

交通运输数据技术 作业二

——地铁 IC 卡数据分析

2251140 范程

一、总述

本次作业与地铁 IC 卡相关的数据提取和分析关系紧密，在进行代码的编写和数据的处理与结果分析时，我采用了 R 语言作为工具，主要目标在于进一步熟悉 tidyverse 库的重要作用和了解 OD 矩阵的构建及其应用，主要分为：IC 卡数据预处理—站点客流量分析—OD 矩阵生成—两个站点之间平均乘车时长的计算等步骤。

以下是一些必要的分析和图表。

二、相关数据分析

（一）预处理

在本次任务中，IC 卡数据的预处理占到了很大的比重，从侧面体现了其是一个不容小觑的环节。的确，在地铁站内，乘客会出现如重复刷卡、短时间内刷进刷出等难以预测和估计的乘车行为，因此为了减小最后结果的误差，增强结果的合理性，我们需要对很多因素进行筛选。

首先需要导入一些必要的库，如 readxl、tidyverse、ggplot、stringr 等。

接着在读取 IC 卡的数据前要把文件中的列名改成英文来避免报错。

预处理大致有几个关键步骤，首先在构建完整的时间序列（年月日时分秒）并按照时间升序排序之后，需要删除异常交易记录：同一 ID 号在相近时间的交易类型都是进站或都是出站（在代码中通过 `type==21 or 22` 来区分进出站）的记录。

通过以下代码：

```
data1 <- data[order(data$ID,data$time),]
diff1 <- diff(data1$ID) #差分卡号（乘客）
diff2 <- diff(data1$type) #区分交易类型
diff3 <- data.frame(diff1,diff2) #合并差分值
```

```
diff4 <- data.frame(1,1)

colnames(diff4)[1] = "diff1"

colnames(diff4)[2] = "diff2" #创建 diff4,更改列名等待填充数据

diff5 <- rbind(diff3,diff4) #diff5 将 diff3&diff4 按行合并

data2 <- filter(data1,diff5$diff1 !=0 | diff5$diff2 !=0) #在 data1 中筛选正常乘车记录
```

data1	68472 obs. of 13 variables
data2	68357 obs. of 13 variables

最终过滤掉 115 条重复记录或异常记录。后面就排除这些数据的影响、针对一个或多个站点进行客流量分析了。

（二） 站点客流量分析

在完成相关数据整合、分类和排序之后，我们通过以下方法绘制进站客流最大的三个站的客流时变图：

```
data6 <- filter(data5, 交易类型 == 21) # 筛选出进站的人

p1 <- ggplot(data = data6, aes(x=时刻, y=人数, group = 站名, color = factor(站名))) +

  ggtitle("进站客流时变曲线") +

  xlab("Time") +

  ylab("Number") +

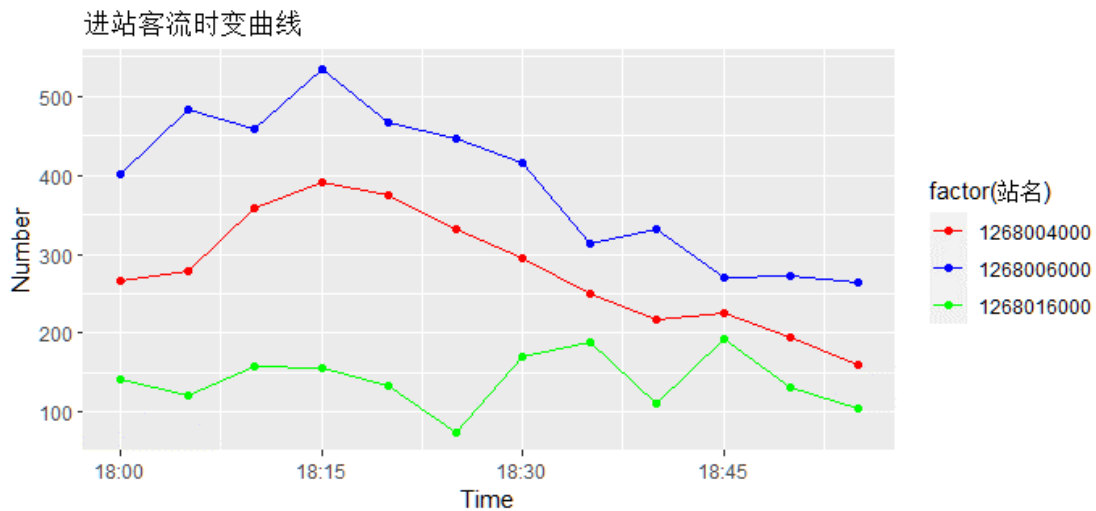
  scale_y_continuous(breaks = seq(0, 600, 100)) + #y 轴范围(0,600),间隔为 100

  geom_line() + #折线图

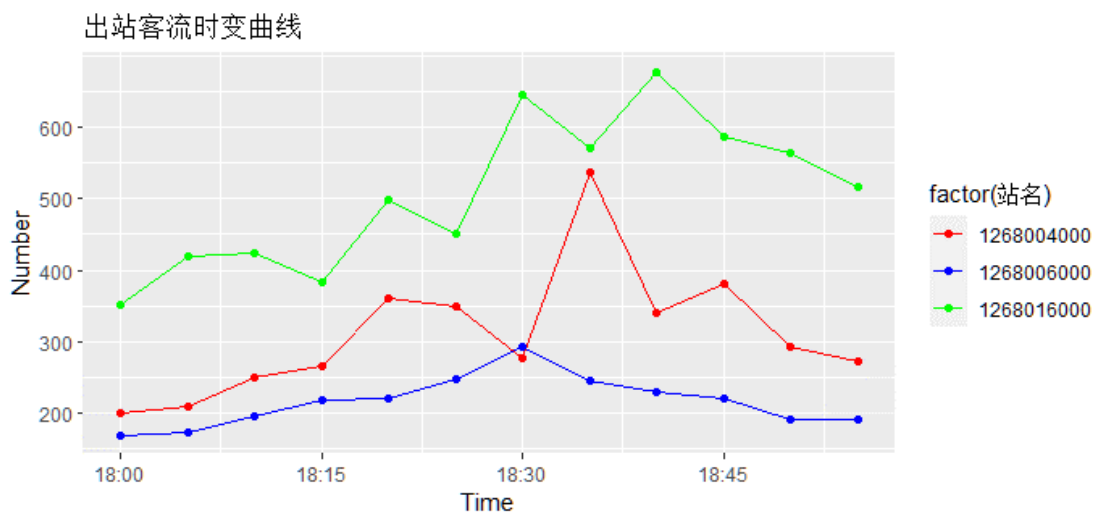
  geom_point() + #散点图

  scale_color_manual(values = c("red", "blue", "green"))

p1
```



同样地也有出站客流最多的三个站点的客流时变图：

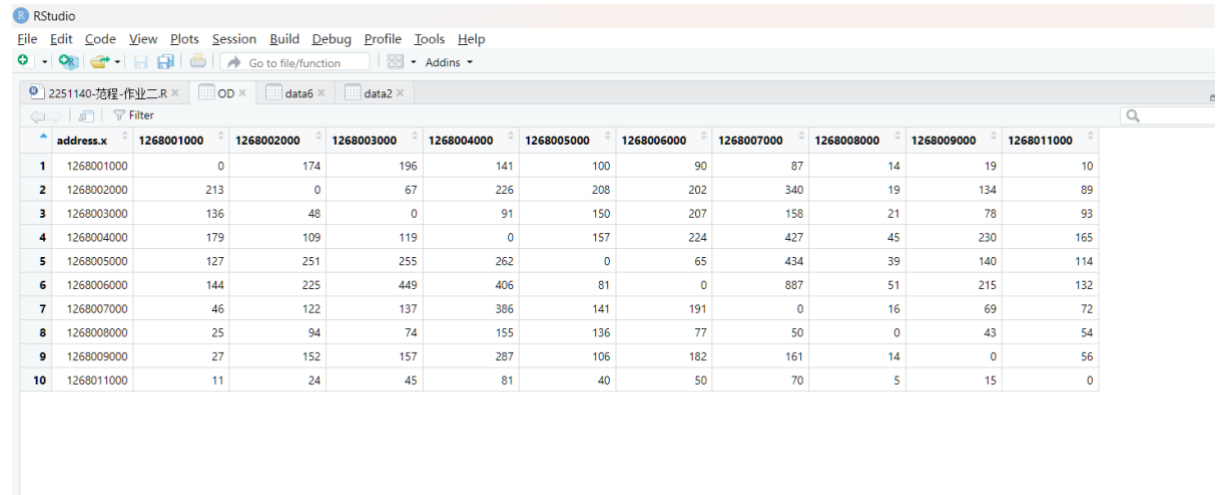


根据进出站的客流，我们可以看出，在工作日（数据来源 2008 年 12 月 1 日是周一）的下班晚高峰时期，编号为 1268016000（蓝）和 1268006000（红）的站点进站人数较多，说明这两个地方靠近工业区或者城市 CBD 等工作岗位较多的地区。而绿色折线代表的站点的进站人数在 18:40 之后开始有一个小幅度上升，可能是由于居民晚间外出导致的。同时，我们发现在 18:33 左右出站客流时变图中红色折线达到最大值，说明从其他地区汇入的客流在此出站，考虑可能是工业区进行夜班的交接班工作等情况

在经过地铁约 25~30 分钟的运转过后，绿色折线代表的 1268016000 所在的站点出站客流占三个站中的主导地位，体现出该站点应该是居民住宅区附近的站点。总体上，进站客流随着时间推移大体上是呈现波动减少的趋势，而进站客流与出站客流数在站点上大致具有此消彼长的互补性。

（三）站点 OD 矩阵的生成

OD 矩阵（Origin-Destination Matrix）可以体现从一个站点到其余站点的流向情况。为了构建一个 10 阶的 OD 方阵，我们选取了 1268001000~1268009000、1268011000 这十个站点来进行统计。首先通过 `filter()` 函数来匹配数据中的 `address`，再用 `inner_join()` 将同一乘客的进出站情况一一对应，再调用 `difftime()` 来过滤掉出站时间早于进站时间的异常记录，最后对



	1268001000	1268002000	1268003000	1268004000	1268005000	1268006000	1268007000	1268008000	1268009000	1268011000
1 1268001000	0	174	196	141	100	90	87	14	19	10
2 1268002000	213	0	67	226	208	202	340	19	134	89
3 1268003000	136	48	0	91	150	207	158	21	78	93
4 1268004000	179	109	119	0	157	224	427	45	230	165
5 1268005000	127	251	255	262	0	65	434	39	140	114
6 1268006000	144	225	449	406	81	0	887	51	215	132
7 1268007000	46	122	137	386	141	191	0	16	69	72
8 1268008000	25	94	74	155	136	77	50	0	43	54
9 1268009000	27	152	157	287	106	182	161	14	0	56
10 1268011000	11	24	45	81	40	50	70	5	15	0

数据进行排序、把同一 ID 的进站和出站在矩阵中对对应好、去除连续进出站的记录，就可以输出 OD 矩阵：

很明显，对角线上代表了同一站点进出站，这部分数据已经被去除，因此 $\text{diag}(\text{OG})=\text{O}$ 。

（四）两个站点间平均乘车时间的计算

对于进出站之间在两个轨道站点间平均的出行时长的计算，我们可以采取以下手段（以在 1268006000 进站和 1268007000 出站为例）：

```
st6 <- "1268006000"
st7 <- "1268007000"
records67 <- v2_data[(v2_data$address.x == st6 & v2_data$address.y == st7),]
intervals <- difftime(records67$ftime.y, records67$ftime.x, units = "mins")
aver_interval = mean(intervals)
aver_interval
```

首先，我们选取两个站点，并且在构建 OD 矩阵之前已经处理好的数据集 `v2_data` 中指定进站和出站的交易地址名称来筛选出这两个站点；然后还是继续调用 `difftime()` 函数，用出站时间 `records67$ftime.y` 减去进站时间 `records67$ftime.x` 并且以分钟为单位表示来得到所有

的从 st6 上车、st7 下车的事件记录，最后再用 `mean()`函数来求出这些数据的平均值，得到在这两个站点的平均乘车时间：

```
> aver_interval  
Time difference of 6.903326 mins
```

求得平均乘车时间约为 6.9 分钟，也就是 7 分钟左右。

三、反思与总结

在本次 IC 卡的数据分析中，我感受到了庞大的城市轨道交通系统的客流情况的一般统计方法。在代码编写上屡次使用到了在一个数据集中定位数据的方法也就是“`data$colname`”的格式，还有管道运算符`%>%`，这些都让我对 R 语言的语法有了更多的一些了解。

与此同时，我还意识到对交通数据进行分析的时候数据筛选是极为苛刻的工作。即使在第一步的数据预处理时已经剔除了很多异常数据，在构建 OD 矩阵或者计算平均乘车时长的时候还是要更进一步进行数据筛选，这体现出数据的庞大和需求的精确之间的矛盾，也要求我对过滤数据的方法如 `filter()`、`difftime()`等有更高的使用水平，这是在后续的数据分析时需要加强的。