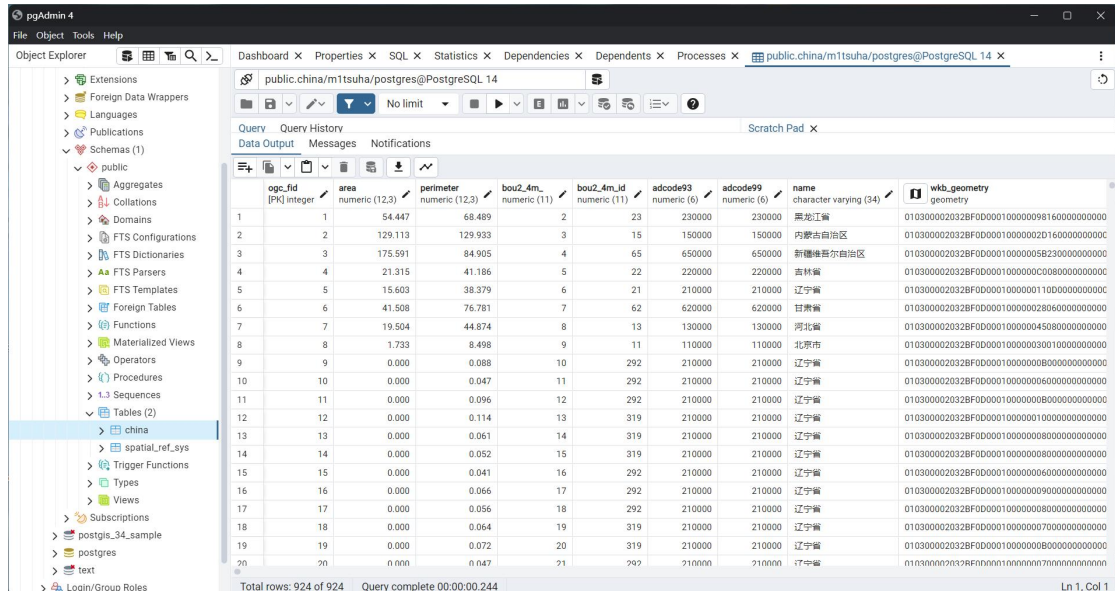


Q1:

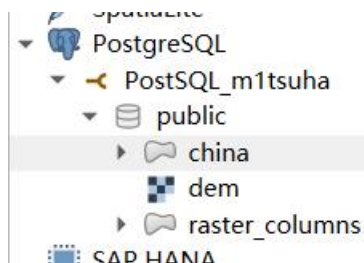
ogr2ogr 矢量数据导入数据库

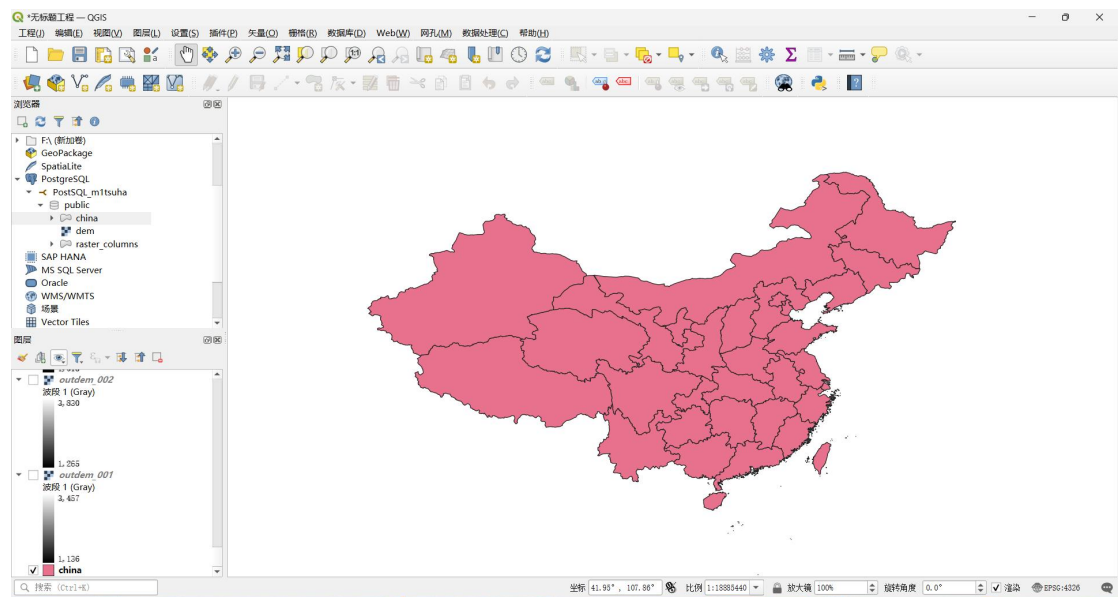
```
ogr2ogr -f PostgreSQL "PG:dbname=m1tsuha user=postgres password=123456" -lco  
PG_USE_COPY=YES -lco SHAPE_ENCODING=GBK -progress -update -append -gt -1 -nln china  
C:/Users/86187/Desktop/Term/sophomore_2/GeoSpatialDataAnalysis/Data/china_shp/china.shp
```



ogr_fid [PK] integer	area numeric (12,3)	perimeter numeric (12,3)	bou2_4m numeric (11)	bou2_4m_id numeric (11)	adcode93 numeric (6)	adcode99 numeric (6)	name character varying (34)	wkb_geometry geometry
1	54.447	68.489	2	23	230000	230000	黑龙江省	010300002032BF0D00010000009816000000000000
2	129.113	129.933	3	15	150000	150000	内蒙古自治区	010300002032BF0D000100000002D160000000000000
3	175.591	84.905	4	65	650000	650000	新疆维吾尔自治区	010300002032BF0D000100000005B230000000000000
4	21.315	41.186	5	22	220000	220000	吉林省	010300002032BF0D00010000000C0C08000000000000
5	15.603	38.379	6	21	210000	210000	辽宁省	010300002032BF0D00010000000110D0000000000000
6	41.508	76.781	7	62	620000	620000	甘肃省	010300002032BF0D0001000000028060000000000000
7	19.504	44.874	8	13	130000	130000	河北省	010300002032BF0D0001000000045080000000000000
8	1.733	8.498	9	11	110000	110000	北京市	010300002032BF0D0001000000030010000000000000
9	0.000	0.088	10	292	210000	210000	辽宁省	010300002032BF0D000100000008B000000000000000
10	0.000	0.047	11	292	210000	210000	辽宁省	010300002032BF0D0001000000064000000000000000
11	0.000	0.096	12	292	210000	210000	辽宁省	010300002032BF0D000100000008B000000000000000
12	0.000	0.114	13	319	210000	210000	辽宁省	010300002032BF0D0001000000010000000000000000
13	0.000	0.061	14	319	210000	210000	辽宁省	010300002032BF0D0001000000080000000000000000
14	0.000	0.052	15	319	210000	210000	辽宁省	010300002032BF0D0001000000080000000000000000
15	0.000	0.041	16	292	210000	210000	辽宁省	010300002032BF0D0001000000064000000000000000
16	0.000	0.066	17	292	210000	210000	辽宁省	010300002032BF0D0001000000099000000000000000
17	0.000	0.056	18	292	210000	210000	辽宁省	010300002032BF0D0001000000080000000000000000
18	0.000	0.064	19	319	210000	210000	辽宁省	010300002032BF0D0001000000070000000000000000
19	0.000	0.072	20	319	210000	210000	辽宁省	010300002032BF0D0001000000080000000000000000
20	0.000	0.047	21	292	210000	210000	辽宁省	010300002032BF0D0001000000070000000000000000

qgis 连接数据库进行显示



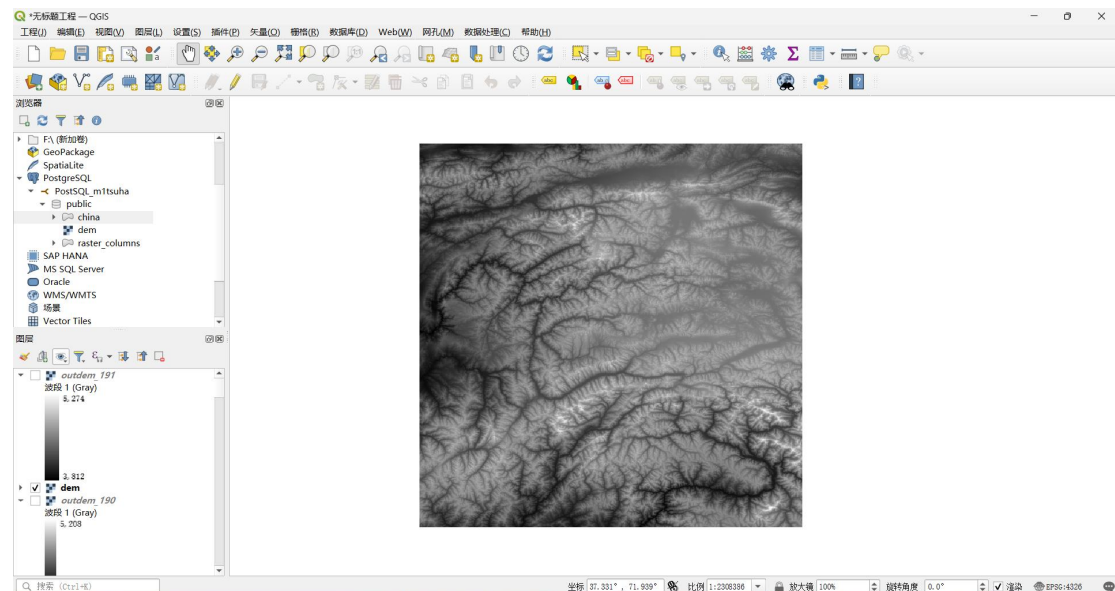


Q2

raster2pgsql 导入 s51.tif 到数据库

```
raster2pgsql -s 4326 -l -C -M
```

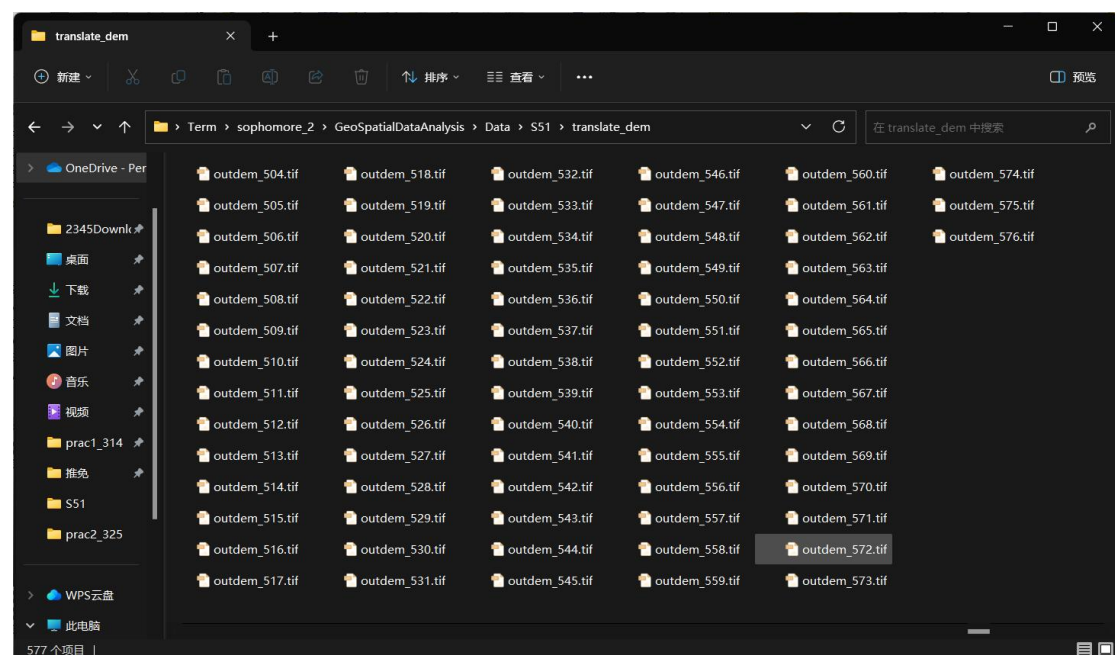
```
C:/Users/86187/Desktop/Term/sophomore_2/GeoSpatialDataAnalysis/Data/S51/s51.tif -F -t  
256x256 public.dem |psql -h localhost -p 5432 -U postgres -d m1tsuha
```



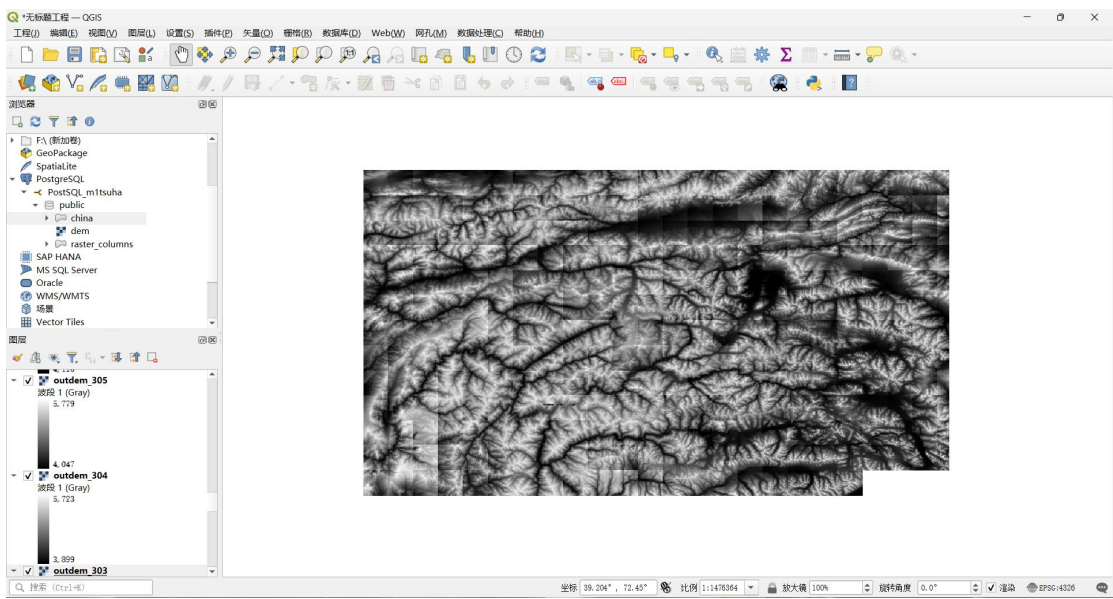
栅格数据用 gdal_translate 命令出到本地文件夹中

```
gdal_translate -of GTiff "PG:dbname=m1tsuha schema=public table=dem user=postgres  
password=123456" -sds
```

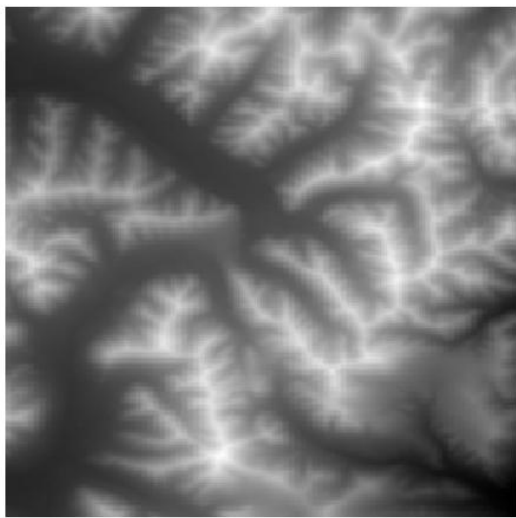
```
C:/Users/86187/Desktop/Term/sophomore_2/GeoSpatialDataAnalysis/Data/S51/translate_dem/  
outdem.tif
```



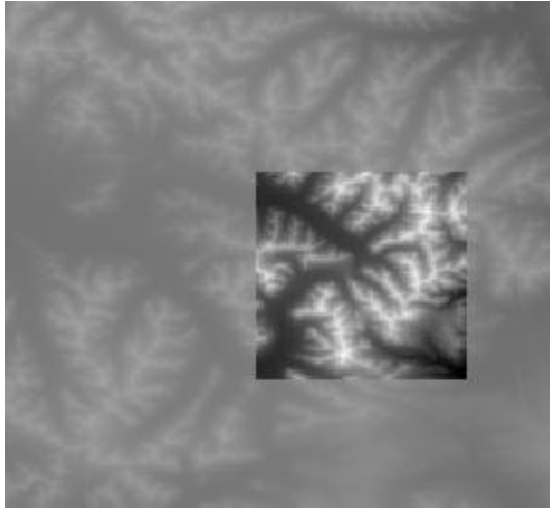
导入 308 张整体预览



索引为 191 的分块细节



在原图像的图层之上展示

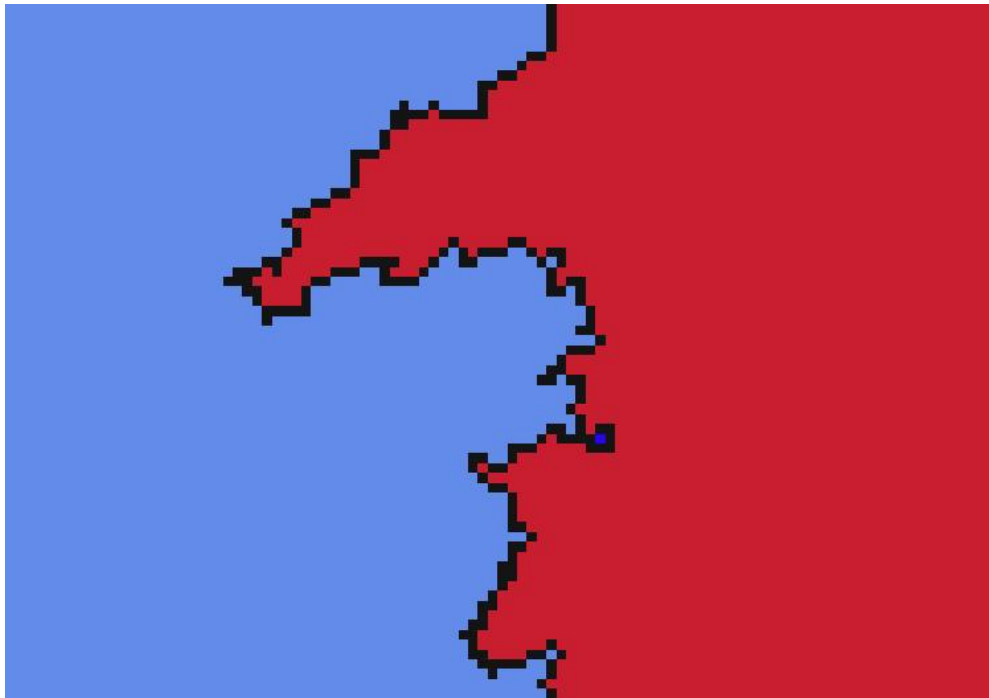


Q3:

处理结果如图



其中湖南省片段细节如图



更改代码如图

```

10 r = shapefile.Reader("china_shp/china.shp")
11 xdist = r.bbox[2] - r.bbox[0]
12 ydist = r.bbox[3] - r.bbox[1]
13 iwidth = 3000
14 iheight = int(iwidth / xdist * ydist)
15 xratio = iwidth / xdist
16 yratio = iheight / ydist
17
18 img = Image.new("RGB", (iwidth, iheight), "white")
19 draw = ImageDraw.Draw(img)
20 idx = 0
21
22 for shape in r.shapes():
23     # Initialize an empty List to store polygons for this shape
24     shape_polys = []
25
26     # Keep track of the last part's end index
27     last_part_end_index = 0
28     idx=idx+1
29     # Iterate over the parts array. part_index:0,1,2,3... start_index:第part_index的起始索引, end_index:第part_index+1的起始索引
30     for part_index in range(len(shape.parts)):#循环遍历parts (是一个列表, 将点集合组合为形状, 如果形状具有多个部分, 则显示的是每个部分的第一点索引。)
31         start_index = shape.parts[part_index]#记录当前shape中某个part的起始索引
32         if part_index < len(shape.parts) - 1:#如果不是最后一个part
33             end_index = shape.parts[part_index + 1]#记录当前shape某个part的结束索引
34         else:
35             # For the last part, use the total number of points
36             end_index = len(shape.points)
37
38         # Extract the points belonging to the current part
39         part_points = shape.points[start_index:end_index]
40
41         if part_index == 1:
42             print(idx)
43         # Convert map coordinates to pixel coordinates and store them in a new list
44         part_pixels = [(int(iwidth - ((r.bbox[2] - x) * xratio)), int((r.bbox[3] - y) * yratio)) for x, y in part_points]
45
46         # Append the part's pixels to the list of polygons for this shape
47         shape_polys.append(part_pixels)
48         shape_polys.reverse()
49
50         # Draw the polygon using the stored pixel coordinates
51         if len(part_pixels) > 2:
52             fill_color_r = int(255 * random.random())
53             fill_color_g = int(255 * random.random())
54             fill_color_b = int(255 * random.random())
55             draw.polygon(part_pixels, outline="rgb(20,20,20)", fill=f"rgb({fill_color_r},{fill_color_g},{fill_color_b})")
56
57         # Continue with the next shape in the Shapefile
58
59 img.save("Output/result.png")

```


功能 1:

```
21 for shape in r.shapes():
22     # Initialize an empty list to store polygons for this shape
23     shape_polys = []
24
25     # Keep track of the last part's end index
26     last_part_end_index = 0
27
28     # Iterate over the parts array. part_index:0,1,2,3... start_index:第part_index的起始索引, end_index:第part_index+1的起始索引
29     for part_index in range(len(shape.parts)):#遍历shape.parts (是一个列表, 将点集合组合为形状, 如果形状具有多个部分, 则显示的是每个部分的第一点索引。)
30         start_index = shape.parts[part_index]#记录当前shape中某个part的起始索引
31         if part_index < len(shape.parts) - 1:#如果不是最后一个part
32             end_index = shape.parts[part_index + 1]#记录当前shape某个part的结束索引
33         else:
34             # For the last part, use the total number of points
35             end_index = len(shape.points)
36
37         # Extract the points belonging to the current part
38         part_points = shape.points[start_index:end_index]
39
40         # Convert map coordinates to pixel coordinates and store them in a new list
41         part_pixels = [(int(iwidth - ((r.bbox[2] - x) * xratio)), int((r.bbox[3] - y) * yratio)) for x, y in part_points]
42
43         # Append the part's pixels to the list of polygons for this shape
44         shape_polys.append(part_pixels)
45
```

读取每个 shape 的所有 parts 中的 points 并根据不同 part 分配其对应的 points 存入列表中
由于 shape.parts()返回的是每个部分的第一点索引的列表
所以我的处理方式是将每个 shape 中的 parts 数目和其对应的 points 索引值存储下来
通过索引值将每个 part 中的 points 存入到 part_points 的列表中
经过画布对应坐标变换后再存入 shape_polys 中便于后面绘制过程。

功能 2:

```
# Draw the polygon using the stored pixel coordinates
if len(part_pixels) > 2:
    fill_color_r = int(255 * random.random())
    fill_color_g = int(255 * random.random())
    fill_color_b = int(255 * random.random())
    draw.polygon(part_pixels, outline="rgb(20,20,20)", fill=f"rgb({fill_color_r},{fill_color_g},{fill_color_b})")

# Continue with the next shape in the Shapefile
```

导入 random 库

通过随机生成 rgb 值来赋予每个 poly 中 fill

使得每个 part 填充的颜色随机。

功能 3:

```
r = shapefile.Reader("china_shp/china.shp")
xdist = r.bbox[2] - r.bbox[0]
ydist = r.bbox[3] - r.bbox[1]
iwidth = 3000
iheight = int(iwidth / xdist * ydist)
xratio = iwidth / xdist
yratio = iheight / ydist
```

先通过 `r.bbox` 读取 `shp` 的数据范围并确定长宽比
通过此宽高比来使得画出的图片宽高比与矢量数据保持一致。