

Python程序设计-大作业

班级: 2019211306

学号: 2019211397

姓名: 毛子恒

1 作业题目

1.1 数据

gpw-v4-population-count-rev11_2020_30_sec_asc.zip是一个全球人口分布数据压缩文件，解压后包括了8个主要的asc后缀文件，他们是全球网格化的人口分布数据文件，这些文件分别是：

- gpw-v4-population-count-rev11_2020_30_sec_1.asc
- gpw-v4-population-count-rev11_2020_30_sec_2.asc
- gpw-v4-population-count-rev11_2020_30_sec_3.asc
- gpw-v4-population-count-rev11_2020_30_sec_4.asc
- gpw-v4-population-count-rev11_2020_30_sec_5.asc
- gpw-v4-population-count-rev11_2020_30_sec_6.asc
- gpw-v4-population-count-rev11_2020_30_sec_7.asc
- gpw-v4-population-count-rev11_2020_30_sec_8.asc

这些文件分布对应地球不同经纬度的范围。

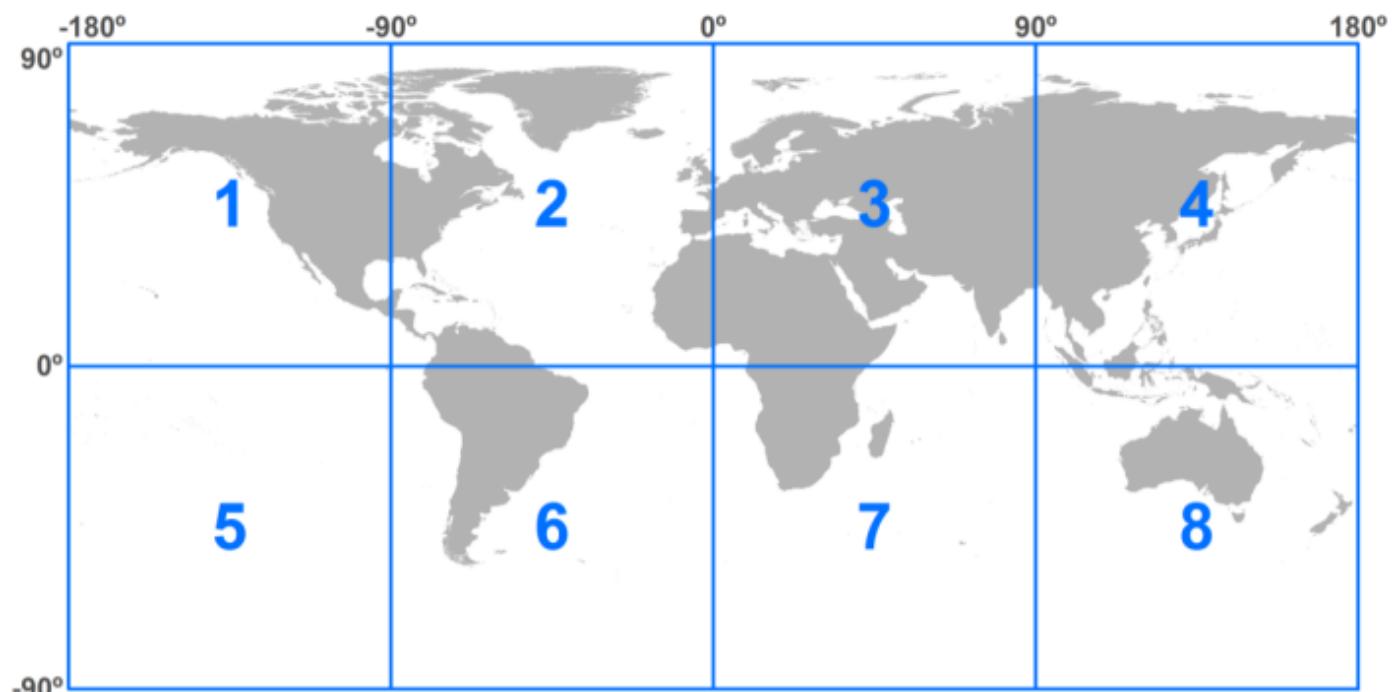


Figure 1. Tiling of 30 arc-second ASCII rasters.

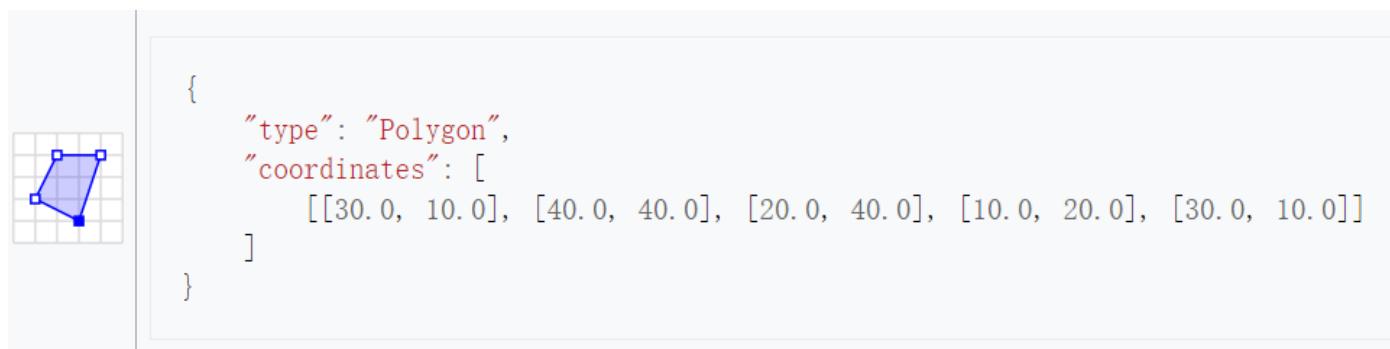
压缩文件下载网页: <https://sedac.ciesin.columbia.edu/data/set/gpw-v4-population-count-rev11/data-download>

1.2 服务端

压缩文件 (gpw-v4-population-count-rev11_2020_30_sec_asc.zip) 是一个全球人口分布数据。基于Sanic实现一个查询服务，服务包括：

- 按给定的经纬度范围查询人口总数，查询结果采用JSON格式。
- 不可以采用数据库，只允许使用文件方式存储数据。
- 可以对现有数据进行整理以便加快查询速度，尽量提高查询速度。

查询参数格式 采用GeoJSON (<https://geojson.org/>) 的多边形（每次只需要查询一个多边形范围，只需要支持凸多边形）



1.3 客户端

针对上面的查询服务，实现一个服务查询客户端，数据获取后使用Matplotlib散点图（Scatter）进行绘制。

- 横坐标（x轴）为经度。
- 纵坐标（y轴）为维度。

2 服务端代码

```
1 import asyncio
2 import math
3 import os
4 import numpy as np
5 from sanic import Sanic
6 from sanic.response import json
7 from sanic.log import logger
8 from shapely.geometry import Polygon
9 from shapely.errors import TopologicalError
10
11 app = Sanic("app")
12
13
14 @app.listener("before_server_start")
15 async def setup(app, loop):
16     for i in range(1, 9):
17         step = 10 # 处理成10度*10度的block
```

```

18     with open(f"./gpw-v4-population-count-
rev11_2020_30_sec_asc/gpw_v4_population_count_rev11_2020_30_sec_{i}.asc",
19             "r") as f:
20         f.readline()
21         f.readline()
22         stx = int(float(f.readline().split()[1])) # 左上角经度
23         sty = int(float(f.readline().split()[1])) + 90 # 左上角纬度
24         cooked = True # 该部分中的所有block是否已经被处理
25         for x_offset in range(0, 90, step):
26             for y_offset in range(0, 90, step):
27                 if not os.path.exists(f"./data/data_{stx + x_offset}_{sty -
y_offset}.npy"):
28                     cooked = False
29                     logger.info(f"Processing data on x: {stx} y: {sty}")
30                     if cooked:
31                         continue
32                     data = np.genfromtxt(
33                         f"./gpw-v4-population-count-
rev11_2020_30_sec_asc/gpw_v4_population_count_rev11_2020_30_sec_{i}.asc",
34                         skip_header=6)
35                     data[data == -9999] = np.nan # 空数据
36                     for x_offset in range(0, 90, step):
37                         for y_offset in range(0, 90, step):
38                             if not os.path.exists(f"./data/data_{stx + x_offset}_{sty -
y_offset}.npy"):
39                                 logger.info(f"Creating data_{stx + x_offset}_{sty -
y_offset}.npy")
40                                 np.save(f"./data/data_{stx + x_offset}_{sty - y_offset}.npy",
41                                         data[y_offset * 120:(y_offset + step) * 120, x_offset *
42                                         120:(x_offset + step) * 120])
43
44     async def get_block_data(block_x, block_y, polygon, step):
45         logger.info(f"Query on block x: {block_x} y: {block_y}")
46         res = []
47         total = 0
48         minx, miny, maxx, maxy = polygon.bounds
49         block_data = np.load(f"./data/data_{block_x}_{block_y}.npy")
50         for second_x in range(max(block_x * 3600, math.floor(minx / 30) * 30),
51                               min((block_x + step) * 3600, math.ceil(maxx / 30) * 30),
52                               30): # 按每30秒枚举经度
53             for second_y in range(min(block_y * 3600, math.ceil(maxy / 30) * 30),
54                               max((block_y - step) * 3600, math.floor(miny / 30) * 30),
55                               -30): # 按每30秒枚举纬度
56                 x_offset = int((block_y * 3600 - second_y) / 30) # 在block中的行偏移量

```

```

55         y_offset = int((second_x - block_x * 3600) / 30) # 在block中的列偏移量
56         cell_polygon = Polygon(((second_x, second_y), (second_x + 30, second_y),
57                                   (second_x + 30, second_y - 30),
58                                   (second_x, second_y - 30))).intersection(polygon)
59         # 该cell和多边形的重合部分
60         if cell_polygon.area > 0: # 如果有重合
61             res.append((second_x, second_y, cell_polygon.area / 900 *
62                         block_data[x_offset, y_offset]))
63             if not np.isnan(res[-1][2]):
64                 total += res[-1][2]
65
66     return res, total
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91 if __name__ == '__main__':
92     app.run()

```

2.1 代码说明

2.1.1 数据预处理

服务端首先读取ASCII格式的源数据，源数据的一个文件保存 $90^\circ \times 90^\circ$ 的人口分布，我将其重新划分为 $10^\circ \times 10^\circ$ 的block，并且处理其中以`-9999`标识的空数据，以`numpy`的二进制格式存储，命名为`data_x_y.npy`，其中`x`和`y`是该block中左上角的经度和纬度。

2.1.2 查询处理

查询参数采用`GeoJSON`格式的多边形数据，坐标单位为角秒。服务端利用`shapely`包根据多边形的顶点创建多边形，使用`bounds`方法获取经度和纬度的范围，以此计算出该多边形跨哪些block，对于每个包含多边形的block，遍历该block中与多边形重合的经纬度，以 $30''$ 为一个跨度，对于其中每个 $30'' \times 30''$ 的cell，计算它和多边形重合的面积，如果面积大于0，则将重合的面积比整个cell面积的权重乘这个cell中的人口数（即实际在多边形中的人口数）加入答案当中。

2.1.3 性能优化

由于数据预处理的时长超过20分钟，在预处理前会检查目录，对于已经预处理的数据会跳过。

查询时对于每个包含多边形的block的处理是并行的，相比较直接从大文件中读取数据，对于小查询可以节省读取文件的时间和内存，对于大查询则可以优化查询时间。

2.1.4 日志

基于`Sanic`的日志系统添加了一些说明信息，一个简单的例子如下：

```
1 [2022-01-10 17:03:30 +0800] [35399] [INFO] Sanic v21.12.1
2 [2022-01-10 17:03:30 +0800] [35399] [INFO] Goin' Fast @ http://127.0.0.1:8000
3 [2022-01-10 17:03:30 +0800] [35399] [INFO] mode: production, single worker
4 [2022-01-10 17:03:30 +0800] [35399] [INFO] server: sanic
5 [2022-01-10 17:03:30 +0800] [35399] [INFO] python: 3.9.7
6 [2022-01-10 17:03:30 +0800] [35399] [INFO] platform: macOS-10.16-x86_64-i386-64bit
7 [2022-01-10 17:03:30 +0800] [35399] [INFO] packages: sanic-routing==0.7.2
8 [2022-01-10 17:03:30 +0800] [35399] [INFO] Processing data on x: -180 y: 90
9 [2022-01-10 17:03:30 +0800] [35399] [INFO] Processing data on x: -90 y: 90
10 [2022-01-10 17:03:30 +0800] [35399] [INFO] Processing data on x: 0 y: 90
11 [2022-01-10 17:03:30 +0800] [35399] [INFO] Processing data on x: 90 y: 90
12 [2022-01-10 17:03:30 +0800] [35399] [INFO] Processing data on x: -180 y: 0
13 [2022-01-10 17:03:30 +0800] [35399] [INFO] Processing data on x: -90 y: 0
14 [2022-01-10 17:03:30 +0800] [35399] [INFO] Processing data on x: 0 y: 0
15 [2022-01-10 17:03:30 +0800] [35399] [INFO] Processing data on x: 90 y: 0
16 [2022-01-10 17:03:30 +0800] [35399] [INFO] Starting worker [35399]
17 [2022-01-10 17:03:32 +0800] [35399] [INFO] Query prarms: [[42441, 79189], [47339, 79189], [43231, 75803]]
18 [2022-01-10 17:03:32 +0800] [35399] [INFO] Query on block x: 10 y: 30
19 [2022-01-10 17:03:33 +0800] - (sanic.access)[INFO][127.0.0.1:51856]: POST http://127.0.0.1:8000/data 200 240779
```

3 客户端代码

```
1 import math
2 import sys
3 import logging
4 import requests
5 import numpy as np
6 import matplotlib.pyplot as plt
7 import cartopy.crs as ccrs
8 import cartopy.feature as cfeature
9 from PyQt5.QtCore import QObject, pyqtProperty, pyqtSlot, pyqtSignal
10 from PyQt5.QtGui import QGuiApplication
11 from PyQt5.QtQml import QQmlApplicationEngine, QQmlListProperty
12
13
14 class Coordinate(QObject):
15     def __init__(self, x: int, y: int, parent=None) -> None:
16         super().__init__(parent)
17         self.x = x
18         self.y = y
19
20     def __str__(self):
21         return f"[{self.x},{self.y}]"
22
23     @pyqtProperty(int, constant=True)
24     def x_deg(self) -> int:
25         return self.x // 3600
26
27     @pyqtProperty(int, constant=True)
28     def x_min(self) -> int:
29         return self.x % 3600 // 60
30
31     @pyqtProperty(int, constant=True)
32     def x_sec(self) -> int:
33         return self.x % 60
34
35     @pyqtProperty(int, constant=True)
36     def y_deg(self) -> int:
37         return self.y // 3600
38
39     @pyqtProperty(int, constant=True)
40     def y_min(self) -> int:
41         return self.y % 3600 // 60
42
```

```
43     @pyqtProperty(int, constant=True)
44     def y_sec(self) -> int:
45         return self.y % 60
46
47
48 # noinspection PyUnresolvedReferences
49 class ClientModel(QObject):
50     def __init__(self, parent=None) -> None:
51         super().__init__(parent)
52         self._coordinate_list = []
53
54     coordinate_list_changed = pyqtSignal()
55
56     @pyqtProperty(QQmlListProperty, notify=coordinate_list_changed)
57     def coordinate_list(self) -> QQmlListProperty:
58         return QQmlListProperty(Coordinate, self, self._coordinate_list)
59
60     @pyqtSlot(int, int)
61     def add_point(self, x: int, y: int) -> None:
62         logging.info(f"Add point x: {x}, y: {y}")
63         self._coordinate_list.append(Coordinate(x, y))
64         self.coordinate_list_changed.emit()
65
66     @pyqtSlot(int)
67     def remove_point(self, index: int) -> None:
68         logging.info(f"Remove point index: {index}")
69         self._coordinate_list.pop(index)
70         self.coordinate_list_changed.emit()
71
72     @pyqtSlot()
73     def submit(self) -> None:
74         if len(self._coordinate_list) == 0: # 消息为空
75             return
76         logging.info(f"Submit points")
77         r = requests.post("http://127.0.0.1:8000/data",
78                           json={"type": "Polygon", "coordinates": [[a.x, a.y] for a
79                               in self._coordinate_list]})
80         if r.status_code == 406:
81             root.errorOccurred("输入有误，必须为多边形")
82             return
83         elif r.status_code != 200:
84             root.errorOccurred("服务器错误")
85             return
86         self._coordinate_list.clear()
87         self.coordinate_list_changed.emit()
```

```

87     if len(r.json().get("res")) == 0:
88         return
89     data = np.array(r.json().get("res")).transpose((1, 0))
90     extent = [math.floor(np.min(data[0]) / 3600), math.ceil(np.max(data[0]) /
91 3600),
92               math.floor(np.min(data[1]) / 3600), math.ceil(np.max(data[1]) /
93 3600)] # 绘图范围
94     fig = plt.figure(figsize=(8, 6))
95     ax = fig.add_subplot(111, projection=ccrs.PlateCarree())
96     ax.set_extent(extent, crs=ccrs.PlateCarree()) # 设置范围
97     ax.add_feature(cfeature.LAND.with_scale('10m')) # 地图背景的陆地标识
98     ax.add_feature(cfeature.COASTLINE.with_scale('10m'), lw=0.25) # 地图背景的海
99     ax.add_feature(cfeature.OCEAN.with_scale('10m')) # 地图背景的海洋标识
100    ax.gridlines(draw_labels=True, dms=True, x_inline=False, y_inline=False)
101    im = ax.scatter([i / 3600 for i in data[0]], [i / 3600 for i in data[1]],
102 s=0.5, c=data[2], cmap='viridis')
103    fig.colorbar(im, ax=ax)
104    ax.title.set_text(f"Total population of the area is
105 {r.json().get('total'):.2f}")
106    plt.show()
107
108
109
110
111
112
113

```

`main.qml`:

```

1 import QtQuick 2.15
2 import QtQuick.Dialogs 1.2
3 import QtQuick.Controls 2.15
4
5 ApplicationWindow {
6     id: root
7     visible: true
8     width: 1440
9     height: 640

```

```
10     maximumHeight: height
11     maximumWidth: width
12     minimumHeight: height
13     minimumWidth: width
14     title: "全球人口分布查询"
15
16     Connections {
17         target: client_model
18     }
19
20     Image {
21         id: map
22         anchors.top: parent.top
23         anchors.left: parent.left
24         anchors.topMargin: 10
25         anchors.leftMargin: 10
26         width: parent.width - 200
27         source: "./map.jpg"
28         fillMode: Image.PreserveAspectFit
29         MouseArea {
30             id: mouse_area
31             anchors.fill: parent
32             onClicked: client_model.add_point(mouseX / width * 1296000 - 648000, -
33             (mouseY / height * 648000 - 324000))
34         }
35
36     Column {
37         anchors.top: parent.top
38         anchors.left: map.right
39         anchors.topMargin: 10
40         anchors.leftMargin: 10
41         height: map.height - 30
42         width: 170
43
44     Row {
45         id: title
46         height: 30
47         Text {
48             text: "经度"
49             width: 70
50             verticalAlignment: Text.AlignVCenter
51         }
52         Text {
53             text: "纬度"
```

```

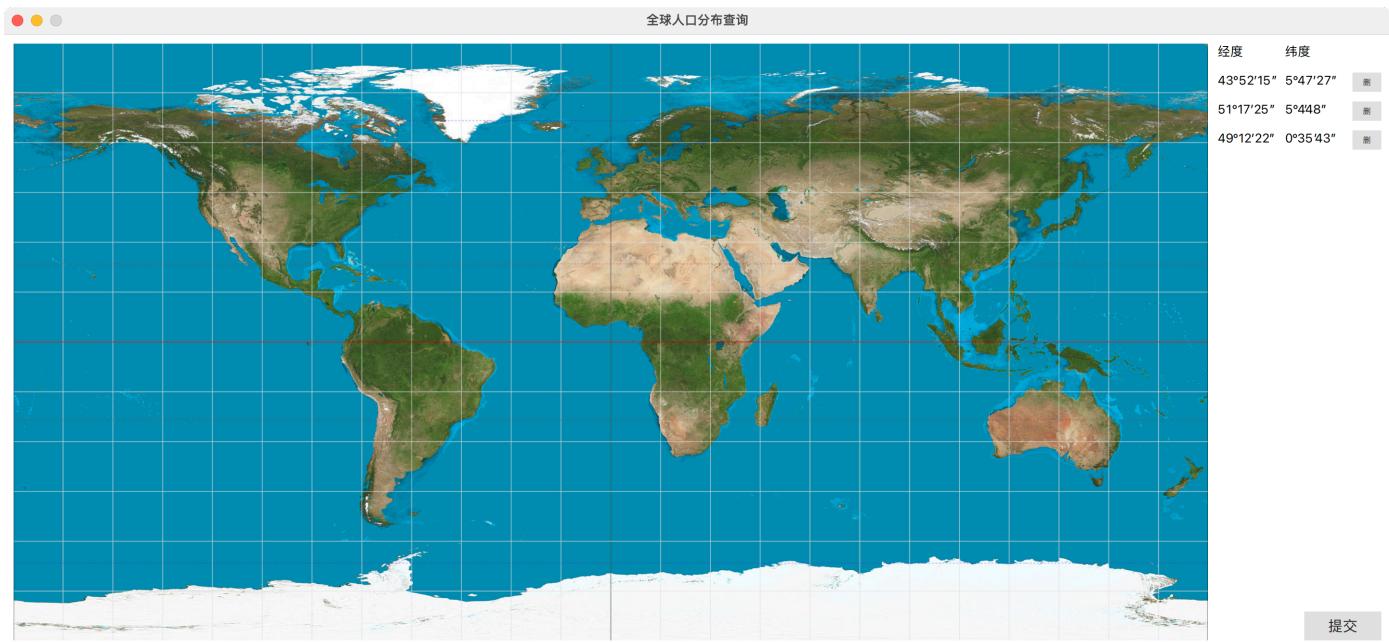
54         width: 70
55         verticalAlignment: Text.AlignVCenter
56     }
57
58     Text {
59         text: ""
60         width: 30
61         verticalAlignment: Text.AlignVCenter
62     }
63
64     ListView {
65         id: list_view
66         anchors.left: parent.left
67         height: parent.height - 30
68         width: parent.width
69         model: client_model.coordinate_list
70         clip: true
71
72         delegate: Row {
73             property int indexOfThisDelegate: index
74             height: 30
75             Text {
76                 text: modelData.x_deg + "°" + modelData.x_min + "'" +
modelData.x_sec + """
77                 width: 70
78                 verticalAlignment: Text.AlignVCenter
79             }
80             Text {
81                 text: modelData.y_deg + "°" + modelData.y_min + "'" +
modelData.y_sec + """
82                 width: 70
83                 verticalAlignment: Text.AlignVCenter
84             }
85             Button {
86                 width: 30
87                 height: 20
88                 text: "删"
89                 font.pixelSize: 8
90                 onClicked: client_model.remove_point(index)
91             }
92         }
93         onCountChanged: {
94             list_view.positionViewAtEnd()
95         }
96         ScrollBar.vertical: ScrollBar {

```

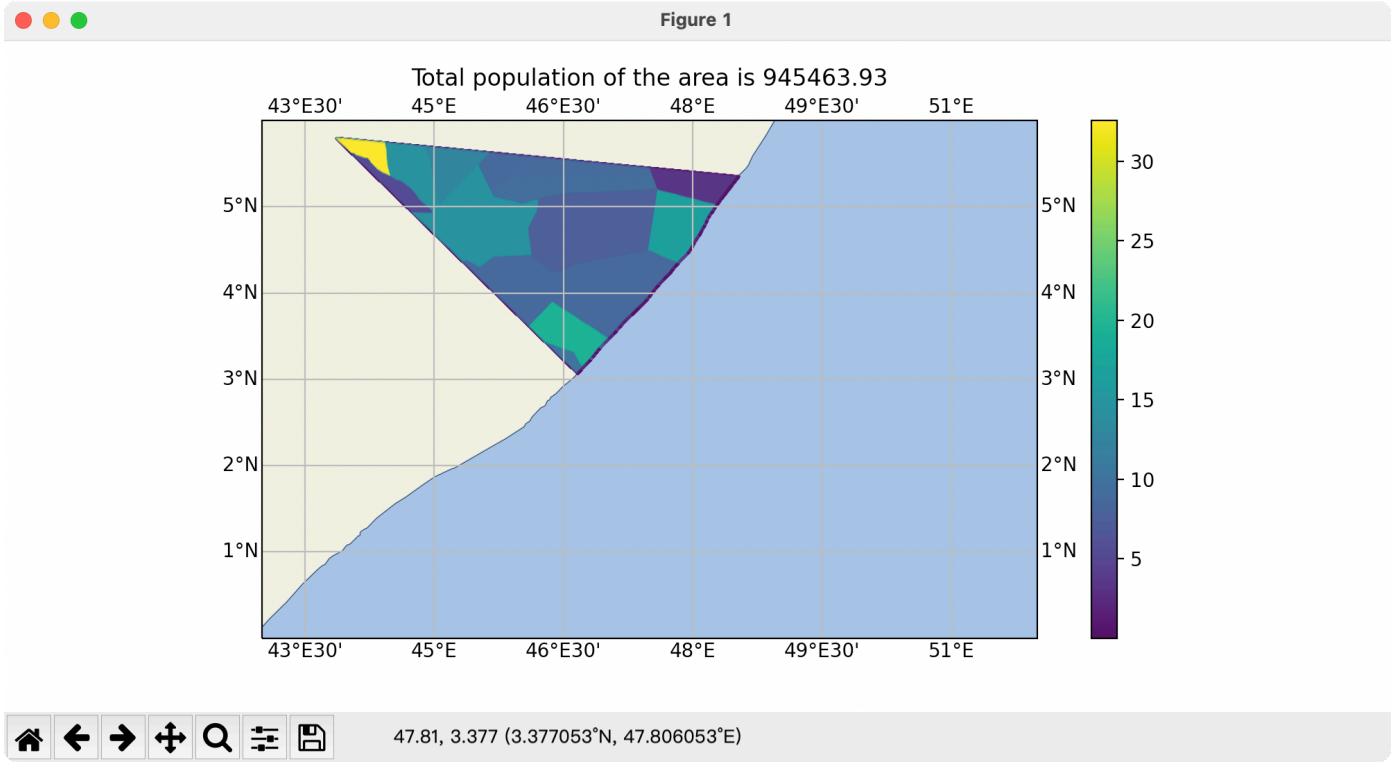
```
97         active: true
98     }
99 }
100
101    Button {
102        anchors.right: list_view.right
103        width: 80
104        height: 30
105        text: "提交"
106        font.pixelSize: 15
107        onClicked: client_model.submit()
108    }
109 }
110
111    MessageDialog {
112        id: error
113        icon: StandardIcon.Warning
114    }
115
116    function errorOccurred(str) {
117        error.text = str
118        error.open()
119    }
120 }
```

3.1 代码说明

客户端采用PyQt5 + QtQuick编写了GUI，效果如下：



针对上述查询的结果的效果图如下：



3.1.1 鼠标位置获取

页面左半部分是世界地图，鼠标在其上左键单击后，客户端会记录当前鼠标位置的偏移量，并且计算出经纬度（以角秒为单位），将该位置添加到多边形的顶点列表中。顶点的坐标由 `Coordinate` 类记录，该类提供访问坐标值各个单位（角度、角分、角秒）的属性。坐标的列表保存在 `ClientModel` 类的 `coordinate_list` 属性中。

注：图片未进行校准，获取到的经纬度可能有些许偏差。

3.1.2 顶点列表

页面的右侧依次显示多边形各顶点的坐标，点击“删”按钮可以删去该顶点，即从 `coordinate_list` 列表中删去一个元素。

3.1.3 绘图

点击“提交”后客户端向服务端发送请求，如果服务端产生报错（非200的状态码），则客户端弹出对话框提示，如果没有报错，则进行绘图。首先采用 `cartopy` 包绘制基本的地形图，限制地图范围为查询的经纬度，绘制海岸线、陆地、海洋的地形，并在其基础上绘制散点图，用 `cmap` 以不同颜色体现不同的人口数量，并且在地图旁添加 `colorbar` 图例。