数据清洗报告：

1. 由于文件信息缺失所以寻找主码 结果是：没有主码 有重复性元祖 1474对 占总数据量的1% 由此推测每一条数据的基本属性是客户购买所拆开的流水单子

Day\_id store\_code item\_id sell\_id 可以决定90%的tuple

+ Unit sold 可以决定 95%的tuple

+sell price 可以决定99%的tuple

具体操作方式，看看day\_id store\_code item\_id sell\_id 所确定下来以后 存在2个tuple 以上的元祖的信息，发现有9735个元祖满足 占总数据的10%。Unit sold 仅可以排除一半的元祖 至 4077个，其他如：alc\_price combo\_flag 均无用，Unit\_sold+sell\_price 可以区分到剩下 1474队元祖

*aggregate1<-sqldf("select day\_id,store\_code,item\_id,sell\_id,count(\*) from origin group by store\_code,day\_id,item\_id,sell\_id having count(\*)>1")*

结论

这也就是说明，基础元祖数据不带聚集属性，也就是元数据。除非以上10%的数据为重复数据，则需要删除。这样我们才可以做关联分析。

#如果是每条记录是transaction的话，一天同种产品sell方式一定存在多种，然而sell\_id 却能够决定90%的tuple，看剩下来的tuple如果销量是以零散单 unit——sold

#只有一点点的话可以认为是散单子？？？

2因为产品的ID数量太多，所以需要根据需求来进行剔除，若销售记录太少，则可以剔除

180款产品太多需要进行整合，发现产品售出记录小于50的有不少

too\_small<-sqldf("select distinct item\_id,count(\*)as n,category\_desc from origin group by item\_id having n<50")

以下下为销售记录小于50的产品id

|  |  |  |  |
| --- | --- | --- | --- |
|  | item\_id | n | category\_desc |
| 1 | 5310 | 15 | Snack |
| 2 | 5976 | 38 | Main |
| 3 | 6067 | 10 | Snack |
| 4 | 6076 | 5 | Drink |
| 5 | 6270 | 20 | Snack |
| 6 | 10961 | 21 | Main |
| 7 | 11778 | 8 | Non-Food |
| 8 | 11857 | 1 | Main |
| 9 | 11956 | 43 | Main |
| 10 | 12778 | 17 | Main |
| 11 | 12801 | 33 | Drink |
| 12 | 13066 | 1 | Non-Food |
| 13 | 13321 | 29 | Side Item |

180款产品-13款 剩余 167款

进一步发现，有些产品虽然销售记录少，但是销售量也有存在一定的异常，例如有一天销量为100

3看各个产品的销售起止时间，与日程表

check\_all\_item\_date<-sqldf("select item\_id,category\_desc,min(calendar\_date)as start\_date,max(calendar\_date)as end\_date from origin group by item\_id")

date\_check<-check\_all\_item\_date

date\_check$date\_count<-(date\_check$end\_date-date\_check$start\_date+1)

date\_check$start\_date<-as.Date(date\_check$start\_date,origin="1970-1-1")

date\_check$end\_date<-as.Date(date\_check$end\_date,origin="1970-1-1")

180款产品总是很多

category\_day number

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 产品销售期限 | 种数 | 描述 | 组别 |  |
| 2年 | 51 | 主打产品 | A | 作为主要分析对象 |
| 1年至2年 | 32 | 副主产品 | B | 看产品对主要产品的冲击 |
| 半年至1年 | 39 | 新产品 | C |  |
| 1个月至6个月 | 44 | 季节性产品 | D |  |
| 少于1个月 | 11 | 淘汰产品 | E |  |
| 少于5天 | 3 |  |  |  |

将销量为0与复数的剔除出去

4 每款产品的日销量矩阵

#4.1 剔除销售时间小于30天 的14款产品

drop\_id<-sqldf("select item\_id,category\_desc from date\_check where date\_count<30")

daily\_sell\_origin<-sqldf("select \* from origin where item\_id not in (select item\_id from drop\_id)")

#4.3 现在是比较麻烦的，需要将数据进行行和列的转变，纵坐标为calendar\_date 横坐标为item\_id 将categorydesc 放在前面然后daily\_sell放在后面

#一般来说这样子操作是有一些粗糙的，因为有部分天数会出现bulk purchasing的情况

library(tidyr)

daily\_sell<-unite(daily\_sell,"new\_id",category\_desc,item\_id,sep="ID")

temp<-melt(daily\_sell,id=c("calendar\_date","new\_id"))

check<-dcast(temp,calendar\_date~new\_id)

write.csv(check,"daily\_sell.csv")