1. **统计7月份每天每个司机每趟行程的开始时间、结束时间、里程、行驶时长**

**思路：**

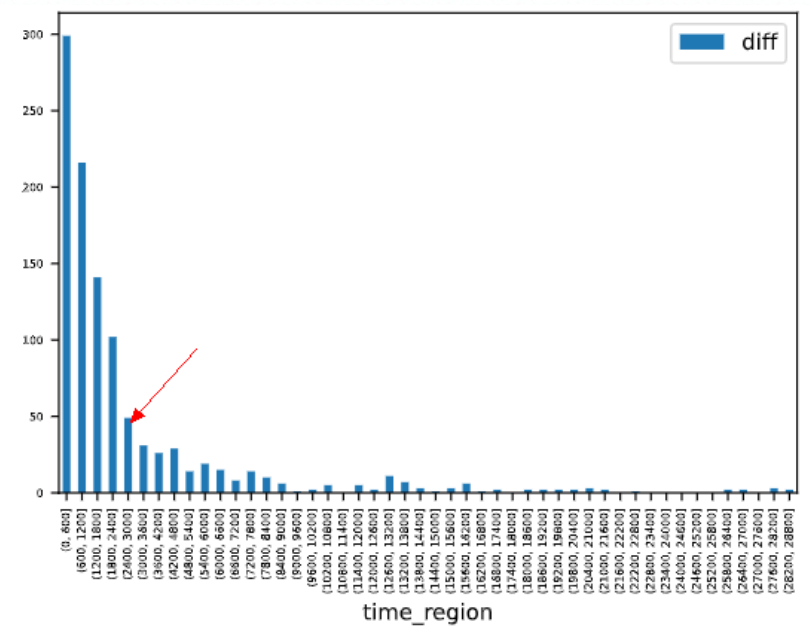
* 一次行程有多个不连续的子时间片构成，将t\_attendance\_detail.csv 考勤表中的同一天且同一个司机的所有时间片筛选出；
* 合并时间片时，需要满足:

Diff = 下一条的start\_time - 上一条的end\_time < 时间阈值

（即两个时间片的间隔时间）

超过时间阈值的，即会被划分到两趟行程中。

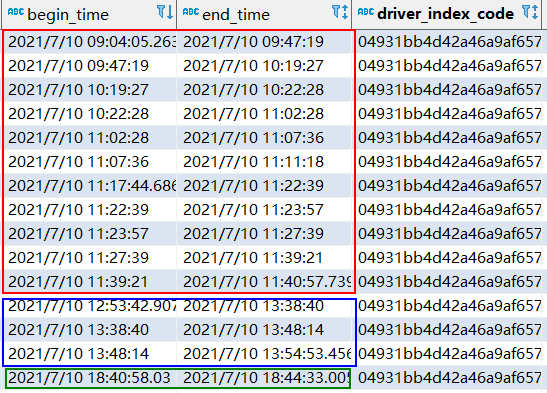
通过统计所有时间片的Diff，确定阈值为50min；



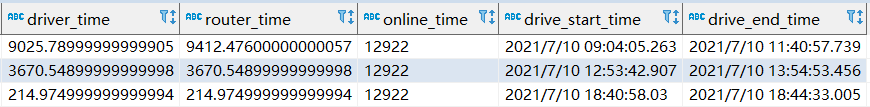
（上图将Diff划分为离散的区间，每个区间长为10min，当阈值取3000s时，可以涵盖绝大部分的时间）

如下例：

在2021/7/10，司机编号为04931bb4d42a46a9af65716d8b818c51的具体行程可以被分为三段；段与段之间的时间间隔超过50min；



整合结果如下：



合并结果详见结果详见t\_task1\_tab

* 属性声明

total\_driver\_time: 来自表t\_attendance\_statistics的driver\_time；

driver\_time: 该趟行程中实际行驶时间；

route\_time：该趟行程时间 = 实际行驶时间 + 时间片间隙（可能下车进行了短暂的休息或是设备发生短暂的异常）

单天所有行程的driver\_time之和应与total\_driver\_time相近记为本次行程的时长。

注意：

* 存在一天出现两次行程、且行程时间相互重叠的异常数据。

尚未处理；

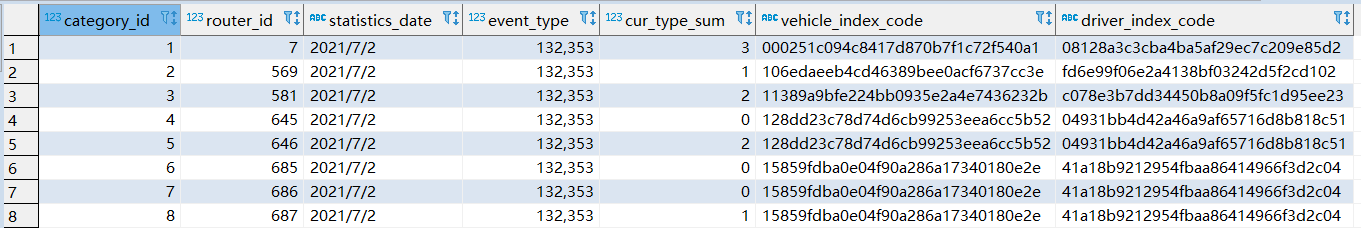
* t\_attendance\_statistics.csv 考勤统计表中存在异常数据：其中driver\_time会小于零，且出发时间晚于结束时间。故此类被筛出。

实现过程详见：task1\_0\_unify.sql 与 task1\_1\_route.sql

1. **统计7月份每天每个司机每趟行程里发生的每类报警类型的数量**

思路：

* 逐一遍历“需求1”的每趟行程，再逐一遍历t\_device\_alarm\_202107xx 报警记录表的每条报警信息，如果报警的发生时间在行程中，就记入该行程；
* 按照报警类型进行记录，获取某类型的总数；



结果详见表t\_device\_alarm\_category

实现过程详见：task2\_0\_pre.sql、task2\_1\_checkEvent.sql 与task2\_2\_t\_riskOut.sql

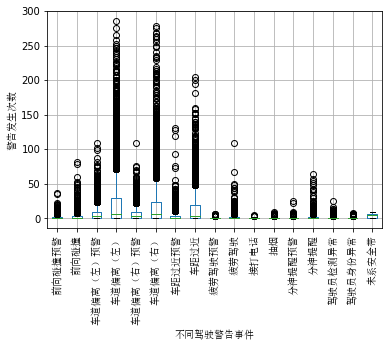
1. **对每个司机每趟行程发生每类报警数量进行分布情况的分析**

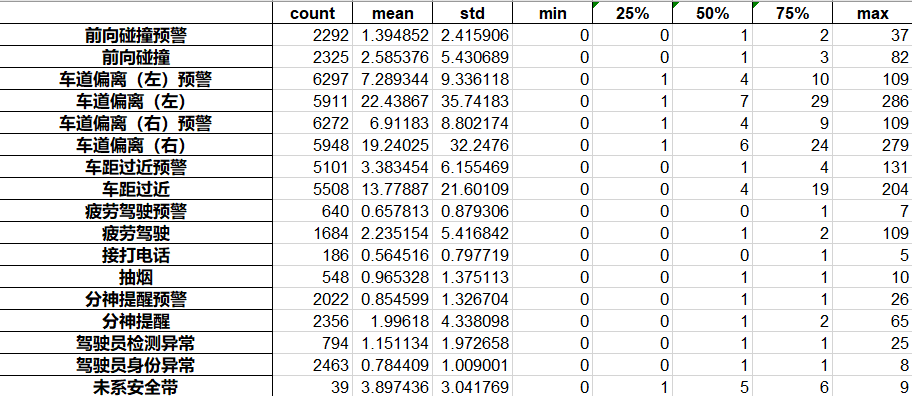
根据7月的报警数据，司机行程中会出现17种报警情况，具体见下图的X轴标注。

对这些告警类型绘制7月1日至7月31日的全量告警记录的在每个司机每趟行程中分布的箱线图，能够得到每趟行程中各告警类型发生次数的中位数、上下四分位数、极端值等统计指标。

从箱线图中看，整体上来看，警告次数出现最高的4中依次为：车道偏离（左）、车道偏离（右）、车距过近、车道偏离（左）预警。同时，每种类型都会出现较多异常点，反映了个别司机告警行为存在异常的情况。

为更好地显示，筛除了每类报警数量 > 300的异常值。





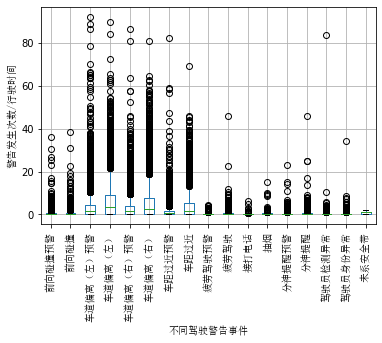
具体describe详见：task3\_describe.xls

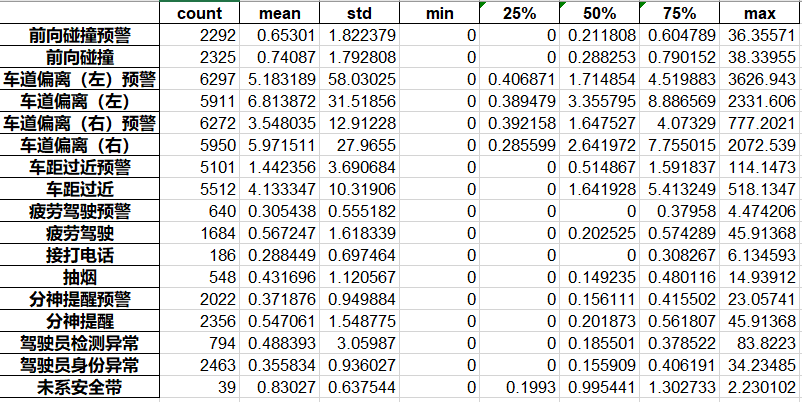
1. **对每个司机每趟行程发生 每类报警数量/行驶时长 进行分布情况的分析行驶时间与发生警告的关系**

对这些告警类型绘制7月1日至7月31日的全量告警记录与行驶时长之商在每个司机每趟行程中分布的箱线图，能够得到每趟行程中该商的中位数、上下四分位数、极端值等统计指标。

考虑到每类报警数量/行驶时长 结果会非常小产生精度损失，所以实际操作中为100000\*每类报警数量/行驶时长。

为更好地显示，筛除了每类报警数量/行驶时长>100的异常值。





具体describe详见：task4\_describe.xls

具体实现详见task3.ipynb

1. **分析连续多种报警类型之间的相关性，对组合报警进行合并分析**
2. **预处理**

现在有司机的每段行程以及各警报的发生时间点。

对于相关性来说，更关注各连续警报发生的序列关系。所以我们需要先以时间阈值，将每段行程中的发生间隔较近的事件聚合成一个报警序列。

**具体处理：**

* 将所有的alarm按照发生时间，归属其发生的行程段中。

具体内容见表t\_alarm\_router\_timestamp；

例如：

以下这条alarm信息就是属于router\_id为7的行程中。



* 将同一行程段中的各报警事件，按照时间阈值（time\_thresh），聚合成报警序列。

基本单位time\_seq:

每一个unit代表alarm事件类型ID的array。

https://tj-superpower.feishu.cn/space/api/box/stream/download/asynccode/?code=NjliNzk4OWU2YTVmY2E2MTYyNzg0YWFkNjkxODY4ZmJfaG5XUDJXOHZGdkpuNmZheVpkb09LdUM4Qk9zdkEzSXNfVG9rZW46Ym94Y25uQ3R2dXZWbXYwUmF2VDBTOTh0NGloXzE2MjkzNjc4MzA6MTYyOTM3MTQzMF9WNA

unit内alarm时间间隔小于time\_thresh，故可以被视为连续发生的警报时间； unit间间隔大于time\_thresh；

**处理技巧：**

1. 部分警报合并。

两个道路偏移合并为一个：左偏、右偏被合并为‘车道偏离’。两个道路偏移预警被合并为一个：左偏预警、右偏预警被合并为“车道偏离预警”。

1. 连续同种去重。连续的多个同种类型的alram只被视为一个！

具体实现见：task5\_0\_timestamp.sql 和task5\_1\_timestamp\_main.sql

* 1. **相关性分析**

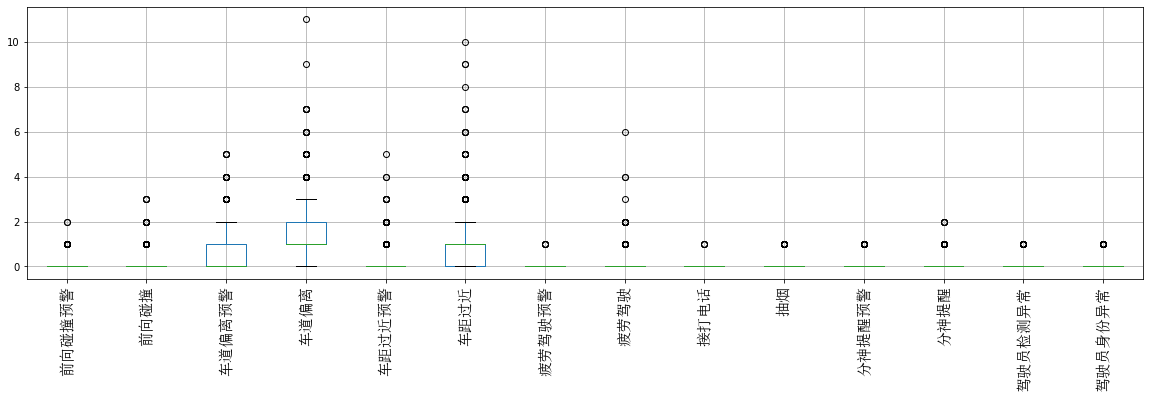
**思路：**

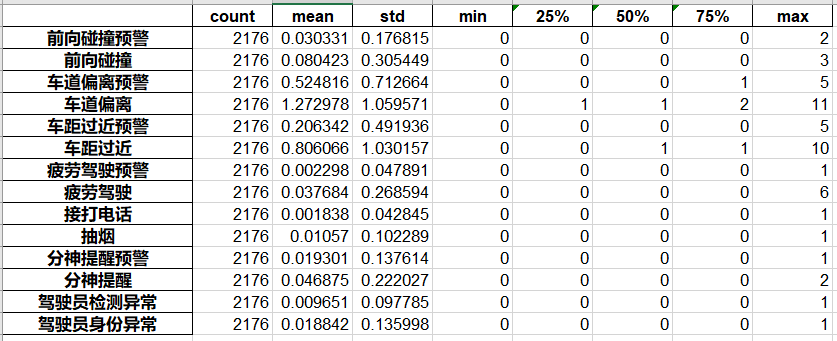
* 1. 统计报警序列中各报警事件发生的频数；
  2. 进行Pearson、Spearman、Kendall相关系数法进行相关性分析；

**具体处理：**

1. 统计报警序列中各报警事件发生的频数；

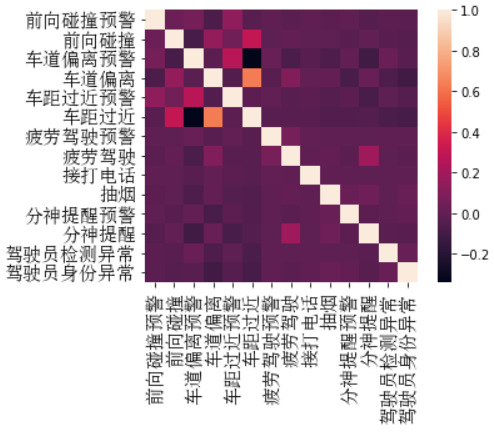
得到相应的箱线图：





详见task5\_0\_stastics.xls;

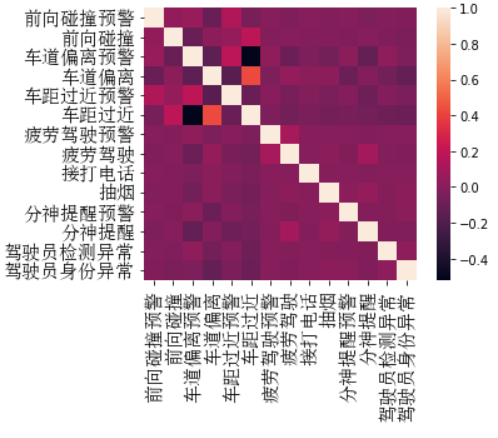
1. 相关系数法
2. Pearson相关系数法



|  |  |  |
| --- | --- | --- |
| 报警A | 报警B | 相关系数 |
| 车距过近 | 车道偏离 | 0.630646981 |
| 车距过近 | 前向碰撞 | 0.303831593 |
| 车距过近预警 | 车道偏离预警 | 0.273247739 |
| 分神提醒 | 疲劳驾驶 | 0.186238273 |
| 车道偏离 | 前向碰撞 | 0.159432224 |
| 车距过近预警 | 前向碰撞预警 | 0.144733566 |
| 疲劳驾驶 | 车道偏离 | 0.096311166 |
| 疲劳驾驶 | 疲劳驾驶预警 | 0.064751085 |
| 车道偏离预警 | 前向碰撞预警 | 0.059700369 |
| 车距过近预警 | 前向碰撞 | 0.0516805 |
| 分神提醒 | 抽烟 | 0.038907496 |
| 前向碰撞 | 前向碰撞预警 | 0.031430804 |
| 驾驶员检测异常 | 车道偏离预警 | 0.02625068 |
| 分神提醒 | 车道偏离 | 0.021803353 |
| 驾驶员身份异常 | 驾驶员检测异常 | 0.020893038 |

详见：task5\_1\_pearson\_sort.xls

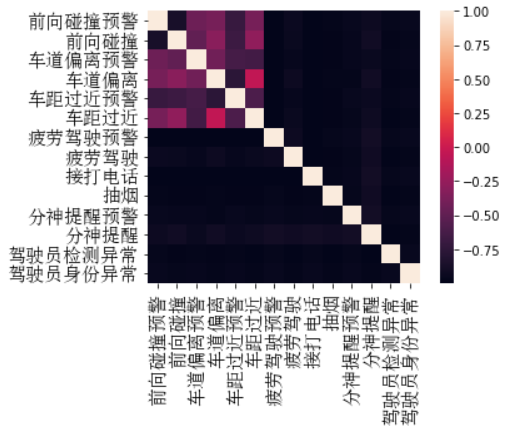
1. Spearman相关系数法



|  |  |  |
| --- | --- | --- |
| 报警A | 报警B | 相关系数 |
| 车距过近 | 车道偏离 | 0.429520413 |
| 车距过近 | 前向碰撞 | 0.182940107 |
| 车距过近预警 | 车道偏离预警 | 0.181671671 |
| 车距过近预警 | 前向碰撞预警 | 0.143394655 |
| 疲劳驾驶 | 疲劳驾驶预警 | 0.108441672 |
| 分神提醒 | 疲劳驾驶 | 0.101539707 |

详见：task5\_1\_spearman\_sort.xls

1. Kendall相关系数法



数据详见：task5\_1\_kendall\_sort.xls

具体实现见：task5\_0\_stastics.ipynb

* 1. **组合报警分析**

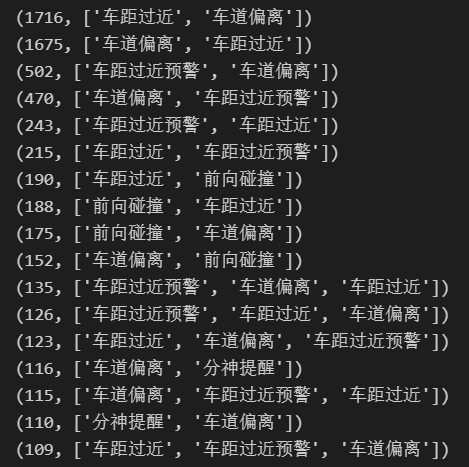
**思路：**

1. 从报警序列中，**挖掘**最常出现的**序列模式**。这些模式体现了报警类型的相关性，即可作为组合报警的分析依据；
2. 最后，根据序列模式，重新遍历每段行程的报警序列，以统计各组合警报出现的频数；

**具体处理：**

1. **挖掘序列模式**

使用PrefixSpan方法寻找序列模式。返回出现频数top-k的序列：



基本单位 result\_seq：(47, ['车距过近', '车道偏离']) 分别代表出现频数和出现事件序列。

* 进一步筛选：会出现这种情况 ，即长序列内容完全包含短序列，故没有产生新的价值，这种情况应该被避免！

(47, ['车距过近', '车道偏离']), (31, ['车道偏离', '车距过近', '车道偏离']) ；

处理方式：最终序列结果存放在filter\_res中。依次遍历result\_seq，每一个进入 filter\_res前，需要先和已经在其中的序列进行以下判断：

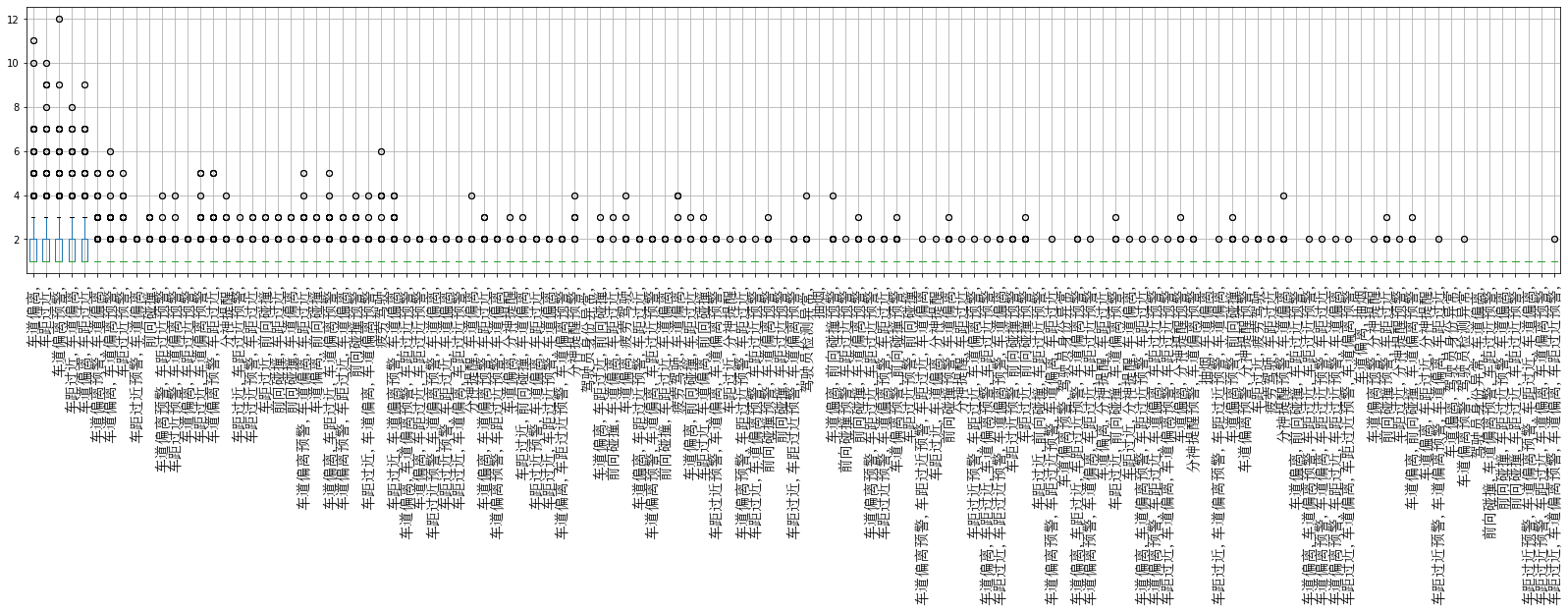
当前序列不可以是已有filter\_res中的子序列，且应该产生新的alarm类型，如果满足则放入filter\_res；

**5.3 序列模式统计**

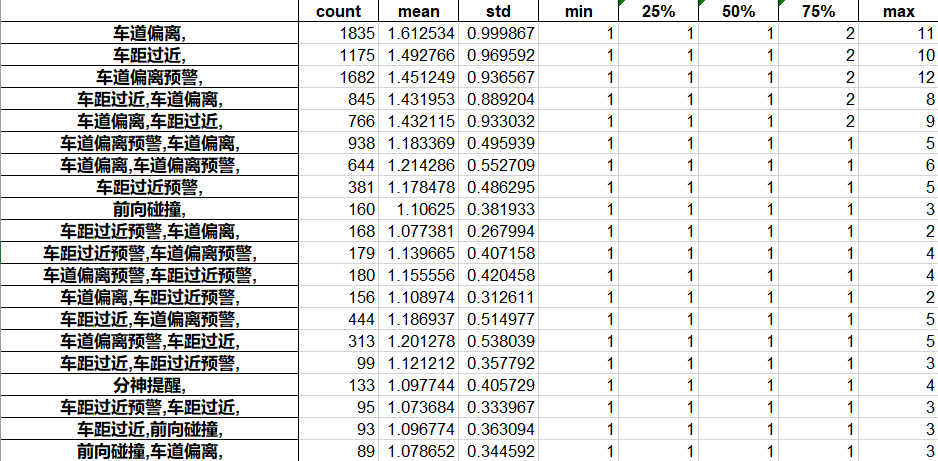
将得到的序列模式与所有的警报序列依次进行匹配，并统计每个序列出现各个模式的次数；

可以用箱线图进行表示：

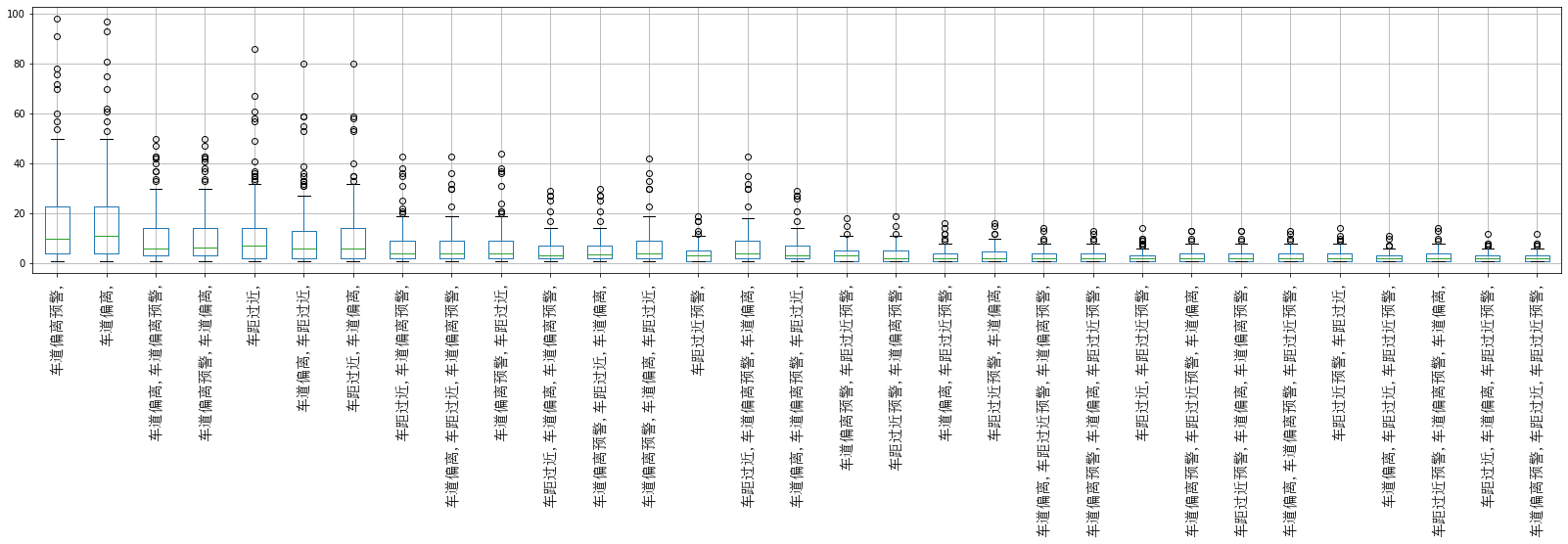
以2min的时间间隔阈值划分的行程序列分析：



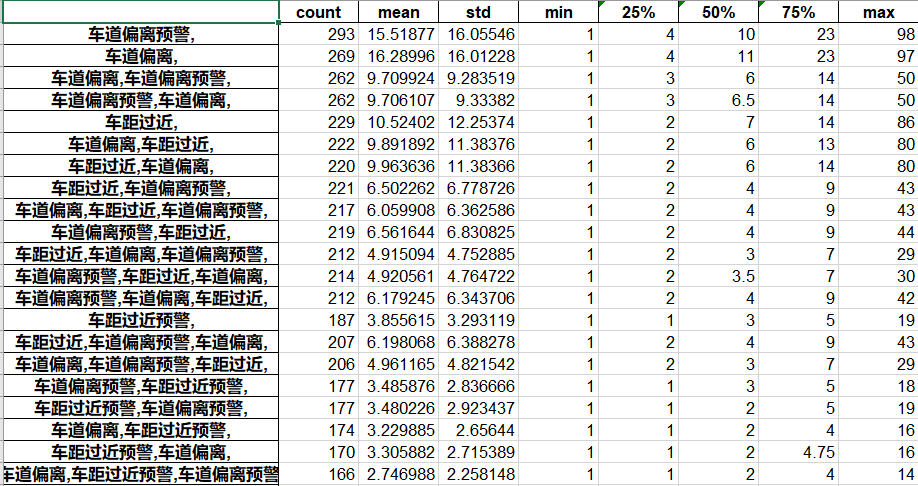
具体的describe信息（见附件：task5\_2\_stastics\_subseq\_2min\_interval.xls）：



以整趟行程进行序列分析：



具体的describe信息（见附件：task5\_2\_stastics\_full\_route.xls）：



具体实现见：task5\_1\_patternSeq.ipynb