# 车载客策略与乘车效率及收益均衡决策模型

## 摘要

本文对机场出租车载客策略与乘车效率提高，出租车收益均衡问题进行了探讨。建立了三个模型对应解决问题中的四个问题。

**模型一 机场出租车选择策略模型**

针对问题（1）（2）建立风险决策模型以到达机场的出租车司机为视角，以出租车司机当日收益最大为最高准则，转化为仅以当前机场乘客数量和蓄车池出租车数量为决策相关因素的决策问题。

对于问题（1），我们将决策背景分为中高低三个时段，然后比较某时段下对应选择放空或进入蓄车池等待的收益，使司机在特定时间段，且观测到确定排队人数和蓄车池出租车数的情况下，做出正确判断成为可能。

对于问题（2），我们收集到了国内某地某机场 24 小时内，按 1 小时为一阶段的飞机降落数量数据，平均每架飞机的载客数以及该区域出租车每次平均机场载客收费价格，该地出租车 1 小时天然气耗量，以及该地天然气价格标准。既运用 K 均值聚类分析算法又运用 C 均值聚类分析算法将一天内乘客人数变换情况分为七段，再利用 r 语言的 hist 命令归类为三个时段，得到高峰，低峰，中峰当日时段分布概率，结合以上数据，设定三种可能收益情形 $d_1,d_2,d_3$ 用决策模型具体给出选择方案。使用出租车司机决策算法，结合高峰，低峰，中峰三段人

数期望，再将该人数除以对应时间段长度得三个时段平均人流速度，当前蓄车池车数，人流速度及排队人数随机的情况下，输出多组“选择留在机场”或“选择回到市区”决策。利用 logistic 回归模型分析发现机场当前排队等待乘车人数和当前蓄车池出租车数量与结果高度相关，人流速度与结果相关性低，logistic 回归证明了选择策略模型的的合理性与适用性，同时说明模型的结果对机场乘客数量和蓄车池车辆数的依赖性高。

**模型二 出租车上车点高效乘车模型**

针对问题（3），该问题应该是一个资源配置问题，建立港湾式停靠站模型，经过讨论我们认为乘客排队上出租车的区域可以分为三部分“蓄车池”，“乘车区车道”，“乘车点”，工作人员主要负责，根据当前排队人数，决定一批蓄车池里的出租车从放入乘车车道的数量，同时，我们将“乘车区”分为主乘车区和辅乘车区，尽量安排乘客均匀分布在各个上车点。为了合理疏导乘客分别进入两边的主乘车区和辅乘车区，机场的人流疏导员需要在出租车乘车点入口对人流进行适当地疏导：在人流量较多时，引导部分乘客前往辅乘车点乘坐出租车；在乘客人数较少时可以选择关闭辅乘车点减少出租车等客时间，同时在出租车进场时需要通过 LED 指示牌告知司机辅站台的开放情况避免司机走错口。

## 模型三 机场出租车短途返程均衡收益模型

针对问题（4），机场的出租车载客收益与载客的行驶里程有关，但乘客的目的地有远有近，出租车司机不能选择乘客和拒载，对于遇上的乘客为短途乘客的司机这样显然不够公平，故需要减少短途载客再次返回的出租车二次排队载客的时间，使得这些出租车的收益尽量均衡，我们使用在现有乘车区基础上增加快速通道的方法（见示意图）。以机场车道设备记录车辆行驶出去到返回的时间差，判断出租车是否为短途返回车，以一轮出发到返回为例列式，当出租车载长途乘客一次不返回所得收益，与出租车载短途乘客一次并返回载第二批乘客所得收益相等，在收益尽量均衡的情况下得出 T 值。建立出租车短途返程优先配客模型，根据相关数据计算出设计短途返回时间差在 T 以内返回的车为短途车，并分析了结果如何。

## 问题重述

大多数乘客下飞机后要去市区（或周边）的目的地，出租车是主要的交通工具之一。送客到机场的出租车司机都将会面临前往到达区排队等待载客返回市区或者直接放空返回市区两个选择，出租车必须到指定的“蓄车池”排队等候，依“先来后到”排队进场载客，等待时间长短取决于排队出租车和乘客的数量多少，需要付出一定的时间成本。

而如果直接放空返回市区，拉客出租车司机会付出空载费用和可能损失潜在的载客收益。

需要解决下面问题：

（1）建立以机场乘客数量和当前蓄车池数量为相关因素的出租车选择决策模型。

（2）收集以上海浦东机场及其出租车的相关数据，结合相关数据完善模型，完成具有实际可行性的算法设计，验证决策模型合理性，以及该模型与机场乘客数量和当前蓄车池内车位数量的相关性。

（3）确定上车点个数及之间距离，合理设计乘车区尽量提高乘车效率。

（4）设计出可以使短途返回汽车减少排队等客时间的方案。

## 问题分析

本题具体要解决的问题可分为决策问题与统计分析问题两种。下面是对四个问题的具体分析：

问题一：为了简化问题，根据题意航班数量和“蓄车池”里已有车辆数为可观测确定信息，假设决策模型仅受机场乘客数量和当前蓄车池车辆数影响，建立相关决策机理方程组，比较当前两种选择情况下决策值的大小，根据比较结果司机可做出选择。

问题二：我们将一天机场乘客到达数量聚类为七个时间段的航班数量表示，归纳为三个时段，结合高峰，低峰，中峰三段人数期望，再将该人数除以对应时间段长度得不同时段平均人流速度，当前蓄车池车数，人流速度及排队人数随机的情况下，输出多组“选择留在机场”或“选择回到市区”决策。利用 logistic 回归模型分析和卡方检验，发现机场当前等待乘车人数和当前出租车数量与模型高度相关，人流速度相关性低以证明选择策略模型的的合理性，以及对机场乘客数量和蓄车池车辆数的依赖性高。

问题三：该问题应该是一个资源配置问题，建立港湾式停靠站模型，经过讨论我们认为乘客排队上出租车的区域可以分为三部分“蓄车池”，“乘车区车道”，“乘车点”，并确定具体尺寸（示意图）。工作人员主要负责，根据当前排队人数，决定一批蓄车池里的出租车从放入乘车车道的数量，同时，我们将“乘车区”分为主乘车区和辅乘车区，尽量安排乘客均匀分布在各个上车点。

问题四：减少短途载客再次返回的出租车二次排队载客的时间，

使得这些出租车的收益尽量均衡，我们使用在现有乘车区基础上增加快速通道的方法（见示意图）。对于判断出租车是否为短途返回车的判断临界时间 T，以一轮出发到返回为例列式，当出租车载长途乘客一次不返回所得收益，假设与出租车载短途乘客一次并返回载第二批乘客所得收益相等，在收益尽量均衡的情况下得出 T 值。

## 模型假设

(1) 假设出租司机仅根据当前排队人数与前方已在排队车辆数来进行选择；

(2) 假设机场到达旅客优先选择出租车作为交通工具；

(3) 假设所查数据真实有效；

(4) 假设出租车司机直接放空返回市区必然会产生损失且损失为定量。

(5) 假设当日与机场无关的收益情况不会影响出租车司机选择策略。

(6) 假设每次载客人数为平均数 1.5 人。

(7) 假设出租车到达机场时，看见的机场乘客仅为当前时段下飞机乘客，上一时段没有等车乘客剩下。

(8) 行车消耗以燃烧天然气地区为例

(9) 假设短途汽车返回后载长途乘客与短途乘客的概率相同

## 符号规定

Q:收益值

S:平均收入（未扣除油耗及其余成本）

s:单辆出租车乘客上车时间

P:平均每架飞机载客数

y:($y_1y_2y_3$) 排在前面的出租车数

t:($t_1t_2t_3$) 等待下一批乘客到来的平均时间

x:($x_1x_2x_3$)候车乘客数量

z=a*r 平均天然气价格 和 候车耗气速率

w=35.75 元/h 出租车返回市区后平均每小时收入

d*:最佳决策值

a:平均耗天然气速率

r:平均天然气价格

w:出租车返回市区后平均每小时收入 35.75 元/h

v:机场到达人数进入等车队列的速率

q:当前队伍中人数减去司机刚好能载到人时的人数

S:载一次长途乘客平均收入 120(元/小时)；

T:司机短途载客往返时间(小时)；

t:司机载一次长途乘客平均使用时间(小时)；

z:平均每小时气耗费 13.75(元)；

## 模型建立与求解

**模型一**

机场出租车选择策略模型的建立与解决

**问题一**

根据题目要求，选择为"放空"和“等待”两种，分别计算司机选择继续等待或放空两个方案的期望收益，然后从中选取期望值最大的方案为最优决策方案。

在期望收益准则下，有

$$d^* = \max\{\sum_j Q_{ij}\mathrm{P(i)}|1\leqslant \mathrm{i} \leqslant 3\}$$

**模型建立**

| | 高峰期$(P_1)$ | 一般期$(P_2)$ | 低峰期$(P_3)$ |
|---|---|---|---|
| 去蓄车池且无等候乘客时期$(a_1)$ $(\frac{3}{2}y_i \leqslant x_i)$ | $Q_{11}$= S-$y_1$sz | $Q_{21}$= S-$y_2$sz | $Q_{31}$= S-$y_3$sz |
| 去蓄车池但有等候时期$(a_2)$ $(\frac{3}{2}y_i > x_i)$ | $Q_{12}$ = S-$(y_1$s+$t_1)$z | $Q_{22}$ =S-$(y_2$s+$t_2)$z | $Q_{32}$ =S-$(y_3$s+$t_3)$z |
| 放空直接回市区$(a_3)$ | $Q_{13}$ = w$y_1$s 或者 w$(y_1$s+$t_1)$ | $Q_{23}$ = w$y_2$s 或者 w$(y_2$s+$t_2)$ | $Q_{33}$ = w$y_3$s 或者 w$(y_3$s+$t_3)$ |

根据模型假设，放空直接回市的间接收益情况有两种，抵消掉放

空回城路上的消耗，根据一天内机场到达人数变化设三个收益时段，高峰时段收益，低峰时段收益，一般期时段收益，同时司机选择去蓄车池后可能面临“不需等候乘客”和“需等候乘客”两种情况，而判断这两种情况需要设定临界条件，$q=xi-1.5yi+vsy_i$,当 q=0 时,即此时司机进入蓄车池刚好能载走该批队伍里最后一组乘客即达到临界条件。

若 q>=0 不需等候乘客（排队人数和“蓄车池”里已有的车辆数是司机可观测到的确定信息，司机可直接通过目测判断出，正在排队人数是否足以轮到自己，该情况下是可以），仅需考虑乘客上车所花时间为 sy，

$$1.5y_i<=x_i+vsy_i,\ Q_{i1}= S-y_jsz$$

将该情况下收益与放空的第一种可能收益进行比较，根据收益大小进行决策。

若 q<0,即司机需要等候乘客（排队人数和“蓄车池”里已有的车辆数是司机可观测到的确定信息，司机可直接通过目测判断出，正在排队人数是否足以轮到自己，该情况下司机需要在无人排队的蓄车区等候一段时间），需考虑等客时间 t ,$t=(1.5y-x-vsy_i)/v$

$$1.5y_i>x_i+vsy_i,\ Q_{i2} = S-(y_js+t_j)z$$

将该情况下收益与放空的第二种可能受益进行比较，根据收益大小进行决策。

同时，根据天然气消耗计算方法得成本为 z=ar。

最终建立出租车选择决策模型,

$$d^* = \max\{\sum_j Q_{ij}\mathrm{P(i)}|1\leqslant \mathrm{i} \leqslant 3\}$$

比较不同时间段上两种情况对应收益，选择收益较高的方式，司机即可在不同时间段做出选择收益最高选择。

**问题二**

收集国内某机场某天 24 小时按 1 小时为一阶段的飞机降落数量数据，平均每辆飞机载客人数以及江北区出租车每次平均机场载客收费价钱，平均每架飞机的载客数和出租车 1 小时天然气耗量，以及重庆天然气价格标准。

**运用 SPSS 进行 K 均值聚类：**

我们从某地的飞机场搜集到了某天到达航班的预期时刻表信息，其中有 24 个数据集，为了将其分为高峰，中峰，低峰三个分段，我们使用了 k 均值聚类和 c 均值聚类来将数据分为 7 个数据点，然后通过 r 语言的 hist()命令进一步将数据分为三个，即为低中高三峰。因为数据量太少了，且不够全面，而 k 均值聚类对数据本身有很高的要求，但是精度很高，故需要将结果与 c 均值聚类相比较，判断是否选择 k 均值聚类。（c 均值聚类对数据要求不高，聚类较为模糊）。考虑到还存在有航班取消的情况我们也收集到了实际的到达航班数，并对其进行了聚类比较。其中“1234”指的时从凌晨 1 点到 4 点，（2 到 4 点没有到达航班）。

根据结果，我们发现 k 均值聚类得到的结果与 c 均值聚类的结果差距不大，故说明二者的都数据比较合理，选择 k 均值聚类的结果来

进行后续的判断。K 均值聚类的点进行线性回归，并发现人数与时间并没有明显的相关性。下图是实际情况的 k 均值聚类的结果。

快速聚类

初始聚类中心

| | 聚类 | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 时间段 | 0 | 1234 | 5 | 6 | 7 | 22 | 16 |
| sum(G) | 645.84 | 149.04 | 99.36 | 249.04 | 50.04 | 430.56 | 331.20 |

迭代历史记录[a]

| | 聚类中心中的变动 | | | | | | |
|---|---|---|---|---|---|---|---|
| 迭代 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | .000 | .000 | .000 | 5.389 | .000 | 17.372 | 22.143 |
| 2 | .000 | .000 | .000 | .000 | .000 | .000 | .000 |

a. 由于聚类中心中不存在变动或者仅有小幅变动，因此实现了收敛。任何中心的最大绝对坐标变动为 .000。当前迭代为 2。初始中心之间的最小距离为 49.361。

最终聚类中心

| | 聚类 | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 时间段 | 0 | 1234 | 5 | 11 | 7 | 17 | 14 |
| sum(G) | 645.84 | 149.04 | 99.36 | 249.81 | 50.04 | 414.00 | 353.28 |

每个聚类中的个案数目

| | | |
|---|---|---|
| 聚类 | 1 | 1.000 |
| | 2 | 1.000 |
| | 3 | 1.000 |
| | 4 | 3.000 |
| | 5 | 1.000 |
| | 6 | 8.000 |
| | 7 | 6.000 |
| 有效 | | 21.000 |
| 缺失 | | 3.000 |

下图是计划人数的 k 均值聚类结果

初始聚类中心

| | 聚类 | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 时间段 | 0 | 1234 | 23 | 21 | 7 | 8 | 22 |
| 坐出租车人数 | 646.00 | 149.00 | 878.00 | 1822.00 | 199.00 | 497.00 | 1441.00 |

迭代历史记录[a]

| | 聚类中心中的变动 | | | | | | |
|---|---|---|---|---|---|---|---|
| 迭代 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 55.392 | .000 | 80.521 | .000 | 82.673 | .000 | 16.530 |
| 2 | 10.263 | .000 | 5.179 | .000 | .000 | .000 | .000 |
| 3 | 10.981 | .000 | 5.963 | .000 | .000 | .000 | .000 |
| 4 | .000 | .000 | .000 | .000 | .000 | .000 | .000 |

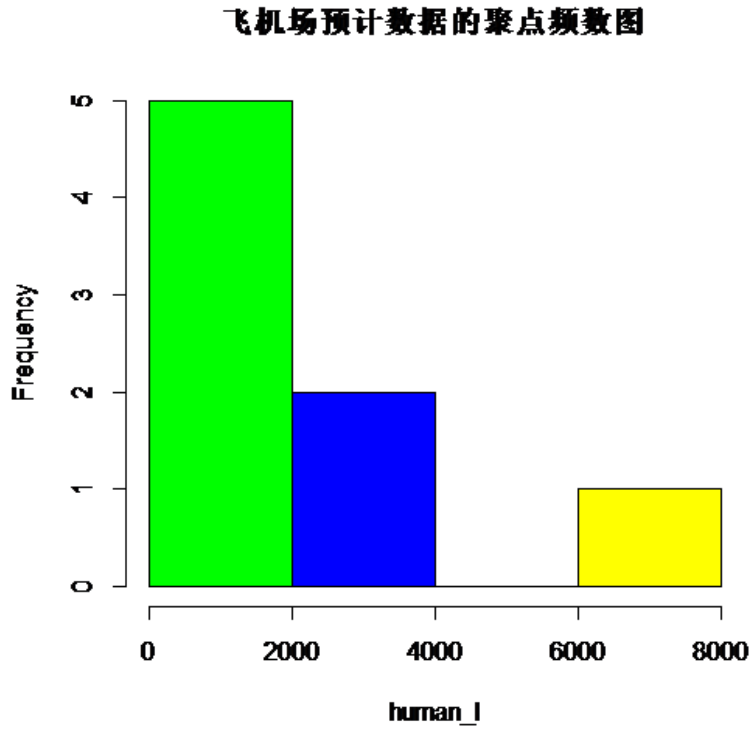
a. 由于聚类中心中不存在变动或者仅有小幅变动，因此实现了收敛。任何中心的最大绝对坐标变动为 .000，当前迭代为 4。初始中心之间的最小距离为 149.215。

最终聚类中心

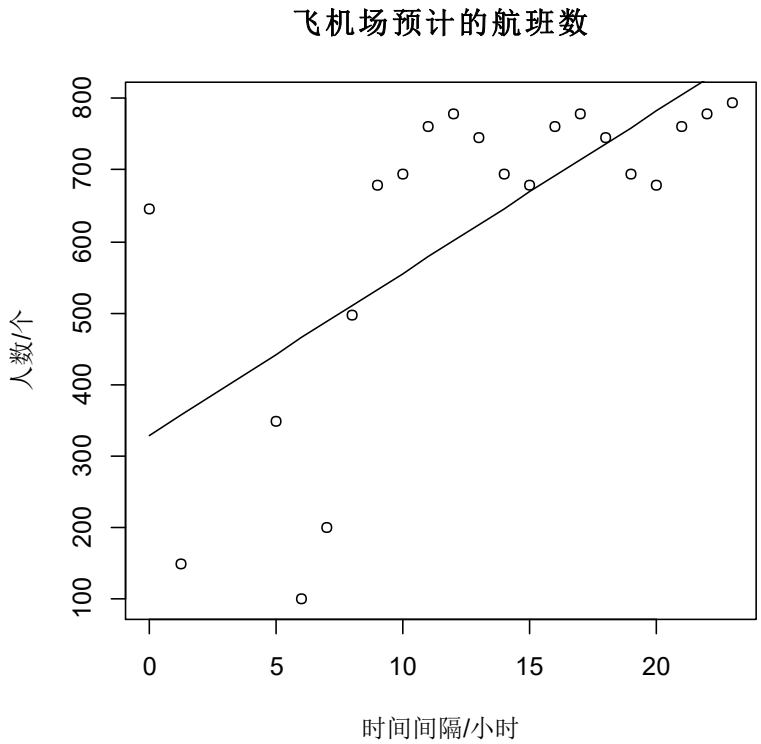
| | 聚类 | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 时间段 | 10 | 1234 | 16 | 21 | 6 | 8 | 21 |
| 坐出租车人数 | 679.20 | 149.00 | 786.63 | 1822.00 | 281.67 | 497.00 | 1457.50 |

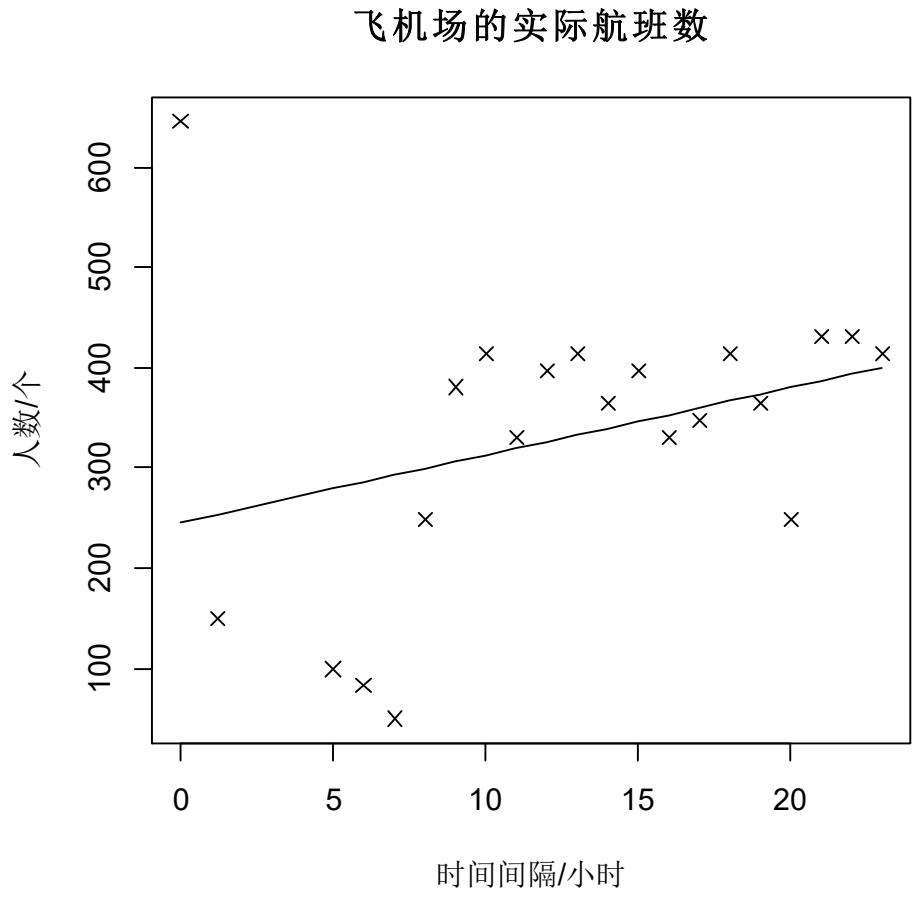
每个聚类中的个案数目

| | | |
|---|---|---|
| 聚类 | 1 | 5.000 |
| | 2 | 1.000 |
| | 3 | 8.000 |
| | 4 | 1.000 |
| | 5 | 3.000 |
| | 6 | 1.000 |
| | 7 | 2.000 |
| 有效 | | 21.000 |
| 缺失 | | 3.000 |




绿色是指低峰，蓝色是指中峰，黄色是指高峰。






以上两个散点图展示的是一天内预计的人数和实际人数的散点图。

在分析中中，因为实际人数和预计的人数相差不大，同时实际的人数仅代表一天的数据，具有临时性，故在误差允许的范围中，我们将飞机场计划的数据来作为基础数据。运用 C 均值聚类分析算法和 K 均值聚类分析算法将一天内乘客人数变换情况分为七段，再聚类为三个时间区，将各区域频数除以总频数作为乘客在各个时间区出现的概率，在结合该机场与该地区出租车相关数据，用决策模型具体给出选择方案。（由于 C 均值聚类分析算法和 K 均值聚类分析算法得出结果较为相近，说明使用模糊聚类与使用精准聚类结果相似，故以精准聚类方法即 k 均值聚类法为主。）

$$d^* = \max\{\sum_j Q_{ij} \mathrm{P(i)} | 1 \leqslant \mathrm{i} \leqslant 3\}$$

$d_1$=120-0.0327($y_1$+$y_2$+4$y_3$)

$d_2$=120-(0.0329$y_1$+0.0367$y_2$+0.2423$y_3$) +0.0023$x_1$+0.0027$x_2$+0.0526$x_3$

$d_3$=0.0851$y_1$+0.0851$y_2$+0.4255$y_3$

或者 0.0911$y_1$+0.0956$y_2$+0.5044$y_3$-0.0040$x_1$-0.0140$x_2$-0.1096$x_3$

$d_1$表示选择“等待”且去蓄车池时无等候乘客时期的收益；

$d_2$表示选择“等待”去蓄车池但有等候时期的收益；

$d_3$表示选择“放空”相对此时选择等待且面临无等候乘客时期的收益，或者表示选择放空相对此时选择等待且面临有等候时期的收益；

针对司机具体观测到的 y 和 x 即可判断 d1,d2 和 d3，即完成：“等待”和“放空”的最终的选择。

## 分析模型

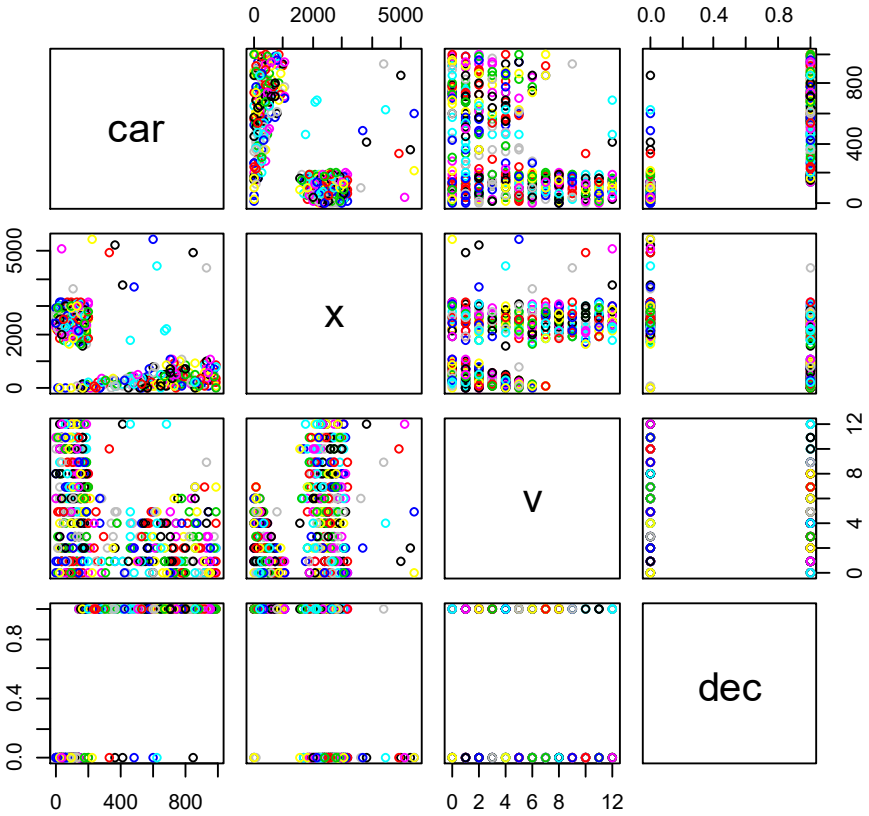
再利用 logistic 模型分析在不同时段下，机场出租车选择策略模型的的合理性，以及对机场乘客数量和蓄车池车辆数的依赖性。

Logistic 回归二值型通过使用其固有的 logistic 函数估计概率，来衡量因变量（我们想要预测的标签）与一个或多个自变量（特征）之间的关系。概率必须二值化才能真地进行预测。这就是 logistic 函数的任务。

我们通过 logistic 回归二值型检验结果的合理性得知，逻辑判断值仅和排在前面的出租车数量以及正在等待的乘客数量呈明显相关，但同时实时人流相关性不明显，因此我们使用的模型控制结果的量与

排在前面的出租车数以及正在排队的乘客数量相关。此外，我们使用的 474 个数据是通过使用两个 python 脚本模拟生成的(分别是模拟数据 1 和模拟数据 2)。下图是这些数据的散点图矩阵

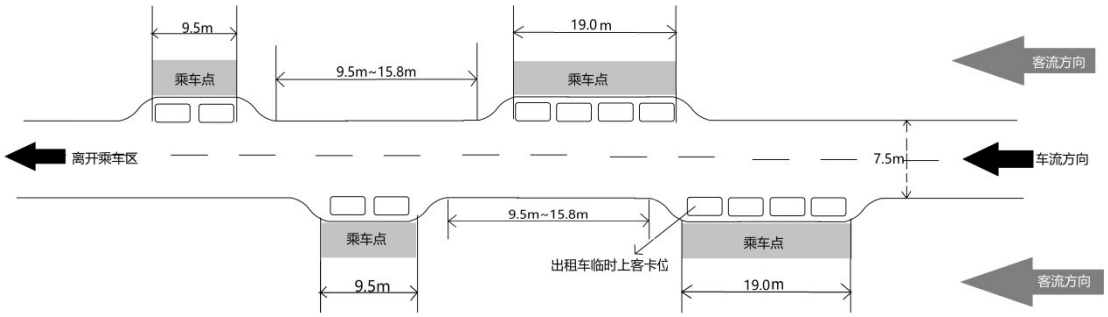
**模拟数据的散点图矩阵**




我们使用 r 语言来进行 logistic 回归，并对得到的结果进行了分析研判。第一次回归中，我们发现决策结果和当时排队人数和车数显著相关，与当时的客流量相关性不显著，在除去客流量后再进行回归后，用卡方检验来衡量两次回归结果的差异，发现客流量对模型的结果贡献度低（p<0.5）。

**问题三**

经过讨论我们认为乘客排队上出租车的区域可以分为三部分“蓄车池”，“乘车区车道”，“上车点”，工作人员主要负责，根据当前排队人数，决定每一批从蓄车池里放入乘车车道的出租车的数量，同时尽量安排乘客均匀分布在各个上车点。根据已有经验，首先设置四个上车点，且将靠左两个上车点定为辅助上车点，靠右两个上车点为主上车点，通常主上车点可暂时容纳的车辆个数大于辅助上车点，根据某地出租车实际长宽设计上车点大小，以在保障乘客与司机安全的情况下，尽量减少乘客和出租车在候车区的时间。




由于从蓄车池内出来的出租车数量庞大 ，为了提高总的乘车效率，在道路条件（双车道）允许的情况下，交叉口进口道可采取双港湾式公交停靠站布置形式，即对从蓄车池里出来的出租车给予一定的分组，在车道的一侧从空间上对公交停靠泊位横向拉开纵向拉开。港湾式停靠站是指在出租车停靠站处将道路适当地拓宽，将出租车的停靠位置设在正常行驶车道之外，以减少出租车停靠时形成的乘车区域内交通瓶颈对后到先走的出租车超车的影响，保证乘车区段内出租车流的正常运行。

由于考虑到我们有两条并行车道，不妨在车道两边各设置主辅两

个乘车点增大乘车效率，同时为了减少出租车进出站的困难及阻塞，我们采取左右两边站点交错排布的方式设置港湾式出租车停泊站。一般来说，纵向拉开的双港湾式出租车停靠站由两个普通的主辅港湾式停靠站组成。为了避免前后两个单港湾停靠站各自车辆停靠时发生冲突而造成出租车进出站的困难及阻塞，应当根据情况(停靠泊位数及出租车身长度等) 适当地设置主辅两个站台之间的间距，保证出租车顺利地停靠上客以及超车。但应当防止乘客在两站台之间步行的距离过大，而导致辅车站的乘客人数过少，主车站的人数过多。因此，根据出租车的尺寸和车道宽以及出租车出入行驶的安全距离，一般设置成 9.5m 最多不超过 15.8m。

为了使总的乘车效率最高，主港湾站台设置 4 个停靠泊位，辅港湾站台设 2 个停靠泊位。同时为了减少停靠泊位多的主站与停靠泊位相对较少的辅站之间进出站的相互干扰与影响，一般应当将辅站设在主站的前方，设置 2 个停靠泊位，而将主站设置在后方,设置 3 个或 4 个停靠泊位。根据出租车的尺寸和出入站的行车安全以及后到先走的出租车的超车需求，同时参考美国的《道路通行能力手册》及卡位的个数，主港湾站台卡位设计总长为 19.0m，辅港湾站台卡位设计总长为 9.5m，标准车道宽 7.5m。

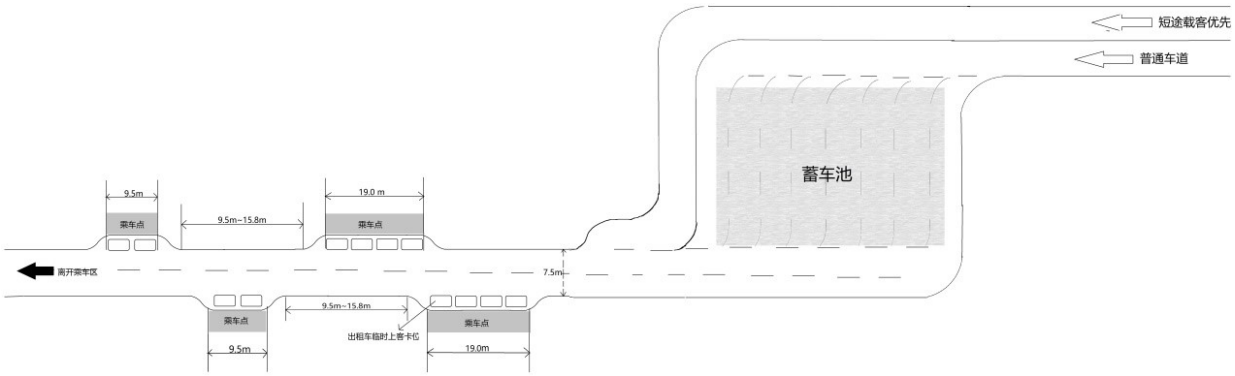
为保障出租车在港湾式停靠站能完全进入港湾停靠，港湾宽度通常与普通标准车道宽度相等，当然也可以略微小于车道宽度。鉴于出租车的车身宽度一般为 1.6m~1.8m，所以若乘车点的港湾若只设置一辆出租车的停车卡位，那么宽度至少要大于 2m 才能满足安全和

方便要求。设置这种港湾式的停靠站需要挤占人行道，因此,设置这样的港湾式乘车点需在用地充足的机场区设置。

此外，为了合理疏导乘客分别进入两边的主乘车区和辅乘车区，机场的人流疏导员需要在出租车乘车点入口对人流进行适当地疏导：在人流量较多时，引导部分乘客前往辅乘车点乘坐出租车；在乘客人数较少时可以选择关闭辅乘车点减少出租车等客时间，同时在出租车进场时需要通过 LED 指示牌告知司机辅站台的开放情况避免司机走错。

## 问题四

出租车短途返程优先配客措施示意图如下：




采用短途载客的司机可以进入快速通道的方案，短途的乘客不用担心会被出租车司机拒载，而出租车司机也不用担心排两三个个小时队却拉了一个短途乘客。机场车道设备快速识别车辆身份，或者短途出租车 GPS 项目，记录了车号和进出站点时间。比如 27 分钟后返回机场，就直接快速进入出租车短途专用道。此外，如果乘客人数过大，现在站点按“长途”和“短途”划分排队，调度员可以根据目的地的远近，告诉乘客应该排哪一队，不仅对驾驶员有利，也方便乘客。

以机场 A 和机场 B 为例，前提背景为出租车到达蓄车池后排队等候时间平均为 2~3 小时，正常长途返回的乘客需要 80km 及以上平均需要一个小时，出租车载到长途乘客或者短途乘客的概率几乎相等。使用第二问的价格和收入的相关数据：

假定司机短途载客后只返回机场一次：

S:　载一次长途乘客平均收入 120(元/小时)；

T：司机短途载客往返时间(小时)；

t：司机载一次长途乘客平均使用时间(小时)；

z: 平均每小时气耗费 13.75(元)；

$$S-\mathrm{z}=\frac{T(S-z)}{2}-\frac{Tz}{2}+\frac{1}{2}\frac{T(S-z)}{2}+\frac{1}{2}(S-z)$$

计算得 T 约为 40.0 分钟。

因此我们采用如下方案：若司机短途载客在一个小时内返回机场，机场方允许该出租车进入短途载客的快速通道，免去蓄车池内的排队。在此过程中，我们同时还采用并联式蓄车：出租车通过驶入区进入蓄车池后，蓄车池的行车路径分成多股并列。每条路径上的车辆首尾相继，每条路径之间并排行驶依次进入上客区，在四处主辅乘客候车区前的港湾式卡位内停车上客，然后通过驶出区离开机场。由于将车辆消解为多条少量车行流线，受到场地约束比单条线的情况小，遇到突发的状况时机动性比较强，较大地缩短出租车辆排队路径，减少蓄车池内的尾气排放量。这种排队方式相比于单条线排队更加灵活，有利于出租车根据具体乘客数量的多少来调整接客时间。

## 模型评价与改进

**模型一：**

**优点：**清楚地说明了当前排队人数与蓄车池车辆数与决策之间的关系，使司机就具体情况进行正确判断成为可能。使用了 Logistic 回归分析模型，Logistic 回归优点之一是它非常高效，不需要太大的计算量，同时又通俗易懂，不需要缩放输入特征，不需要任何调整，且很容易调整，并且输出校准好的预测概率。与线性回归一样，当你去掉与输出变量无关的属性以及相似度高的属性时，logistic 回归效果确实会更好。因此特征处理在 Logistic 和线性回归的性能方面起着重要的作用。Logistic 回归的另一个优点是它非常容易实现，且训练起来很高效。在研究中，我们通常以 Logistic 回归模型作为基准，再尝试使用更复杂的算法。但它的一个缺点就是不能用来解决非线性问题，因为它的决策面是线性的。

**缺点：**缺乏对未来预判的能力，对节假日等特殊情况等的预测，面对特殊数据，决策正确性减弱。

**模型二：**

**优点：**总体上讲，港湾式出租车停靠站与非港湾式出租车停靠站相比，在机场乘车区来说，相对提高了乘车效率，同时可以缓解停靠站的交通阻塞问题；同时可以减少对乘车区外的交通干扰，尤其对机场人数高峰时期更有成效，不仅能够保证路段出租车车流的正常运行，还能使总的乘车效率最高，间接地减少车辆延误时间。

**缺点：**尽管港湾式公交停靠站有许多优点，但仍存在一些问题:

(1) 港湾式停靠站的经济投资较；

(2) 港湾式停靠站对道路的要求比较高，需要占用非机动车道与人行道的港湾式停靠站，人行道较窄时，停靠站处的行人与非机动车较拥挤，可能需要拓宽道路断面；

**模型三：**

**优点：**并联式蓄车是大型城市高铁、机场枢纽的典型蓄车模式。这种蓄车方式的优点有很多：一是节约利用场地，二是可根据出租车多少及车的状况随时调节，三则是大大缩短了车辆在蓄车场内的行驶距离，节约行驶能耗的同时，也可减少尾气排放量。

**缺点：**该设计较为理想化，未具体考虑短途车返程后，二次载客对象为长途或短途的概率，以及工组人员一次要从蓄车池中放出出租车的数目。

# 参考文献

【1】Robert I.Kabacoff 著 R 语言实战 人民邮电出版社 2013

【2】中国知网 浅谈港湾式公交停靠站的设置 http://xuewen.cnki.net/CJFD-JTYH200602013.html 《交通与运输(学术版)》2006 年 02 期

【3】中国知网 林路 城市地下综合交通枢纽中出租车蓄车场流线设计 http://www.wanfangdata.com.cn/search/searchList.do?searchType=all&showType=&pageSize=&searchWord=城市地下综合交通枢纽中出租车蓄车场流线设计&isTriggerTag=

【4】《城市建筑》规划·设计 2016 年 06 月 20 日 任福田, 等译.《道路通行能力手册》( HCM ). 北京: 中国建筑工业出版社，1991

# 附录

**1. R 语言 画图（散点图，直方图）**

```
time<-c(0,1234,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23)
>
plane<-c(13,3,7,6,12,30,41,42,46,47,45,42,41,46,47,45,42,41,46,47,48,48,89,110,87,53)
> human<-c(plane[1:3]*0.8*0.45*138,plane[4:21]*0.15*0.8*138)
>  plot(time,human,xlab="时间间隔/小时",ylab="人数/个",main="飞机场预计的航班数")
> time<-c(0,1.234,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23)
>  plot(time,human,xlab="时间间隔/小时",ylab="人数/个",main="飞机场预计的航班数")
```

**2. R 语言 C 均值聚类算法**

```
FCM <- function(x, K, mybeta = 2, nstart = 1, iter_max = 100, eps = 1e-06)
{

    FCM_onetime <- function(x, init_centers, mybeta = 2, iter_max = 100, eps = 1e-06) {
        n = dim(x)[1]
        d = dim(x)[2]
        g = init_centers
```

```
K = dim(g)[1]
histJ = c()
pasfini = 1
Jold = Inf
D = matrix(0, n, K)
for (j in 1:K) {
        D[, j] = rowSums(sweep(x, 2, g[j, ], "-")^2)
}
iter = 1
J_old = Inf
while (pasfini) {
        s = (1/(D + eps))^(1/(mybeta - 1))
        u = s/(s %*% matrix(1, K, K))
        t1 = t(u^mybeta) %*% x
        t2 = t(u^mybeta) %*% matrix(1, n, d)
        V = t1/t2
        g = V
        D = matrix(0, n, K)
        for (j in 1:K) {
                D[, j] = rowSums(sweep(x, 2, g[j, ], "-")^2)
        }
        J = sum(u^mybeta * D)
```

```
            pasfini = abs(J - Jold) > 0.001 && (iter < iter_max)
            Jold = J
            histJ = c(histJ, J)
            iter = iter + 1
        }
        cluster_id = apply(u, 1, which.max)
        re = list(u, J, histJ, g, cluster_id)
        names(re) = c("u", "J", "histJ", "g", "cluster_id")
        return(re)
    }
    x = as.matrix(x)
    seeds = 1:nrow(x)
    id = sample(seeds, K)
    g = as.matrix(x[id, ])
    re_best = FCM_onetime(x = x, init_centers = g, mybeta = mybeta,
iter_max = iter_max, eps = eps)
    if (nstart > 1) {
        minJ = 0
        i = 2
        while (i <= nstart) {
            init_centers_id = sample(seeds, K)
            init_centers = as.matrix(x[init_centers_id, ])
```

```
                run = FCM_onetime(x, init_centers = init_centers, mybeta
= mybeta, iter_max = iter_max)
                if (runJ<=rebestJ) {
                    re_best = run
                }
                i = i + 1
            }
        }
        return(re_best)
```

在使用时，调用格式：FCM(data,K=聚点个数)

**3. R 语言 C 均值算法输出结果**

**对实际数据的操作**

```
>    time<-c(0,1234,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23)
#各个时刻
>plane<-c(13,3,2,5,3,15,23,25,20,24,25,22,24,20,21,25,22,15,26,26,25)
#各个时刻飞机数量
>  pe<-c(plane[1:3]*0.8*0.45*138,plane[4:21]*0.8*0.15*138)          #计
算各个时段的人数分布
> f<-data.frame(time,pe)
> a<-FCM(f,K=7)                        #进行 c 均值聚类分析
> round(a$g)
```

```
      time    pe
[1,]    6    74
[2,]   14 372
[3,]    0 646
[4,]   17 417
[5,]   11 248
[6,]   14 335
[7,] 1234 149
> a
$u
                  [,1]                [,2]                [,3]                [,4]
[,5]
 [1,] 1.189518e-09 5.174132e-09 1.000000e+00 7.362156e-09
2.460862e-09
 [2,] 8.169572e-13 8.039434e-13 6.987779e-13 7.966788e-13
8.216104e-13
 [3,] 9.448584e-01 8.211464e-03 2.045722e-03 6.061529e-03
2.746625e-02
 [4,] 8.977556e-04 1.781526e-03 1.729376e-04 9.614098e-04
9.925877e-01
 [5,] 9.668615e-01 5.490000e-03 1.605214e-03 4.234599e-03
1.444554e-02
```

[6,] 3.452889e-04 6.865428e-04 6.651087e-05 3.703525e-04 9.971417e-01

[7,] 9.755520e-04 8.802813e-01 1.307876e-03 6.839956e-02 5.232990e-03

[8,] 4.889516e-04 3.173080e-02 1.050255e-03 9.555185e-01 2.061148e-03

[9,] 4.168812e-04 1.650766e-02 2.784670e-04 3.764120e-03 4.022652e-03

[10,] 2.172155e-03 3.485125e-01 3.677120e-03 5.763566e-01 1.023224e-02

[11,] 2.097148e-04 1.372965e-02 4.500130e-04 9.807898e-01 8.842431e-04

[12,] 6.333432e-04 9.115302e-01 6.727175e-04 1.947826e-02 3.972709e-03

[13,] 2.094793e-03 3.378418e-01 3.543037e-03 5.896099e-01 9.866039e-03

[14,] 2.970634e-04 1.181274e-02 1.983854e-04 2.699056e-03 2.860323e-03

[15,] 1.619245e-03 2.042287e-01 1.364214e-03 2.565855e-02 1.226802e-02

[16,] 6.294305e-05 4.089757e-03 1.347884e-04 9.942670e-01 2.651998e-04

[17,] 8.700759e-04 8.804739e-01 9.233974e-04 2.686464e-02 5.443156e-03

[18,] 2.450864e-03 4.903850e-03 4.740565e-04 2.652572e-03 9.795821e-01

[19,] 1.497913e-03 5.482198e-02 4.079360e-03 9.127962e-01 5.732546e-03

[20,] 1.553498e-03 5.662657e-02 4.227879e-03 9.098237e-01 5.943054e-03

[21,] 3.341934e-04 2.097795e-02 7.138417e-04 9.703559e-01 1.405096e-03

| | [,6] | [,7] |
|---|---|---|
| [1,] | 4.027441e-09 | 2.198741e-10 |
| [2,] | 8.122605e-13 | 1.000000e+00 |
| [3,] | 1.095282e-02 | 4.038200e-04 |
| [4,] | 3.580632e-03 | 1.799941e-05 |
| [5,] | 6.986621e-03 | 3.764979e-04 |
| [6,] | 1.382646e-03 | 6.946162e-06 |
| [7,] | 4.374361e-02 | 5.913001e-05 |
| [8,] | 9.114317e-03 | 3.605404e-05 |
| [9,] | 9.749922e-01 | 1.805077e-05 |
| [10,] | 5.890315e-02 | 1.462272e-04 |
| [11,] | 3.921022e-03 | 1.554091e-05 |

[12,] 6.367797e-02 3.481988e-05

[13,] 5.690269e-02 1.417473e-04

[14,] 9.821195e-01 1.298086e-05

[15,] 7.547813e-01 7.996366e-05

[16,] 1.175575e-03 4.704940e-06

[17,] 8.537655e-02 4.827732e-05

[18,] 9.885943e-03 5.059320e-05

[19,] 2.094899e-02 1.230630e-04

[20,] 2.169746e-02 1.278606e-04

[21,] 6.187838e-03 2.520942e-05

$J

[1] 2644.482

$histJ

[1] 256392.443 245246.738 235906.404 214078.665 112054.741 19792.681

[7] 3821.324 3213.015 3120.213 2968.129 2832.632 2774.895

[13] 2749.422 2735.554 2727.038 2721.281 2717.041 2713.675

[19] 2710.821 2708.263 2705.855 2703.493 2701.090

2698.571

[25] 2695.864 2692.899 2689.611 2685.948 2681.880

2677.426

[31] 2672.665 2667.759 2662.939 2658.470 2654.589

2651.444

[37] 2649.066 2647.379 2646.249 2645.529 2645.088

2644.826

[43] 2644.675 2644.589 2644.540 2644.514 2644.499

2644.491

[49] 2644.487 2644.484 2644.483 2644.482

$g

time pe

[1,] 6 74

[2,] 14 372

[3,] 0 646

[4,] 17 417

[5,] 11 248

[6,] 14 335

[7,] 1234 149

$cluster_id

[1] 3 7 1 5 1 5 2 4 6 4 4 2 4 6 6 4 2 5 4 4 4

对预期数据的聚类：

```
>    time<-c(0,1234,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23)
#各个时刻
>
plane<-c(13,3,7,6,12,30,41,42,46,47,45,42,41,46,47,45,42,41,46,47,48,4
8,89,110,87,53)   #飞机数
> table_c <-data.frame(time,plane)
> table_c
      time plane
1        0       13
2    1234         3
3        5         7
4        6         6
5        7       12
6        8       30
7        9       41
8       10       42
9       11       46
10      12       47
11      13       45
```

12 14 42

13 15 41

14 16 46

15 17 47

16 18 48

17 19 48

18 20 89

19 21 110

20 22 87

21 23 53

>human<-c(plane[1:3]*0.8*0.45*138,plane[4:21]*0.15*0.8*138) # 计算各个时段的人数

> sum(human)

[1] 15715.44

> a<-FCM(table_c,K=7) #使用 c 均值聚类进行分析

> round(a$g)

time plane

[1,] 1234 3

[2,] 21 88

[3,] 12 42

[4,] 5 10

[5,] 1234 3

[6,] 21 110

[7,] 18 48 #输出分类的点

> a$g<-round(a$g) #对 a$g 取整

> a

$u

[,1] [,2] [,3] [,4] [,5]

[1,] 4.349923e-03 1.723114e-02 6.093418e-04 1.933556e-02 8.929370e-01

[2,] 1.098632e-07 1.055114e-07 9.999995e-01 8.267898e-08 9.303315e-08

[3,] 5.258232e-02 9.066794e-01 1.352644e-03 7.366410e-03 1.907858e-02

[4,] 9.560547e-01 2.588296e-02 1.506694e-03 3.723511e-03 6.767363e-03

[5,] 9.146999e-01 6.024534e-02 1.552812e-03 5.028146e-03 1.014094e-02

[6,] 5.220094e-02 6.221457e-01 3.909626e-03 4.296903e-02 1.908556e-01

[7,] 3.073450e-06 1.080183e-05 4.864057e-07 2.115508e-05 9.998688e-01

[8,] 8.491390e-04 2.832309e-03 1.416807e-04 7.375069e-03 9.478233e-01

[9,] 4.564311e-04 1.283249e-03 9.190254e-05 1.200629e-02 2.517180e-02

[10,] 6.119548e-05 1.660832e-04 1.287139e-05 2.287804e-03 2.469508e-03

[11,] 1.926801e-03 5.625594e-03 3.722842e-04 3.715559e-02 1.570552e-01

[12,] 8.987094e-04 2.996485e-03 1.507944e-04 7.826556e-03 9.445783e-01

[13,] 9.153281e-05 3.214833e-04 1.460973e-05 6.322059e-04 9.960793e-01

[14,] 4.165649e-04 1.170726e-03 8.444207e-05 1.103219e-02 2.285744e-02

[15,] 4.575422e-05 1.241287e-04 9.688093e-06 1.724527e-03 1.838192e-03

[16,] 9.273132e-04 2.434259e-03 2.047761e-04 5.237328e-02 2.882145e-02

[17,] 9.407368e-04 2.469251e-03 2.080191e-04 5.319782e-02 2.920310e-02

[18,] 3.490386e-03 5.278221e-03 1.900877e-03 1.744412e-02 9.712482e-03

[19,] 1.997201e-02 2.751288e-02 1.310838e-02 6.319759e-02 4.285097e-02

[20,] 6.907867e-03 1.057001e-02 3.680909e-03 3.682032e-02 1.989657e-02

[21,] 1.840405e-05 4.215627e-05 4.909073e-06 9.987506e-01 2.475377e-04

| | [,6] | [,7] |
|---|---|---|
| [1,] | 1.305383e-03 | 6.423169e-02 |
| [2,] | 4.803924e-08 | 8.820189e-08 |
| [3,] | 1.439424e-03 | 1.150118e-02 |
| [4,] | 1.074675e-03 | 4.990126e-03 |
| [5,] | 1.273767e-03 | 7.059135e-03 |
| [6,] | 5.674729e-03 | 8.224433e-02 |
| [7,] | 1.130288e-06 | 9.451889e-05 |
| [8,] | 3.450125e-04 | 4.063344e-02 |
| [9,] | 2.736458e-04 | 9.607167e-01 |
| [10,] | 4.036703e-05 | 9.949622e-01 |
| [11,] | 1.049489e-03 | 7.968151e-01 |
| [12,] | 3.652427e-04 | 4.318389e-02 |
| [13,] | 3.367462e-05 | 2.827148e-03 |
| [14,] | 2.498208e-04 | 9.641888e-01 |
| [15,] | 3.019068e-05 | 9.962275e-01 |

[16,] 6.729130e-04 9.145660e-01

[17,] 6.826990e-04 9.132984e-01

[18,] 9.496311e-01 1.254282e-02

[19,] 7.823448e-01 5.101343e-02

[20,] 8.961214e-01 2.600295e-02

[21,] 2.141206e-05 9.149514e-04

$J

[1] 90148.3

$histJ

[1] 220963.11 177338.71 123169.34 102637.98 93217.34 91202.78 90794.99

[8] 90624.76 90519.26 90442.52 90382.13 90332.62 90291.38 90257.09

[15] 90229.01 90206.59 90189.24 90176.28 90166.93 90160.42 90156.02

[22] 90153.13 90151.28 90150.12 90149.40 90148.96 90148.69 90148.53

[29] 90148.44 90148.38 90148.35 90148.33 90148.32 90148.31 90148.31

[36] 90148.30 90148.30 90148.30

$g

| | time | human |
|---|---|---|
| [1,] | 6 | 148 |
| [2,] | 6 | 396 |
| [3,] | 1234 | 149 |
| [4,] | 23 | 881 |
| [5,] | 10 | 679 |
| [6,] | 21 | 1554 |
| [7,] | 15 | 774 |

$cluster_id

[1] 5 3 2 1 1 2 5 5 7 7 7 5 5 7 7 7 7 6 6 6 4

**4. SPSS K 均值聚类算法结果**

快速聚类

初始聚类中心

| | 聚类 | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 时间段 | 0 | 1234 | 5 | 6 | 7 | 22 | 16 |
| sum(G) | 645.84 | 149.04 | 99.36 | 249.04 | 50.04 | 430.56 | 331.20 |

迭代历史记录[a]

| | 聚类中心中的变动 | | | | | | |
|---|---|---|---|---|---|---|---|
| 迭代 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | .000 | .000 | .000 | 5.389 | .000 | 17.372 | 22.143 |
| 2 | .000 | .000 | .000 | .000 | .000 | .000 | .000 |

a. 由于聚类中心中不存在变动或者仅有小幅变动，因此实现了收敛。任何中心的最大绝对坐标变动为 .000。当前迭代为 2。初始中心之间的最小距离为 49.361。

最终聚类中心

| | 聚类 | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 时间段 | 0 | 1234 | 5 | 11 | 7 | 17 | 14 |
| sum(G) | 645.84 | 149.04 | 99.36 | 249.81 | 50.04 | 414.00 | 353.28 |

每个聚类中的个案数目

| | | |
|---|---|---|
| 聚类 | 1 | 1.000 |
| | 2 | 1.000 |
| | 3 | 1.000 |
| | 4 | 3.000 |
| | 5 | 1.000 |
| | 6 | 8.000 |
| | 7 | 6.000 |
| 有效 | | 21.000 |
| 缺失 | | 3.000 |

## 5. Python 算法 当乘客人数 x 大于临界值

```
import random

v=random.randint(0,12)

b=random.randint(0,200)

x1=random.randint(0,2000)

x2=random.randint(2000,4000)
```

```
x3=random.randint(6000,7000)
count=1
while (4*x1+2*x2+x3)/7-1.5*b+v*0.2*b > 0 :
    if 120-0.0327*(7*b) > (0.0851+0.1702+0.3404)*b and count < 100000:
        print(b,x1,x2,x3,v,0)
        x1=random.randint(0,2000)
        x2=random.randint(2000,4000)
        x3=random.randint(6000,8000)
        b=random.randint(0,200)
        v=random.randint(0,12)
        count=count+1
    elif 120-0.0327*(7*b) <= (0.0851+0.1702+0.3404)*b and count < 100000:
        print(b,x1,x2,x3,v,1)
        x1=random.randint(0,2000)
        x2=random.randint(2000,4000)
        x3=random.randint(6000,8000)
        b=random.randint(0,200)
        v=random.randint(0,12)
        count=count+1
        if count > 100000:
```

```
                break
```

**6. Python 算法　当乘客人数小于临界值**

```
import random
v=random.randint(0,12)
b=random.randint(0,1000)
x=random.randint(0,6000)
count=1
while count<10000000:
    while x-1.5*b+v*0.2*b <= 0:
        while
120-(0.0329+0.0734+0.1938)*b+(0.0023+0.0053+0.0421)*x >
(-0.004-0.014-0.1096)*x+(0.1912+0.5044+0.0911)*b and count <
100000000:
            print(b,x,v,0)
            v=random.randint(0,12)
            b=random.randint(0,1000)
            x=random.randint(0,6000)
            count=count+1
        else:
            print(b,x,v,1)
            v=random.randint(0,12)
```

```
                b=random.randint(0,1000)
                x=random.randint(0,6000)
                count=count+1
                if count > 100000000:
                        break
        else:
                v=random.randint(0,12)
                b=random.randint(0,1000)
                x=random.randint(0,6000)
                count=count+1
```

7，logistic 回归代码

```
> data1 <- read.table("data.txt",header=TRUE)          #将模拟的 474
```

组数据导入到 r 语言中

```
>table(data1$dec)
```

#其中选择留在机场的策略有 200 个，选择回去的策略有 274 个

```
  0   1
200 274
>  fit<-glm(dec~car+x+v,data=data1,family=binomial())          #使用
```

logistic 回归进行研究

```
> summary(fit)
```

Call:

glm(formula = dec ~ car + x + v, family = binomial(), data = data1)

Deviance Residuals:

| Min | 1Q | Median | 3Q | Max |
|---|---|---|---|---|
| -4.3899 | -0.4650 | 0.0004 | 0.0318 | 1.8323 |

Coefficients:

| | Estimate | Std. Error | z value | Pr(>\|z\|) | |
|---|---|---|---|---|---|
| (Intercept) | -0.2706622 | 0.6427969 | -0.421 | 0.674 | |
| car | 0.0194281 | 0.0028615 | 6.790 | 1.12e-11 | *** |
| x | -0.0013655 | 0.0002501 | -5.459 | 4.79e-08 | *** |
| v | 0.0444011 | 0.0423618 | 1.048 | 0.295 | |

---

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 645.50 on 473 degrees of freedom

Residual deviance: 268.13 on 470 degrees of freedom

AIC: 276.13

Number of Fisher Scoring iterations: 8

#可以看出 car 和 x 对方程的贡献是很显著的，v 对方程的贡献一般。

> fit.reduce <-glm(dec~car+x,data=data1,family=binomial()) #此处将 v 速度变量去除，检验去除 v 之后的新模型是否拟合好。

> anova (fit.reduce,fit,test="Chisq")

Analysis of Deviance Table

Model 1: dec ~ car + x

Model 2: dec ~ car + x + v

| | Resid. Df | Resid. Dev | Df | Deviance | Pr(>Chi) |
|---|---|---|---|---|---|
| 1 | 471 | 269.24 | | | |
| 2 | 470 | 268.13 | 1 | 1.1061 | 0.2929 |

#对于广义线性回归，使用卡方检验，发现 p=0.2929 表明卡方值不显著。说明加上 v 变量之后不会显著提高模型的预测精度

> coef(fit.reduce)

| (Intercept) | car | x |
|---|---|---|
| -0.070247727 | 0.019570048 | -0.001344096 |

> exp(coef(fit.reduce)) #解释模型的系数

| (Intercept) | car | x |
|---|---|---|
| 0.9321629 | 1.0197628 | 0.9986568 |

> plot(data1,col=c(1:25),main="模拟数据的散点图矩阵") #画出散点图矩阵