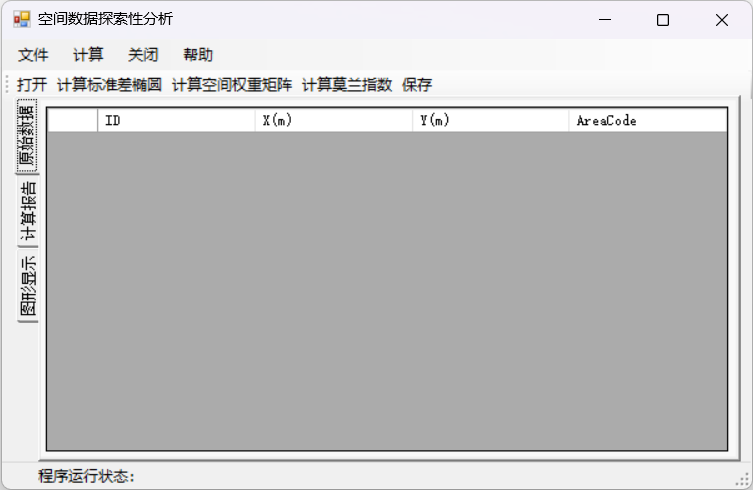
**一、程序优化性说明**

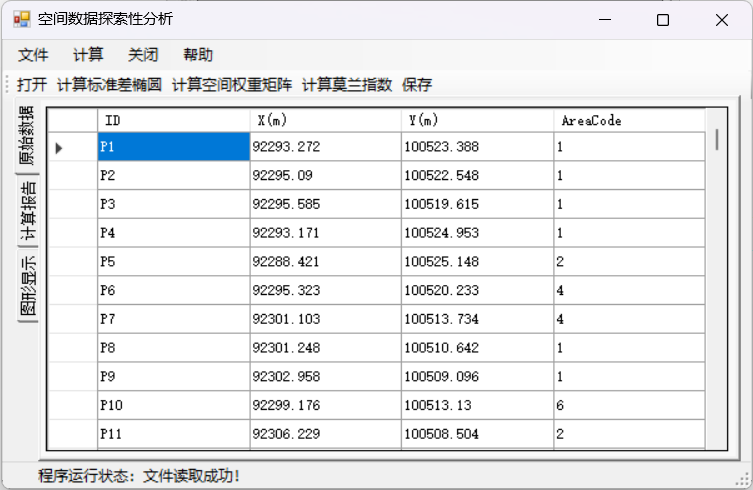
1．用户交互界面说明



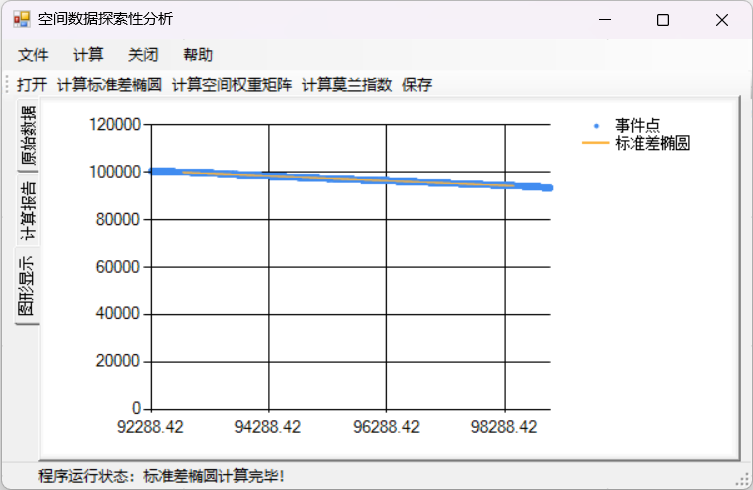
程序界面上方包括菜单栏和工具栏、通过点击按钮可以实现程序的不同功能；中间部分包含三个选项卡：原始数据（表格形式）、计算报告（文本形式）以及图形显示（图表形式）；下方为状态栏，指示程序运行状态。

2．程序运行过程说明

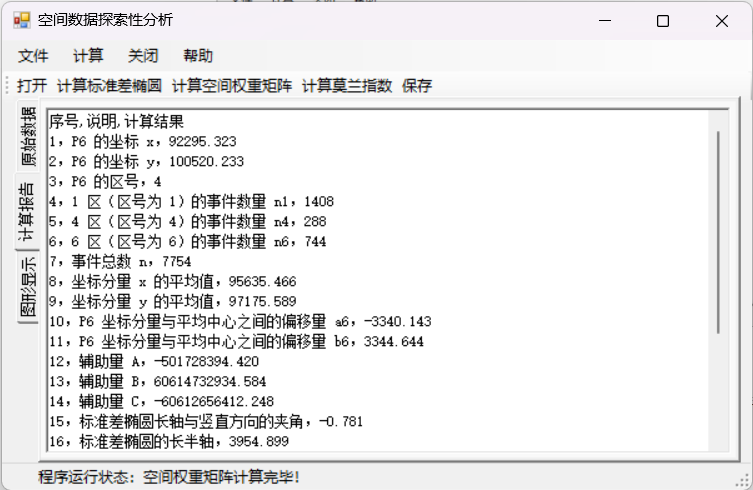
（1）点击“打开”按钮，选择正式数据.txt，读取到的数据将以表格形式展示：



（2）点击“计算标准差椭圆”按钮，计算完成后自动跳转到“图形显示”选项卡：



（3）点击“计算空间权重矩阵”按钮，计算完成后自动跳转到“计算报告”选项卡：



（4）点击“计算莫兰指数”按钮，计算完成后自动更新“计算报告”选项卡中的文本内容：

（5）点击“保存”按钮，输入保存文件名称result.txt：



（6）点击“关闭”按钮，程序结束运行。

3．程序运行结果

序号,说明,计算结果

1，P6 的坐标 x，92295.323

2，P6 的坐标 y，100520.233

3，P6 的区号，4

4，1 区（区号为 1）的事件数量 n1，1408

5，4 区（区号为 4）的事件数量 n4，288

6，6 区（区号为 6）的事件数量 n6，744

7，事件总数 n，7754

8，坐标分量 x 的平均值，95635.466

9，坐标分量 y 的平均值，97175.589

10，P6 坐标分量与平均中心之间的偏移量 a6，-3340.143

11，P6 坐标分量与平均中心之间的偏移量 b6，3344.644

12，辅助量 A，-501728394.420

13，辅助量 B，60614732934.584

14，辅助量 C，-60612656412.248

15，标准差椭圆长轴与竖直方向的夹角，-0.781

16，标准差椭圆的长半轴，3954.899

17，标准差椭圆的短半轴，94.495

18，1 区平均中心的坐标分量 X，95577.112

19，1 区平均中心的坐标分量 Y，97233.212

20，4 区平均中心的坐标分量 X，95554.001

21，4 区平均中心的坐标分量 Y，97263.180

22，1 区和 4 区的空间权重，26.424462

23，6 区和 7 区的空间权重，3.705788

24，研究区域犯罪事件的平均值，1107.714286

25，全局莫兰指数辅助量 S0，1019.281195

26，全局莫兰指数 I，-0.031250

27，1 区的局部莫兰指数 I1，-46.333278

28，3 区的局部莫兰指数 I3，16.572870

29，5 区的局部莫兰指数 I5，25.203687

30，7 区的局部莫兰指数 I7，26.353771

31，局部莫兰指数的平均数，-2.411543

32，局部莫兰指数的标准差，32.285871

33，1 区局部莫兰指数的 Z 得分 Z1，-1.360401

34，3 区局部莫兰指数的 Z 得分 Z3，0.588010

35，5 区局部莫兰指数的 Z 得分 Z5，0.855335

36，7 区局部莫兰指数的 Z 得分 Z7，0.890957

**二、程序规范性说明**

1．程序功能与结构设计说明

**程序功能：**

程序运用空间探索性分析，对美国7区犯事件展开研究。通过标准差椭圆来呈现数据的分布方向与范围，借助空间权重来界定各区间之间的相互关系；利用空间莫兰指数，判别数据在空间上的分布状况。

**结构设计说明：**

（1）Point.cs（事件点类），主要功能为设计点的特征，为后续数据存储提供便利

|  |  |  |
| --- | --- | --- |
| **成员变量** | **类型** | **解释** |
| ID | int | 事件点的编号 |
| X, Y | double | 事件点的平面位置 |
| AreaCode | int | 事件点所在区号 |

（2）Area.cs（区域类），主要功能为设计区域的特征，为后续数据存储提供便利

|  |  |  |
| --- | --- | --- |
| **成员变量** | **类型** | **解释** |
| Code | int | 区号 |
| PointsNum | int | 区域内包含事件点的数量 |
| AreaPoints | List<Point> | 区域内包含事件点 |
| AxAver, AyAver | double | 区域事件中心 |
| I | double | 局部莫兰指数 |
| Z | double | Z 得分 |

（3）DataCenter.cs（数据中心），主要功能为数据存储，包括原始数据以及计算结果

|  |  |  |
| --- | --- | --- |
| **成员变量** | **类型** | **解释** |
| Points | List<Point> | 事件点列表 |
| Areas | List<Area> | 区域列表 |
| Number | int | 题目编号 |

（4）FileCenter.cs（文件中心），主要功能为文件读取，并将文件中的数据存储进数据中心

|  |  |
| --- | --- |
| **成员函数** | **功能** |
| ReadFile | 读取文件并将数据存储进数据中心 |

（5）Algorithm.cs（算法类），主要功能为具体的算法实现以及图形绘制

|  |  |
| --- | --- |
| **成员函数** | **功能** |
| CalEllipse | 计算标准差椭圆 |
| CalWeightMatrix | 计算空间权重矩阵 |
| CalMolanIndex | 计算莫兰系数 |

2．核心算法源码

class Algorithm

{

public static double[,] WeightMatrix;

/// <summary>

/// 计算标准差椭圆

/// </summary>

/// <param name="data"></param>

/// <param name="chart1"></param>

/// <returns></returns>

public static string CalEllipse(DataCenter data,Chart chart1)

{

string text = null;

for(int i = 1; i < 8; i++)

{

Area a = new Area();

a.Code = i;

data.Areas.Add(a);

}

//数据统计

foreach(Point p in data.Points)

{

for(int i = 1; i < 8; i++)

{

if (p.AreaCode == i)

{

data.Areas[i - 1].AreaPoints.Add(p);

data.Areas[i - 1].PointsNum++;

continue;

}

}

}

int n = 0;

foreach(Area a in data.Areas)

{

n += a.PointsNum;

}

//计算平均中心

double XAver = data.Points.Sum(p => p.X / n);

double YAver = data.Points.Sum(p => p.Y / n);

//计算标准差椭圆

double sum1 = data.Points.Sum(p => Pow(p.X - XAver, 2));

double sum2 = data.Points.Sum(p => Pow(p.Y - YAver, 2));

double sum3 = data.Points.Sum(p => (p.Y - YAver) \* (p.X - XAver));

double A = sum1 - sum2;

double B = Sqrt(Pow(sum1 - sum2, 2) + 4 \* sum3 \* sum3);

double C = 2 \* sum3;

double theta = Atan((A + B) / C);

double up1 = data.Points.Sum(p => Pow((p.X - XAver) \* Cos(theta) + (p.Y - YAver) \* Sin(theta), 2));

double SDEx = Sqrt((2 \* up1) / n);

double up2 = data.Points.Sum(p => Pow((p.X - XAver) \* Sin(theta) - (p.Y - YAver) \* Cos(theta), 2));

double SDEy = Sqrt((2 \* up2) / n);

//绘图

chart1.Series.Clear();

Series s1 = new Series();//创建系列

s1.Name = "事件点";

for (int i = 0; i < data.Points.Count; i++)

{

DataPoint p = new DataPoint(data.Points[i].X, data.Points[i].Y);//创建数据点（X, Y）

s1.Points.Add(p);//数据点加入系列

}

s1.ChartType = SeriesChartType.Point;

s1.MarkerStyle = MarkerStyle.Circle;

chart1.Series.Add(s1);//系列加入图表

Series s2 = new Series();//创建系列

s2.Name = "标准差椭圆";

for (double theta0 = 0; theta0 <= 2 \* PI; theta0 += 0.1)

{

double x = XAver + SDEx \* Cos(theta0) \* Cos(theta) - SDEy \* Sin(theta0) \* Sin(theta);

double y = YAver + SDEx \* Cos(theta0) \* Sin(theta) - SDEy \* Sin(theta0) \* Cos(theta);

DataPoint dp = new DataPoint(x, y);

s2.Points.Add(dp);

}

s2.ChartType = SeriesChartType.Line;

//2.MarkerStyle = MarkerStyle.Circle;

chart1.Series.Add(s2);//系列加入图表

text += $"{data.Number}，1 区（区号为 1）的事件数量 n1，{data.Areas[0].PointsNum}\r\n"; data.Number++;

text += $"{data.Number}，4 区（区号为 4）的事件数量 n4，{data.Areas[3].PointsNum}\r\n"; data.Number++;

text += $"{data.Number}，6 区（区号为 6）的事件数量 n6，{data.Areas[5].PointsNum}\r\n"; data.Number++;

text += $"{data.Number}，事件总数 n，{n}\r\n"; data.Number++;

text += $"{data.Number}，坐标分量 x 的平均值，{XAver:f3}\r\n"; data.Number++;

text += $"{data.Number}，坐标分量 y 的平均值，{YAver:f3}\r\n"; data.Number++;

text += $"{data.Number}，P6 坐标分量与平均中心之间的偏移量 a6，{data.Points[5].X - XAver:f3}\r\n"; data.Number++;

text += $"{data.Number}，P6 坐标分量与平均中心之间的偏移量 b6，{data.Points[5].Y - YAver:f3}\r\n"; data.Number++;

text += $"{data.Number}，辅助量 A，{A:f3}\r\n"; data.Number++;

text += $"{data.Number}，辅助量 B，{B:f3}\r\n"; data.Number++;

text += $"{data.Number}，辅助量 C，{C:f3}\r\n"; data.Number++;

text += $"{data.Number}，标准差椭圆长轴与竖直方向的夹角，{theta:f3}\r\n"; data.Number++;

text += $"{data.Number}，标准差椭圆的长半轴，{SDEx:f3}\r\n"; data.Number++;

text += $"{data.Number}，标准差椭圆的短半轴，{SDEy:f3}\r\n"; data.Number++;

return text;

}

/// <summary>

/// 计算空间权重矩阵

/// </summary>

/// <param name="data"></param>

/// <returns></returns>

public static string CalWeightMatrix(DataCenter data)

{

string text = null;

//计算各区平均中心

foreach(Area a in data.Areas)

{

a.AxAver = a.AreaPoints.Sum(p => p.X / a.PointsNum);

a.AyAver = a.AreaPoints.Sum(p => p.Y / a.PointsNum);

}

//计算空间权重矩阵

WeightMatrix = new double[7, 7];

for(int i = 0; i < 7; i++)

{

for(int j = 0; j < 7; j++)

{

if (i == j) WeightMatrix[i, j] = 0.0;

else

{

WeightMatrix[i, j] = 1000.0 / Sqrt(Pow(data.Areas[i].AxAver - data.Areas[j].AxAver, 2)

+ Pow(data.Areas[i].AyAver - data.Areas[j].AyAver, 2));

}

}

}

text += $"{data.Number}，1 区平均中心的坐标分量 X，{data.Areas[0].AxAver:f3}\r\n"; data.Number++;

text += $"{data.Number}，1 区平均中心的坐标分量 Y，{data.Areas[0].AyAver:f3}\r\n"; data.Number++;

text += $"{data.Number}，4 区平均中心的坐标分量 X，{data.Areas[3].AxAver:f3}\r\n"; data.Number++;

text += $"{data.Number}，4 区平均中心的坐标分量 Y，{data.Areas[3].AyAver:f3}\r\n"; data.Number++;

text += $"{data.Number}，1 区和 4 区的空间权重，{WeightMatrix[0, 3]:f6}\r\n"; data.Number++;

text += $"{data.Number}，6 区和 7 区的空间权重，{WeightMatrix[5, 6]:f6}\r\n"; data.Number++;

return text;

}

/// <summary>

/// 计算莫兰系数

/// </summary>

/// <param name="data"></param>

/// <returns></returns>

public static string CalMolanIndex(DataCenter data)

{

string text = null;

//数据整理

int n = 0;

foreach (Area a in data.Areas)

{

n += a.PointsNum;

}

double XAve = n / 7.0;

//全局莫兰指数

double S0 = 0.0;

for (int i = 0; i < 7; i++)

{

for (int j = 0; j < 7; j++)

{

S0 += WeightMatrix[i, j];

}

}

double down = data.Areas.Sum(p => Pow(p.PointsNum - XAve, 2));

double up = 0.0;

for (int i = 0; i < 7; i++)

{

for (int j = 0; j < 7; j++)

{

up += WeightMatrix[i, j] \* (data.Areas[i].PointsNum - XAve) \* (data.Areas[j].PointsNum - XAve);

}

}

double I = (7.0 \* up) / (S0 \* down);

//局部莫兰指数

for(int i = 0; i < 7; i++)

{

double Si\_2 = 0.0;

for (int j = 0; j < 7; j++)

{

if (i == j) continue;

else

{

Si\_2 += Pow(data.Areas[j].PointsNum - XAve, 2);

}

}

Si\_2 /= 6.0;

double sum = 0.0;

for (int j = 0; j < 7; j++)

{

if (i == j) continue;

else

{

sum += WeightMatrix[i, j] \* (data.Areas[j].PointsNum - XAve);

}

}

data.Areas[i].I = (data.Areas[i].PointsNum - XAve) \* sum / Si\_2;

}

//计算局部莫兰指数的 Z 得分

double u = data.Areas.Sum(p => p.I / 7.0);

double sum0 = data.Areas.Sum(p => Pow(p.I - u, 2));

double sigma = Sqrt(sum0 / 6.0);

foreach(Area a in data.Areas)

{

a.Z = (a.I - u) / sigma;

}

text += $"{data.Number}，研究区域犯罪事件的平均值，{XAve:f6}\r\n"; data.Number++;

text += $"{data.Number}，全局莫兰指数辅助量 S0，{S0:f6}\r\n"; data.Number++;

text += $"{data.Number}，全局莫兰指数 I，{I:f6}\r\n"; data.Number++;

text += $"{data.Number}，1 区的局部莫兰指数 I1，{data.Areas[0].I:f6}\r\n"; data.Number++;

text += $"{data.Number}，3 区的局部莫兰指数 I3，{data.Areas[2].I:f6}\r\n"; data.Number++;

text += $"{data.Number}，5 区的局部莫兰指数 I5，{data.Areas[4].I:f6}\r\n"; data.Number++;

text += $"{data.Number}，7 区的局部莫兰指数 I7，{data.Areas[6].I:f6}\r\n"; data.Number++;

text += $"{data.Number}，局部莫兰指数的平均数，{u:f6}\r\n"; data.Number++;

text += $"{data.Number}，局部莫兰指数的标准差，{sigma:f6}\r\n"; data.Number++;

text += $"{data.Number}，1 区局部莫兰指数的 Z 得分 Z1，{data.Areas[0].Z:f6}\r\n"; data.Number++;

text += $"{data.Number}，3 区局部莫兰指数的 Z 得分 Z3，{data.Areas[2].Z:f6}\r\n"; data.Number++;

text += $"{data.Number}，5 区局部莫兰指数的 Z 得分 Z5，{data.Areas[4].Z:f6}\r\n"; data.Number++;

text += $"{data.Number}，7 区局部莫兰指数的 Z 得分 Z7，{data.Areas[6].Z:f6}\r\n"; data.Number++;

return text;

}

}