**深圳68路公交客流量分析报告**

何梓维

**一、加载包**

#载入模块  
import os  
import csv   
import pandas as pd  
import numpy as np  
from sklearn.cluster import DBSCAN  
import matplotlib.pyplot as plt   
import math

**二、导入68路公交GPS数据**

### 提取68路的GPS数据以及数据预处理  
# 设置当前工作路径  
os.chdir('C:/Users/Administrator/Desktop/python深圳68路公交客流量分析')  
# 读取数据以及数据预处理  
data = pd.DataFrame()  
file\_list = os.listdir('gps') #设置需要读取文件的文件夹  
for file\_name in file\_list:  
 csv\_data = pd.read\_csv('gps/'+file\_name,delimiter=',',encoding='gbk') #对从9日到13日逐个文件读取  
 csv\_data.dropna() #去除任何有空值的行  
 csv\_data.drop\_duplicates(['业务时间', '卡片记录编码', '车牌号'], keep='first', inplace=True) # 去除在'业务时间','卡片记录编码','车牌号'上重复的行  
 data = pd.concat([data,csv\_data],axis=0) #合并所有文件数据  
print('所有线路刷卡记录数据预处理后的数据形状：',data.shape) #查看预处理后的数据形状  
#在预处理后的总GPS数据中取出68路的数据  
data\_68 = data.loc[(data['线路名称']=='68路'),:]  
print('68路的刷卡记录数据为：',data\_68)  
print('68路的刷卡记录数据形状：',data\_68.shape)

**任务结果：**

所有线路刷卡记录数据预处理后的数据形状： (53774, 6)

68路的刷卡记录数据为： 经度 纬度 业务时间 卡片记录编码 线路名称 车牌号

1111 114.021003 22.531967 2014/6/9 7:34 320830455 68路 粤BL7071

1112 114.021003 22.531967 2014/6/9 7:34 297671449 68路 粤BL7071

1113 114.021003 22.531967 2014/6/9 7:34 329849352 68路 粤BL7071

1114 113.940399 22.549150 2014/6/9 7:29 296173669 68路 粤BL7115

1115 113.940399 22.549150 2014/6/9 7:29 285082248 68路 粤BL7115

... ... ... ... ... ... ...

13927 114.062332 22.563600 2014/6/13 14:04 361384480 68路 粤BL7385

13928 114.054031 22.518291 2014/6/13 16:20 151016056 68路 粤BL7385

13929 114.066215 22.659215 2014/6/13 18:35 293681482 68路 粤BL7385

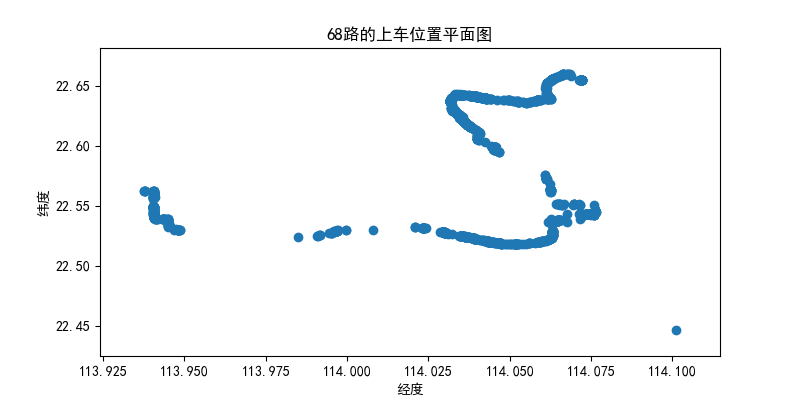
13930 114.061569 22.638833 2014/6/13 18:48 328786553 68路 粤BL7385

13931 114.042137 22.639516 2014/6/13 19:03 20482131 68路 粤BL7385

**三、绘制68路刷卡位置散点图**

### 绘制68路刷卡位置的散点图  
plt.rcParams['figure.figsize'] = (8.0, 6.0)  
plt.rcParams['font.sans-serif']=['SimHei'] #这行代码用于显示中文，'SimHei'就是黑体  
plt.figure(figsize=(8,4)) #画布大小  
plt.scatter(data\_68["经度"],data\_68["纬度"])  
plt.xlabel('经度')  
plt.ylabel('纬度')  
plt.title('68路的上车位置平面图')  
plt.savefig('68路的上车位置平面图.png')  
plt.show()

**任务结果：**



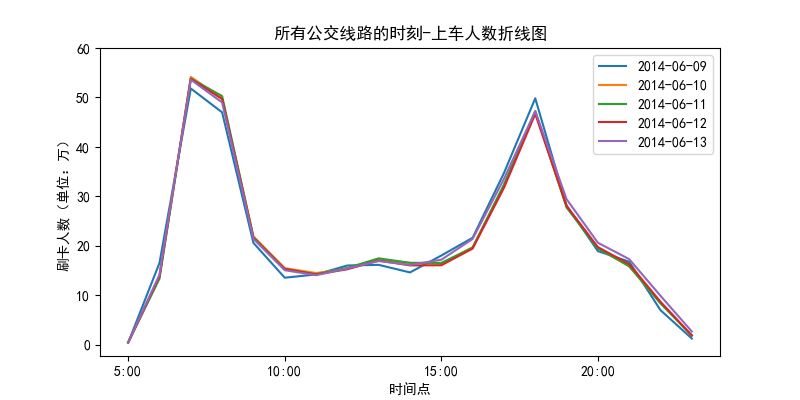
**结果分析：**

**由68路的上车位置平面图可以看到上车客流量集中在左中右三个部分，中间部分最多，左边部分最少。**

**四、绘制所有公交线路的时刻-上车人数折线图**

### 绘制所有公交线路的时刻-上车人数折线图  
file\_list = os.listdir('time') #设置需要读取文件的文件夹  
plt.figure(figsize=(8,4)) #画布大小  
for file\_name in file\_list:  
 data\_time = pd.read\_csv('time/'+file\_name) #对从9日到13日逐个文件读取  
 plt.plot(data\_time['date'],data\_time['num'],label=file\_name) #以一日内的时刻date为x轴，以上车人数num为y轴  
plt.xticks([5,10,15,20],['5:00','10:00','15:00','20:00']) #将时刻改为时间  
plt.xlabel('时间点')  
plt.yticks([0,100000,200000,300000 ,400000,500000,600000],['0','10','20','30','40','50','60']) #将人数改为以“万”单位  
plt.ylabel('刷卡人数（单位：万）')  
plt.title('所有公交线路的时刻-上车人数折线图')  
plt.legend(("2014-06-09","2014-06-10","2014-06-11","2014-06-12","2014-06-13")) #图例  
plt.savefig('所有公交线路的时刻-上车人数折线图.png')  
plt.show()

**任务结果：**



**结果分析：**

**由所有公交线路的时刻-上车人数折线图可以看出，一天内的客流量高峰集中在7点到8点和17点到18点两个时间段。符合心理预期，因为大部分公司的7点到8点是上班时间，17点到18点是下班时间。建议这两个时间段多发班车或疏导交通，提高运输能力。**

**五、对所有刷卡记录密度聚类，得到实际站点**

**完成思路：在DBSCAN聚类中，如果区域D中，点X的邻域N(X)内至少包含MinPts个点，则点X为D的核心点。我们对所有刷卡记录所在的经纬度进行DBSCAN聚类，将得到的核心点看作实际上存在的车站点。然后将非核心点、非边界点的噪声点除去。**

###密度聚类  
data\_68.reset\_index(drop=True,inplace = True) #由于后面需要横向合并，重新创建序列  
db = DBSCAN(eps=0.0011,min\_samples=3).fit(data\_68.iloc[:,0:2]) # DBSCAN聚类，半径为0.0011（度），3代表聚类中心的区域必须至少有3个才能聚成一类  
flag = pd.Series(db.labels\_, name=('flag')) #db.labels\_是聚类数据的簇标签,labels\_为-1为噪声点  
df\_cluster\_result = pd.concat([data\_68, flag], axis=1) #横向合并簇标签和原数据  
df\_cluster\_result = df\_cluster\_result[df\_cluster\_result["flag"] >= 0] #去掉噪声点  
print('68路的刷卡记录数据及其所在核心点flag为：',df\_cluster\_result) #flag值表示该乘客上车时所在站点的聚类核心点，一共有从0到40共41个核心点，将该flag值看作实际乘客上车站点。  
df\_cluster\_result.to\_csv('68\_cluster\_result.csv')  
#绘制密度聚类后的68路核心上车点的散点图  
df\_station = df\_cluster\_result.copy()  
df\_station = df\_station.drop\_duplicates("flag") # 去除重复值  
df\_station = df\_station.reset\_index(drop=False,inplace=False) #创建一个新序列，默认为drop=False,inplace=False  
plt.rcParams['figure.figsize'] = (8.0, 6.0)  
plt.rcParams['font.sans-serif']=['SimHei'] #这行代码用于显示中文，'SimHei'就是黑体  
plt.scatter(df\_station["经度"],df\_station["纬度"],c=df\_station["flag"])  
plt.xlabel('经度')  
plt.ylabel('纬度')  
plt.title('68路的上车位置DBSCAN聚类核心点平面图')  
plt.savefig('68路的上车位置DBSCAN聚类核心点平面图.png')  
plt.show()

**任务结果：**

68路的刷卡记录数据形状： (1649, 6)

68路的刷卡记录数据及其所在核心点flag为： 经度 纬度 业务时间 卡片记录编码 线路名称 车牌号 flag

0 114.021003 22.531967 2014/6/9 7:34 320830455 68路 粤BL7071 0

1 114.021003 22.531967 2014/6/9 7:34 297671449 68路 粤BL7071 0

2 114.021003 22.531967 2014/6/9 7:34 329849352 68路 粤BL7071 0

3 113.940399 22.549150 2014/6/9 7:29 296173669 68路 粤BL7115 1

4 113.940399 22.549150 2014/6/9 7:29 285082248 68路 粤BL7115 1

... ... ... ... ... ... ... ...

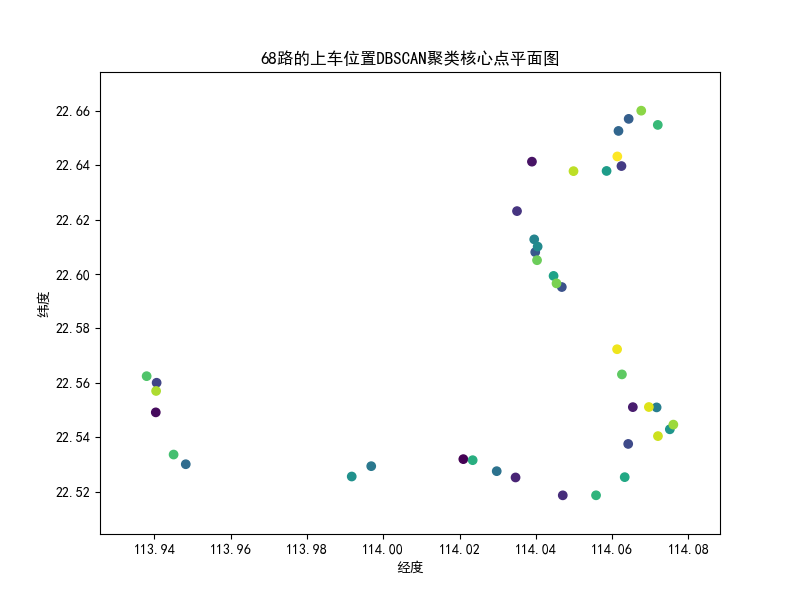
1644 114.062332 22.563600 2014/6/13 14:04 361384480 68路 粤BL7385 30

1645 114.054031 22.518291 2014/6/13 16:20 151016056 68路 粤BL7385 5

1646 114.066215 22.659215 2014/6/13 18:35 293681482 68路 粤BL7385 12

1647 114.061569 22.638833 2014/6/13 18:48 328786553 68路 粤BL7385 7

1648 114.042137 22.639516 2014/6/13 19:03 20482131 68路 粤BL7385 2



**六、计算每个站点的上车人数**

###计算每个站点的上车人数，将df\_cluster\_result中的flag值看作实际乘客上车站点  
get\_on\_num = df\_cluster\_result.groupby('flag').count().iloc[:,1] #以flag值分组并计算每组记录的个数，这就是该flag的上车人数  
get\_on\_num.name = ('get\_on\_num') #将列名改为get\_on\_num  
get\_on\_num = get\_on\_num.reset\_index()  
print('各个站点的上车人数为：',get\_on\_num)

**任务结果：**

各个站点的上车人数为： flag get\_on\_num

0 0 3

1 1 108

2 2 257

3 3 13

... ...

38 38 5

39 39 10

40 40 6

**七、OD矩阵与估计下车人数**

**完成思路：**

**1.计算每个站点对乘客的吸引权重（第j个站点的吸引权重=该站上车人数/总上车人数），**

**2.下车概率的泊松部分，（），其中，为公交线路途经站点数的数学期望，为i站上车途经j-i站下车的概率。**

**3.下车概率，，其中，为i站上车的乘客在j站下车的概率，为泊松部分，为j站的吸引权重。**

**4.OD矩阵，i行j列的值表示“从i站上车到j站下车”的人数,最后一列是该站上车总人数，最后一行是该站下车总人数。比如，第二行最后一列的值表示站点1的上车总人数。**

###下面我们尝试估计出每个站点的下车人数  
#计算每个站点对乘客的吸引权重（吸引权重=该站上车人数/总上车人数）。由于多人上车的点自然也会多人下车，所以吸引权重越大，意味着乘客下车的概率越大。  
attraction\_weight = get\_on\_num['get\_on\_num']/sum(get\_on\_num['get\_on\_num'])  
attraction\_weight = pd.DataFrame({'weight':attraction\_weight.values,'flag':attraction\_weight.index}) #将attraction\_weight由series类型转换为dataframe类型  
print('各个站点的吸引权重为：',attraction\_weight)  
attraction\_weight.to\_csv('attraction\_weight.csv')  
  
#构建下车概率矩阵，i行j列表示从i站上车的乘客在j站下车的概率  
#当不受权重影响时，自乘客上车起，乘客经过几站下车服从泊松分布，分布均值为，68路公交出行途经的站点数的数学期望  
flag\_num = len(attraction\_weight['flag']) #站点数  
lam = sum(attraction\_weight['flag'])/flag\_num #68路公交出行途经的站点数的数学期望  
get\_off\_probability = pd.DataFrame(np.zeros((flag\_num,flag\_num+1))) #构建下车概率矩阵，i行j列表示从i站上车的乘客在j站下车的泊松部分概率.第flag\_num+1列将为后面for循环计算i站上车乘客在所有站下车概率的总和起作用。  
for i in range(flag\_num):  
 for j in range(flag\_num):  
 if(i<j): #i<j意味上车站点一定在下车站点之前  
 get\_off\_probability.iloc[i,j] = (math.e\*\*(-lam)\*lam\*\*(j-i))/math.factorial(j-i) #从i站上车的乘客在j站下车的泊松部分  
 get\_off\_probability.iloc[i,flag\_num] = sum(get\_off\_probability.iloc[i,0:flag\_num]\*attraction\_weight.iloc[:,0]) #第flag\_num+1列为从i站上车到所有站点的泊松概率乘以吸引权重的总和  
#下车概率矩阵补上吸引权重部分  
for i in range(flag\_num):  
 for j in range(flag\_num):  
 if(get\_off\_probability.iloc[i,flag\_num]!=0):  
 get\_off\_probability.iloc[i, j] =(get\_off\_probability.iloc[i, j] \* attraction\_weight.iloc[j, 0])/get\_off\_probability.iloc[i,flag\_num] #加上吸引权重后，i行j列表示从i站上车的乘客在j站下车的概率  
print('下车概率矩阵：',get\_off\_probability)  
get\_off\_probability.to\_csv('get\_off\_probability.csv')  
  
#构建OD矩阵,求出一个站点到另一个站点的下车人数。OD矩阵i行j列表示从i站上车到j站下车的人数,最后一列是该站上车总人数，最后一行是该站下车总人数  
OD = pd.DataFrame(np.zeros((flag\_num+1,flag\_num+1)))  
for i in range(flag\_num):  
 for j in range(flag\_num):  
 if(i<j):  
 OD.iloc[i,j] = round(get\_on\_num.iloc[i,1]\*get\_off\_probability.iloc[i,j]) #从i站上车到j站下车的人数=从i站上车人数\*从i站到j站的下车概率  
for i in range(flag\_num):  
 OD.iloc[i,flag\_num] = sum(OD.iloc[i,0:flag\_num]) #从i站上车的总人数  
for j in range(flag\_num):  
 OD.iloc[flag\_num,j] = sum(OD.iloc[0:flag\_num,j]) #从j站下车的总人数  
OD.iloc[flag\_num,flag\_num] = sum(OD.iloc[flag\_num,0:flag\_num]) #总乘客数  
print('OD矩阵为：',OD)  
OD.to\_csv('68\_OD.csv')

**任务结果：**

各个站点的吸引权重为： weight flag

0 0.001839 0

1 0.066217 1

2 0.157572 2

3 0.007971 3

... ...

37 0.001839 37

38 0.003066 38

39 0.006131 39

40 0.003679 40

下车概率矩阵： 0 1 2 ... 39 40 41

0 0.0 1.625670e-07 3.868493e-06 ... 0.000020 0.000006 1.679105e-02

1 0.0 0.000000e+00 3.806436e-07 ... 0.000039 0.000012 1.706479e-02

2 0.0 0.000000e+00 0.000000e+00 ... 0.000073 0.000023 1.729885e-02

3 0.0 0.000000e+00 0.000000e+00 ... 0.000134 0.000044 1.739127e-02

... ...

37 0.0 0.000000e+00 0.000000e+00 ... 0.198020 0.792079 1.276373e-08

38 0.0 0.000000e+00 0.000000e+00 ... 0.142857 0.857143 1.769231e-09

39 0.0 0.000000e+00 0.000000e+00 ... 0.000000 1.000000 1.516483e-10

40 0.0 0.000000e+00 0.000000e+00 ... 0.000000 0.000000 0.000000e+00

[41 rows x 42 columns]

OD矩阵为： 0 1 2 3 4 5 ... 36 37 38 39 40 41

0 0.0 0.0 0.0 0.0 0.0 0.0 ... 0.0 0.0 0.0 0.0 0.0 1.0

1 0.0 0.0 0.0 0.0 0.0 0.0 ... 0.0 0.0 0.0 0.0 0.0 106.0

2 0.0 0.0 0.0 0.0 0.0 0.0 ... 0.0 0.0 0.0 0.0 0.0 260.0

... ...

37 0.0 0.0 0.0 0.0 0.0 0.0 ... 0.0 0.0 0.0 1.0 2.0 3.0

38 0.0 0.0 0.0 0.0 0.0 0.0 ... 0.0 0.0 0.0 1.0 4.0 5.0

39 0.0 0.0 0.0 0.0 0.0 0.0 ... 0.0 0.0 0.0 0.0 10.0 10.0

40 0.0 0.0 0.0 0.0 0.0 0.0 ... 0.0 0.0 0.0 0.0 0.0 0.0

41 0.0 0.0 0.0 0.0 0.0 0.0 ... 45.0 18.0 44.0 138.0 150.0 1603.0

**八、绘制各个站点的上下车人数柱状图**

###绘制各个站点的上下车人数柱状图  
#创建一个dataframe，将每个站点的估计下车人数和实际上车人数放一起  
get\_off\_num = []  
for j in range(flag\_num):  
 get\_off\_num.append(int(OD.iloc[flag\_num,j]))  
get\_off\_num = pd.DataFrame(get\_off\_num,columns=['get\_off\_num'])  
sum\_get\_num = pd.concat([get\_on\_num['get\_on\_num'], get\_off\_num], axis=1) #将下车人数与之前的上车人数get\_on\_num横向合并  
print('各个站点的上车人数和下车人数为：',sum\_get\_num)  
sum\_get\_num.to\_csv('sum\_get\_num.csv')  
#创建柱状图  
plt.rcParams['figure.figsize'] = (8.0, 6.0)  
plt.rcParams['font.sans-serif']=['SimHei'] #这行代码用于显示中文，'SimHei'就是黑体  
sum\_get\_num.plot(kind='bar',figsize=(12,4))  
plt.legend(['上车人数','下车人数'])  
plt.xlabel('站点')  
plt.ylabel('人数')  
plt.title('68路各个站点的上车人数和下车人数')  
plt.savefig('68路各个站点的上车人数和下车人数.png')  
plt.show()

**任务结果：**

各个站点的上车人数和下车人数为： get\_on\_num get\_off\_num

0 3 0

1 108 0

2 257 0

3 13 0

4 76 0

5 137 0

6 261 0

7 27 0

8 28 0

9 81 0

10 3 0

11 4 0

12 52 5

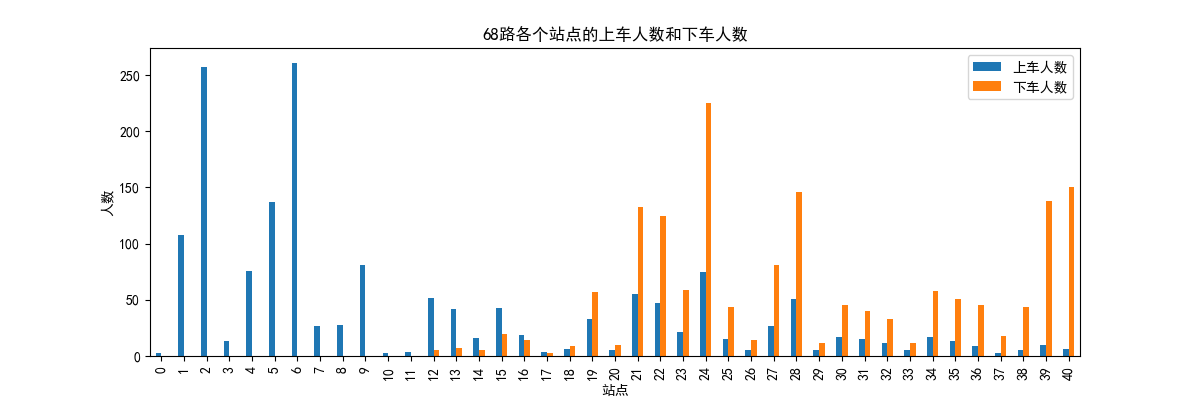
. . . . . .

37 3 18

38 5 44

39 10 138

40 6 150



**结果分析：**

**可能是刚开始发车的缘故，0到9站点上车人数很多，但下车人数很少；中间部分21到24站点、27、28站点都是高客流量站点，下车人数比上车人数要大；而到线路后面部分30到40站点时，上车人数减少，下车人数逐渐增多。最多人上车的是2站点和6站点，5天累计达250人，平均一天50人；最多人下车的站点是24站点，平均一天46人左右；客流量最小的站点是10和11站点，上下车都非常少。建议有关部门增开短线车，只停靠如2、5、6、21等高客流量站点，以提高公交运载能力。**