计算结果为0的情况。

2.9 车辆轨迹数据的切片与分段

2.9.1 轨迹的切片

通过前面的代码，成功地从数据中提取了停车和出行信息。然而，在所得到的出行信息中，只包含每次出行的起始点和终止点的时间以及经纬度信息，而没有出行轨迹信息。为了进一步分析车辆的出行轨迹，需要从每次出行的时间段内提取轨迹数据，即根据出行信息对轨迹数据集进行切片（slice）。在之前计算得到的出行信息中，每条出行信息都具有出行id、起始时间和结束时间列。轨迹切片的结果是提取出行过程中的轨迹点，并为每个轨迹点分配出行id标签。

轨迹切片的思路如图 2.40所示。为轨迹数据创建一个flag列，用于标记该行是否位于所需切片的时间段内。然后将出行数据中的每次出行记录分解为开始时间记录（flag标签为1）和结束时间记录（flag标签为-1），并插入到轨迹数据中。接下来，按车辆对flag列进行分组求和，得到flag1列。在结果中，flag1列值为1（出行）且flag列值为0（非临时插入的数据）的行即为所需的轨迹数据。

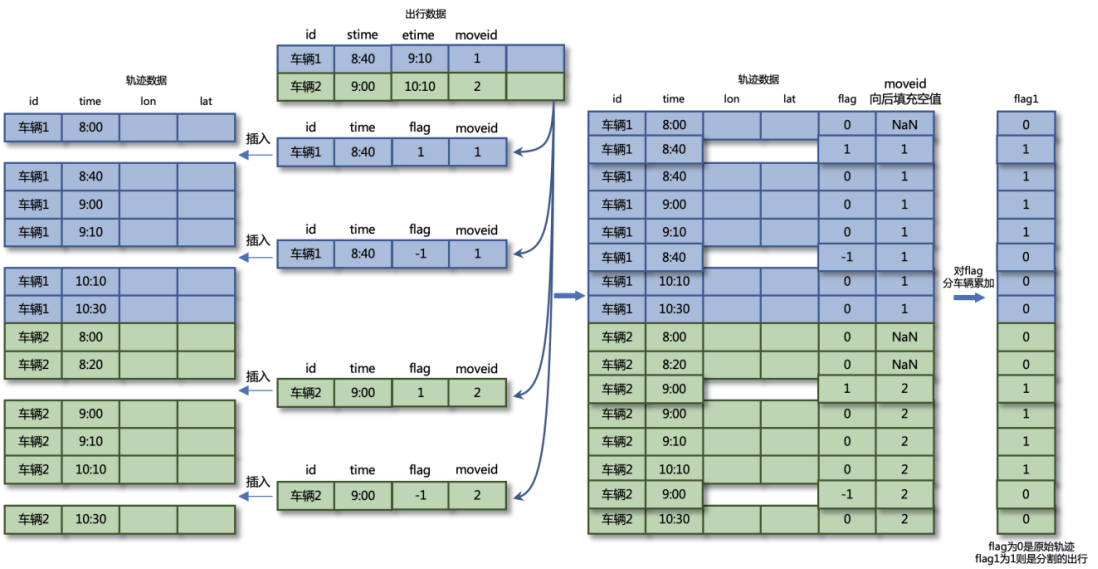

图2.40 轨迹的切片思路

需要注意的是，这里的切片思路要求用于切片的时间段不存在重叠的时间段，也就是说，如果一个时间段内存在两次出行，那么这种切片方法就无法实现。实现的代码如下：

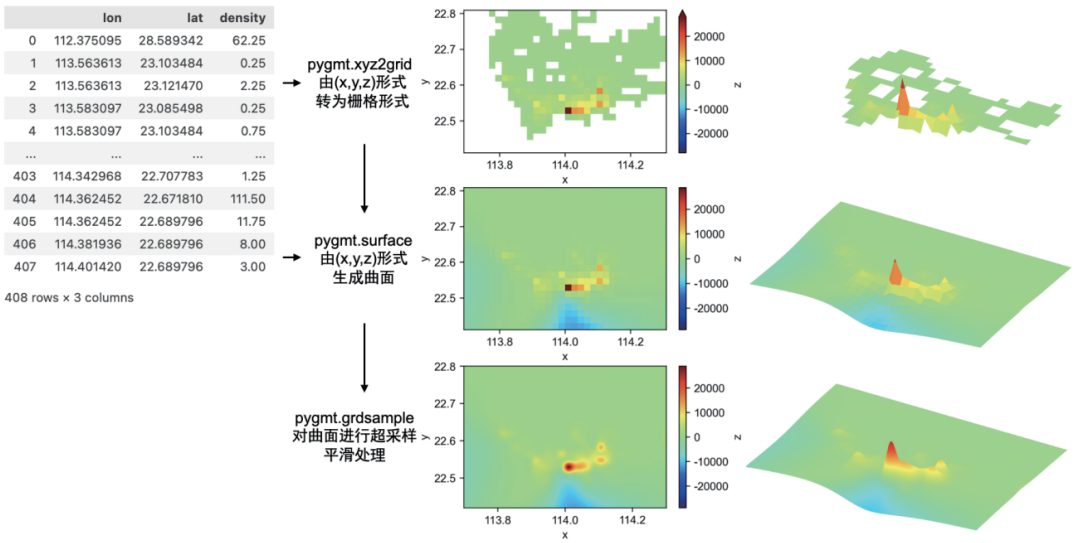

图3.9　pyGMT对空间数据的操作

下面基于前面的思路对数据分布数据进行数据处理与热力图绘制，首先用前面TransBigData栅格统计的方法计算数据分布密度，也就是获得整个曲面上的“控制点”。这里将数据统计量换算为数据的分布密度，单位为（条/km^2）：

```
data = pd.read_csv(r'Data/GPSData.csv')
# 将GPS栅格化，获得栅格参数
bounds = [113.7, 22.42, 114.3, 22.8]
grid_size = 2000
params = tbd.area_to_params(bounds,accuracy = grid_size)
# 对应的栅格编号
data['LONCOL'], data['LATCOL'] = tbd.GPS_to_grid(data['lon'], data['lat'],
params)
# 统计每一栅格数据量
grid_agg = data.groupby(['LONCOL', 'LATCOL'])['id'].count().rename('count').
reset_index()
# 转换为数据密度，单位为（条/km^2）
grid_agg['density'] = grid_agg['count'] / (grid_size/1000) ** 2
# 生成栅格中心点经纬度
grid_agg['lon'],grid_agg['lat'] = tbd.grid_to_centre([grid_agg['LONCOL'], grid_
agg['LATCOL']], params)
grid_agg = grid_agg[['lon','lat','density']]
grid_agg
```

接下来，需要用PyGMT包中的方法对数据地理分布进行计算，首先需要导入PyGMT，并定义研究范围：

```
import pygmt
# 通过region参数设置区域范围，经纬度的顺序为[west, east, south, north]，与bounds的顺序
不同
region = [bounds[0],bounds[2],bounds[1],bounds[3]]
# [113.7, 114.3, 22.42, 22.8]
```

5级别约为燃料消耗量的5%～7%，在Euro 6级别约为燃料消耗量的3%～4%。如果要计算这一项所产生的CO_2，需要首先计算每个排气系统技术类别中的燃料消耗量，然后计算使用的尿素总量，再计算CO_2排放量。

5.1.7　COPERT 模型小结

本节介绍了COPERT模型的基本原理和各类排放污染物的方法，为了方便理解，将本节的内容整理为思维导图，如图 5.1所示。

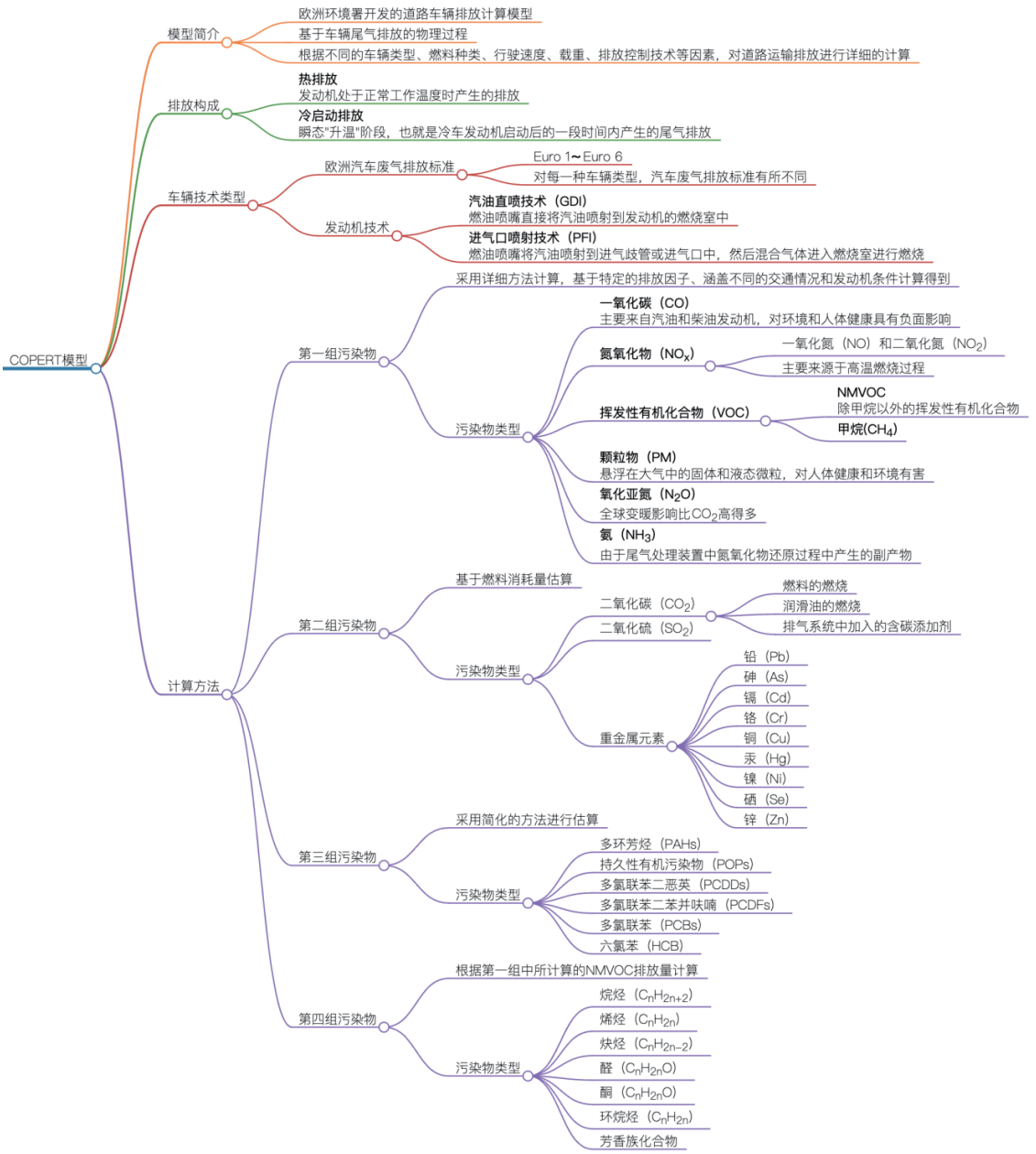

图5.1　COPERT排放计算模型

```
df_sb2_l1['geometry'] = gpd.points_from_xy(df_sb2_l1['Global_X'], df_sb2_
l1['Global_Y'])
df_sb2_l1 = gpd.GeoDataFrame(df_sb2_l1)

# 导入路段拓扑结构
nb_s2_l1 = gpd.read_file(r'nb_s2_l1/nb_s2_l1.shp')
nb_s2_l1_shp = nb_s2_l1['geometry'].iloc[0]

# 进行路段投影
df_nb_s2_l1['project'] = df_nb_s2_l1.apply(lambda r: nb_s2_l1_shp.
project(r['geometry']), axis=1)
```

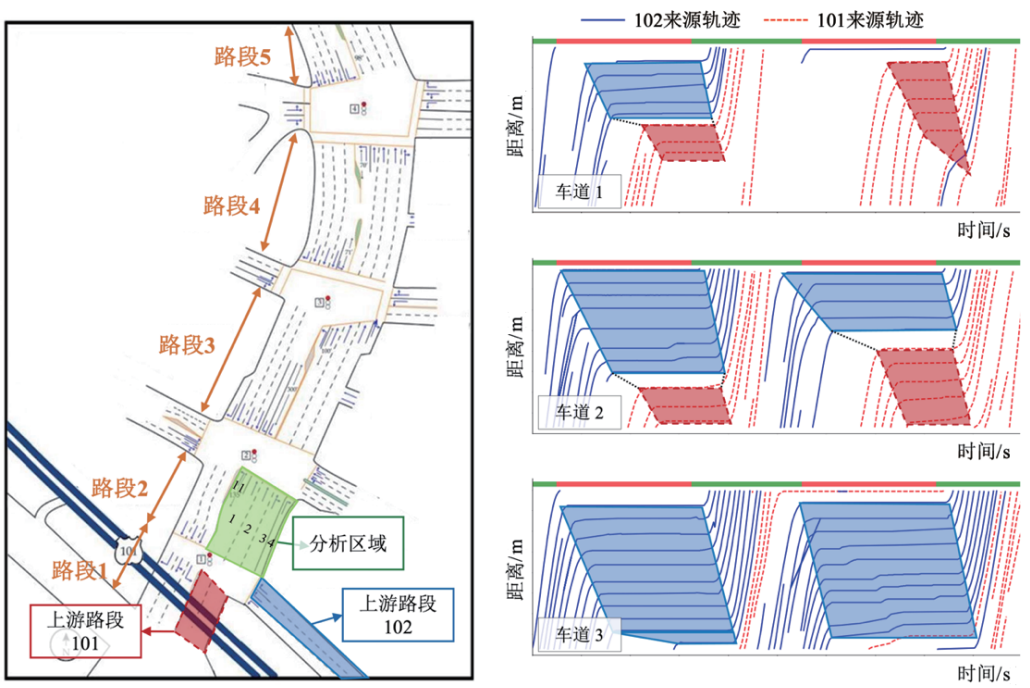

图8.9　NGSIM交通波可视化示例

得到处理好的数据如图 8.10所示。

	Vehicle_ID_new	Time_Second	Section_ID	Direction	Lane_ID	O_Zone	Global_X	Global_Y	v_Vel	geometry	project
12285	20	20.1	2	2	1	102	6451955.903	1872881.910	25.52	POINT (6451955.903 1872881.910)	252.704892
12286	20	20.2	2	2	1	102	6451956.372	1872884.382	25.25	POINT (6451956.372 1872884.382)	250.260016
12287	20	20.3	2	2	1	102	6451956.964	1872886.829	25.14	POINT (6451956.964 1872886.829)	247.787078
12288	20	20.4	2	2	1	102	6451957.628	1872889.267	25.37	POINT (6451957.628 1872889.267)	245.292584
12289	20	20.5	2	2	1	102	6451958.314	1872891.637	26.10	POINT (6451958.314 1872891.637)	242.850922

图8.10　NGSIM处理后数据实例

步骤2：绘制车辆时空轨迹。

首先我们构建一个字典vehicle_dict，其键值对分别表示车辆来源与车辆ID。在本案例中，来源为两条路段：101、102。相关代码如下所示：

```
# 来源路段列表
```