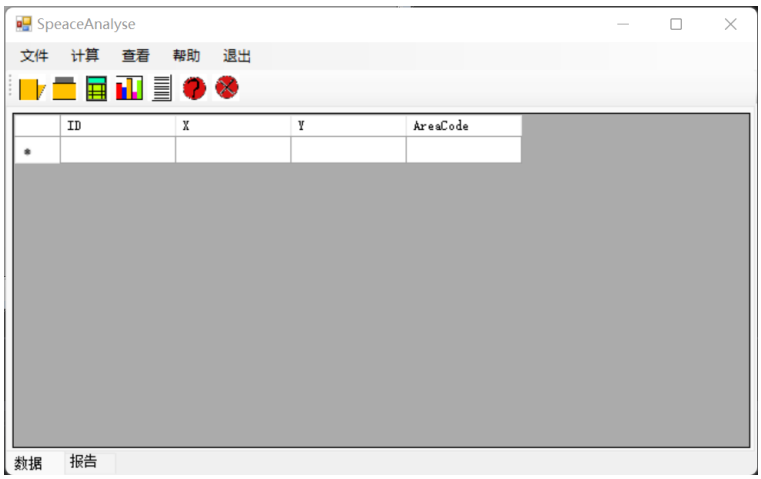
报告文档

一、程序优化性说明

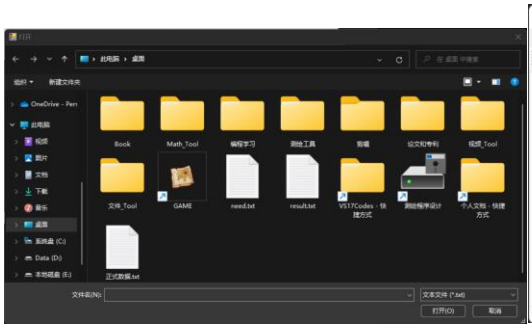
1．用户交互界面说明

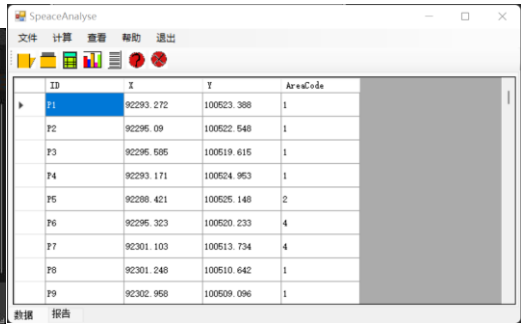


包括打开数据、保存报告、一键处理、查看数据、查看报告、问题解决、退出程序

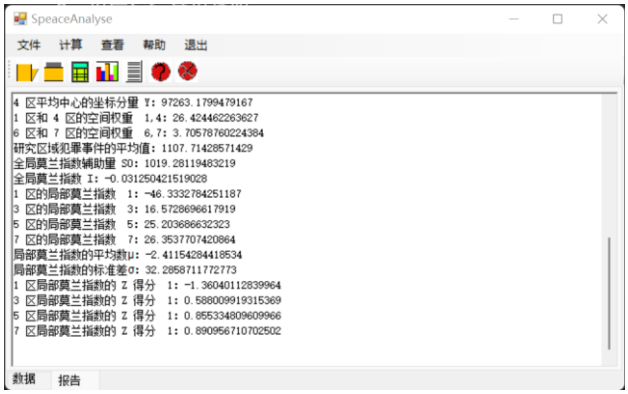
2．程序运行过程说明

（1）打开数据，查看数据

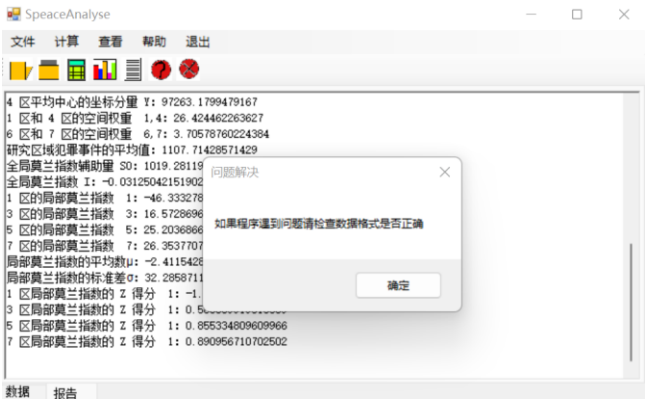




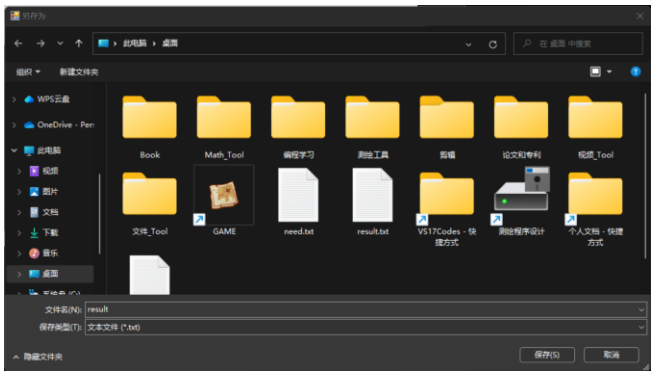
（2）一键计算，查看计算报告



（3）如果有问题点击帮助



（4）导出计算报告



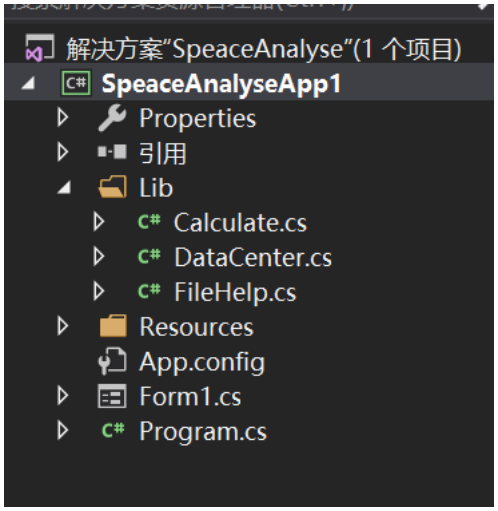
3．程序运行结果


ed.t
result.txt
文件
编辑
查看
序号，说明，计算结果
P6 的坐标 x：92295.323
P6 的坐标 y：100520.233
P6 的区号：4
1 区（区号为 1）的事件数量 n1：1408
4 区（区号为 4）的事件数量 n4：288
6 区（区号为 6）的事件数量 n6：744
事件总数 n：7754
坐标分量 x 的平均值：95635.4660526181
坐标分量 y 的平均值：97175.5892972663
P6 坐标分量与平均中心之间的偏移量 a6：-3340.14305261809
P6 坐标分量与平均中心之间的偏移量 b6：3344.64370273371
辅助量 A：7075.74624464818
辅助量 B：60612656412.2482
辅助量 C：-60612656412.2478
标准差椭圆长轴与竖直方向的夹：-0.785398221766003
标准差椭圆的长半轴SDEx：7075.74625809524
标准差椭圆的长半轴SDEy：230.385936883299
1 区平均中心的坐标分量 X：95577.1120823862
1 区平均中心的坐标分量 Y：97233.212497159
4 区平均中心的坐标分量 X：95554.0014965278
4 区平均中心的坐标分量 Y：97263.1799479167
1 区和 4 区的空间权重??1,4：26.424462263627
6 区和 7 区的空间权重??6,7：3.70578760224384
研究区域犯罪事件的平均值：1107.71428571429
全局莫兰指数辅助量 S0：1019.28119483219
全局莫兰指数 I：-0.031250421519028
1 区的局部莫兰指数??1：-46.3332784251187
3 区的局部莫兰指数??3：16.5728696617919
5 区的局部莫兰指数??5：25.203686632323
7 区的局部莫兰指数??7：26.3537707420864
局部莫兰指数的平均数μ：-2.41154284418534
局部莫兰指数的标准差σ：32.2858711772773
1 区局部莫兰指数的 Z 得分??1：-1.36040112839964
3 区局部莫兰指数的 Z 得分??1：0.588009919315369
5 区局部莫兰指数的 Z 得分??1：0.855334809609966
7 区局部莫兰指数的 Z 得分??1：0.890956710702502
行 1，列 1
1,061 个字符
100%
Windows (CRLF)
ANSI

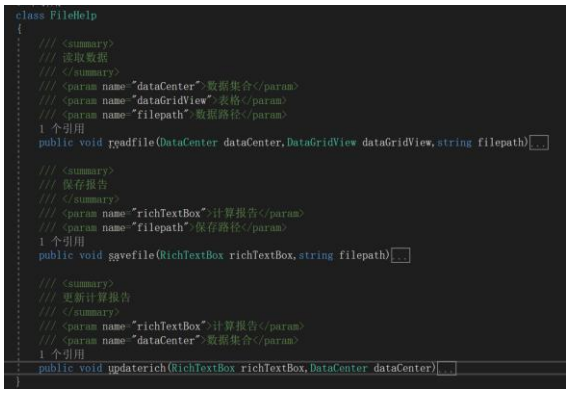
二、程序规范性说明

1．程序功能与结构设计说明

核心类放在 FIleHelp,DataCenter 和 Calculate 中

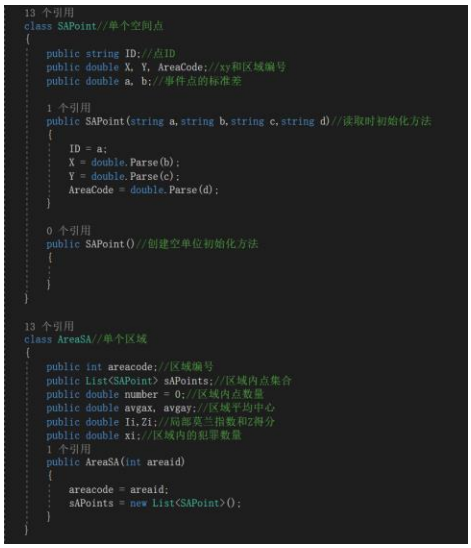


（1） FileHelp 包括读取数据、保存数据、更新控件

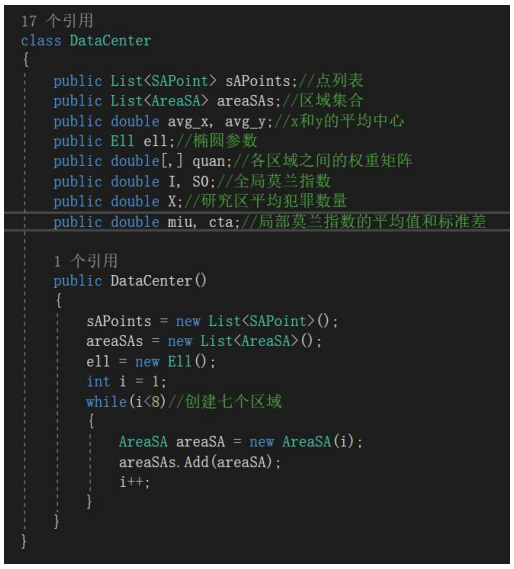


```
class FileHelp
{
    /// <summary>
    /// 读取数据
    /// </summary>
    /// <param name="dataCenter">数据集合</param>
    /// <param name="dataGridView">表格</param>
    /// <param name="filepath">数据路径</param>
    1 个引用
    public void readfile(DataCenter dataCenter,DataGridView dataGridView,string filepath)...

    /// <summary>
    /// 保存报告
    /// </summary>
    /// <param name="richTextBox">计算报告</param>
    /// <param name="filepath">保存路径</param>
    1 个引用
    public void savefile(RichTextBox richTextBox,string filepath)...

    /// <summary>
    /// 更新计算报告
    /// </summary>
    /// <param name="richTextBox">计算报告</param>
    /// <param name="dataCenter">数据集合</param>
    1 个引用
    public void updaterich(RichTextBox richTextBox,DataCenter dataCenter)...
}
```

（2） DataCenter 包含不同数据类型，点数据、椭圆参数等等

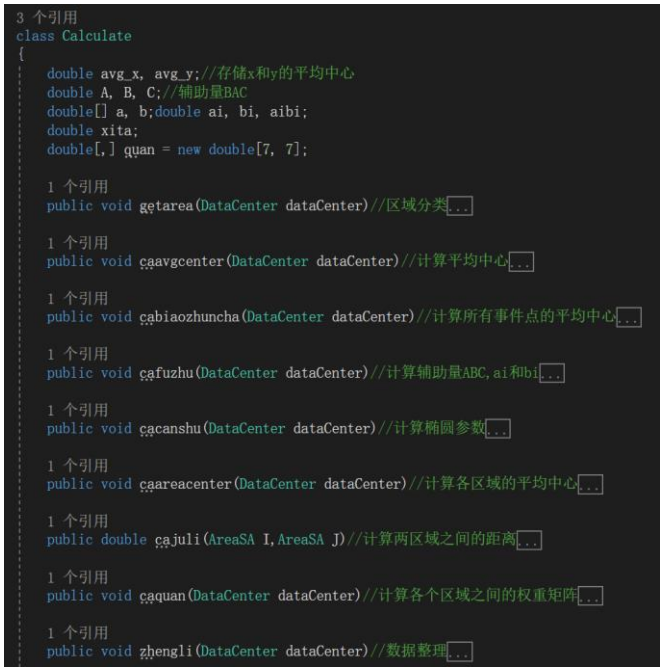


```
13 个引用
class SAPoint//单个空间点
{
    public string ID;//点ID
    public double X, Y, AreaCode;//xy和区域编号
    public double a, b;//事件点的标准差

    1 个引用
    public SAPoint(string a,string b,string c,string d)//读取时初始化方法
    {
        ID = a;
        X = double.Parse(b);
        Y = double.Parse(c);
        AreaCode = double.Parse(d);
    }

    0 个引用
    public SAPoint()//创建空单位初始化方法
    {

    }
}

13 个引用
class AreaSA//单个区域
{
    public int areacode;//区域编号
    public List<SAPoint> sAPoints;//区域内点集合
    public double number = 0;//区域内点数量
    public double avgax, avgay;//区域平均中心
    public double Ii,Zi;//局部莫兰指数和Z得分
    public double xi;//区域内的犯罪数量
    1 个引用
    public AreaSA(int areaid)
    {
        areacode = areaid;
        sAPoints = new List<SAPoint>();
    }
}
```

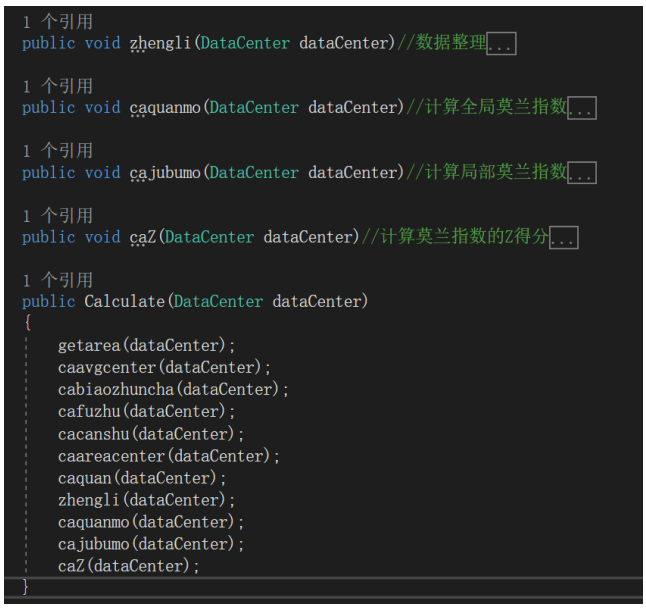


```
17 个引用
class DataCenter
{
    public List<SAPoint> sAPoints;//点列表
    public List<AreaSA> areaSAs;//区域集合
    public double avg_x, avg_y;//x和y的平均中心
    public Ell ell;//椭圆参数
    public double[,] quan;//各区域之间的权重矩阵
    public double I, S0;//全局莫兰指数
    public double X;//研究区平均犯罪数量
    public double miu, cta;//局部莫兰指数的平均值和标准差

    1 个引用
    public DataCenter()
    {
        sAPoints = new List<SAPoint>();
        areaSAs = new List<AreaSA>();
        ell = new Ell();
        int i = 1;
        while(i<8)//创建七个区域
        {
            AreaSA areaSA = new AreaSA(i);
            areaSAs.Add(areaSA);
            i++;
        }
    }
}
```

（3） Calculate 包含了计算函数



```
3 个引用
class Calculate
{
    double avg_x, avg_y;//存储x和y的平均中心
    double A, B, C;//辅助量BAC
    double[] a, b;double ai, bi, aibi;
    double xita;
    double[,] quan = new double[7, 7];

    1 个引用
    public void getarea(DataCenter dataCenter)//区域分类...

    1 个引用
    public void caavgcenter(DataCenter dataCenter)//计算平均中心...

    1 个引用
    public void cabiaozhuncha(DataCenter dataCenter)//计算所有事件点的平均中心...

    1 个引用
    public void cafuzhu(DataCenter dataCenter)//计算辅助量ABC,ai和bi...

    1 个引用
    public void cacanshu(DataCenter dataCenter)//计算椭圆参数...

    1 个引用
    public void caareacenter(DataCenter dataCenter)//计算各区域的平均中心...

    1 个引用
    public double cajuli(AreaSA I,AreaSA J)//计算两区域之间的距离...

    1 个引用
    public void caquan(DataCenter dataCenter)//计算各个区域之间的权重矩阵...

    1 个引用
    public void zhengli(DataCenter dataCenter)//数据整理...
```



```
1 个引用
public void zhengli(DataCenter dataCenter)//数据整理...

1 个引用
public void caquanmo(DataCenter dataCenter)//计算全局莫兰指数...

1 个引用
public void cajubumo(DataCenter dataCenter)//计算局部莫兰指数...

1 个引用
public void caZ(DataCenter dataCenter)//计算莫兰指数的Z得分...

1 个引用
public Calculate(DataCenter dataCenter)
{
    getarea(dataCenter);
    caavgcenter(dataCenter);
    cabiaozhuncha(dataCenter);
    cafuzhu(dataCenter);
    cacanshu(dataCenter);
    caareacenter(dataCenter);
    caquan(dataCenter);
    zhengli(dataCenter);
    caquanmo(dataCenter);
    cajubumo(dataCenter);
    caZ(dataCenter);
}
```

2．核心算法源码

下面是计算类 Calculate 中的主要代码:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Math;

namespace SpeaceAnalyseApp1.Lib
```

```
{
    class Calculate
    {
        double avg_x, avg_y;//存储 x 和 y 的平均中心
        double A, B, C;//辅助量 BAC
        double[] a, b;double ai, bi, aibi;
        double xita;
        double[,] quan = new double[7, 7];

        public void getarea(DataCenter dataCenter)//区域分类
        {
            foreach(SAPoint sAPoint in dataCenter.sAPoints)
            {
                if (sAPoint.AreaCode == 1) dataCenter.areaSAs[0].sAPoints.Add(sAPoint);
                else if (sAPoint.AreaCode == 2)
dataCenter.areaSAs[1].sAPoints.Add(sAPoint);
                else if (sAPoint.AreaCode == 3)
dataCenter.areaSAs[2].sAPoints.Add(sAPoint);
                else if (sAPoint.AreaCode == 4)
dataCenter.areaSAs[3].sAPoints.Add(sAPoint);
                else if (sAPoint.AreaCode == 5)
dataCenter.areaSAs[4].sAPoints.Add(sAPoint);
                else if (sAPoint.AreaCode == 6)
dataCenter.areaSAs[5].sAPoints.Add(sAPoint);
                else if (sAPoint.AreaCode == 7)
dataCenter.areaSAs[6].sAPoints.Add(sAPoint);
                else MessageBox.Show("不存在该区域");
            }
            foreach(AreaSA areaSA in dataCenter.areaSAs)
            {
                areaSA.number = areaSA.sAPoints.Count;
            }
        }

        public void caavgcenter(DataCenter dataCenter)//计算平均中心
        {
            dataCenter.avg_x = dataCenter.sAPoints.Average(p => p.X);
            dataCenter.avg_y = dataCenter.sAPoints.Average(p => p.Y);
            avg_x = dataCenter.avg_x;
            avg_y = dataCenter.avg_y;
        }

        public void cabiaozhuncha(DataCenter dataCenter)//计算所有事件点的平均中心
        {
```

```
        foreach (SAPoint sAPoint in dataCenter.sAPoints)
        {
            sAPoint.a = sAPoint.X - avg_x;
            sAPoint.b = sAPoint.Y - avg_y;
        }
    }

    public void cafuzhu(DataCenter dataCenter)//计算辅助量 ABC,ai 和 bi
    {
        a = new double[dataCenter.sAPoints.Count];
        b = new double[dataCenter.sAPoints.Count];
        int i = 0;
        foreach (SAPoint sAPoint in dataCenter.sAPoints)
        {
            a[i] = sAPoint.a;
            b[i] = sAPoint.b;
            aibi += a[i] * b[i];
        }
        ai = a.Sum();bi = b.Sum();

        //A
        A = ai - bi; dataCenter.ell.A = A;
        //B
        double need1, need2;
        need1 = Pow((ai - bi), 2);
        need2 = 4 * Pow(aibi, 2);
        B = Sqrt(need1 + need2);dataCenter.ell.B = B;
        //C
        C = 2 * aibi;dataCenter.ell.C = C;
    }

    public void cacanshu(DataCenter dataCenter)//计算椭圆参数
    {
        dataCenter.ell.xita = Atan((A + B) / C);
        xita = dataCenter.ell.xita;
        double up1 = 0;double up2 = 0;
        int i = 0;
        while(i<a.Count())
        {
            up1 += Pow((ai * Cos(xita) + bi * Sin(xita)), 2);
            up2 += Pow((ai * Sin(xita) - bi * Cos(xita)), 2);
            i++;
        }
        dataCenter.ell.SDEx = Sqrt(2) * Sqrt(up1 / a.Count());
```

```
            dataCenter.ell.SDEy = Sqrt(2) * Sqrt(up2 / a.Count());
        }

        public void caareacenter(DataCenter dataCenter)//计算各区域的平均中心
        {
            foreach(AreaSA areaSA in dataCenter.areaSAs)
            {
                double sumx = 0;double sumy = 0;
                foreach(SAPoint sAPoint in areaSA.sAPoints)
                {
                    sumx += sAPoint.X;
                    sumy += sAPoint.Y;
                }
                areaSA.avgax = sumx / areaSA.number;
                areaSA.avgay = sumy / areaSA.number;
            }
        }

        public double cajuli(AreaSA I,AreaSA J)//计算两区域之间的距离
        {
            double d = 0;
            double need1 = Pow((I.avgax - J.avgax), 2);
            double need2 = Pow((I.avgay - J.avgay), 2);
            d = Sqrt(need1 + need2);
            return d;
        }

        public void caquan(DataCenter dataCenter)//计算各个区域之间的权重矩阵
        {
            dataCenter.quan = new double[7, 7];
            for(int i=0; i<7; i++)
            {
                for(int j=0;j<7;j++)
                {
                    if (i == j) quan[i, j] = 0.0;
                    else quan[i, j] = 1000 / cajuli(dataCenter.areaSAs[i],
dataCenter.areaSAs[j]);
                }
            }
            dataCenter.quan = quan;
        }

        public void zhengli(DataCenter dataCenter)//数据整理
        {
```

```
        double n = 0;
        foreach(AreaSA areaSA in dataCenter.areaSAs)
        {
            areaSA.xi = areaSA.sAPoints.Count;
            n += areaSA.xi;
        }
        dataCenter.X = n / 7;
    }

    public void caquanmo(DataCenter dataCenter)//计算全局莫兰指数
    {
        double N = 7;
        for(int i=0;i<7; i++)
        {
            for(int j=0;j<7;j++)
            {
                dataCenter.S0 += quan[i, j];
            }
        }
        double need1 = 0;double need2 = 0;
        for(int i=0;i<7;i++)
        {
            need2 += Pow((dataCenter.areaSAs[i].xi - dataCenter.X),2);
        }
        for (int i = 0; i < 7; i++)
        {
            for (int j = 0; j < 7; j++)
            {
                need1 += (quan[i, j] *
                    (dataCenter.areaSAs[i].xi - dataCenter.X) *
                    (dataCenter.areaSAs[j].xi - dataCenter.X));
            }
        }
        dataCenter.I = (N / dataCenter.S0) * (need1 / need2);
    }

    public void cajubumo(DataCenter dataCenter)//计算局部莫兰指数
    {
        for (int i = 0; i < 7; i++)//7 个区域
        {
            double sum1 = 0; double sum2 = 0;
            for (int j = 0; j < 7; j++)//求两种分子
            {
                if(i!=j)
```

```
                {
                    sum1 += quan[i, j] * (dataCenter.areaSAs[j].xi - dataCenter.X);
                    sum2 += Pow((dataCenter.areaSAs[j].xi - dataCenter.X),2);
                }
            }
            double Si2 = sum2 / 6;
            dataCenter.areaSAs[i].li = (dataCenter.areaSAs[i].xi - dataCenter.X) / Si2 *
sum1;
        }
    }

    public void caZ(DataCenter dataCenter)//计算莫兰指数的 Z 得分
    {
        double i = 0;
        foreach(AreaSA areaSA in dataCenter.areaSAs)
        {
            i += areaSA.li;
        }
        dataCenter.miu = i / 7;

        double need1 = 0;
        foreach (AreaSA areaSA in dataCenter.areaSAs)
        {
            need1 += Pow((areaSA.li-dataCenter.miu),2);
        }
        dataCenter.cta = Sqrt(need1 / 6);

        foreach (AreaSA areaSA in dataCenter.areaSAs)
        {
            areaSA.Zi = (areaSA.li - dataCenter.miu) / dataCenter.cta;
        }
    }

    public Calculate(DataCenter dataCenter)
    {
        getarea(dataCenter);
        caavgcenter(dataCenter);
        cabiaozhuncha(dataCenter);
        cafuzhu(dataCenter);
        cacanshu(dataCenter);
        caareacenter(dataCenter);
        caquan(dataCenter);
        zhengli(dataCenter);
        caquanmo(dataCenter);
```

```
                cajubumo(dataCenter);
                caZ(dataCenter);
            }
        }
    }
```