所属类别		参赛编号
本科	**2024 年第十六届“华中杯”大学生数学建模**	202406600250

使用行车轨迹估计交通信号灯周期问题

摘　要

红绿灯位置是道路上行人和车辆的交会点，极大影响着道路结构和交通运行，在城市路网中起着重要作用。用大量司机的行车轨迹数据来精准有效地估计交通信号灯的红绿周期，对便利司机的行车规划，改善交通拥挤等均有着重要意义。

针对问题一，为分析得到信号灯的红绿周期，我们要先对题目给出的数据进行**数据清洗**，来得到有效数据。我们分别对五个路口做出各个时刻车辆所处位置进行 **DBSCAN 聚类分析**，通过截取红绿灯位置附近的数据作为有效数据，减少数据处理量，同时减弱了异常值对问题的影响。接着针对每一时刻是红灯还是绿灯做出判断。对于红灯时间（t_r）的估计，我们采用最大值的估计模型；对于绿灯时间（t_g）的估计，我们采用最小值进行估计。之后，将t_r与t_g求和后得出总时长（T)。由于t_r会小于等于实际时长，而t_g会大于等于实际时长，且二者在一个周期中和为定值。因此得到的总时长十分准确。

针对问题二，为解决实际问题，我们需要考虑车辆比例不符合总车量、车流量过小导致数据集较为分散、轨迹数据的误差产生异常值等诸多现实影响因素。由于问题一的求解已经考虑到了异常值的影响，而车辆比例、车流量所带来的影响难以在单个样本数据集上解决，故沿用问题一的求解方法，得到红绿灯的周期。

针对问题三，为判断周期的变化以及变化后的周期，我们引入了**滑动窗口**算法来检验周期的变化，检测出周期变化后通过周期的变化点来截断时间轴，对周期变化前后的红绿灯周期进行求解。

针对问题四，为判断所有方向的路口信号灯周期，我们首先需要考虑信号灯的位置与车辆轨迹点的大致分布情况，据此来判断路口的形状。接着，通过将车辆的起点与终点所在的区域划分来判断车辆的大致运动，从而判断存在几种方向。最后，通过研究某一方向所有车辆的轨迹点来得到该方向的红绿灯周期，进而可以得到该路口的红绿灯周期

关键词： DBSCAN 聚类；数据清洗；滑动窗口法

目　录

一、问题提出 1
（一） 问题背景 1
（二） 问题重述 1
二、问题分析 1
（一） 问题一的分析 1
（二） 问题二的分析 2
三、模型假设 3
（一） 假设 3
四、符号说明 3
五、问题一的求解 4
（一） 数据的分析与处理 4
1. 提取给出的各类信息 4
2.热力图绘制散点的数据特征 4
3.使用 DBSCSAN 聚类方法来确定红绿灯的大致位置，并进行数据筛选 5
（二） 建立数学模型并求解 6
1. 建立数学模型 6
2. 红绿灯周期的估计 6
3. 阈值 ε 的调节 6
（三） 红、绿灯时长的估计 7
1. 估计原理 7
2. 输出 A1-A5 的“0”，“1”表 7
3. 对“0”、“1”表的处理 8
4. 红、绿灯时长的平均估计 9
（四）识别结果 9
六、问题二的求解 10
（一） 统计学分析 10
1. 车辆比例带来的误差分析 10
2. 车流量带来的误差分析 10
3. 定位偏误带来的误差分析 10
（二） 对模型的影响及求解 10
七、问题三的求解 11
（一） 数据的分析处理 11
（二） 模型建立 11
（三） 模型结果 12
八、问题四的求解 14
（一）数据的可视化处理与分析 14
（二） 车辆行驶方向的确定 14
（三） 模型结果 15
九、模型评价 16
（一） 模型的优点 16
（二） 模型的缺点 16
十、参考文献 16

一、问题提出

（一） 问题背景

红绿灯位置是道路上行人和车辆的交会点，在城市路网中起着重要作用。在进行导航时，提前得知红绿灯的信号能够极大地方便司机驾驶。但是由于许多信号灯未接入网络，无法直接从交通管理部门获取所有信号灯的数据，也不可能在所有路口安排人工读取信号灯周期信息，这样十分费时费力，效率很低。

如今数据科学飞速发展，数据与生活已经成为密不可分的一部分。因此，我们是否能通过大量车辆行驶的时间与轨迹数据，来推算估计得到信号灯的红绿周期成为了一个重要问题。

（二） 问题重述

某电子地图服务商希望获取城市路网中所有交通信号灯的红绿周期，以便为司机提供更好的导航服务。由于许多信号灯未接入网络，无法直接从交 通管理部门获取所有信号灯的数据，也不可能在所有路口安排人工读取信号灯周期信息。所以，该公司计划使用大量客户的行车轨迹数据估计交通信号灯的周期。

本文待解决的问题如下：

问题一：若信号灯周期固定不变，且已知所有车辆的行车轨迹，建立模型，利用车辆行车轨迹数据估计信号灯的红绿周期。

问题二：实际上，只有部分用户使用该公司的产品，即只能获取部分样本车辆的行车轨迹。同时，受各种因素的影响，轨迹数据存在定位误差，误差大小未知。讨论样本车辆比例、车流量、定位误差等因素对上述模型估计精度的影响。

问题三：如果信号灯周期有可能发生变化，能否尽快检测出这种变化，以及变化后的新周期？尝试求出周期切换的时刻，以及新旧周期参数，并指明识别出周期变化所需的时间和条件。

问题四：附件 4 是某路口连续 2 小时内所有方向样本车辆的轨迹数据，请尝试识别出该路口信号灯的周期。

二、问题分析

（一） 问题一的分析

问题一要求我们根据附件 1 中给出的，5 个不相关路口各自一个方向连续 1

小时内车辆的轨迹数据，来尝试求出这些路口相应方向的信号灯周期，并按格式要求填入表 1。首先，在所有数据中，只要有车辆的速度小于给出的阈值，我们认为此时刻信号灯为红灯，而与每辆车的具体位置无关。因此，我们的想法是先从每一个时刻上判断此时信号灯的状态，再用得到的数据来估计红灯和绿灯的时间。同时，我们还注意到给出的数据中存在一些无效数据，所以要对数据进行预处理。因为车辆在路口信号灯处停下时视为红灯时间，而不处于路口的车辆不受信号灯的影响。所以我们只需要使用 DBSCAN 聚类分析法，来截取路口附近的车辆位置数据作为用于研究的有效数据。在得出对红灯和绿灯时间的估计值后，对得到的数据进行平均处理，以削弱误差带来的影响。

（二） 问题二的分析

问题二要求我们根据实际情况，加入了车辆比例，车流量，定位偏误等因素来讨论模型对信号灯周期的影响。首先，我们对影响因素进行分析。车辆比例有误可以认为是轨道数据不能很好地代表总体的数据特征，导致红绿灯周期数据有偏；车流量偏低，可以认为是数据集的样本量过少，在计算红绿灯周期时，由于周期性变化，导致的原存在的多个周期因为数据缺失导致合成为一个周期，进而使红绿灯时间偏大；定位偏误，可以认为是轨迹数据集中存在异常数据，可以通过对数据进行聚类分析以剔除噪声，实现周期估计的准确性。接着，我们再对问题一中提出的模型进行分析，发现可以很好地解决定位偏误问题，用于求解。

（三） 问题三的分析

第三问要求我们考虑路口红绿灯的周期性变化问题，需要找出周期变化的时间点和变化前后的红绿灯时长。红绿灯时长的判断可以沿用前两问的模型，对于周期性判断，要选取合适的数据框来对所有数据进行分割。这可以采用滑动窗口法，不断平移较小的数据框，对数据框平移前后进行比较，再选择合适的阈值进行周期性判断。对于阈值的选取可以考虑使用加权平均，峰态系数等指标综合考虑。

（四） 问题四的分析

问题四要求我们对路口多个方向上进行红绿灯周期的估计。首先，我们先对路口位置和车辆轨迹进行热力图分析，确定路口位置和形状后，再通过不同车辆的轨迹来确认不同方向，对行驶不同方向的车辆进行划分，得到多个特定方向上车辆的轨迹的数据集，最后利用已经建立的数学模型处理方向不同的数据集，得到该路口不同方向红绿灯的周期。

三、模型假设

（一） 假设

1. 信号灯只有红、绿两种状态，且不考虑信号灯失灵的情况。
2. 所有车辆均正常行驶，无突发事故。
3.忽略驾驶员的反应时间，驾驶员会针对信号灯和其他车辆行为做出及时反应。

四、符号说明

表 1 本文符号说明

符号	说明	单位
v_{id,t_i}	编号为 id 的车在t_i时速度	米每秒（m/s）
x_{id,t_i}	编号为 id 的车在t_i时 x 位置	米（m）
y_{id,t_i}	编号为 id 的车在t_i时 y 位置	米（m）
ε	判断车辆是否停止的速度阈值	米每秒（m/s）
t_r	估计的红灯持续时间	秒（s）
t_g	估计的绿灯持续时间	秒（s）
T	周期时间	秒（s）
Δt_r	红灯估计时间的误差	秒（s）
Δt_g	绿灯估计时间的误差	秒（s）
T_r	红灯实际持续时间	秒（s）
T_g	绿灯实际持续时间	秒（s）
μ	其他干扰带来的误差时间	秒（s）
R_i	红灯第 i 段持续时长	秒（s）
G_i	绿灯第 i 段持续时长	秒（s）
RWS_j	红灯第 j 个窗口下的时长加权和	秒（s）
GWS_j	绿灯第 j 个窗口下的时长加权和	秒（s）
ΔRWS_k	红灯第 k+1 个窗口下的时长加权和与第 k 个窗口下的时长加权和之差	秒（s）
ΔGWS_k	绿灯第 k+1 个窗口下的时长加权和与第 k 个窗口下的时长加权和之差	秒（s）
$F(x)$	用于计算加权值的函数	秒（s）
ε_2	判断周期变动阈值	秒（s）

五、问题一的求解

（一） 数据的分析与处理

车辆的行驶状态和道路息息相关，车辆在不同道路上的运动模式也会存在不同特点，因此，我们首先对车辆行驶过程中的轨迹数据特征进行表达和分析。

1. 提取给出的各类信息

原始数据由车辆编号 id，时间 time，以及位置信息 x，y 组成，每个轨迹T_n由 id 相同的(x,y)组成，通过分析轨迹数据所包含的时间信息和空间信息，进一步提取车辆运动轨迹中隐含的行驶特征，包括当前的行驶速度、行驶方向、与上一轨迹点之间的间距和时间间隔。

2.热力图绘制散点的数据特征

原始数据集过于庞大，如果能够将轨迹点的分布可视化，将大大提高路口位置的准确性，为此，我们将轨迹点的散点用热力图进行可视化，方便进一步分析，得到的附件 1 中 A1~A5 的数据热力图如图 1：

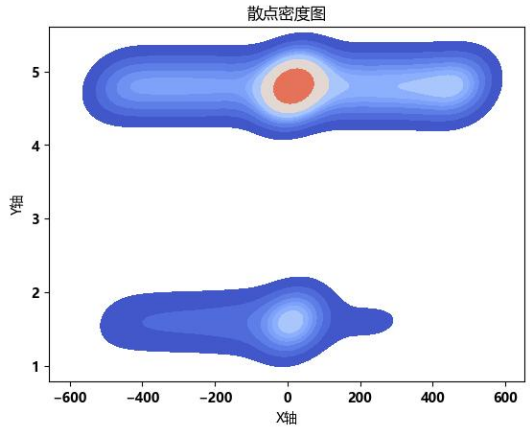

图 1-1 A1 数据热力图

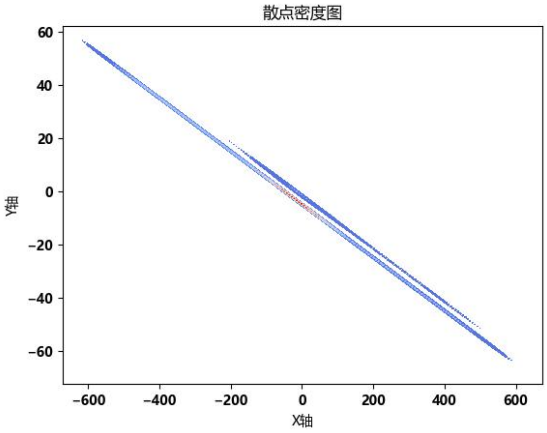

图 1-2 A2 数据热力图

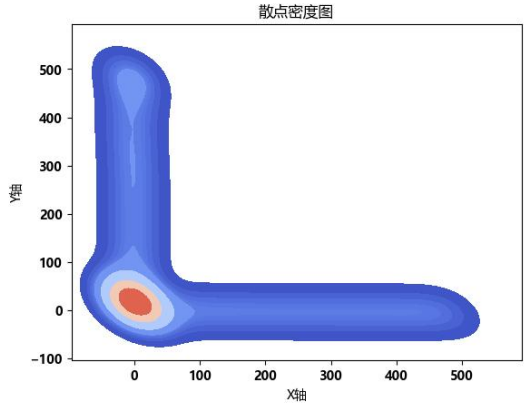

图 1-3 A3 数据热力图

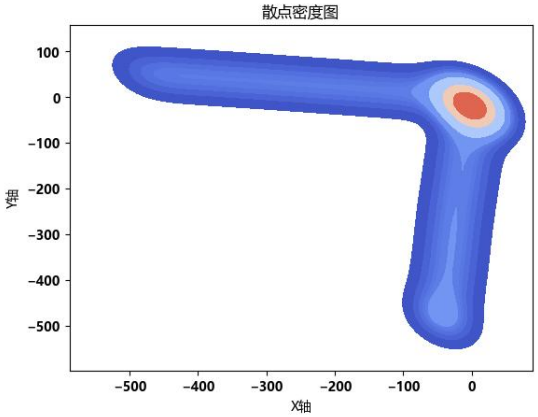

图 1-4 A4 数据热力图

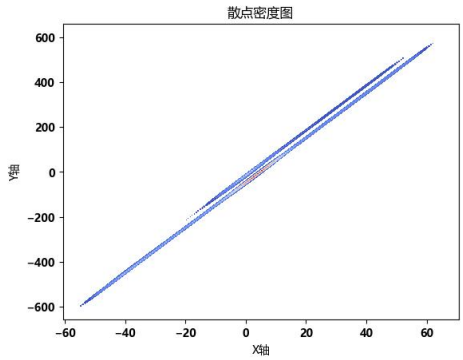

图 1-5 A5 数据热力图

3.使用 DBSCSAN 聚类方法来确定红绿灯的大致位置，并进行数据筛选

红绿灯位置常出现车辆转向和排队等待等情况，反映到数据集中，即在某一位置时车辆轨迹出现密集现象，因此可以通过筛选轨迹点密集的位置作为红绿灯位置的候选集。又由于已经假设路口只存在一个，并只研究单个方向，故确定红绿灯位置的大致范围。

(1) DBSCAN 聚类

DBSCAN 是一种基于密度的聚类算法，主要用于发现任意形状的聚类簇，并且对噪声数据也有较好的容忍度。在进行 DBSCAN 聚类时，只需要两个核心参数：

i. ε，即社区的最大半径，它是 DBSCAN 用来确定两个点是否相似和属于同一类的距离。更大的 epsilon 将产生更大的簇(包含更多的数据点)，更小的 epsilon 将构建更小的簇。

ii. minPts，即点的最小个数，在一个邻域的半径内 minPts 数的邻域被认为是一个簇。

有了这两项核心参数，我们就可以对原数据集进行聚类分析，首先，选择一个在其半径内至少有 minPts 的随机点。然后对核心点的邻域内的每个点进行评估，以确定它是否在 ε 距离内有 minPts (minPts 包括点本身)。如果该点满足 minPts 标准，它将成为另一个核心点，集群将扩展。如果一个点不满足 minPts 标准，它成为边界点。随着过程的继续，算法开始发展成为核心点“a”是“b”的邻居，而“b”又是“c”的邻居，以此类推。当集群被边界点包围时，这个聚类簇已经搜索完全，因为在距离内没有更多的点。选择一个新的随机点，并重复该过程以识别下一个簇，最后能将数据集划分。

(2) DBSCAN 用于数据清洗

在本题中，由于只存在一个路口的一个行驶方向，在红灯亮起时，大量轨迹点将密集分布在红绿灯附近，在绿灯时亮起时，轨迹点不会出现大量密集分布的情况，故可以通过 DBSCAN 聚类来选择轨迹点密集分布处作为红绿灯的大致位置。

在确定大致位置后，由于 DBSCAN 算法对噪声能够很好处理，我们在 DBSCAN 聚类处理结果的基础上，选择红绿灯的大致位置作为中心点，以 ε 作

为社区半径，以 minPts 作为最小点数，得到在路口位置附近的轨迹点数据。相较于其他簇，基于红绿灯位置的簇中，数据更加密集，数据集更能代表总体特征；同时由于靠近红绿灯位置，该簇内轨迹点也能用于估计车辆由于红灯亮起所停止的时间。对这部分数据集建立数学模型并求解，能够使红绿灯周期的估计更加有效。

（二） 建立数学模型并求解

1. 建立数学模型

为分析车辆的运动与红灯亮起之间的关系，将 id 相同的经过 DBSCAN 聚类处理后的数据集合：

$$(X_{id}, Y_{id}) = \{(x_{id,t_1}, y_{id,t_1}), (x_{id,t_2}, y_{id,t_2}), (x_{id,t_3}, y_{id,t_3}), \cdots, (x_{id,t_n}, y_{id,t_n})\}$$

在平面直角坐标系下，通过下列公式转化进行分析：

$$\Delta x_{id,t_i} = x_{id,t_{i+1}} - x_{id,t_i}$$

$$\Delta y_{id,t_i} = y_{id,t_{i+1}} - y_{id,t_i}$$

$$v_{id,t_i} = \frac{\sqrt{\Delta x_{id,t_i}{}^2 + \Delta y_{id,t_i}{}^2}}{t_{i+1} - t_i}$$

为了研究各个时刻的红绿灯状态，需要设置阈值 ε 以判断车辆的运动情况：当车辆速度低于阈值 ε 时，认为车辆因红灯停止，以判断红绿灯的周期。

2. 红绿灯周期的估计

由于存在车流量过小、车辆异常位置变动、车辆等问题，无法单一地用 0 来判断车辆由于红灯亮起而停车，故我们引入函数 f 通过对v_{id,t_i}进行分类，从而判断相对应的时间段t_i是否为红灯亮起时间

$$f(t_i) = \begin{cases} 1, v_{id,t_i} < \varepsilon \\ 0, v_{id,t_i} \geq \varepsilon \end{cases}$$

由函数 f 易知，当 $f(t_i) = 1$ 时，表示为红灯时刻；当 $f(t_i) = 0$ 时，表示为绿灯时刻。我们针对每个路口进行计算，得到各个路口的“0”、“1”表。

3. 阈值ε的调节

对于阈值 ε，应当设置在合适区间内：当阈值设置较小时，由于抽样导致的误差及异常数据的影响，会导致能够反映红灯时刻的函数个数减少，从而使t_r

估计偏小；当阈值设置较大时，同理会带来能够反映红灯时刻的函数个数增加，从而使t_r估计偏大。因此，我们对阈值 ε 进行调参，通过 R 文件最终判定阈值的较优值，接着我们对红灯时长进行估计。

（三） 红、绿灯时长的估计

1. 估计原理

由于我们考虑的都是车辆无违反交通规则的情况，所以数据中红灯时间会小于等于实际的红灯时间（司机看到红灯刹车的时间与红灯转为绿灯后汽车启动的时间相互对冲一部分后可以忽略不计），因此我们选择用红灯持续时间的最大值来估计实际时间，再对升序排序后的后三个值求平均，以此来削弱异常值带来的影响。对于红灯实际时间和估计值的关系有

$$T_r = t_r + \Delta t_r + \mu$$

而绿灯时间会大于等于实际的绿灯时间。比如：车辆在移动但是离信号灯较远，此时驾驶员的驾驶行为不受信号灯影响；或者驾驶员看到信号灯后，根据经验判断在红灯转变为绿灯之后才到达路口，因而在看到远处信号灯为红灯时选择稍微减速或者保持原状态行驶；又或是车辆数据比较稀疏，导致长时间无车辆遇到红灯等情况也会发生。因此我们选择用绿灯持续时间的最小值来估计绿灯的实际时间，再对升序排序后且去除异常值的前三个值求平均，以此来削弱异常值带来的影响。对于绿灯实际时间和估计值的关系，有

$$T_g = t_g + \Delta t_g + \mu$$

而在一、二题的情景下，各组数据的信号灯周期是保持不变的。因此，每个周期的时长是保持不变的。当我们使用红灯最大值和绿灯最小值作为估计值时，红灯最大值与绿灯最小值是同一个周期内的，此时有

$$\Delta t_r + \Delta t_g = 0$$

$$T = T_r + T_g = t_r + t_g + \mu$$

即，对于周期时长的估计误差是非常小的。

2. 输出 A1-A5 的“0”，“1”表

在模型建立完成之后，我们用 R 语言将其实现，并做出一定的处理。

首先，我们使用经过数据清理后得到的数据，根据上文中建立的模型来输出每个路口的“0”、“1”表，如图 2：

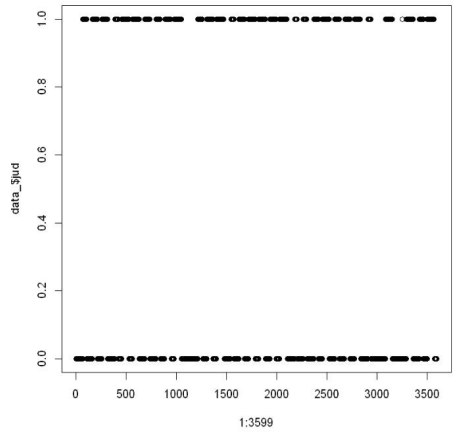

图 2-1 A1“0”、“1”表

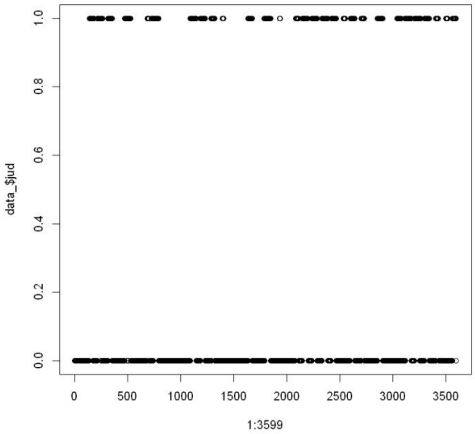

图 2-2 A2“0”、“1”表

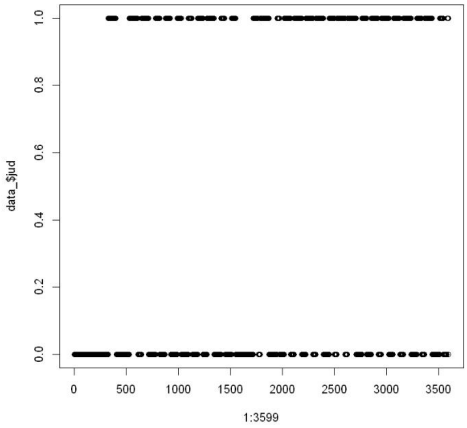

图 2-3 A3“0”、“1”表

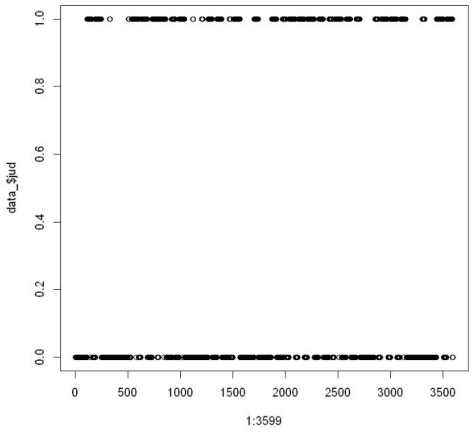

图 2-4 A4“0”、“1”表

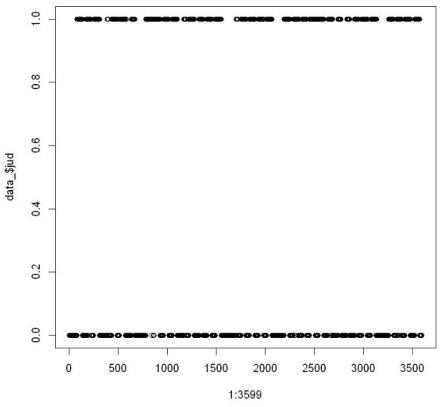

图 2-5 A5“0”、“1”表

因为对于其他非路口处，红绿灯不影响其运动，所以我们运用 DBSCAN 聚类分析法删去了一部分数据导致的缺失时刻数据用“0”，即绿灯来填充。

3. 对“0”、“1”表的处理

为了得到排序好的红灯和绿灯的时长，我们把完整的“0”、“1”表切割为局部“0”、“1”值连续的列表，例如将（0，0，0，1，1，1，0，0，0）切割

为（0，0，0）、（1，1，1）和（0，0，0）。经过处理后，我们可以很容易得到红灯、绿灯在时刻表上的持续时间，并将其进行排序。

由于数据中存在一些异常现象，如图 3 以下情况：

59	3.64	-11.83		
59	3.64	-11.83		
59	3.64	-11.83		
59	0.46	-11.51		
59	0.46	-11.51		
59	0.46	-11.51		
59	0.46	-11.51		
59	0.46	-11.51		
59	0.46	-11.51		
59	0.46	-11.51		
59	0.46	-11.51		
59	3.64	-11.83		
59	3.64	-11.83		
59	3.64	-11.83		
59	3.64	-11.83		

图 3 异常情况举例

59 号车在一段时间内进行了两次疑似变道的行为，且同时刻没有其他车辆在路口处等待。这导致原本应该设置为“1”，即红灯的时刻被误判为“0”，打断了红灯的持续时间，造成最大值的估计错误。因此，我们编写了一个修改此类误差值的 R 语言函数，将一段时间内有少于 3 次突然变动的情况修正为前后的共同情况，比如将（1，1，1，0，1，1，1）修正为（1，1，1，1，1，1，1），以此来提高模型的估计精度。

4. 红、绿灯时长的平均估计

最后，我们输出红、绿灯升序的持续时间。由于我们设置的修正为一段时间内的少于 3 次的变动，所以还是会有少量异常值的出现。为解决此类问题并削弱异常值带来的影响，我们首先删去 10s 以内的红、绿灯持续时长，将其视为异常值，然后取红灯持续时长中前三大的数据以及绿灯持续时长中前三小的数据，分别求平均，用这个平均时长来作为我们在此模型下估计的红、绿灯时长。

（四）识别结果

经过上述模型的求解，我们得到了基于附件 1 的 A1~A5 数据集各自的信号灯周期，结果见下表 2：

表 2 第一题结果

路口	A1	A2	A3	A4	A5
红灯时长（秒）	75.33	57	82	68.67	62.67
绿灯时长	30.33	28	21.33	16	22

（秒）					

六、问题二的求解

（一） 统计学分析

1. 车辆比例带来的误差分析

假设车辆的实际轨迹点集为T_{real}，研究的轨迹点的数据集为T_n。我们基于样本的数据集，实际上是对总体T_{real}的进行了抽样，得到的T_n。当抽样时车辆比例出现偏差时，得到的样本可能会失去能够反映原数据集的数据特征的某些良好属性，进而导致估计的误差变大。

2. 车流量带来的误差分析

车流量能够一定程度上代表总体数据的分布：车流量越大，轨迹数据集越密集，所获取的信息更全面，数据特征更突出，易于判断红绿灯；车流量越小，轨迹数据越稀疏，所获得的信息不够完整，可能出现长时间无车辆遇到红灯的情况。虽然这并不影响我们使用红灯持续时间的最大值和绿灯持续时间的最小值来预测红灯、绿灯的时长，但是样本数据量的减少会造成预测的红灯时间偏小以及绿灯的持续时间偏大的误差更大。以对红灯的时长预测为例，如果样本量变小，可能会导致 μ 变大，在实际时长不变的情况下，预测的红灯时长偏差会更大。

$$T_r = t_r + \Delta t_r + \mu$$

3. 定位偏误带来的误差分析

定位偏差在轨迹数据集上的表现为出现异常值。异常值会导致我们算法对车辆是否运动产生误判，从而出现瞬时的红灯时刻或在红灯时刻出现车辆移动情况。基于此，我们引入 DBSCAN 聚类算法，剔除部分噪声，在原数据集上进行数据清洗后得到新数据集后再进行求解。新数据集对比原数据集，不仅剔除异常值，而且选用了路口处的轨迹数据，简化了数据处理的难度，同时也能集中路口处的轨迹数据，再对其进行异常值修正，这样能够很好地解决定位偏误对模型求解的影响。还是以红灯预测为例，这种情况类似第一题中图 3 的数据异常情况，此时Δt_r会变大，导致t_r偏小得更多，偏差会更大。

$$T_r = t_r + \Delta t_r + \mu$$

（二） 对模型的影响及求解

通过 DBSCAN 聚类算法，我们删除了噪声点，减少了定位偏误所带来的异常值影响，而车辆比例、车流量所带来的影响是抽样时产生的偏误，难以在单个样本数据集上解决，所以在其他影响因素无法解决的条件下，我们认为通过问题

一所得到的数学模型有较为优良的性能，能够剔除异常值所带来的影响。现在，我们基于附件 2 的 B1~B5 数据集，仍采用问题一的算法，得到了各自的信号灯周期，结果见下表 3：

表 3 第二题结果

路口	B1	B2	B3	B4	B5
红灯时长（秒）	79.67	88	66	76.33	98.33
绿灯时长（秒）	26	28.33	22	39.33	19

七、问题三的求解

（一） 数据的分析处理

在第三问中，我们沿用前两问的思路，对第三问给出的六个路口数据进行处理，得到各个路口的“0”、“1”表，再得出红绿灯交替的持续时间。由于红、绿灯对比无法体现周期变化，所以我们将红灯时间与绿灯时间分离，得到各自分段的持续时间，再对红、绿灯的持续时间分别做分析。

由于样本数据并不完美，我们还需要去除掉一些异常值。根据我们的思路，我们需要用于判断的数据应该是绿灯时间中的较小值以及红灯时间中的较大值，因此，我们需要清洗掉一些异常值，比如一些极大的绿灯持续时间，和一些极小的红灯数据。

我们分别计算出每一段红、绿灯持续时间的中位数，再分别清洗掉红灯中小于中位数的数据，以及绿灯中大于中位数的数据。

由于存在恰好长时间没有车辆等红灯以及红灯时车辆变道等现象，这会导致出现很大的绿灯持续时间和极短暂的红灯持续时间。所以如果我们不去除掉这些异常值，在后续建模求解中就会受到很大的干扰

（二） 模型建立

我们对各个路口的红、绿灯持续时间分别进行分析。在处理完异常数据之后，我们使用滑动窗口法，用 4 作为数据框的大小，遍历所有数据。

对于每一个区间内的数据，我们计算它们的加权和：

$$RWS_j = \sum_{i=j}^{j+4} \mathrm{F}(x) = \sum_{i=j}^{j+4} \mathrm{F}(R_i) \quad i = 1,2,3\ldots\ldots$$

$$GWS_j = \sum_{i=j}^{j+4} \mathrm{F}(x) = \sum_{i=j}^{j+4} \mathrm{F}(G_i) \quad i = 1,2,3\ldots\ldots$$

接下来，我们再分别对红、绿灯的相邻两个窗口加权和做差

$$\Delta RWS_k = |RWS_{k+1} - RWS_k| \quad k = 1,2,3\ldots\ldots$$
$$\Delta GWS_k = |GWS_{k+1} - GWS_k| \quad k = 1,2,3\ldots\ldots$$

得到窗口加权和之差后，判断差值是否小于我们找到的阈值 ε_2。

如果

$$\Delta RWS_k = |RWS_{k+1} - RWS_k| < \varepsilon_2$$

则认为周期未发生变化；
如果

$$\Delta RWS_k = |RWS_{k+1} - RWS_k| \geq \varepsilon_2$$

则认为周期发生了变化。

同时，对于每个周期内的红、绿灯持续时间，我们仍然采用最值的均值作为估计值，来削弱误差带来的影响。

（三） 模型结果

路口	C1	C2	C3	C4	C5	C6
周期 1 红灯时长（秒）	54.67	67	68.33	78	53	75
周期 1 绿灯时长（秒）	35.33	50.5	28	31.33	81	35
周期切换时刻	387	472	816	2234	2195	1198
周期 2 红灯时长（秒）	54.67	67	79.33	78	53	75
周期 2 绿灯时长（秒）	70.33	66.33	28	42.33	63.5	45.33

周期 切换时刻	939	2871	2010	2640	2649	4570
周期 3 红灯时长 （秒）	54.67	67	79.33	78	53	75
周期 3 绿灯时长 （秒）	38.33	24.33	41.67	34	57	36.67
周期 切换时刻	3564	4331	2753	6628	3552	
周期 4 红灯时长 （秒）	54.67	67	79.33	78	53	
周期 4 绿灯时长 （秒）	56.33	75.5	27	55	47	
周期 切换时刻	4214	5699			4690	
周期 5 红灯时长 （秒）	54.67	67			53	
周期 5 绿灯时长 （秒）	43.33	39.5			73.33	
周期 切换时刻					6464	
周期 6 红灯时长 （秒）					53	

周期 6 绿灯时长 （秒）					47	

八、问题四的求解

（一）数据的可视化处理与分析

我们先对题目中给出的数据进行可视化处理和分析。对于所有车在每一时刻的位置，我们绘制了热力图（如图 4），显然，我们可以发现该路口为一个十字路口。

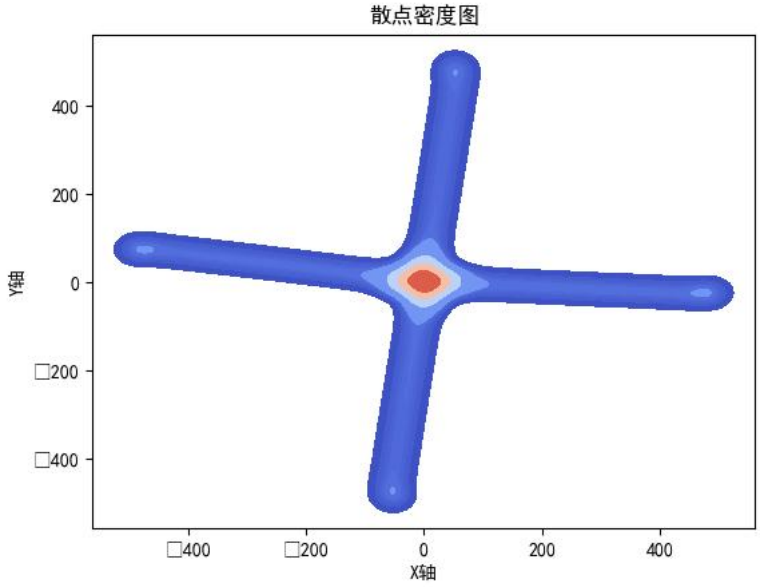

图 4 第四题车辆位置热力图

接着，我们对所有数据进行筛选。由于在第四题中可以存在不同的行驶方向，为了针对不同的行驶方向得出各方向的信号灯时间，我们需要把所有方向中的对应车辆的时间与位置信息提取出来，对每个方向分别进行分析。

（二） 车辆行驶方向的确定

为了确定车辆行驶的方向，我们把每辆车行驶的起点与终点提取出来，对每辆车的行驶方向进行判读。基于热力图，我们可以看出，大量的车辆轨迹点集中在点(0,0)附近，且轨迹分布成十字型。因此，我们引入两条直线l_1：y = x 与l_2:$y = -x$ 对轨迹点进行划分，划分标准及结果如下：

$$\text{region(x,y)} = \begin{cases} 1, y > x \; and \; y > -x \\ 2, y < x \; and \; y > -x \\ 3, y < x \; and \; y < -x \\ 4, y > x \; and \; y < -x \end{cases}$$

通过划分出的四个区域，能够大致判断起点与终点的区域变化，分别以 $region_{begin}$-$region_{end}$ 的数据来表现车辆的移动情况，从而确定车辆的行驶方向，分别为 i-j (i = 1,2,3,4; j = 1,2,3,4) 16 个方向。通过对数据的进一步观察，我们发现 1-1，2-2，3-3，4-4 这四个方向的数据的样本量相较于其他方向的数据量过小，存在严重的数据缺失情况，且由于相同区域内产生的变化也无法判断红绿灯的周期，故删除这四个方向，对剩下的 12 个方向逐个分析：通过研究某一方向上所有车辆的轨迹数据集，用问题三所构建的数学模型进行求解得到单个方向的路口信号灯的周期T_{i-j}，最后对所有方向的周期进行整合得到该路口信号灯周期的数据集 T = $\{T_{1-2}, T_{1-3}, T_{1-4}, ..., T_{4-3}\}$

（三） 模型结果

方向	周期1红灯时长	周期1绿灯时长	周期切换时刻	周期2红灯时长	周期2绿灯时长	周期切换时刻	周期3红灯时长	周期3绿灯时长	周期切换时刻	周期4红灯时长	周期4绿灯时长	周期切换时刻	周期5红灯时长	周期5绿灯时长
D1	122.5	18	4056	122.5	14.5									
D2	86.71	44.75												
D3	40.75	97.6	979	40.75	69.25	1280	84.33	69.25	2488	94	69.25	3801	94	54
D4	100.67	56.75	2018	84.75	56.75	2175	84.75	62.5						
D5	107.25	31	2018	93.5	31									
D6	97	35												
D7	100	50	5629	95	50									

D8	90	40	2250	80	55	3098	60	85	4332	80	45			
D9	120	30	1929	120	20									
D10	105	25	822	115	25	6074	105	25						
D11	100	40												
D12	100	45	784	100	60	2180	80	60	4201	100	60			

九、模型评价

（一） 模型的优点

本文所使用的模型，不仅仅是对每一辆车进行分析，更通过时间截面的思想，同时观察每一个时间点上所有车的运动状态。通过这种方法我们可以得到一个连贯的红绿灯时间时间序列进行处理。

其次，考虑到车辆占比以及车流量，还有定位的误差，我们利用热力图展示出红绿灯路口的位置，极大的清洗掉了无用的数据。

再者，本文使用的模型可以针对性的修改参数，在极端情况下可以通过调整参数使模型发挥更好的作用

（二） 模型的缺点

本文使用的模型对于变异的抵抗性较差，部分极端值如果没有在数据清洗的时候清除，会对最后模型的结果产生极大的影响。

其次，本文在对于周期判断时需要的数据量较大，如果在实际生活里运用，难以在短时间内做出判断。

后续如果要增强模型的判定速度，可以考虑进一步用机器学习的窗口滑动法来判断周期的改变。

而针对变异值的问题，后续可以进一步利用好变异对 T 影响较小的优点，将红绿灯时间通过公式

$$t_{r_i} + T = t_{r_{i+1}}$$

进行迭代后，得到更广泛和准确的数据

十、参考文献

[1]李志聪,孙旭阳.基于离群点检测和自适应参数的三支 DBSCAN 算法[J/OL].计算机应用研究:1-7[2024-04-21]
[2]赵肄江,方辰昱,廖祝华.一种基于轨迹数据的红绿灯位置检测方法[J].测绘地理信息,2024,49(02):122-130
[3]李晓明,顾钰培,张俊涛.一种滑动窗口的 GPS 轨迹点地图匹配算法[J].西安工业大学学报,2017,37(06):459-462

附件：
本文中需要用到的 R 语言代码如下：

```
fs1 = function(A, len, dat){
  n = 0
  for (j in len){
    if (jud_v(j, datA1)){n = n + 1}
    A1 = append(A1, jud_v(j, datA1))
  }
  return(A1)
}

jud_in = function(i, dat){
  for (m in dat){
    if (i == m){return(TRUE)}
  }
  return(FALSE)
}

jud_v = function(t, dat){
  dat_tmp1 = subset(dat, time == t, select = c(x, y, vehicle_id))
  dat_tmp2 = subset(dat, time == t + 1, select = c(x, y, vehicle_id))
  # print(dat_tmp2)

  for (i in dat_tmp1$vehicle_id){
    tmp1_x = subset(dat_tmp1, vehicle_id == i, select = c(x))
    tmp1_y = subset(dat_tmp1, vehicle_id == i, select = c(y))
    if (jud_in(i, dat_tmp2$vehicle_id)){
      tmp2_x = subset(dat_tmp2, vehicle_id == i, select = c(x))
      tmp2_y = subset(dat_tmp2, vehicle_id == i, select = c(y))
    }else{next}

    # 参数
    if (sqrt((tmp1_x - tmp2_x) ** 2 + (tmp1_y - tmp2_y) ** 2) < 1) {return(1)}
  }
  return(0)
}
```

```
f0 = function(cve){
   tmp = c()
   cve_new = cve
   if (cve[1] == 0){
      for (i in cve){
         if(i == 1){
            return(list(tmp, cve_new))} else {
               tmp = append(tmp, 0)
               cve_new = cve_new[-1]
               if (length(cve_new) == 0){return(list(tmp, cve_new))}
            }
      }
   }

   if (cve[1] == 1){
      for (i in cve){
         if(i == 0){
            return(list(tmp, cve_new))} else {
               tmp = append(tmp, 1)
               cve_new = cve_new[-1]
               if (length(cve_new) == 0){return(list(tmp, cve_new))}
            }
      }
   }
}

f1 = function(a){
   A = c()
   while(length(a) != 0){
      m = f0(a)
      m1 = m[1]
      a = unlist(m[2])
      A = append(A, m1)
   }
   return(A)
}

fchange = function(A){
   if (A == 0){return(1)}
   if (A == 1){return(0)}
}
```

```
fwash = function(A){
   n = length(A)
   m = 0
   for (i in 1:n){
      i = i - m
      if (i + 1 > length(A)){
         print("-------分割线 3-------")
         break}
      if (length(A[[i]]) <= 3){
         print("-------分割线 4-------")
         print(i)
         for (j in 1:(length(A[[i]]))){
            A[[i]][j] = fchange(A[[i]][j])
         }
         A[[i - 1]] = append(A[[i - 1]], A[[i]])
         A[[i - 1]] = append(A[[i - 1]], A[[i + 1]])
         A[[i]] = NULL
         A[[i]] = NULL
         m = m + 1
      }
   }
   return(A)
}

fjud01 = function(m){
   if (m[1] == 0){return(0)}
   if (m[1] == 1){return(1)}
}

fflt0 = function(A){
   lis = list()
   for (i in A){
      if (fjud01(i) == 0){
         lis = append(lis, length(i))
      }
   }
   lis = unlist(lis)
   return(sort(lis))
}

fflt1 = function(A){
   lis = list()
   for (i in A){
      if (fjud01(i) == 1){
```

```
            lis = append(lis, length(i))
        }
    }
    lis = unlist(lis)
    return(sort(lis))
}

fjud01 = function(m){
    if (m[1] == 0){return(0)}
    if (m[1] == 1){return(1)}
}

Fflt0 = function(A){
    lis = list()
    for (i in A){
        if (fjud01(i) == 0){
            lis = append(lis, length(i))
        }
    }
    lis = unlist(lis)
    return(lis)
}

Fflt1 = function(A){
    lis = list()
    for (i in A){
        if (fjud01(i) == 1){
            lis = append(lis, length(i))
        }
    }
    lis = unlist(lis)
    return(lis)
}

FF = function(A){
    lis = c()
    for (i in A){
        lis = append(lis, length(i))
    }
    return(unlist(lis))
}
```

```
jud_v = function(t, dat){
   dat_tmp1 = subset(dat, time == t, select = c(x, y, vehicle_id))
   dat_tmp2 = subset(dat, time == t + 1, select = c(x, y, vehicle_id))
   # print(dat_tmp2)

   for (i in dat_tmp1$vehicle_id){
      tmp1_x = subset(dat_tmp1, vehicle_id == i, select = c(x))
      tmp1_y = subset(dat_tmp1, vehicle_id == i, select = c(y))
      if (jud_in(i, dat_tmp2$vehicle_id)){
         tmp2_x = subset(dat_tmp2, vehicle_id == i, select = c(x))
         tmp2_y = subset(dat_tmp2, vehicle_id == i, select = c(y))
      }else{next}

      # 参数
      if (sqrt((tmp1_x - tmp2_x) ** 2 + (tmp1_y - tmp2_y) ** 2) < 0.7) {return(1)}
   }
   return(0)
}

fjd = function(dvec, dat){
   C1 = c()
   n1 = 0
   for (j in dvec){
      if (jud_v(j, dat)){n1 = n1 + 1}
      C1 = append(C1, jud_v(j, dat))
   }
   return(C1)
}

frc0 = function(dt, param){
   long = length(dt)

   # 分位数参数
   rt = quantile(dt, 0.6)

   record = c()

   for (i in 1:long){
      m = 0



      dtf_1 = c()
      dtf_2 = c()
      while (dt[i] >= rt | dt[i] < 10) {
```

```
      i = i + 1
    }
    if (i + 4 >= long){break}
    dtf_1 = c(dtf_1, dt[i])

    while (dt[i + 1] >= rt | dt[i + 1] < 10) {
      i = i + 1
      if (i + 4 >= long){break}
    }
    dtf_1 = c(dtf_1, dt[i + 1])
    dtf_2 = c(dtf_2, dt[i + 1])
    m = i + 1
    if (i + 4 >= long){break}


    while (dt[i + 2] >= rt | dt[i + 2] < 10) {
      i = i + 1
      if (i + 4 >= long){break}
    }
    dtf_1 = c(dtf_1, dt[i + 2])
    dtf_2 = c(dtf_2, dt[i + 2])
    if (i + 4 >= long){break}

    while (dt[i + 3] >= rt | dt[i + 3] < 10) {
      i = i + 1
      if (i + 4 >= long){break}
    }
    dtf_1 = c(dtf_1, dt[i + 3])
    dtf_2 = c(dtf_2, dt[i + 3])
    if (i + 4 >= long){break}

    while (dt[i + 4] >= rt | dt[i + 4] < 10) {
      i = i + 1
      if (i + 4 >= long){break}
    }
    dtf_2 = c(dtf_2, dt[i + 4])

    # 调参
    m1_ss = sum(mean(dtf_1))
    m2_ss = sum(mean(dtf_2))

    if (abs(m1_ss - m2_ss) > param){record = c(record, m)}
  }
  return(unique(record))
```

```
}


frc1 = function(dt, param){
  long = length(dt)

  # 分位数参数
  rt = quantile(dt, 0.4)

  record = c()

  for (i in 1:long){
    m = 0


    dtf_1 = c()
    dtf_2 = c()
    while (dt[i] <= rt | dt[i] > 200) {
      i = i + 1
    }
    if (i + 4 >= long){break}
    dtf_1 = c(dtf_1, dt[i])

    while (dt[i + 1] <= rt | dt[i + 1] > 200) {
      i = i + 1
      if (i + 4 >= long){break}
    }
    dtf_1 = c(dtf_1, dt[i + 1])
    dtf_2 = c(dtf_2, dt[i + 1])
    m = i + 1
    if (i + 4 >= long){break}


    while (dt[i + 2] <= rt | dt[i + 2] > 200) {
      i = i + 1
      if (i + 4 >= long){break}
    }
    dtf_1 = c(dtf_1, dt[i + 2])
    dtf_2 = c(dtf_2, dt[i + 2])
    if (i + 4 >= long){break}

    while (dt[i + 3] <= rt | dt[i + 3] > 200) {
      i = i + 1
      if (i + 4 >= long){break}
```

```
    }
    dtf_1 = c(dtf_1, dt[i + 3])
    dtf_2 = c(dtf_2, dt[i + 3])
    if (i + 4 >= long){break}

    while (dt[i + 4] <= rt | dt[i + 4] > 200) {
      i = i + 1
      if (i + 4 >= long){break}
    }
    dtf_2 = c(dtf_2, dt[i + 4])

    # 调参
    m1_ss = sum(mean(dtf_1))
    m2_ss = sum(mean(dtf_2))

    if (abs(m1_ss - m2_ss) > param){record = c(record, m)}
  }
  return(unique(record))
}
```

调用方法以 C1 为例

```
datA1 = read.csv("F:/hzb/B 题：使用行车轨迹估计交通信号灯周期问题_1713424222679/附件
/附件 3/C1.csv")
C3_st = c()
C3_st = fs1(C3_st, seq(19, 3599), datC3)
C3 = f1(C3_st)
C1_Ex = fwash(f1(C1))
C1_0 = Fflt0(C1_Ex)
C1_1 = Fflt1(C1_Ex)
```