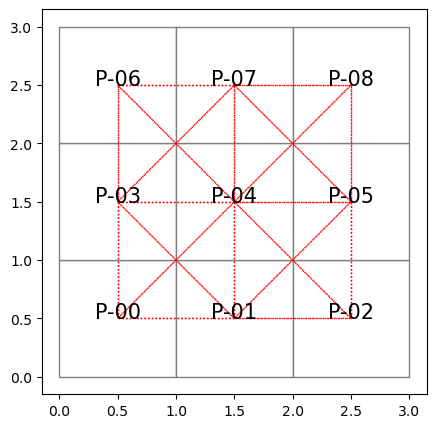
neighbors: {0: [1, 3], 1: [0, 2, 4], 2: [1, 5], 3: [0, 4, 6], 4: [1, 3, 5, 7], 5: [8, 2, 4], 6: [3, 7], 7: [8, 4, 6], 8: [5, 7]};  
nonzeros: 24

|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **1** | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| **2** | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| **3** | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| **4** | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| **5** | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| **6** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **7** | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| **8** | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

Queen连通如下图，单元间共享一个或多个顶点，因此对于单元0的Queen邻接除了1和3单元的Rook边邻接外，还包括顶点邻接单元4，其它单元以此类推。对于的矩阵，空间权重非零值有40个，较之Rook邻接增加了16个顶点邻接单元。w.cardinalities方法报告了每个单元的邻接数量，w.histogram方法则显示了邻接单元数的频数，例如包含有3个邻接单元的单元数有4个，分别为单元0、2、6和8。

wq=LW.contiguity.Queen.from\_dataframe(p\_gdf)  
print(f"neighbors: {wq.neighbors};\nnonzeros: {wq.nonzero};\nweights:\n{wq.weights} ;\ncardinalities: {wq.cardinalities};\nhistogram: {wq.histogram}")  
weights\_plot(p\_gdf,wq,annotate\_column="id",figsize=(5,5),annotate\_fontsize=15)

neighbors: {0: [1, 3, 4], 1: [0, 2, 3, 4, 5], 2: [1, 4, 5], 3: [0, 1, 4, 6, 7], 4: [0, 1, 2, 3, 5, 6, 7, 8], 5: [1, 2, 4, 7, 8], 6: [3, 4, 7], 7: [3, 4, 5, 6, 8], 8: [4, 5, 7]};  
nonzeros: 40;  
weights:  
{0: [1.0, 1.0, 1.0], 1: [1.0, 1.0, 1.0, 1.0, 1.0], 2: [1.0, 1.0, 1.0], 3: [1.0, 1.0, 1.0, 1.0, 1.0], 4: [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0], 5: [1.0, 1.0, 1.0, 1.0, 1.0], 6: [1.0, 1.0, 1.0], 7: [1.0, 1.0, 1.0, 1.0, 1.0], 8: [1.0, 1.0, 1.0]} ;  
cardinalities: {0: 3, 1: 5, 2: 3, 3: 5, 4: 8, 5: 5, 6: 3, 7: 5, 8: 3};  
histogram: [(3, 4), (4, 0), (5, 4), (6, 0), (7, 0), (8, 1)]  
  
  
  
  
  
<AxesSubplot:>

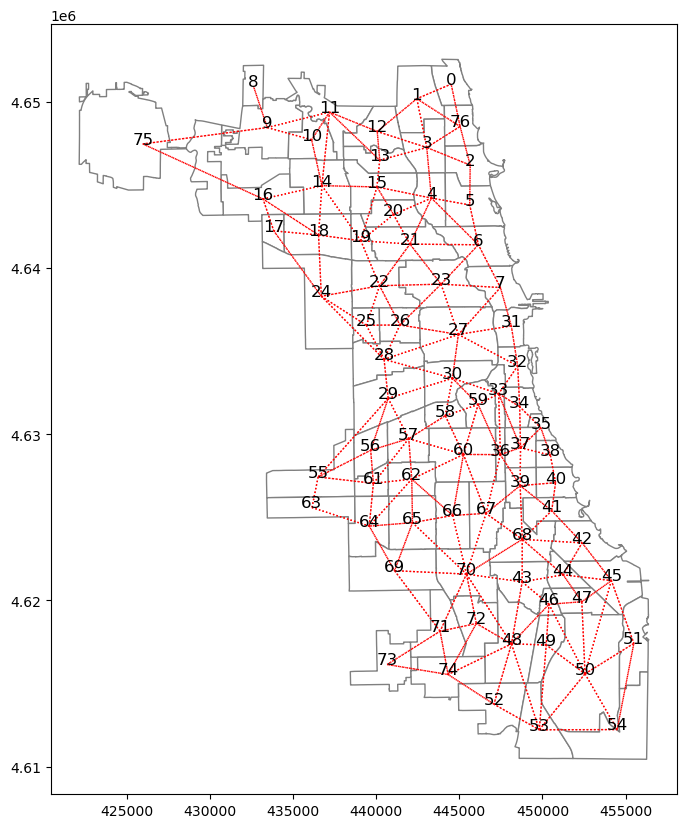


#### 2） Rook neighbors 空间权重——公共健康数据

仅提取索引值为2的邻接单元和对应的社区名称，有单元3、76和5。计算的稀疏矩阵，非零数位358个，用w.pct\_nonzeeros方法可以返回非零数占总数的百分比为。

import libpysal.weights as LW  
  
w\_rook\_pubStats=LW.Rook.from\_dataframe(pubicHealth\_Statistic\_gdf)  
idx=2  
print(f"neighbors num:{w\_rook\_pubStats.n};\nnonzero:{w\_rook\_pubStats.nonzero};\npct\_nonzero:{w\_rook\_pubStats.pct\_nonzero};\nneighbors with index {idx}:{w\_rook\_pubStats.neighbors[idx]};\nneighbor names:\n {pubicHealth\_Statistic\_gdf['Community Area Name'][w\_rook\_pubStats.neighbors[idx]+[idx]]}")  
weights\_plot(pubicHealth\_Statistic\_gdf,w\_rook\_pubStats,"index",annotate\_fontsize=12)

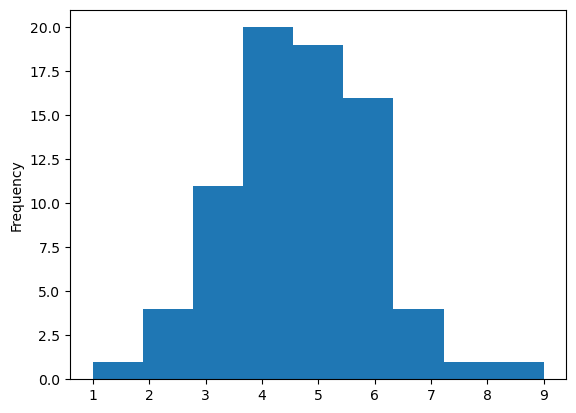
neighbors num:77;  
nonzero:358;  
pct\_nonzero:6.038117726429415;  
neighbors with index 2:[3, 76, 5];  
neighbor names:  
 3 Lincoln Square  
76 Edgewater  
5 Lake View  
2 Uptown  
Name: Community Area Name, dtype: object



绘制w.cardinalities方法返回的邻接数的频数，可以观察到邻接数为4的单元最多，其次为5、6、3等。

w\_rook\_cardinalities\_series=pd.Series(w\_rook\_pubStats.cardinalities)  
w\_rook\_cardinalities\_series.plot.hist(bins=w\_rook\_cardinalities\_series.unique().shape[0])

<AxesSubplot:ylabel='Frequency'>



提取观察前5个单元的权重值，只要毗邻的就为1，否则为0（不存储）。

{k:w\_rook\_pubStats.weights[k] for k in [0,1,2,3,4]}

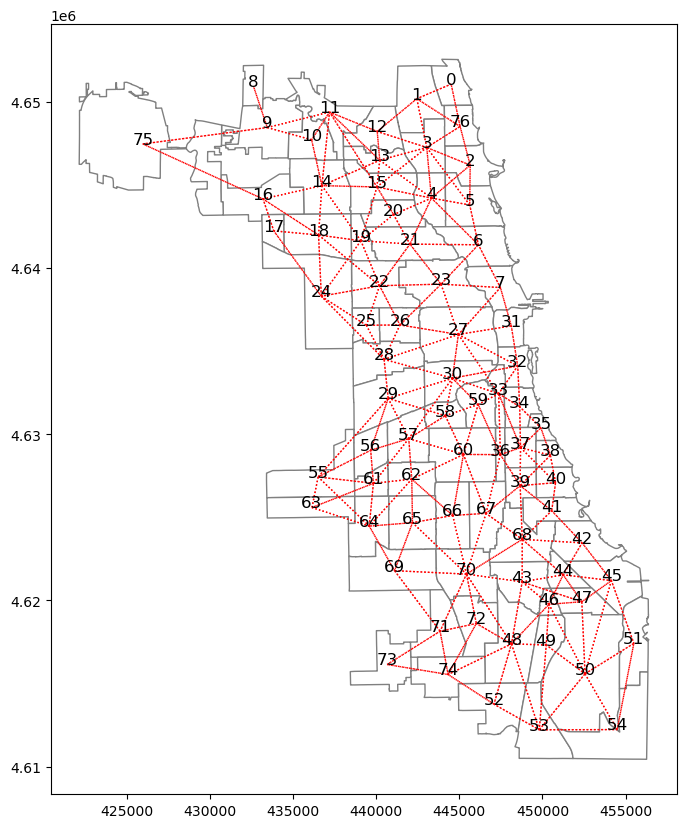
{0: [1.0, 1.0],  
 1: [1.0, 1.0, 1.0, 1.0],  
 2: [1.0, 1.0, 1.0],  
 3: [1.0, 1.0, 1.0, 1.0, 1.0, 1.0],  
 4: [1.0, 1.0, 1.0, 1.0, 1.0, 1.0]}

#### 3） Queen neighbors 空间权重——公共健康数据

类似Rook neighbors方法计算，可以获取返回空间权重值及相关属性统计量。

w\_queen\_pubStats=LW.Queen.from\_dataframe(pubicHealth\_Statistic\_gdf)  
print(f"neighbors num:{w\_queen\_pubStats.n};\nnonzero:{w\_queen\_pubStats.nonzero};\npct\_nonzero:{w\_queen\_pubStats.pct\_nonzero};\nneighbors with index {idx}:{w\_queen\_pubStats.neighbors[idx]};\nneighbor names:\n {pubicHealth\_Statistic\_gdf['Community Area Name'][w\_queen\_pubStats.neighbors[idx]+[idx]]}")  
weights\_plot(pubicHealth\_Statistic\_gdf,w\_queen\_pubStats,"index",annotate\_fontsize=12)

neighbors num:77;  
nonzero:394;  
pct\_nonzero:6.645302749198853;  
neighbors with index 2:[3, 4, 5, 76];  
neighbor names:  
 3 Lincoln Square  
4 North Center  
5 Lake View  
76 Edgewater  
2 Uptown  
Name: Community Area Name, dtype: object  
  
  
  
  
  
<AxesSubplot:>



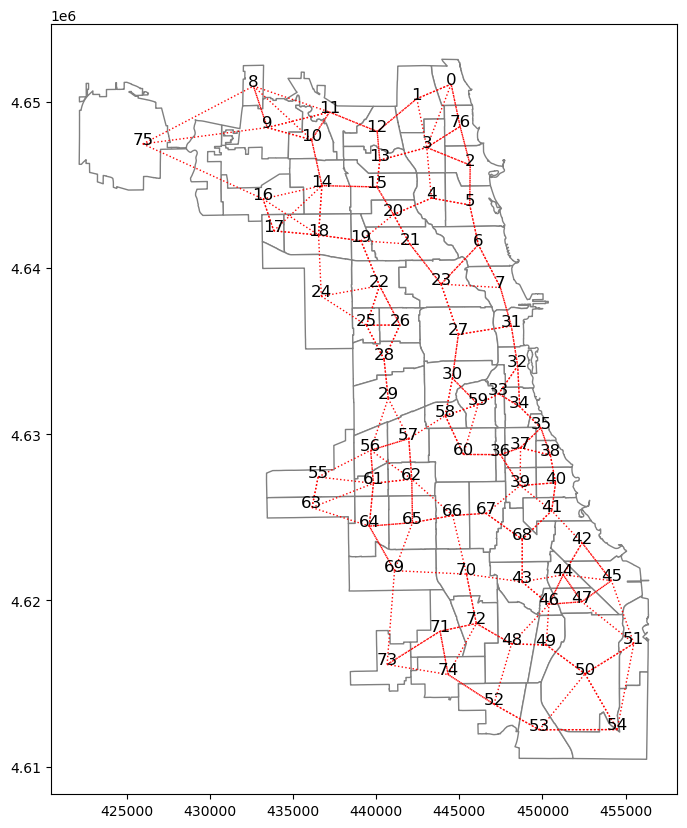
print(f"histogram: {w\_queen\_pubStats.histogram};\nneighbors with index {idx}:{w\_queen\_pubStats.neighbors[idx]};")

histogram: [(1, 1), (2, 3), (3, 8), (4, 19), (5, 13), (6, 16), (7, 11), (8, 5), (9, 1)];  
neighbors with index 2:[3, 4, 5, 76];

#### 4） KNN neighbors 空间权重 （K-Nearest Neighbor weights）

除了毗邻性（contiguity），地理空间距离（geospatial distance）的权重为观察单元与邻里单元间的空间距离，中心点由GeoDataFrame中的空间信息计算出。PySAL库提供了KNN方法，可以指定邻里单元数（参数k）确定最近距离单元连通数量。

w\_knn\_pubStats=LW.KNN.from\_dataframe(pubicHealth\_Statistic\_gdf,k=3)  
weights\_plot(pubicHealth\_Statistic\_gdf,w\_knn\_pubStats,"index",annotate\_fontsize=12)



提取空间权重值，可以观察到指定k数量最近空间距离单元的权重值均为1，否则为0（不存储）。

w\_knn\_df=pd.DataFrame(w\_knn\_pubStats.weights).T  
w\_knn\_df

|  | **0** | **1** | **2** |
| --- | --- | --- | --- |
| **0** | 1.0 | 1.0 | 1.0 |
| **1** | 1.0 | 1.0 | 1.0 |
| **2** | 1.0 | 1.0 | 1.0 |
| **3** | 1.0 | 1.0 | 1.0 |
| **4** | 1.0 | 1.0 | 1.0 |
| **...** | ... | ... | ... |
| **72** | 1.0 | 1.0 | 1.0 |
| **73** | 1.0 | 1.0 | 1.0 |
| **74** | 1.0 | 1.0 | 1.0 |
| **75** | 1.0 | 1.0 | 1.0 |
| **76** | 1.0 | 1.0 | 1.0 |

77 rows × 3 columns

#### 5） Kernal 空间权重（Kernel weights）

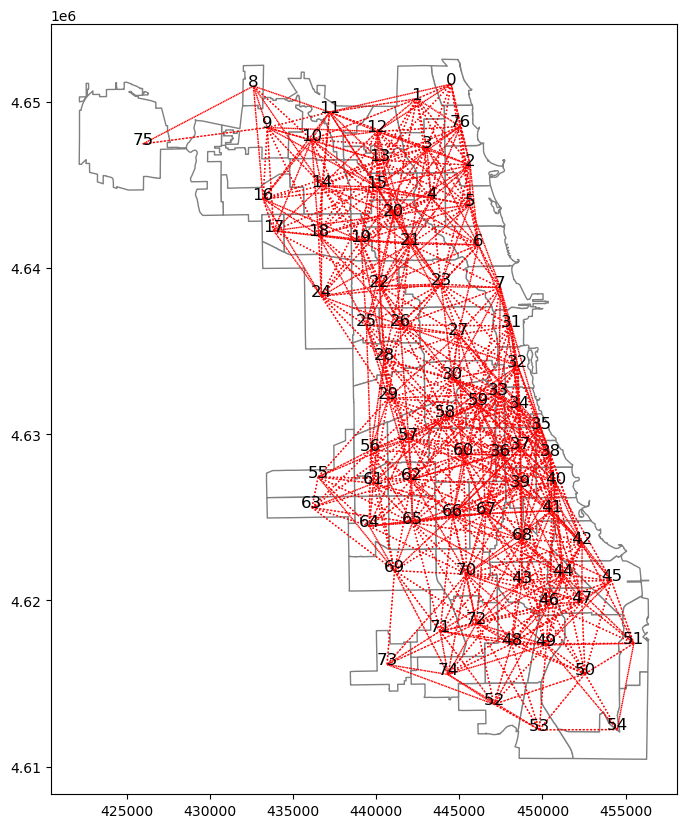
根据Waldo Tobler的说法，地理学第一定律是“一切事物都与其他事物相关，但近处的事物比远处的事物更相关”[3]。Tobler的第一定律是空间依赖性和空间自相关的基本概念的基础，用于空间插值的反距离加权法，并支持kriging（克里格）区域化变量理论[4]，是所有空间分析中的基本假设。PySAL支持连续值权重，例如Kernal 空间权重（距离权重），以更直接的方式反应地理学第一定律。

Kernal 空间权重是最常用的一种距离权重，反映了空间单元相似性（接近性）随距离增加衰减的情况（特性），即观测值（单元值）与观测值之间的空间权重是基于它们之间的距离，但是被具有某些特性的kernel函数优化，使其具有距离衰减的特性，但可能会以不同的速度相对于距离进行衰减。

w.distance.Kernal包含bandwidth可选参数，配置缓冲距离，超过该参数值的单元权重为0；k可选参数，同KNN 空间权重，配置满足bandwith参数之下，最近的k个单元，其余的权重均为0。

默认bandwidth和k参数，计算结果如下。

w\_kernel\_pubStats=LW.distance.Kernel.from\_dataframe(pubicHealth\_Statistic\_gdf)  
weights\_plot(pubicHealth\_Statistic\_gdf,w\_kernel\_pubStats,"index",annotate\_fontsize=12)



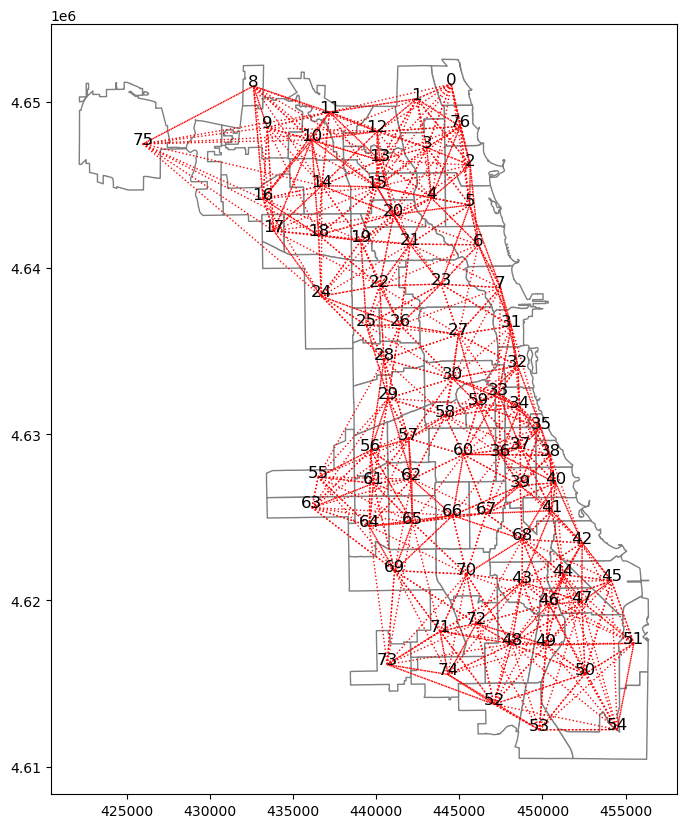
print("Neighborhood cells with index values of [0,1]: %s;weights:%s"%({k:w\_kernel\_pubStats.neighbors[k] for k in [0,75]},{k:w\_kernel\_pubStats.weights[k] for k in [0,75]}))  
print(f"bandwidth:\n {w\_kernel\_pubStats.bandwidth[:3]}")

Neighborhood cells with index values of [0,1]: {0: [13, 12, 11, 5, 4, 2, 3, 76, 1, 0], 75: [75, 9, 8]};weights:{0: [0.16866186834038388, 0.30053319508986687, 0.005684039422874099, 0.02596010826224393, 0.0785766333091692, 0.33703077271339976, 0.45886912756515374, 0.6550849730275031, 0.7029166480913323, 1.0], 75: [1.0, 9.99999900663795e-08, 0.005604407259564681]}  
bandwidth:  
 [[7551.42606251]  
 [7551.42606251]  
 [7551.42606251]]

固定bandwidth可能并不是最佳选择，例如观测区域单元密度变化较大的情况，较大密度范围内的邻接密度高，而稀疏范围内的邻接密度低，因此可以配置fixed参数值为False，使用K-近邻规则自适应调整缓冲距离。一旦考虑给定k参数值邻近观测值，剩下的观测值的空间权重则为0。下述计算配置k=10，可以提取bandwidth信息，发现缓冲距离并不是固定一个值。

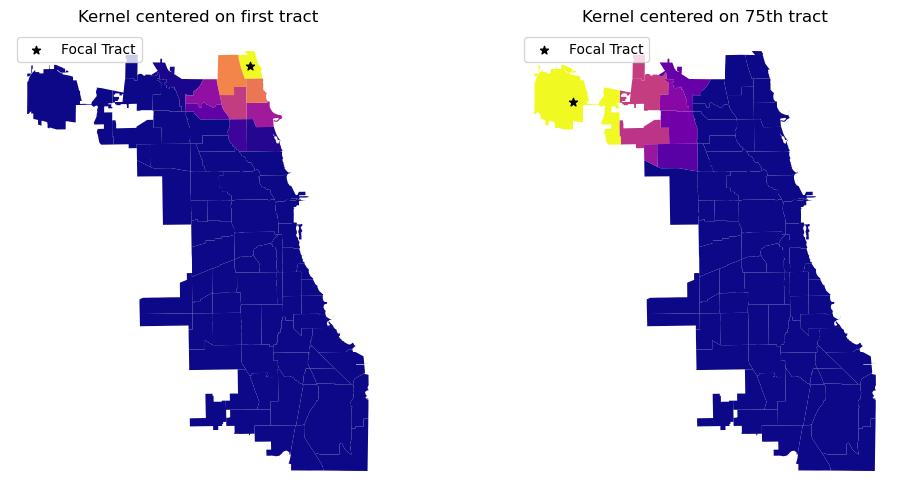
w\_kernel\_adaptive\_bandwidthK\_pubStats=LW.distance.Kernel.from\_dataframe(pubicHealth\_Statistic\_gdf,fixed=False,k=10)  
print("Neighborhood cells with index values of [0,1]: %s;weights:%s"%({k:w\_kernel\_adaptive\_bandwidthK\_pubStats.neighbors[k] for k in [0,75]},{k:w\_kernel\_adaptive\_bandwidthK\_pubStats.weights[k] for k in [0,75]}))  
print(f"bandwidth:\n {w\_kernel\_adaptive\_bandwidthK\_pubStats.bandwidth[:3]}")  
weights\_plot(pubicHealth\_Statistic\_gdf,w\_kernel\_adaptive\_bandwidthK\_pubStats,"index",annotate\_fontsize=12)

Neighborhood cells with index values of [0,1]: {0: [0, 1, 76, 3, 2, 12, 13, 4, 5, 11, 15], 75: [75, 8, 9, 16, 17, 10, 14, 11, 18, 24, 12]};weights:{0: [1.0, 0.7059466557922132, 0.6586028247723479, 0.4643882207762944, 0.3437925176958082, 0.30766718557358774, 0.17714083871950848, 0.08797439955526709, 0.03589452011659111, 0.01582525062943063, 9.99999900663795e-08], 75: [1.0, 0.46823640098244257, 0.4652394381840269, 0.4366454123752175, 0.3304402909469787, 0.2806741382356108, 0.21664465623407503, 0.19315892507283117, 0.15670867768694086, 0.0003303421393521244, 9.99999900663795e-08]}  
bandwidth:  
 [[7629.23806353]  
 [6794.76626129]  
 [5762.32953627]]



可以通过地图打印直观查看给定单元的空间权重值分布情况，上述k值配置为10，因此只有邻近（含自身）的10个单元权重，其余空间权重值为0.

import matplotlib.pyplot as plt  
  
full\_matrix, ids=w\_kernel\_adaptive\_bandwidthK\_pubStats.full()  
f, ax=plt.subplots(1, 2, figsize=(12, 6), subplot\_kw=dict(aspect="equal"))  
pubicHealth\_Statistic\_gdf.assign(weight\_0=full\_matrix[0]).plot("weight\_0", cmap="plasma", ax=ax[0])  
pubicHealth\_Statistic\_gdf.assign(weight\_0=full\_matrix[75]).plot("weight\_0", cmap="plasma", ax=ax[1])  
  
# Add centroid of focal tracts  
pubicHealth\_Statistic\_gdf.iloc[[0], :].centroid.plot(ax=ax[0], marker="\*", color="k", label="Focal Tract")  
pubicHealth\_Statistic\_gdf.iloc[[75], :].centroid.plot(ax=ax[1], marker="\*", color="k", label="Focal Tract")  
  
ax[0].set\_title("Kernel centered on first tract")  
ax[1].set\_title("Kernel centered on 75th tract")  
# Remove axis  
[ax\_.set\_axis\_off() for ax\_ in ax]  
# Add legend  
[ax\_.legend(loc="upper left") for ax\_ in ax];



#### 6）距离缓冲带和混合权重（Distance bands and hybrid Weights）

配置目标单元一个给定半径距离的圆形缓冲区域，只有落在该圆形缓冲区域内的单元计算空间权重，参数binary可以配置空间权重为二进制或连续值的形式。

w\_distanceband\_binary\_pubStats=LW.distance.DistanceBand.from\_dataframe(pubicHealth\_Statistic\_gdf,10000,binary=True)  
w\_distanceband\_binary\_pubStats.weights[0]

[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]

w\_distanceband\_numerical\_pubStats=LW.distance.DistanceBand.from\_dataframe(pubicHealth\_Statistic\_gdf,10000,binary=False)  
w\_distanceband\_numerical\_pubStats.weights[0]

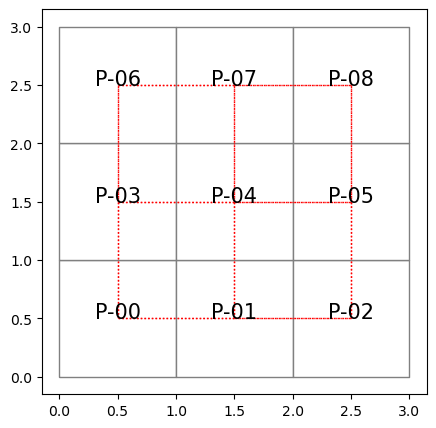
[0.0004457513942007562,  
 0.0001997457994379509,  
 0.00024471957719075977,  
 0.00014371821151303165,  
 0.00013595471749197705,  
 0.00010200262902818481,  
 0.00011065100827986884,  
 0.0001331823319322558,  
 0.00018932323503777486,  
 0.00015929176500392128,  
 0.00010113809568491048,  
 0.00013107470125748466,  
 0.00011653491130540765,  
 0.00010061185161683936,  
 0.0003839360652665388]

#### 7）Lattice 空间权重 （Lattice Weights）

创建一个用于规则网格的权重对象，类似于“空间权重的基本解释”部分给出的构建简单的网格状地理几何数据并计算空间权重的方法。

lat2w=LW.lat2W(4,3)  
print(f"num: {lat2w.n};\npct\_nonzero: {lat2w.pct\_nonzero};\nneighbors: {lat2w.neighbors}")  
print(lat2w.weights)  
weights\_plot(p\_gdf,wr,annotate\_column="id",figsize=(5,5),annotate\_fontsize=15)

num: 12;  
pct\_nonzero: 23.61111111111111;  
neighbors: {0: [3, 1], 3: [0, 6, 4], 1: [0, 4, 2], 4: [1, 3, 7, 5], 2: [1, 5], 5: [2, 4, 8], 6: [3, 9, 7], 7: [4, 6, 10, 8], 8: [5, 7, 11], 9: [6, 10], 10: [7, 9, 11], 11: [8, 10]}  
{0: [1.0, 1.0], 3: [1.0, 1.0, 1.0], 1: [1.0, 1.0, 1.0], 4: [1.0, 1.0, 1.0, 1.0], 2: [1.0, 1.0], 5: [1.0, 1.0, 1.0], 6: [1.0, 1.0, 1.0], 7: [1.0, 1.0, 1.0, 1.0], 8: [1.0, 1.0, 1.0], 9: [1.0, 1.0], 10: [1.0, 1.0, 1.0], 11: [1.0, 1.0]}



#### 8）地球表面的曲率

上述在计算空间权重前已经将公共健康数据投影到epsg=32616的芝加哥区域，但是如果仍为wgs84，又不可能或不方便进行投影，则可以考虑非投影参考系统（如经度/纬度）中隐含曲率的近似值配置radius参数进行计算，可以发现计算结果约等于上述对应的计算结果。

w.to\_adjlist()方法可以将空间权重返回为各单元对应邻接单元权重的DataFrame格式数据。

from pysal.lib import cg as geometry  
radius=geometry.sphere.RADIUS\_EARTH\_MILES  
  
w\_kernel\_adaptive\_bandwidthK\_radius\_pubStats=LW.distance.Kernel.from\_dataframe(pubicHealth\_Statistic\_gdf.to\_crs(args.gi.epsg\_wgs84),fixed=False,k=10,radius=radius)  
w\_kernel\_adaptive\_bandwidthK\_radius\_pubStats.to\_adjlist()

|  | **focal** | **neighbor** | **weight** |
| --- | --- | --- | --- |
| **0** | 0 | 0 | 1.000000e+00 |
| **1** | 0 | 1 | 7.065094e-01 |
| **2** | 0 | 2 | 3.431105e-01 |
| **3** | 0 | 3 | 4.639753e-01 |
| **4** | 0 | 4 | 8.693914e-02 |
| **...** | ... | ... | ... |
| **842** | 76 | 12 | 2.498819e-01 |
| **843** | 76 | 13 | 2.123117e-01 |
| **844** | 76 | 15 | 6.983849e-02 |
| **845** | 76 | 20 | 9.999999e-08 |
| **846** | 76 | 76 | 1.000000e+00 |

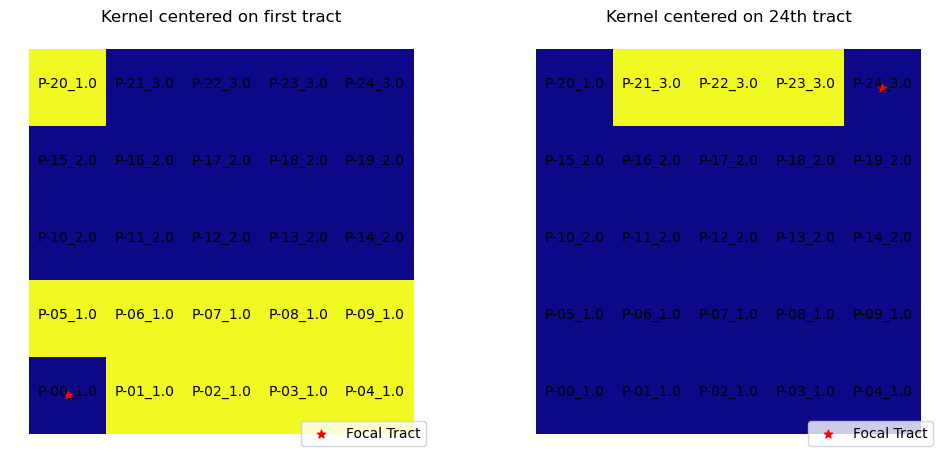
847 rows × 3 columns

#### 9）Block 空间权重 （Block Weights）

Block 空间权重是按照分组/分类/区块定义邻接关系，为各组内单元彼此连通，即使单元间不一定在地理空间位置上彼此连接，例如单元0（图左黄色标识邻接具有权重的单元），而非组内的单元则被赋予0值。

#import warnings  
#warnings.filterwarnings('ignore')  
import numpy as np  
import matplotlib.pyplot as plt  
  
p\_g\_gdf=building\_meshgrid\_gdf(5,5)  
regimes=np.ones(25)  
regimes[range(10,20)]=2  
regimes[range(21,25)]=3  
p\_g\_gdf["regime"]=regimes  
p\_g\_gdf["id\_regime"]=p\_g\_gdf.apply(lambda row:f"{row.id}\_{row.regime}",axis=1)  
  
w\_b=LW.block\_weights(p\_g\_gdf["regime"])  
full\_matrix, ids=w\_b.full()  
  
f, ax=plt.subplots(1, 2, figsize=(12, 6), subplot\_kw=dict(aspect="equal"))  
p\_g\_gdf.assign(weight\_0=full\_matrix[0]).plot("weight\_0", cmap="plasma", ax=ax[0])  
p\_g\_gdf.assign(weight\_0=full\_matrix[24]).plot("weight\_0", cmap="plasma", ax=ax[1])  
  
# Add centroid of focal tracts  
p\_g\_gdf.iloc[[0], :].centroid.plot(ax=ax[0], marker="\*", color="r", label="Focal Tract")  
p\_g\_gdf.iloc[[24], :].centroid.plot(ax=ax[1], marker="\*", color="r", label="Focal Tract")  
  
ax[0].set\_title("Kernel centered on first tract")  
ax[1].set\_title("Kernel centered on 24th tract")  
# Remove axis  
[ax\_.set\_axis\_off() for ax\_ in ax]  
# Add legend  
[ax\_.legend(loc="lower right") for ax\_ in ax];  
  
# annotate  
\_=p\_g\_gdf.apply(lambda x: ax[0].annotate(text=x["id\_regime"], xy=x.geometry.centroid.coords[0], ha='center',fontsize=10),axis=1)  
\_=p\_g\_gdf.apply(lambda x: ax[1].annotate(text=x["id\_regime"], xy=x.geometry.centroid.coords[0], ha='center',fontsize=10),axis=1)

C:\Users\richi\anaconda3\envs\usda\lib\site-packages\libpysal\weights\weights.py:172: UserWarning: The weights matrix is not fully connected:   
 There are 3 disconnected components.  
 warnings.warn(message)



### 2.7.3.3 空间自相关 (Spatial Autocorrelation)

相关性描述了两个变量之间的关系；自相关描述了在两个不同的对象或同一对象不同时间上对同一变量进行的两个观测之间的关系；空间自相关是一种自相关，其中成对的对象由它们之间的距离定义，表明空间分布上的不同。空间自相关的目的是为了寻找某一变量（属性）在空间分布上的特征（模式），即空间结构，这涉及到两个内容，一个是衡量空间分布的空间相似性，通常用空间权重衡量；另一个是匹配空间相似性的衡量标准属性相似性，通常用空间滞后（延迟）衡量。

#### 1）空间相似性（Spatial Similarity）与属性相似性（Attribute Similarity）

对公共健康数据采用queen空间权重，考虑邻边和一个到多个顶点的连通，实现空间相似性的计算。对于属性相似性，根据公式： wijijy\_{j}$为连通单元的属性值。如果空间权重为1或0表述，则目标单元的空间滞后为其连通单元的属性值和均值；如果为连续变化值，则目标单元的空间滞后为连通单元属性值空间权重加权后的和均值。

为了清晰计算过程，移除不相关的属性字段，仅保留测量值Lung Cancer字段，地理空间几何geometry字段和用于标识单元（社区）的ID。空间滞后使用w.lag\_spatial方法计算。

LungCancer\_gdf=pubicHealth\_Statistic\_gdf[["geometry","Community Area","Community Area Name","Lung Cancer"]]  
LungCancer\_gdf[:3]

|  | **geometry** | **Community Area** | **Community Area Name** | **Lung Cancer** |
| --- | --- | --- | --- | --- |
| **0** | POLYGON ((445789.555 4649779.811, 445691.750 4... | 1 | Rogers Park | 36.7 |
| **1** | POLYGON ((443315.818 4652166.308, 443316.880 4... | 2 | West Ridge | 36.0 |
| **2** | POLYGON ((446874.222 4644956.738, 446627.798 4... | 3 | Uptown | 50.5 |

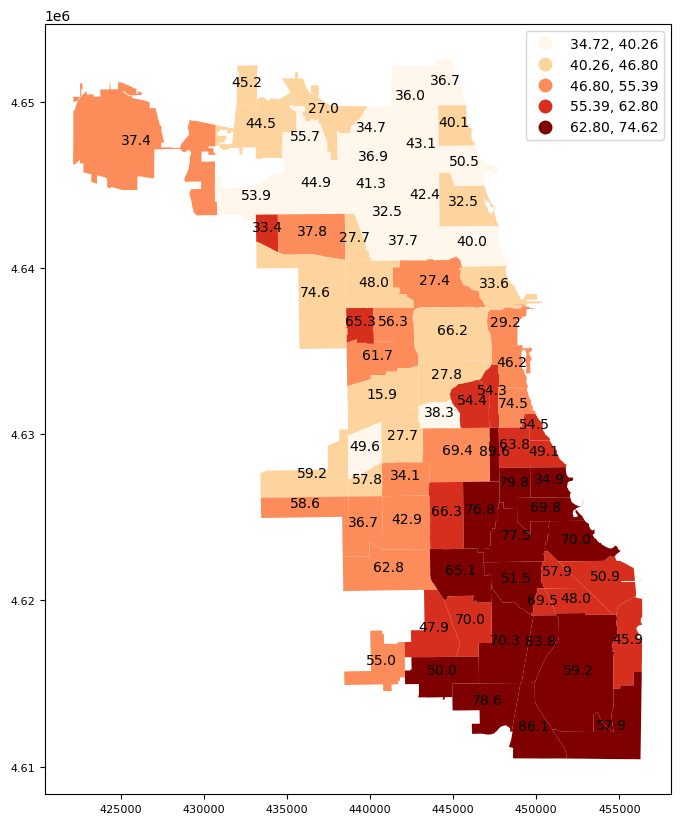
import pandas as pd  
pd.options.mode.chained\_assignment = None # default='warn'  
  
w\_queen\_pubStats=LW.Queen.from\_dataframe(pubicHealth\_Statistic\_gdf)  
w\_queen\_pubStats.transform='r'  
LungCancer\_gdf["lag"]=list(LW.lag\_spatial(w\_queen\_pubStats,LungCancer\_gdf['Lung Cancer']))  
LungCancer\_gdf

|  | **geometry** | **Community Area** | **Community Area Name** | **Lung Cancer** | **lag** |
| --- | --- | --- | --- | --- | --- |
| **0** | POLYGON ((445789.555 4649779.811, 445691.750 4... | 1 | Rogers Park | 36.7 | 38.0500 |
| **1** | POLYGON ((443315.818 4652166.308, 443316.880 4... | 2 | West Ridge | 36.0 | 38.6500 |
| **2** | POLYGON ((446874.222 4644956.738, 446627.798 4... | 3 | Uptown | 50.5 | 39.5250 |
| **3** | POLYGON ((444126.139 4647342.956, 444126.361 4... | 4 | Lincoln Square | 43.1 | 39.3000 |
| **4** | POLYGON ((444174.409 4642483.583, 444169.878 4... | 5 | North Center | 42.4 | 39.3125 |
| **...** | ... | ... | ... | ... | ... |
| **72** | POLYGON ((447293.327 4619865.787, 447295.721 4... | 73 | Washington Heights | 70.0 | 58.3250 |
| **73** | POLYGON ((442056.996 4617496.001, 442058.132 4... | 74 | Mount Greenwood | 55.0 | 48.9500 |
| **74** | POLYGON ((446556.956 4615011.389, 446528.614 4... | 75 | Morgan Park | 50.0 | 64.3600 |
| **75** | MULTIPOLYGON (((430701.343 4648604.251, 430701... | 76 | O'Hare | 37.4 | 49.2000 |
| **76** | POLYGON ((445789.555 4649779.811, 445789.556 4... | 77 | Edgewater | 40.1 | 41.5750 |

77 rows × 5 columns

配置参数scheme，按分位数（Quantiles）给定参数k划分数量，显示空间滞后地图。

gdf\_plot\_annotate(LungCancer\_gdf,"lag","Lung Cancer",annotate\_fontsize=10,scheme="Quantiles",k=5)



#### 2）全局空间自相关（Global Spatial Autocorrelation ）

* 二进制方式 （Binary Case）

二进制方式是指测量数据的值只包含0和1，一类情况是测量值本身即为0或1；另一类是将连续数值按照中位数或均值等切分为0或1的值。二进制方式是最简单观察数值空间分布的手段，其中大于中位数的值数量有38个单元，通过显示高值和低值的集聚或分散情况，可以初步判断感染肺癌人数的分布是存在集聚关系的，即如果一个单元为高值，则邻近的单元也趋近于高值；反之，如果一个单元为低值，则邻近单元也趋近于低值的关系。

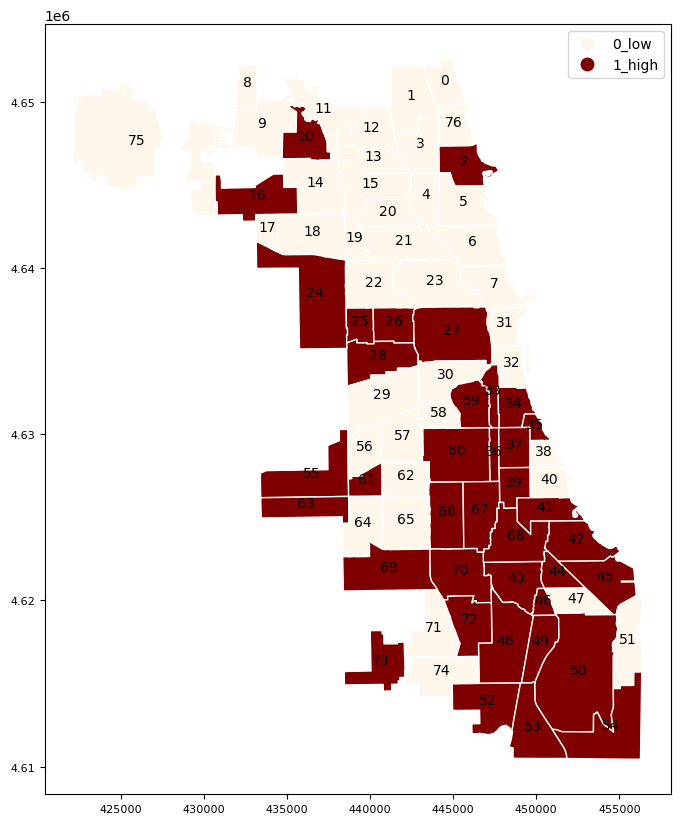
LC\_val=LungCancer\_gdf['Lung Cancer']  
LC\_val\_binary=LC\_val>LC\_val.median()  
print(f"number greater than median: {sum(LC\_val\_binary)}")

number greater than median: 38

LC\_binary\_labels=["0\_low","1\_high"]  
LC\_val\_binary=[LC\_binary\_labels[i] for i in 1\*LC\_val\_binary]  
LungCancer\_gdf["binary\_median"]=LC\_val\_binary  
LungCancer\_gdf[:3]

|  | **geometry** | **Community Area** | **Community Area Name** | **Lung Cancer** | **lag** | **binary\_median** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | POLYGON ((445789.555 4649779.811, 445691.750 4... | 1 | Rogers Park | 36.7 | 38.050 | 0\_low |
| **1** | POLYGON ((443315.818 4652166.308, 443316.880 4... | 2 | West Ridge | 36.0 | 38.650 | 0\_low |
| **2** | POLYGON ((446874.222 4644956.738, 446627.798 4... | 3 | Uptown | 50.5 | 39.525 | 1\_high |

gdf\_plot\_annotate(LungCancer\_gdf,"binary\_median","index",annotate\_fontsize=10,categorical=True)



* 邻接数 （Join counts）

为量化高值集聚或低值集聚，引入连接数计算，计数两两一对单元是高值-高值（Black-Black，BB）、低值-低值（White-white，WW）还是高值-低值（Black-White，BW）。使用jc.bb，jc.ww和jc.bw方法计算邻接数，从计算结果来看，测量值趋向于“同类”集聚。

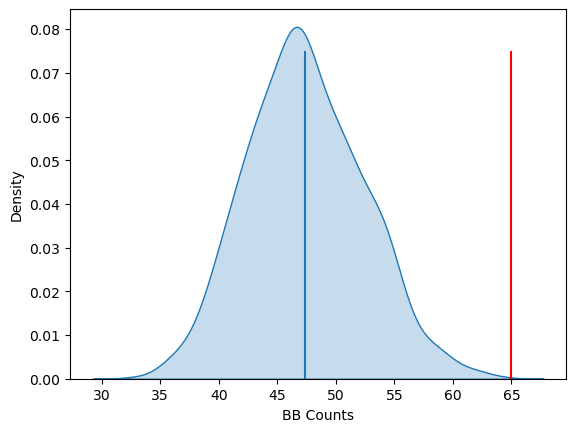
import esda  
import numpy as np  
  
LC\_val\_binary=1\*(LC\_val>LC\_val.median())  
np.random.seed(7777777)  
w\_queen\_pubStats.transform='b'  
LC\_jc=esda.join\_counts.Join\_Counts(LC\_val\_binary,w\_queen\_pubStats)  
print(f"BB: {LC\_jc.bb};\nWW: {LC\_jc.ww};\nBW: {LC\_jc.bw}\nall possibilities: {w\_queen\_pubStats.s0/2}\nmean\_BB: {LC\_jc.mean\_bb}")

BB: 65.0;  
WW: 64.0;  
BW: 68.0  
all possibilities: 197.0  
mean\_BB: 47.3973973973974

如何确定BB或WW分布是趋于集聚而不是完全随机分布的？PySAL库的jc.sim\_bb方法使用观察到的属性值的随机空间排列生成完全空间随机（complete spatial randomness，CSR）无效情况下的数组数据，重复了大量次数（默认为999次），以构建一个参考分布来评估计数的统计意义。从计算结果来看，每次重新随机分布已有属性值到各个空间单元（相当于打乱实际属性值分布，使其空间分布随机），并统计每次的BB连接数，发现BB连接数位于33到64之间，均值为47.397，而实际BB连接数为65，高于CRS分布最大值，且p-value=0.001，小于显著性水平0.05，拒绝原假设，即实际BB的空间分布不是空间随机的，可以说明具有一定的集聚性。

import seaborn as sbn  
import matplotlib.pyplot as plt  
  
print(f"p-value sim\_bb: {LC\_jc.p\_sim\_bb}")  
print(f"sim\_bb min: {LC\_jc.sim\_bb.min()},sim\_bb max: {LC\_jc.sim\_bb.max()},sim\_bb mean: {LC\_jc.mean\_bb}")  
  
sbn.kdeplot(LC\_jc.sim\_bb, shade=True)  
plt.vlines(LC\_jc.bb, 0, 0.075, color='r')  
plt.vlines(LC\_jc.mean\_bb, 0,0.075)  
plt.xlabel('BB Counts')  
  
plt.show()

p-value sim\_bb: 0.001  
sim\_bb min: 33.0,sim\_bb max: 64.0,sim\_bb mean: 47.3973973973974



* [权重变换](https://pysal.org/libpysal/_modules/libpysal/weights/weights.html#W.set_transform) （Transformations of weights）——w.transform

可以直接对空间权重执行权重转换，参数包括：

B – Binary，二进制。目标单元的连通单元均为1

R – Row-standardization (global sum=n)，行-标准化。因为值为1，因此和为行（连通样本数）数。例如单元1含有4个连通单元，因此值为1/4=0.25

D – Double-standardization (global sum=1)，加倍-标准化。为测量值除以单元数，和为1。例如，本例有77个单元，1/77=0.012987012987012988

V – Variance stabilizing，方差稳定化

O – Restore original transformation (from instantiation)，恢复原始转换（从实例化对象）

w\_queen\_pubStats.transform="b"  
keys\_lst=[0,1,2]  
{k:w\_queen\_pubStats.weights[k] for k in keys\_lst}

{0: [1.0, 1.0], 1: [1.0, 1.0, 1.0, 1.0], 2: [1.0, 1.0, 1.0, 1.0]}

w\_queen\_pubStats.transform="r"  
{k:w\_queen\_pubStats.weights[k] for k in keys\_lst}

{0: [0.5, 0.5], 1: [0.25, 0.25, 0.25, 0.25], 2: [0.25, 0.25, 0.25, 0.25]}

w\_queen\_pubStats.transform="d"  
  
print(w\_queen\_pubStats.nonzero)  
{k:w\_queen\_pubStats.weights[k] for k in keys\_lst}

394  
  
  
  
  
  
{0: [0.012987012987012988, 0.012987012987012988],  
 1: [0.012987012987012988,  
 0.012987012987012988,  
 0.012987012987012988,  
 0.012987012987012988],  
 2: [0.012987012987012988,  
 0.012987012987012988,  
 0.012987012987012988,  
 0.012987012987012988]}

w\_queen\_pubStats.transform="v"  
{k:w\_queen\_pubStats.weights[k] for k in keys\_lst}

{0: [0.31728195118702557, 0.31728195118702557],  
 1: [0.22435221923244497,  
 0.22435221923244497,  
 0.22435221923244497,  
 0.22435221923244497],  
 2: [0.22435221923244497,  
 0.22435221923244497,  
 0.22435221923244497,  
 0.22435221923244497]}

w\_queen\_pubStats.transform="o"  
{k:w\_queen\_pubStats.weights[k] for k in keys\_lst}

{0: [1.0, 1.0], 1: [1.0, 1.0, 1.0, 1.0], 2: [1.0, 1.0, 1.0, 1.0]}

LungCancer\_gdf.shape

(77, 6)

* 连续值方式（Continuous Case）

**全局 Moran’s I指数（全局莫兰指数）**

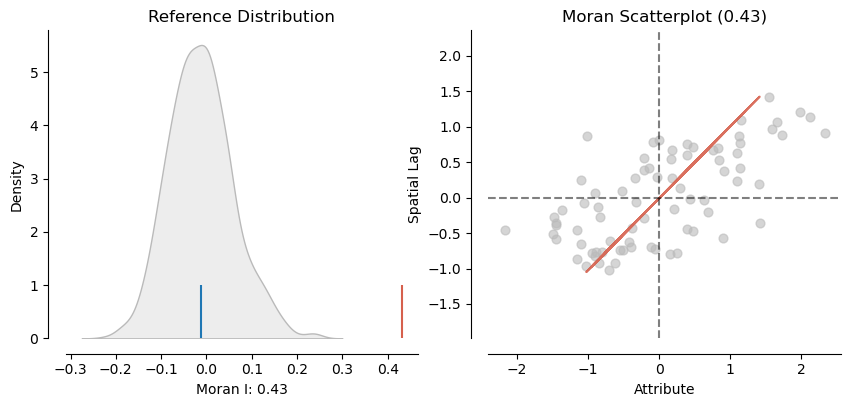
对于Moran’s I指数，若代表某一变量的样本总数，$x\_{i} i xi iwij为空间权重矩阵。Moran’s I指数的值域为[-1,1]，小于0表示负相关，大于0表示正相关，等于0则表示研究区域个单元间彼此独立。Moran’s I值越趋近于1，表明研究对象属性测量值在空间分布上呈现集聚效应越显著；Moran’s I值越趋近于-1，则表明研究对象属性测量值在空间分布上呈现异质性（趋异性）越显著。

计算公共健康数据的全局Moran’s I指数，结果为0.431，可以判断感染肺癌人数的分布是存在一定集聚关系的。同样，将Moran’s I指数对照CRS无效情况下的数组数据，可以发现p-value=0.001，拒绝原假设，表明单元测量值之间具有集聚性。

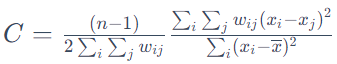
PySAL的splot模块提供了plot\_moran方法，可以直接打印Moran’s I指数，这包括CRS参考分布图表，及属性值和空间滞后的散点图，含回归曲线。

from splot.esda import plot\_moran  
  
w\_queen\_pubStats.transform='r'  
  
LC\_mi=esda.moran.Moran(LC\_val,w\_queen\_pubStats)  
print(f"LC mi: {LC\_mi.I}; p-value p\_sim: {LC\_mi.p\_sim}")  
  
plot\_moran(LC\_mi)

LC mi: 0.43157676011326235; p-value p\_sim: 0.001  
  
  
  
  
  
(<Figure size 1000x400 with 2 Axes>,  
 array([<AxesSubplot:title={'center':'Reference Distribution'}, xlabel='Moran I: 0.43', ylabel='Density'>,  
 <AxesSubplot:title={'center':'Moran Scatterplot (0.43)'}, xlabel='Attribute', ylabel='Spatial Lag'>],  
 dtype=object))



* 其它全局指数——Geary’s C和 Getis and Ord’s G

在 The contiguity ratio and statistical mapping[5]中提出的Geary’s C指数，公式为： nwijxii为样本均值。与Moran’s I比较，两种测量方法都是将每个观察点的本地邻域内的关系与整个样本的关系进行比较，Moran’s I在标准化数值上采用交叉乘积，而Geary’s C在没有任何标准化的数值上采用差异。

Geary’s C计算结果为0.561，p-value为0.001，因此同Global Moran’s I一样，表明单元测量值之间具有集聚性。

w\_queen\_pubStats.transform="b"  
geary=esda.geary.Geary(LungCancer\_gdf["Lung Cancer"],w\_queen\_pubStats)  
print(f"Geary's C: {geary.C}; Pseudo p-value: {geary.p\_sim}")

Geary's C: 0.5618485483281203; Pseudo p-value: 0.001

在The analysis of spatial association by use of distance statistics[6]中提出的Getis and Ord’s G指数，是针对点基于距离的空间自相关统计，如果构建二元空间权重矩阵，也可用于多边形（polygon）数据计算，其公式为： wij (d )是对观测值（单元）之间关系分配的二元权重，并遵循距离缓冲带标准。该指数最初是作为集中度而不是空间自相关的方法提出，因此该指数非常适合于测量值（高或低）趋向于集聚的程度，为正空间相关统计值，因此不能发现负空间相关关系。

Getis & Ord G计算结果为0.221，p-value值为0.037，小于显著性水平0.05，表明单元之间具有一定的集聚性。

pts=LungCancer\_gdf.centroid  
xys=pd.DataFrame({"X": pts.x, "Y": pts.y})  
min\_thr=LW.util.min\_threshold\_distance(xys)  
print(f"min\_thr: {min\_thr}")  
w\_db=LW.DistanceBand.from\_dataframe(LungCancer\_gdf, min\_thr)  
gao=esda.getisord.G(LungCancer\_gdf["Lung Cancer"], w\_db)  
print( "Getis & Ord G: %.3f; Pseudo p-value: %.3f" % (gao.G, gao.p\_sim))

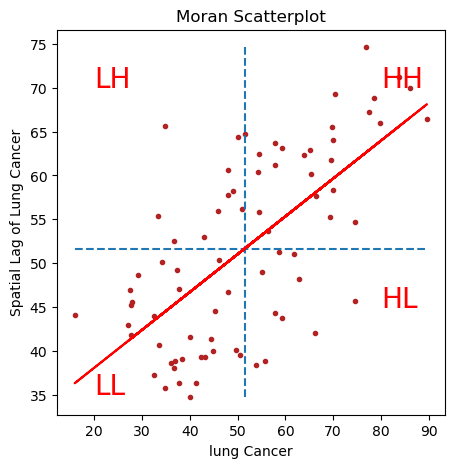
min\_thr: 7509.104795460789  
Getis & Ord G: 0.221; Pseudo p-value: 0.037

#### 3）局部空间自相关（Local Autocorrelation）——热点（Hot Spots）、冷点（Cold Spots）和空间异常值（Spatial Outliers）

全局空间自相关对研究区域提供了一个单一数值结果，并没有指出单元（高、低测量值）集聚的区域，或明确分散的情况，而局部空间自相关侧重于观察单元与邻近单元之间的关系，从而了解属性单元分布的空间结构。

首先建立测量值与空间滞后之间的散点图，并按照各自均值位置分为四象限图观察数值分布情况，指定四个象限分布：右上角为高-高（HH），高值集聚；左上角为低-高（LH），高值包含低值异常；左下方为低-低（LL），低值集聚；右下方为高-低（HL），低值包含高值异常。

LC\_val=LungCancer\_gdf["Lung Cancer"]  
LC\_lag=LungCancer\_gdf["lag"]  
b,a=np.polyfit(LC\_val,LC\_lag,1)  
  
f, ax=plt.subplots(1, figsize=(5, 5))  
plt.plot(LC\_val, LC\_lag, '.', color='firebrick')  
  
# dashed vert at mean of the price  
plt.vlines(LC\_val.mean(), LC\_lag.min(), LC\_lag.max(), linestyle='--')  
# dashed horizontal at mean of lagged price   
plt.hlines(LC\_lag.mean(), LC\_val.min(), LC\_val.max(), linestyle='--')  
  
# red line of best fit using global I as slope  
plt.plot(LC\_val, a + b\*LC\_val, 'r')  
plt.title('Moran Scatterplot')  
plt.ylabel('Spatial Lag of Lung Cancer')  
plt.xlabel('lung Cancer')  
  
plt.text(80, 70, "HH", fontsize=20, c="r")  
plt.text(80, 45, "HL", fontsize=20, c="r")  
plt.text(20, 70, "LH", fontsize=20, c="r")  
plt.text(20, 35, "LL", fontsize=20, c="r")  
  
plt.show()



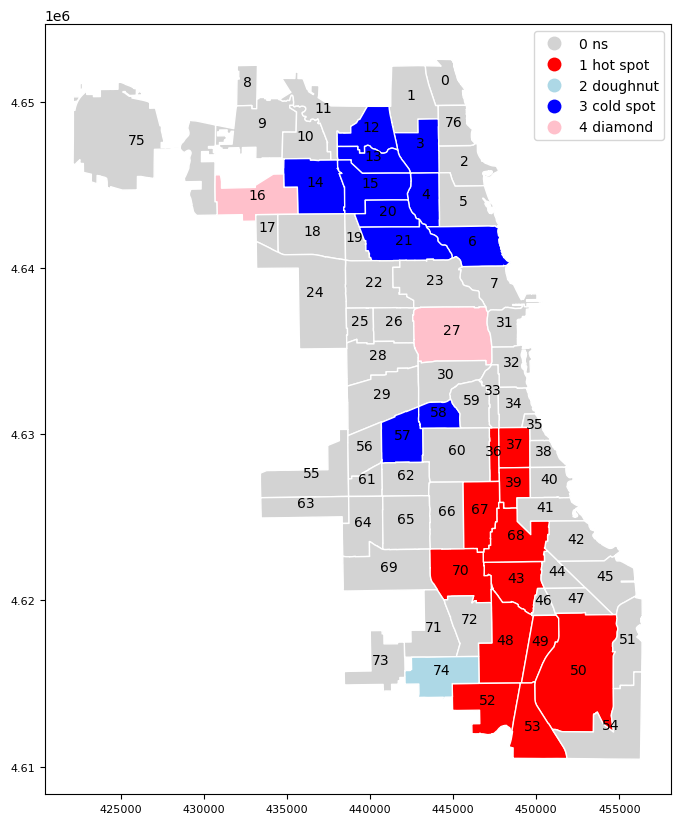
**局部 Moran’s I指数（局部莫兰指数）**

对于空间位置单元的局部Moran’s I指数定义为： LI= ，局部 Moran’s I取值范围不限于[-1,1]。使用PySAL库提供的esda.moran.Moran\_Local方法计算局部 Moran’s I指数，其属性Is为local Moran’s I值，q则是划分为四象限后的分类，包括HH、LH、LL和HL。用p\_sim查看各单元对应p-value值，计算满足显著性水平0.05的单元数量为28个，不满足的单元地图打印是标识为灰色。从计算结果来看，HH区域主要集中于芝加哥城南部，LL区间集中于城北部，这样的分布与城市用地类型存在关联，南部区域多工业用地，例如矿物提取、一般工业、制造加工、仓储配送和通用电气工业等，而北部工业用地较少，多植被和水体。

w\_queen\_pubStats.transform='r'  
LC\_li=esda.moran.Moran\_Local(LC\_val,w\_queen\_pubStats)  
print(LC\_li.q) # values indicate quandrant location 1 HH, 2 LH, 3 LL, 4 HL  
print(LC\_li.Is)  
print(f"p\_value<0.05 num: {(LC\_li.p\_sim<0.05).sum()}")  
  
LungCancer\_gdf["li"]=LC\_li.q  
LungCancer\_gdf["p\_value\_li"]=LC\_li.p\_sim  
LungCancer\_gdf["li\_005"]=LungCancer\_gdf.apply(lambda row:row.li if row.p\_value\_li<0.05 else 0,axis=1)  
spot\_labels=[ '0 ns', '1 hot spot', '2 doughnut', '3 cold spot', '4 diamond']  
LungCancer\_gdf['cl\_li']=LungCancer\_gdf.apply(lambda row:spot\_labels[row.li\_005],axis=1)

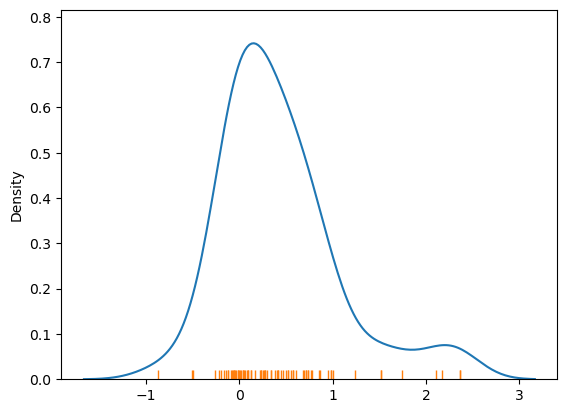
[3 3 3 3 3 3 3 3 3 3 4 3 3 3 3 3 4 2 3 3 3 3 3 3 4 1 1 4 4 3 3 3 3 1 1 1 1  
 1 2 1 2 1 1 1 1 2 1 2 1 1 1 2 1 1 1 4 3 3 3 1 1 4 3 4 2 2 1 1 1 4 1 2 1 4  
 2 3 3]  
[ 7.34620122e-01 7.35038893e-01 4.34357030e-02 3.77790044e-01  
 4.08920807e-01 5.25466603e-01 7.12085185e-01 7.13351926e-01  
 1.62275911e-01 2.62377369e-01 -1.97927882e-01 7.68303455e-01  
 9.76827976e-01 6.78820959e-01 2.78568895e-01 5.69785716e-01  
 -1.17269854e-01 -2.64366188e-01 2.22662497e-01 5.47724185e-01  
 9.97779992e-01 7.72459773e-01 6.17866759e-02 4.02310865e-01  
 -4.98836025e-01 4.43178899e-01 3.99005339e-02 -5.12948531e-01  
 -1.75860468e-02 9.78415607e-01 5.16474374e-01 2.32087932e-01  
 2.11416015e-02 9.25450975e-02 2.74082781e-01 1.22670357e-01  
 2.10636271e+00 4.96968056e-01 -5.96459256e-02 1.51368595e+00  
 -8.67719651e-01 9.53499561e-01 8.59421513e-01 8.90341145e-04  
 2.31571055e-01 -1.01569221e-02 6.86771446e-01 -8.13951200e-02  
 1.24236761e+00 2.36680617e+00 3.33414230e-01 -9.33124407e-02  
 1.73765287e+00 2.36681139e+00 2.90939023e-01 -2.21655560e-01  
 7.89000752e-02 8.48155414e-01 6.06819219e-01 4.74881975e-02  
 2.54081230e-01 -1.67501098e-01 8.67244258e-02 -6.54438964e-03  
 -5.93331085e-02 -4.82062395e-02 3.38731639e-01 2.16720170e+00  
 1.52001725e+00 -1.39526956e-01 5.74769112e-01 -1.20575288e-01  
 4.68874466e-01 -3.29572490e-02 -7.06073043e-02 1.18888522e-01  
 4.17201629e-01]  
p\_value<0.05 num: 26

from matplotlib import colors  
cmap=colors.ListedColormap([ 'lightgrey', 'red', 'lightblue', 'blue', 'pink'])  
gdf\_plot\_annotate(LungCancer\_gdf,"cl\_li","index",annotate\_fontsize=10,categorical=True,cmap=cmap)



打印局部 Moran’s I指数核密度估计图，从分布情况也可以判断出，高于0的部分相对于低于0的部分数量多，但HH和LL均存在。

import seaborn   
ax=seaborn.kdeplot(LC\_li.Is)  
seaborn.rugplot(LC\_li.Is, ax=ax);



### 2.7.3.3 地理轮廓 (Geosilhouettes)

聚类契合度的地理测量，用于测量观测值与特定聚类的拟合程度，在Geosilhouettes: Geographical measures of cluster fit[7]中提出有两个轮廓系数，分别为：

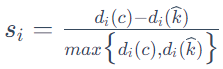
1. Path Silhouettes（路径轮廓，组内/组间相似度），表征聚类中联合地理和特征的相似度；
2. Boundary Silhouettes（边缘轮廓，组缘相似度）, 表征聚类中地理边界的定义程度。

上述轮廓指数已经纳入到了PySAL库的edsa模块中，可以调用计算。

#### 1）轮廓指数/系数（Silhouette Score）

在聚类问题中使用的轮廓系数，用来衡量观测结果与当前聚类分组的匹配程度，常用来评价聚类算法模型的效果。轮廓系数组成部分：

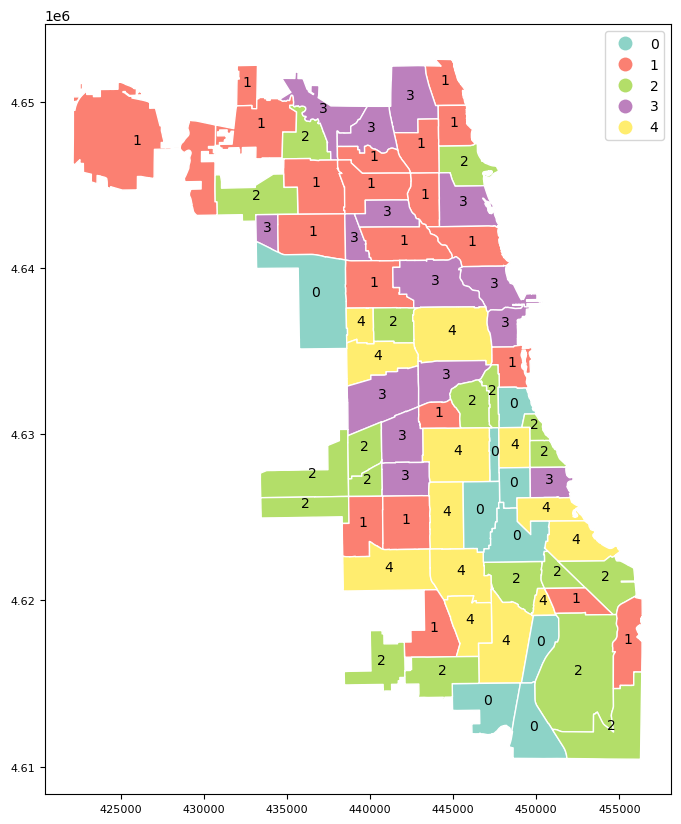
* ，为单元所处的群组（簇）；
* ，为不包含单元的群组；
* di(c) ic$之间的不相似度（反之相似度），等于观测单元测量值与同一群组中所有其它单元测量值之间的平均距离，为组内差异；
* di(k) iik之间的不相似度（反之相似度），等于观测单元测量值与其它群组中所有单元测量值之间的平均距离，为组间差异。

根据聚类的要求，通常为“簇内差异小，而簇外差异大”，即di(k) di(c) iiiiici ickiksi接近于-1。

评估聚类结果的轮廓系数可以使用Sklearn库提供的silhouette\_samples方法。这里用Sklearn库提供的KMeans算法执行聚类，计算轮廓系数时，给出两种情形，一种是直接的聚类结果，这包括为同一簇而分散的单元；另一种是将同一簇按照地理空间位置进一步切分为单独的群组，通过定义函数gdf\_parent\_child\_dissolve()实现切分。

下述代码执行了KMeans聚类，并打印聚类结果，这里配置参数n\_clusters为5，即聚类5类。

import warnings  
warnings.filterwarnings('ignore')  
  
from sklearn.cluster import KMeans  
  
clusterer=KMeans(n\_clusters=5)  
LungCancer\_gdf["cluster\_labels"]=clusterer.fit\_predict(LungCancer\_gdf[['Lung Cancer']].values)  
  
gdf\_plot\_annotate(LungCancer\_gdf,"cluster\_labels","cluster\_labels",annotate\_fontsize=10,categorical=True,cmap="Set3")



def gdf\_parent\_child\_dissolve(gdf,cluster\_column,prefix="parent\_child"):  
 '''  
 将聚类簇结果按照地理空间邻接性切分，即同一簇中，邻接的单元划分为1组，结果信息存储与列parent\_child\_idx或parent\_child\_int中  
  
 Parameters  
 ----------  
 gdf : GeoDataFrame  
 含有簇/分组标签列的数据.  
 cluster\_column : string  
 簇/分组列名.  
 prefix : string, optional  
 列名前缀. The default is "parent\_child".  
  
 Returns  
 -------  
 overlay\_gdf : GeoDataFrame  
 重新切分的簇为一个polygon.  
 explode\_gdf : GeoDataFrame  
 炸开重新切分的簇为多个单元.  
  
 '''   
 from sklearn import preprocessing  
   
 parent\_child\_idx="%s\_idx"%prefix  
 parent\_child\_int="%s\_int"%prefix  
   
 dissolve\_gdf=gdf[['geometry',cluster\_column]].dissolve(by=cluster\_column)  
 explode\_gdf=dissolve\_gdf.explode()  
 explode\_gdf[parent\_child\_idx]=explode\_gdf.index   
 label\_encoder=preprocessing.LabelEncoder()   
 print(explode\_gdf[parent\_child\_idx].to\_list())  
   
 explode\_gdf[parent\_child\_int]=label\_encoder.fit\_transform([str(i) for i in explode\_gdf[parent\_child\_idx].to\_list()])  
 overlay\_gdf=gpd.overlay(gdf,explode\_gdf, how='intersection',make\_valid=True)  
   
 return overlay\_gdf,explode\_gdf

LungCancer\_overlay\_gdf,\_=gdf\_parent\_child\_dissolve(LungCancer\_gdf,'cluster\_labels')  
LungCancer\_overlay\_gdf

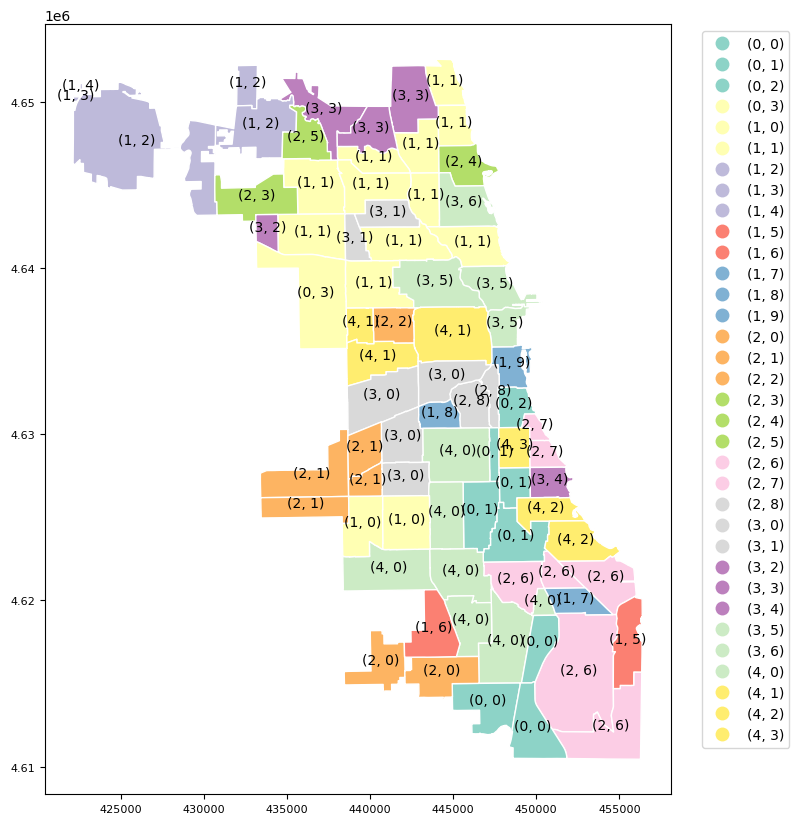
[(0, 0), (0, 1), (0, 2), (0, 3), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8), (1, 9), (2, 0), (2, 1), (2, 2), (2, 3), (2, 4), (2, 5), (2, 6), (2, 7), (2, 8), (3, 0), (3, 1), (3, 2), (3, 3), (3, 4), (3, 5), (3, 6), (4, 0), (4, 1), (4, 2), (4, 3)]

|  | **Community Area** | **Community Area Name** | **Lung Cancer** | **lag** | **binary\_median** | **li** | **p\_value\_li** | **li\_005** | **cl\_li** | **cluster\_labels** | **parent\_child\_idx** | **parent\_child\_int** | **geometry** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | Rogers Park | 36.7 | 38.050000 | 0\_low | 3 | 0.128 | 0 | 0 ns | 1 | (1, 1) | 5 | POLYGON ((445691.750 4649781.565, 445688.751 4... |
| **1** | 4 | Lincoln Square | 43.1 | 39.300000 | 0\_low | 3 | 0.018 | 3 | 3 cold spot | 1 | (1, 1) | 5 | POLYGON ((444126.361 4647333.168, 444126.692 4... |
| **2** | 5 | North Center | 42.4 | 39.312500 | 0\_low | 3 | 0.018 | 3 | 3 cold spot | 1 | (1, 1) | 5 | POLYGON ((444169.878 4642483.548, 444159.821 4... |
| **3** | 7 | Lincoln Park | 40.0 | 34.720000 | 0\_low | 3 | 0.006 | 3 | 3 cold spot | 1 | (1, 1) | 5 | POLYGON ((447619.016 4642483.907, 447618.814 4... |
| **4** | 14 | Albany Park | 36.9 | 38.900000 | 0\_low | 3 | 0.025 | 3 | 3 cold spot | 1 | (1, 1) | 5 | POLYGON ((441669.914 4647075.747, 441672.157 4... |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **74** | 65 | West Lawn | 36.7 | 52.566667 | 0\_low | 2 | 0.466 | 0 | 0 ns | 1 | (1, 0) | 4 | POLYGON ((440745.000 4623085.184, 440729.460 4... |
| **75** | 66 | Chicago Lawn | 42.9 | 53.000000 | 0\_low | 2 | 0.442 | 0 | 0 ns | 1 | (1, 0) | 4 | POLYGON ((443592.071 4623091.859, 443588.072 4... |
| **76** | 72 | Beverly | 47.9 | 60.580000 | 0\_low | 2 | 0.113 | 0 | 0 ns | 1 | (1, 6) | 10 | POLYGON ((444300.698 4620227.572, 444349.926 4... |
| **77** | 76 | O'Hare | 37.4 | 49.200000 | 0\_low | 3 | 0.421 | 0 | 0 ns | 1 | (1, 3) | 7 | POLYGON ((422147.083 4649993.382, 422145.649 4... |
| **78** | 76 | O'Hare | 37.4 | 49.200000 | 0\_low | 3 | 0.421 | 0 | 0 ns | 1 | (1, 4) | 8 | POLYGON ((422686.092 4650740.455, 422686.092 4... |

79 rows × 13 columns

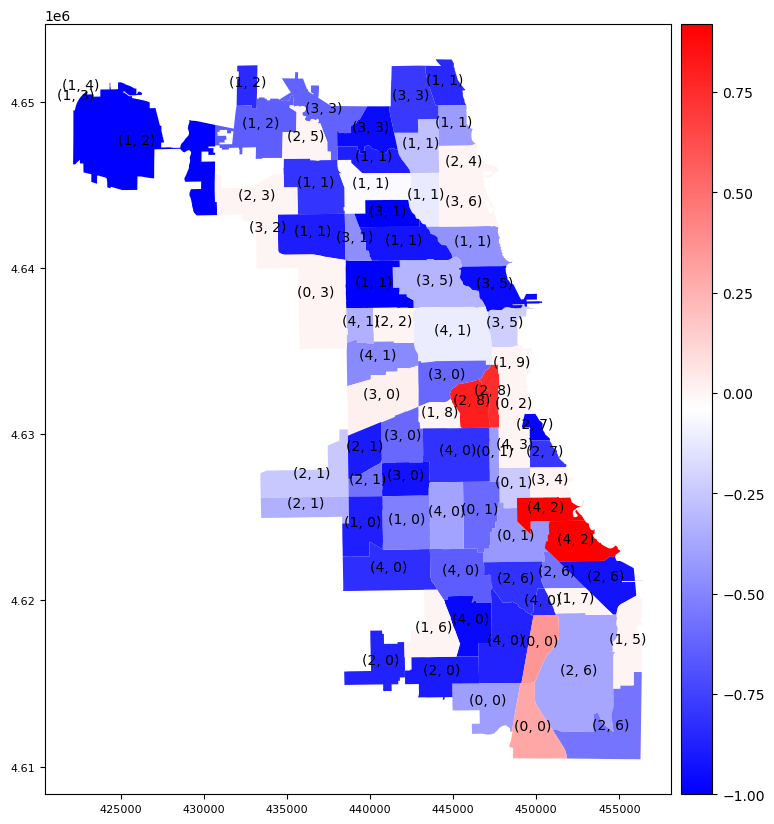
按照地理空间位置邻接切分聚类簇，并以元组形式表述，例如(1,1)，（1,0）等，第1个数值为原始聚类标签，第2个数值为同一簇切分后的子类标签。

gdf\_plot\_annotate(LungCancer\_overlay\_gdf,"parent\_child\_idx","parent\_child\_idx",annotate\_fontsize=10,categorical=True,cmap="Set3",legend\_bbox\_to\_anchor=(1.2, 1))



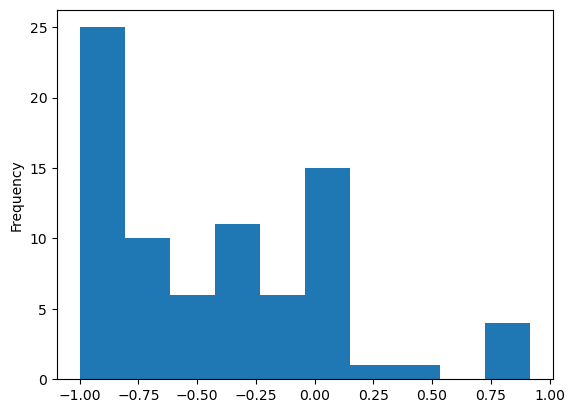
按照地理空间位置邻接切分聚类簇结果计算轮廓系数如下图，可以观察到，多数单元轮廓系数为负数（蓝色），有较强的异质性，相对组内相似度低而组间相似度高；个别单元为正数（红色），相对组内相似度高而组间相似度低。对聚类结果按照地理邻接切分后，使得各单元的轮廓系数变低，聚类效果不明显。

from sklearn.metrics import silhouette\_samples  
  
LungCancer\_overlay\_gdf["silhouettes\_pci"]=silhouette\_samples(LungCancer\_overlay\_gdf[['Lung Cancer']].values, LungCancer\_overlay\_gdf.parent\_child\_int)  
gdf\_plot\_annotate(LungCancer\_overlay\_gdf,"silhouettes\_pci","parent\_child\_idx",annotate\_fontsize=10,cmap="bwr")



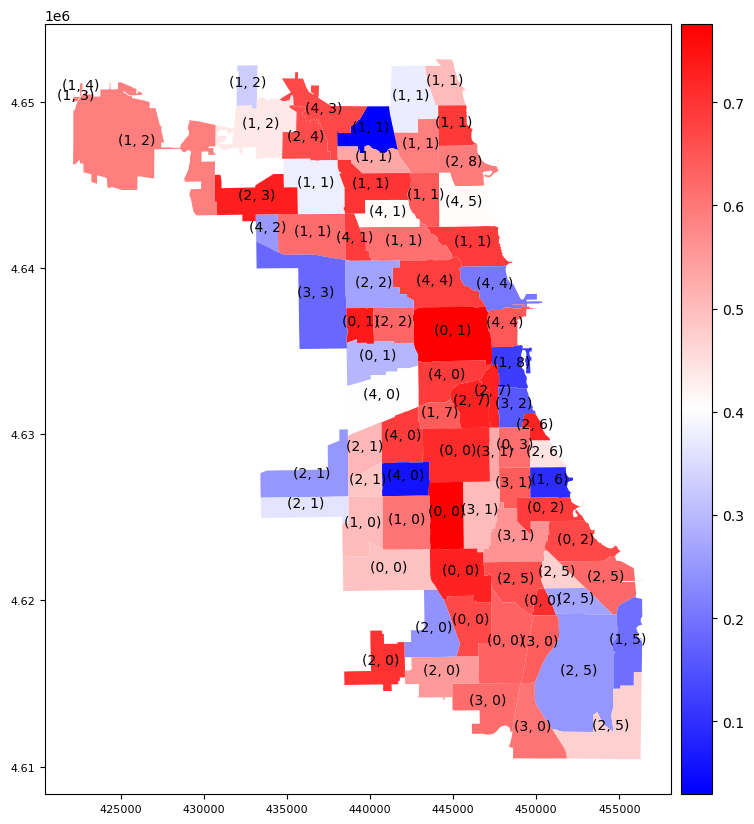
统计结果直方图的轮廓系数数值分布，也表明了较多单元的轮廓系数值小于0，为负值。

LungCancer\_overlay\_gdf["silhouettes\_pci"].plot.hist();

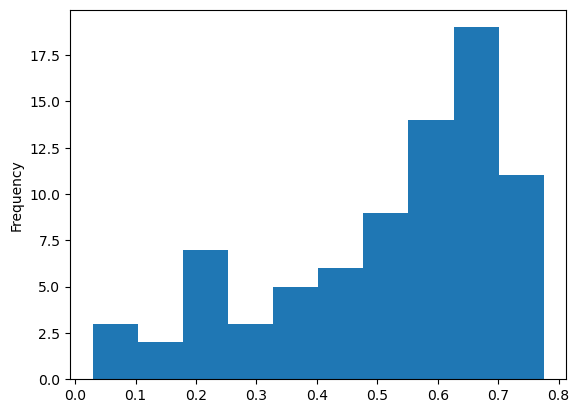


对未切分，最初的聚类结果执行轮廓系数计算，则可以看到多数为红色，聚类结果较好，但也有少数单元轮廓系数低，组内相似度低而组件高。

LungCancer\_overlay\_gdf["silhouettes\_cluster"]=silhouette\_samples(LungCancer\_overlay\_gdf[['Lung Cancer']].values, LungCancer\_overlay\_gdf.cluster\_labels)  
gdf\_plot\_annotate(LungCancer\_overlay\_gdf,"silhouettes\_cluster","parent\_child\_idx",annotate\_fontsize=10,cmap="bwr")



LungCancer\_overlay\_gdf["silhouettes\_cluster"].plot.hist();



#### 2）次优簇标签/最近邻吻合分布（Nearest Label ）

轮廓系数表述了单元组内和组间的相似度，并评估聚类的结果，但并未给出次优簇/第2最佳选择群组的标签，即，为位于中单元第2最佳选择群组。次优簇显示了目标单元和其它聚类簇之间潜在的关系，反映了如果单元选择次优簇可能对聚类结果的影响，从而可以辅助构建一个衡量标准，明确当前优于的程度。

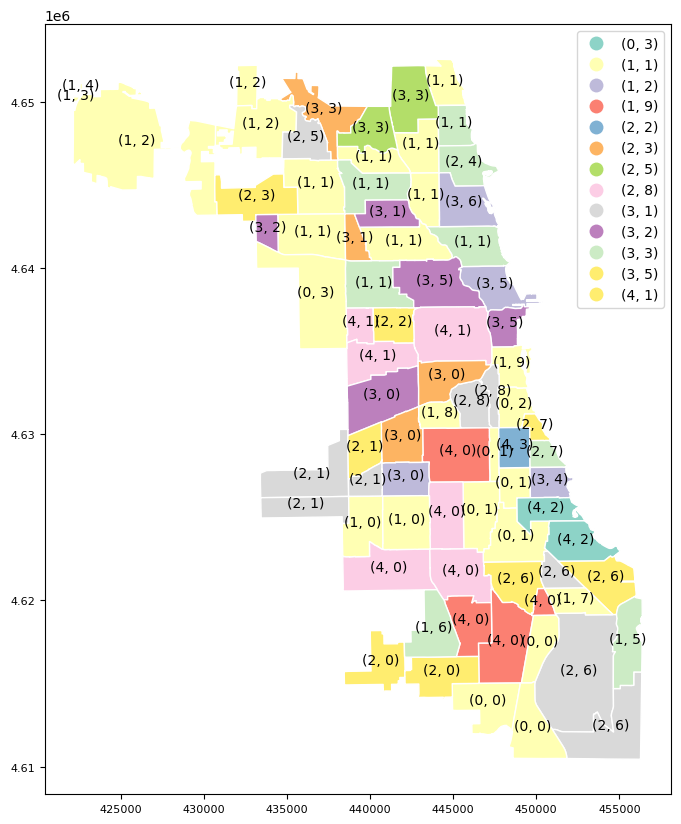
PySAL提供了esda.silhouettes.nearest\_label方法计算次优簇标签。

import esda  
import numpy as np  
  
LungCancer\_overlay\_gdf["nearest\_label"]=esda.silhouettes.nearest\_label(LungCancer\_overlay\_gdf[['Lung Cancer']].values, LungCancer\_overlay\_gdf.parent\_child\_int.values)  
LungCancer\_overlay\_gdf["nearest\_culster"]=np.asarray(LungCancer\_overlay\_gdf.parent\_child\_idx)[LungCancer\_overlay\_gdf.nearest\_label]  
LungCancer\_overlay\_gdf[:3]

|  | **Community Area** | **Community Area Name** | **Lung Cancer** | **lag** | **binary\_median** | **li** | **p\_value\_li** | **li\_005** | **cl\_li** | **cluster\_labels** | **parent\_child\_idx** | **parent\_child\_int** | **geometry** | **silhouettes\_pci** | **nearest\_label** | **nearest\_culster** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | Rogers Park | 36.7 | 38.0500 | 0\_low | 3 | 0.128 | 0 | 0 ns | 1 | (1, 1) | 5 | POLYGON ((445691.750 4649781.565, 445688.751 4... | -0.845133 | 7 | (1, 1) |
| **1** | 4 | Lincoln Square | 43.1 | 39.3000 | 0\_low | 3 | 0.018 | 3 | 3 cold spot | 1 | (1, 1) | 5 | POLYGON ((444126.361 4647333.168, 444126.692 4... | -0.274611 | 9 | (1, 1) |
| **2** | 5 | North Center | 42.4 | 39.3125 | 0\_low | 3 | 0.018 | 3 | 3 cold spot | 1 | (1, 1) | 5 | POLYGON ((444169.878 4642483.548, 444159.821 4... | -0.116809 | 9 | (1, 1) |

图中图例为次优簇，单元标签为当前簇。

gdf\_plot\_annotate(LungCancer\_overlay\_gdf,"nearest\_culster","parent\_child\_idx",annotate\_fontsize=10,categorical=True,cmap="Set3")

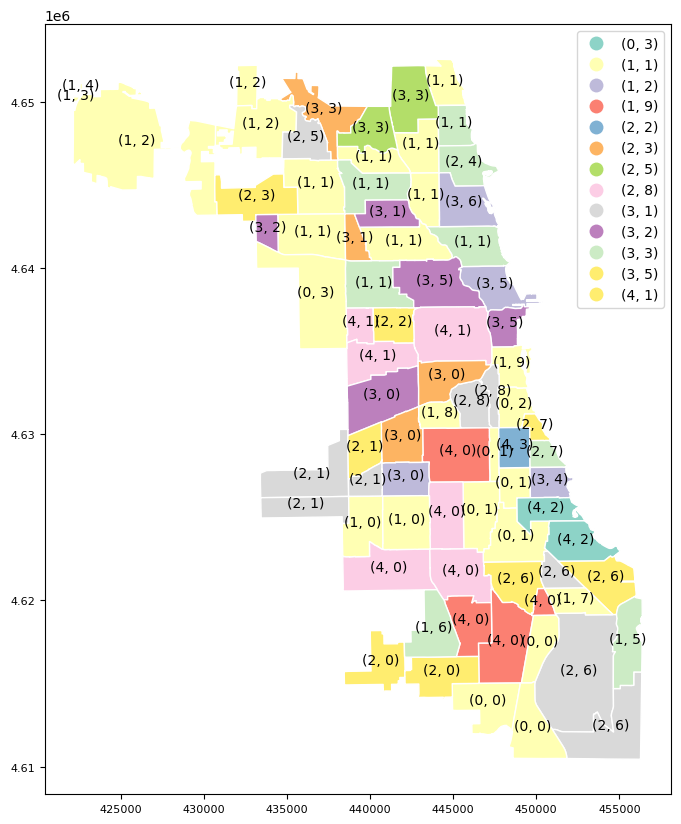


配置keep\_self参数为True，则包含当前单元所在的簇，对于按地理空间邻接切分后的群组，因为轮廓系数多数降低为负值，则单元的次优簇通常不会选择当前簇。

LungCancer\_overlay\_gdf["nearest\_label\_self"]=esda.silhouettes.nearest\_label(LungCancer\_overlay\_gdf[['Lung Cancer']].values, LungCancer\_overlay\_gdf.parent\_child\_int.values,keep\_self=True)  
LungCancer\_overlay\_gdf["nearest\_culster\_self"]=np.asarray(LungCancer\_overlay\_gdf.parent\_child\_idx)[LungCancer\_overlay\_gdf.nearest\_label]  
LungCancer\_overlay\_gdf[:3]

|  | **Community Area** | **Community Area Name** | **Lung Cancer** | **lag** | **binary\_median** | **li** | **p\_value\_li** | **li\_005** | **cl\_li** | **cluster\_labels** | **parent\_child\_idx** | **parent\_child\_int** | **geometry** | **silhouettes\_pci** | **nearest\_label** | **nearest\_culster** | **nearest\_label\_self** | **nearest\_culster\_self** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | Rogers Park | 36.7 | 38.0500 | 0\_low | 3 | 0.128 | 0 | 0 ns | 1 | (1, 1) | 5 | POLYGON ((445691.750 4649781.565, 445688.751 4... | -0.845133 | 7 | (1, 1) | 7 | (1, 1) |
| **1** | 4 | Lincoln Square | 43.1 | 39.3000 | 0\_low | 3 | 0.018 | 3 | 3 cold spot | 1 | (1, 1) | 5 | POLYGON ((444126.361 4647333.168, 444126.692 4... | -0.274611 | 9 | (1, 1) | 9 | (1, 1) |
| **2** | 5 | North Center | 42.4 | 39.3125 | 0\_low | 3 | 0.018 | 3 | 3 cold spot | 1 | (1, 1) | 5 | POLYGON ((444169.878 4642483.548, 444159.821 4... | -0.116809 | 9 | (1, 1) | 9 | (1, 1) |

gdf\_plot\_annotate(LungCancer\_overlay\_gdf,"nearest\_culster\_self","parent\_child\_idx",annotate\_fontsize=10,categorical=True,cmap="Set3")

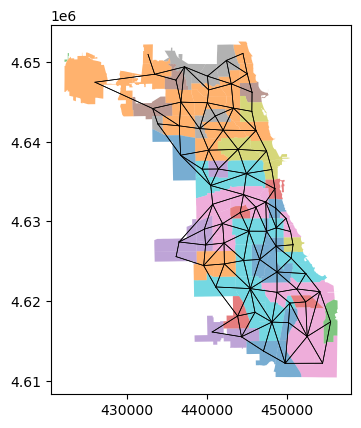


#### 3）地理结构（Geographical Structure）

地理结构为所用空间权重单元间空间连接的图或网络形式，每个单元（或几何中心点）为一个观测点（单元），边代表观察点之间的地理联系。

import libpysal.weights as LW  
w\_rook\_LC=LW.Rook.from\_dataframe(LungCancer\_overlay\_gdf)

import matplotlib.pyplot as plt  
  
f,ax=w\_rook\_LC.plot(LungCancer\_overlay\_gdf,edge\_kws=dict(linewidth=.5), node\_kws=dict(s=0))  
LungCancer\_overlay\_gdf.plot('parent\_child\_idx', ax=ax, alpha=.6)  
plt.show()



#### 4）Path Silhouettes （路径轮廓）

路径轮廓将两个观测单元之间的空间权重（地理位置关系）和属性特征相似度（异同）结合起来，表达为：，式中，为观测单元和间的属性特征相似度；为空间地理位置差异。通常由测量单元间的欧式距离衡量，但有时单元通过与之间的其它单元（假设有单元，）连接到，路径为最短路径，因此公式修正为： 。

PySAL库提供了esda.silhouettes.path\_silhouette方法计算路径轮廓。

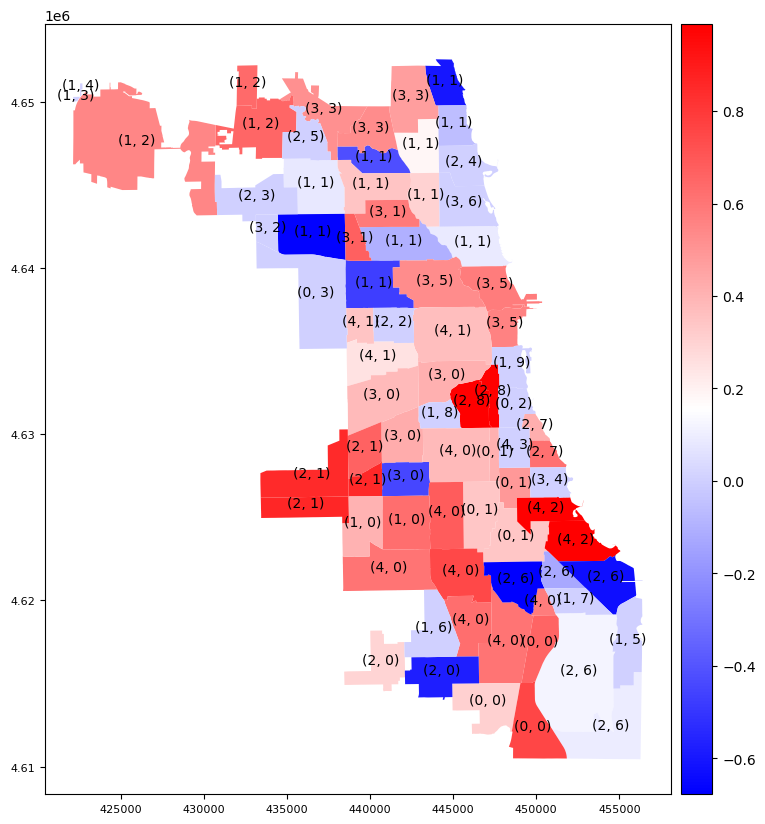
w\_queen\_overlay=LW.Queen.from\_dataframe(LungCancer\_overlay\_gdf)  
  
LungCancer\_overlay\_gdf["path\_silhouettes"]=esda.silhouettes.path\_silhouette(LungCancer\_overlay\_gdf[['Lung Cancer']].values, LungCancer\_overlay\_gdf.parent\_child\_int.values,w\_queen\_overlay)  
LungCancer\_overlay\_gdf[:3]

|  | **Community Area** | **Community Area Name** | **Lung Cancer** | **lag** | **binary\_median** | **li** | **p\_value\_li** | **li\_005** | **cl\_li** | **cluster\_labels** | **...** | **parent\_child\_int** | **geometry** | **silhouettes\_pci** | **nearest\_label** | **nearest\_culster** | **nearest\_label\_self** | **nearest\_culster\_self** | **path\_silhouettes** | **centroid** | **dbscan\_cluster** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | Rogers Park | 36.7 | 38.0500 | 0\_low | 3 | 0.128 | 0 | 0 ns | 1 | ... | 5 | POLYGON ((445691.750 4649781.565, 445688.751 4... | -0.845133 | 7 | (1, 1) | 7 | (1, 1) | -0.600258 | POINT (444506.602 4651061.851) | 0 |
| **1** | 4 | Lincoln Square | 43.1 | 39.3000 | 0\_low | 3 | 0.018 | 3 | 3 cold spot | 1 | ... | 5 | POLYGON ((444126.361 4647333.168, 444126.692 4... | -0.274611 | 9 | (1, 1) | 9 | (1, 1) | 0.186486 | POINT (443039.322 4647248.058) | 1 |
| **2** | 5 | North Center | 42.4 | 39.3125 | 0\_low | 3 | 0.018 | 3 | 3 cold spot | 1 | ... | 5 | POLYGON ((444169.878 4642483.548, 444159.821 4... | -0.116809 | 9 | (1, 1) | 9 | (1, 1) | 0.300000 | POINT (443320.001 4644205.717) | 1 |

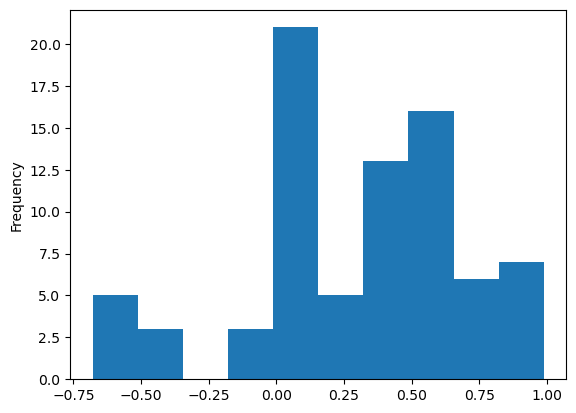
3 rows × 21 columns

从打印地图中可以得知，对于依据地理邻接切分后的簇，与前文计算的轮廓系数比较，在考虑了地理空间（空间权重）因素后，多数单元的路径轮廓为正值，表明具有较好的聚类结果；其中也包括少数蓝色单元，为组内相似度低，而组间相似度高，可以考虑选择次优簇。

\_=gdf\_plot\_annotate(LungCancer\_overlay\_gdf,"path\_silhouettes","parent\_child\_idx",annotate\_fontsize=10,cmap="bwr")



LungCancer\_overlay\_gdf["path\_silhouettes"].plot.hist();



为与KMeans聚类结果地理邻接切分的路径轮廓比较，下述实验则使用DBSCAN，不考虑测量（属性）值前提下，执行空间距离聚类将邻近的单元组合为簇，并计算该聚类结果的路径轮廓，可以发现多数单元为蓝色，表明仅考虑空间距离而忽略单元特征值的聚类，在考虑特征值计算路径轮廓，则路径轮廓值往往小于0，不能反映单元的特征属性值的聚类情况。

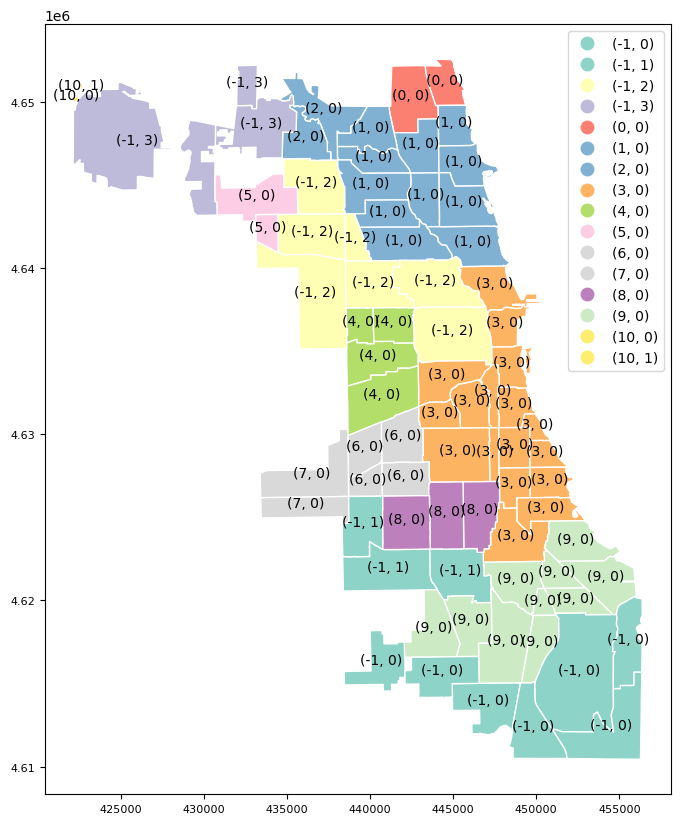
from sklearn.cluster import DBSCAN  
  
LungCancer\_overlay\_gdf['centroid']=LungCancer\_overlay\_gdf.geometry.apply(lambda row:row.centroid)  
centroids=LungCancer\_overlay\_gdf['centroid'].to\_list()  
centroids\_coordinates=np.array([[p.x,p.y] for p in centroids])  
clustering\_dbscan=DBSCAN(eps=2500, min\_samples=2).fit(centroids\_coordinates)  
LungCancer\_overlay\_gdf["dbscan\_cluster"]=clustering\_dbscan.labels\_  
LungCancer\_overlay\_dbscan\_gdf,\_=gdf\_parent\_child\_dissolve(LungCancer\_overlay\_gdf,'dbscan\_cluster',prefix="dbscan\_parent\_child")  
LungCancer\_overlay\_dbscan\_gdf[:3]

[(-1, 0), (-1, 1), (-1, 2), (-1, 3), (0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (5, 0), (6, 0), (7, 0), (8, 0), (9, 0), (10, 0), (10, 1)]

|  | **Community Area** | **Community Area Name** | **Lung Cancer** | **lag** | **binary\_median** | **li** | **p\_value\_li** | **li\_005** | **cl\_li** | **cluster\_labels** | **...** | **nearest\_label** | **nearest\_culster** | **nearest\_label\_self** | **nearest\_culster\_self** | **path\_silhouettes** | **centroid** | **dbscan\_cluster** | **dbscan\_parent\_child\_idx** | **dbscan\_parent\_child\_int** | **geometry** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | Rogers Park | 36.7 | 38.05 | 0\_low | 3 | 0.128 | 0 | 0 ns | 1 | ... | 7 | (1, 1) | 7 | (1, 1) | -0.600258 | POINT (444506.602 4651061.851) | 0 | (0, 0) | 4 | POLYGON ((445688.751 4649783.865, 445686.199 4... |
| **1** | 2 | West Ridge | 36.0 | 38.65 | 0\_low | 3 | 0.052 | 0 | 0 ns | 3 | ... | 27 | (2, 5) | 27 | (2, 5) | 0.469072 | POINT (442441.974 4650184.259) | 0 | (0, 0) | 4 | POLYGON ((443319.829 4652121.144, 443322.049 4... |
| **2** | 4 | Lincoln Square | 43.1 | 39.30 | 0\_low | 3 | 0.018 | 3 | 3 cold spot | 1 | ... |  |  |  |  |  |  |  |  |  |  |

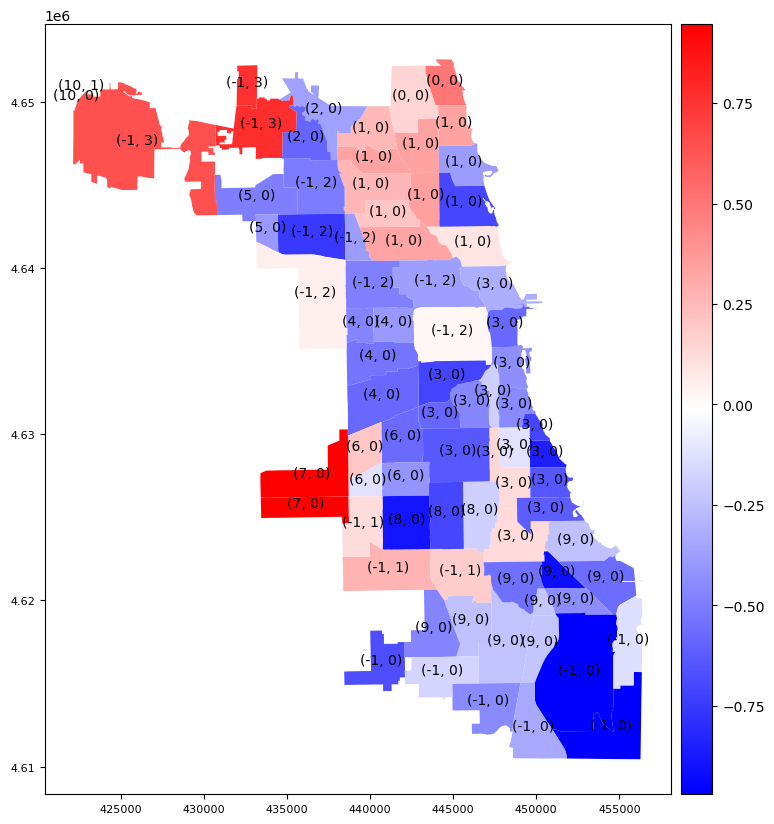
3 rows × 23 columns

\_=gdf\_plot\_annotate(LungCancer\_overlay\_dbscan\_gdf,"dbscan\_parent\_child\_idx","dbscan\_parent\_child\_idx",annotate\_fontsize=10,categorical=True,cmap="Set3")



LungCancer\_overlay\_dbscan\_gdf["path\_silhouettes\_dbscan"]=esda.silhouettes.path\_silhouette(LungCancer\_overlay\_dbscan\_gdf[['Lung Cancer']].values, LungCancer\_overlay\_dbscan\_gdf.dbscan\_parent\_child\_int.values,w\_rook\_LC)

gdf\_plot\_annotate(LungCancer\_overlay\_dbscan\_gdf,"path\_silhouettes\_dbscan","dbscan\_parent\_child\_idx",annotate\_fontsize=10,cmap="bwr")



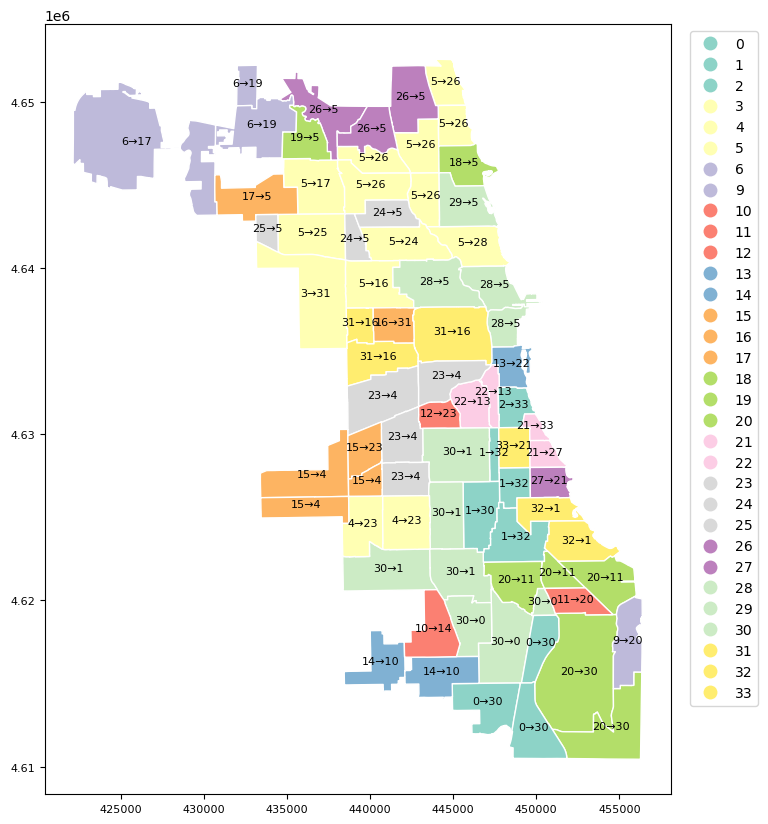
计算路径轮廓时，如果配置参数return\_nbfc为True，则会返回次优簇。

注意，在按地理邻近分解簇时，将位于簇1的单元75也分解为了小片区域，其中新产生的77和78单元并不与其它单元邻接，即不共边，也无点连通，因此计算Queen空间权重时，值为空值，如果为空值，在返回次优簇时esda.silhouettes.path\_silhouette方法会提示错误，因此先移除了这两个无空间权重值的单元。

LungCancer\_overlay\_dropped\_gdf=LungCancer\_overlay\_gdf.drop(index=[77,78])  
w\_queen\_overlay\_dropped=LW.Queen.from\_dataframe(LungCancer\_overlay\_dropped\_gdf)  
  
LungCancer\_overlay\_dropped\_gdf["path\_silhouette"], LungCancer\_overlay\_dropped\_gdf["next\_best\_path"]=esda.silhouettes.path\_silhouette(LungCancer\_overlay\_dropped\_gdf[['Lung Cancer']].values, LungCancer\_overlay\_dropped\_gdf.parent\_child\_int.values,w\_queen\_overlay\_dropped,return\_nbfc=True)

图中图例为当前簇标签，单元标注为当前簇→次优簇。

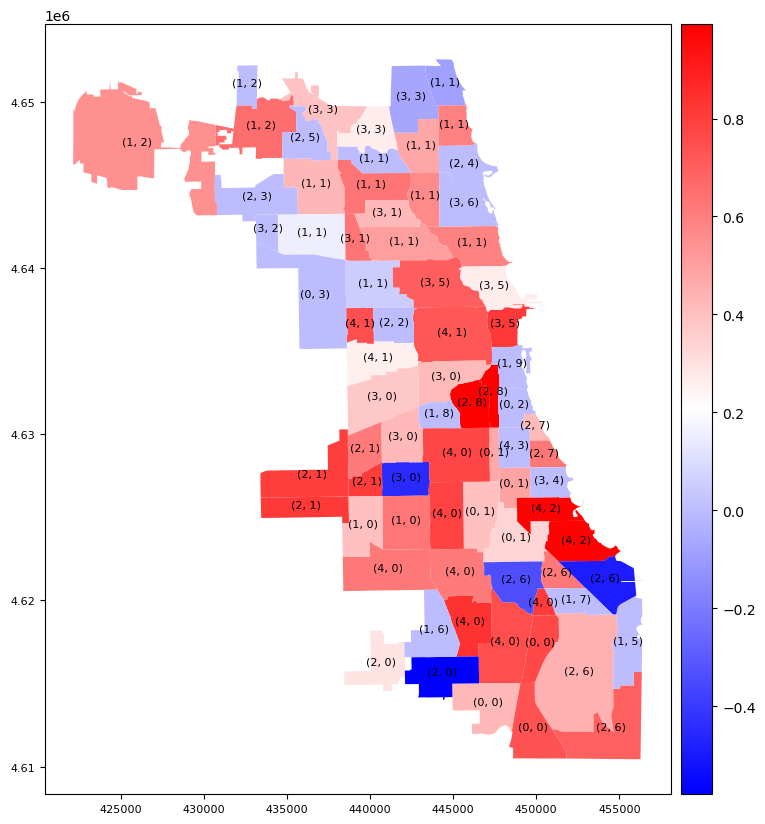
LungCancer\_overlay\_dropped\_gdf["pci\_nbp"]=LungCancer\_overlay\_dropped\_gdf.apply(lambda row:f"{row.parent\_child\_int}→{row.next\_best\_path}",axis=1)  
  
gdf\_plot\_annotate(LungCancer\_overlay\_dropped\_gdf,"parent\_child\_int","pci\_nbp",annotate\_fontsize=8,categorical=True,cmap="Set3",legend\_bbox\_to\_anchor=(1.15, 1))



#### 5）Boundary Silhouettes（边缘轮廓）

目标单元路径轮廓指示的次优簇可能并不在当前簇邻近的簇中，如果考虑地理空间邻接的特性，则分析可能并不现实，因此边缘轮廓则考虑了这一需求，将注意力限制在邻近的簇中，从而将目标单元归于次优簇时，也不会在地理空间上被孤立。当一个单元位于邻近簇c时，具有更好的相似度，则边缘轮廓系数值为负值；反之为正值。从下述打印的地图中可以观察到，大部分单元都具有较高的组内相似度和较低的组间相似度；而蓝色小于0的单元则可以考虑归于邻接的次优簇。

LungCancer\_overlay\_dropped\_gdf["boundary\_silhouette"]=esda.silhouettes.boundary\_silhouette(LungCancer\_overlay\_dropped\_gdf[['Lung Cancer']].values, LungCancer\_overlay\_dropped\_gdf.parent\_child\_int.values,w\_queen\_overlay\_dropped)  
gdf\_plot\_annotate(LungCancer\_overlay\_dropped\_gdf,"boundary\_silhouette","parent\_child\_idx",cmap='bwr',annotate\_fontsize=8)



注释（Notes）：

① 芝加哥公共健康数据，该数据集包含了按芝加哥社区区域划分的27个具有公共卫生意义的指标（<https://data.cityofchicago.org/Health-Human-Services/Public-Health-Statistics-Selected-public-health-in/iqnk-2tcu>）。

② Chicago Data Portal，为芝加哥城开放数据门户，可免费下载数据用于相关分析，其中许多数据集每天至少更新一次或数次（<https://data.cityofchicago.org/>）。

③ PySAL，支持空间数据科学计算的开源项目（<https://pysal.org/>）。

参考文献（References）:

[1] Sergio J. Rey, Dani Arribas-Bel, Levi J. Wolf. Geographic Data Science with Python[M]. licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.](https://creativecommons.org/licenses/by-nc-nd/4.0/), 2020.

[2] [PySAL Notebooks Book](http://pysal.org/notebooks/intro.html)，http://pysal.org/notebooks/intro.html.

[3] Tobler W. A computer movie simulating urban growth in the Detroit region[J]. Economic Geography, 1970, 46(Supplement): 234–240.

[4] Kemp, Karen. Encyclopedia of Geographic Information Science, SAGE, 2008, pp 146–147.

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

[6] Arthur Getis and JK Ord. The analysis of spatial association by use of distance statistics. Geographical Analysis, 24(3):189–206, 1992.

[7] Wolf, L. J., Knaap, E., & Rey, S. (2021). Geosilhouettes: Geographical measures of cluster fit. Environment and Planning B: Urban Analytics and City Science, 48(3), 521–539. https://doi.org/10.1177/2399808319875752