互评作业2: 频繁模式挖掘

本次作业中，对于数据进行模式挖掘分析，一共分为下面4个任务。

由于仅涉及到有关购买记录的分析，为了减少冗余数据对读写、内存的占用，因此对30G数据进行处理，仅保存有关的数据。

同时，考虑到任务要求，忽略掉categories列，而是使用items里面的ids对应的category作为订单记录的商品种类。

预处理：数据重整

这一步进行数据重整，对于拥有全量信息的parquet，摘取有用信息并另存。选择的条目有：

```
avg_price           int64
categories         object
category_number     int64
payment_method     object
payment_status     object
purchase_date      object
```

其中categories使用items的id，category_number使用items的数量。整理完后读取一个parquet元信息示例如下：

```
Index(['avg_price', 'categories', 'category_number', 'payment_method',
       'payment_status', 'purchase_date'],
      dtype='object')
avg_price           int64
categories         object
category_number     int64
payment_method     object
payment_status     object
purchase_date      object
dtype: object
(8437500, 6)
[8437500 rows x 6 columns]
```

avg_price	categories	category_number	payment_method	payment_status	purchase_date
7847	[笔记本电脑, 零食, 智能手机]	3	支付宝	部分退款	2023-05-05
7561	[智能手机, 婴儿用品, 儿童课外读物, 模型, 玩具]	5	现金	部分退款	2022-01-29
184	[蔬菜]	1	现金	已支付	2025-01-26
1067	[笔记本电脑]	1	支付宝	已支付	2024-09-06
2542	[平板电脑, 裤子, 调味品, 耳机, 水产]	5	云闪付	已支付	2020-07-21

使用这种简化，可以大大减少不必要的条目，将**2G**的parquet文件压缩至**55M**！！！

因此，使用这个处理后的数据，便可以完成所有下面涉及到的4个模式挖掘任务，大大减少IO和存储的开销。

任务一：商品类别关联规则挖掘

这一任务需要分析用户在同一订单中购买的不同商品类别之间的关联关系。

1.加载合并数据

对于处理后的16个parquet进行读取、合并。由于简化后数据非常小，16个parquet共计800M内存，因此可以安心放入内存中。读取速度较快，共计用时86.51s完成读取。

```
[START] Loading and merging Parquet files...
[END] Loading and merging Parquet files in 86.51 seconds
Data shape: (135000000, 6)
```

2.提取商品类别

对于数据，我们只需要读取categories列，就可以完成分析。同时，在读取数据时，我们忽略掉每次进购买了一种商品的订单记录，因为这对关联规则挖掘并无作用。最终数据示例如下：

```
0                [笔记本电脑, 零食, 智能手机]
1     [智能手机, 婴儿用品, 儿童课外读物, 模型, 玩具]
4          [平板电脑, 裤子, 调味品, 耳机, 水产]
5       [益智玩具, 文具, 蛋奶, 婴儿用品, 平板电脑]
6                  [帽子, 米面, 床上用品]
7          [音响, 儿童课外读物, 笔记本电脑, 水产]
9                  [围巾, 蔬菜, 婴儿用品]
10                   [办公用品, 汽车装饰]
12                       [内衣, 相机]
14              [智能手机, 帽子, 音响, 裤子]
Name: categories, dtype: object
```

对于1.35亿条原始数据，最终得到107,997,513(1.07亿)条多种商品购买的记录。这一过程用时164.32s。

```
[START] Preprocessing 'categories' field...
[END] Preprocessing 'categories' field in 164.32 seconds
After filtering, data shape: (107997513, 6)
```

3.关联规则挖掘

我们使用FP-growth算法进行挖掘。因为Apriori相对而言对大批量数据处理性能较差。

首先我们统计频繁项集，这涉及到单一频繁项集和后续的多项集，在这里我们利用python的mlxtend库，置信度0.02。

```
frequent_itemsets = fpgrowth(df_te, min_support=min_support, use_colnames=True)
```

之后利用关联规则，提取支持度>0.5的规则。

```
rules = association_rules(frequent_itemsets, metric="confidence",
min_threshold=min_confidence)
```

在这一过程中，我们提取到0条符合要求的规则，因此后续电子设备也不能够提取到相关数据。

由于频繁项集有单项和多项，因此我们保存了一些中间结果到csv，同时方便后续可视化处理，如下：

- frequent_itemsets_all.csv，所有频繁项集
- frequent_itemsets_multi.csv，除去单项的频繁项集
- association_rules_top.csv，关联规则

这一过程执行较慢，对于1亿多条数据，共计用时44min13s。共计有42个频繁项集（支持度>0.02）。根据分析，对于1亿条数据，最小支持度0.02来说，需要有200万次组合出现，才能够符合频繁项集，代码是没有挖掘到相关频繁项集的。

```
Top-20 frequent itemsets with ≥2 items (by support count):

No rules found with confidence ≥ 0.5. Showing Top-20 rules without filtering...

Top-20 association rules:

[Saved] All rules saved to association_rules_top.csv
[END] FP-Growth mining in 2646.77 seconds
```

```
Found 42 frequent itemsets

Top-20 frequent itemsets (by support count):
['模型']: 9542977
['围巾']: 9504119
['文具']: 9472810
['卫浴用品']: 9295653
['水果']: 9259872
['饮料']: 9183825
['智能手机']: 9151838
['裙子']: 9076932
['零食']: 9054424
['耳机']: 8977060
['游戏机']: 8943939
['裤子']: 8940869
['益智玩具']: 8912537
['笔记本电脑']: 8874020
['健身器材']: 8873289
['智能手表']: 8838529
['儿童课外读物']: 8800545
['内衣']: 8798714
['外套']: 8764536
['婴儿用品']: 8694650

Top-20 frequent itemsets with ≥2 items (by support count):

No rules found with confidence ≥ 0.5. Showing Top-20 rules without filtering...

Top-20 association rules:

[Saved] All rules saved to association_rules_top.csv
```

```
[END] FP-Growth mining in 2646.77 seconds
```

4.筛选电子产品相关规则

根据要求，电子产品如下：

```
electronic_keywords = ['笔记本电脑', '平板电脑', '智能手机', '耳机', '相机', '电视']
```

但是由于上面没有提取到任何符合要求的关联规则，所以这里的分析也无法获取有效的规则。

5.可视化处理

空白规则，没有需要可视化的地方。

任务二：支付方式与商品类别的关联分析

这一任务需要挖掘不同支付方式与商品类别之间的关联规则。

由于Apriori和FP-Growth算法都用了事务矩阵，这个矩阵过于庞大、开销甚为不能接受（对于一个文件的8437500行事务构成的矩阵分析支持度0.01需要数分钟，16个文件耗时过久），并且考虑到我们在上一次作业（互评作业1）中对于数据初步分析时确定有7种支付方式和42种商品类别，因此可以构建一个7*42的dict，用于存储每种事务出现的次数，进而只需要分析dict就可以计算支持度、置信度、提升度。

同样，如果是用这样的方式进行挖掘，那么就不需要全量数据存入内存，只需要一边读取一边统计，逐行处理计数，对于内存及其友好。

```
# 固定支付方式和商品类（避免杂项干扰）
PAYMENT_METHODS = ['银联', '现金', '支付宝', '微信支付', '储蓄卡', '信用卡', '云闪付']
CATEGORIES = [
    '智能手机', '笔记本电脑', '平板电脑', '智能手表', '耳机', '音响', '相机', '摄像机',
'游戏机',
    '上衣', '裤子', '裙子', '内衣', '鞋子', '帽子', '手套', '围巾', '外套',
    '零食', '饮料', '调味品', '米面', '水产', '肉类', '蛋奶', '水果', '蔬菜',
    '家具', '床上用品', '厨具', '卫浴用品',
    '文具', '办公用品',
    '健身器材', '户外装备',
    '玩具', '模型', '益智玩具',
    '婴儿用品', '儿童课外读物',
    '车载电子', '汽车装饰'
]
```

```
# 初始化统计量
joint_counts = defaultdict(int)     # {(pay, cate): count}
payment_counts = defaultdict(int)   # {pay: count}
category_counts = defaultdict(int)  # {cate: count}
total_rows = 0
```

通过这样计量，只需要遍历一遍数据，就可以获取所有事物的频率。

1.加载合并数据

同上。加载很快，每个文件只需要大约5s。

2.提取商品类别和支付方式

同上，处理categories为list，然后增加payment_method（string）列。

```
pay = row['payment_method']
cats = [c for c in row['categories'] if c in CATEGORIES]
```

每个文件处理约490s，16个文件共计130min。

```
处理文件：part-00015.parquet

读取文件 part-00015.parquet 用时：5.45 秒
100%|██████████████████████████████████████████████████████████████████████████████████████████████████| 8437500/8437500 [08:01<00:00, 17523.21it/
统计处理行数：8437500，计数更新用时：481.52 秒
```

3.构建事务

不同于上面的商品种类，由于涉及到不同种类，所以需要通过生成的方式，组合出每一个订单对应的事务矩阵。

```
# 初始化统计量
joint_counts = defaultdict(int)     # {(pay, cate): count}
payment_counts = defaultdict(int)   # {pay: count}
category_counts = defaultdict(int)  # {cate: count}
total_rows = 0
```

4.关联规则挖掘

要求找出支持度 ≥ 0.01、置信度 ≥ 0.6的规则，使用公式计算即可进行挖掘。

```
sup = joint / total_rows
conf = joint / payment_counts[pay] if payment_counts[pay] > 0 else 0
lift = conf / (category_counts[cat] / total_rows) if category_counts[cat] > 0
else 0
```

保存结果设置为csv。

```
results.append({
    "支付方式": pay,
    "商品类别": cat,
    "共现次数": joint,
    "支持度": round(sup, 5),
    "置信度": round(conf, 5),
    "提升度": round(lift, 3)
})
```

	支付方式	商品类别	共现次数	支持度	置信度	提升度
162	微信支付	模型	1517932	0.01124	0.07871	1.001
288	云闪付	模型	1517449	0.01124	0.07869	1.001
204	储蓄卡	模型	1516621	0.01123	0.07863	1.000
36	银联	模型	1516743	0.01124	0.07863	1.000
120	支付宝	模型	1515893	0.01123	0.07861	1.000
..	...	...	...	...	...	...
49	现金	摄像机	1232210	0.00913	0.06389	1.000
217	信用卡	摄像机	1231555	0.00912	0.06387	1.000
7	银联	摄像机	1231714	0.00912	0.06386	1.000
91	支付宝	摄像机	1230984	0.00912	0.06384	0.999
175	储蓄卡	摄像机	1231293	0.00912	0.06383	0.999

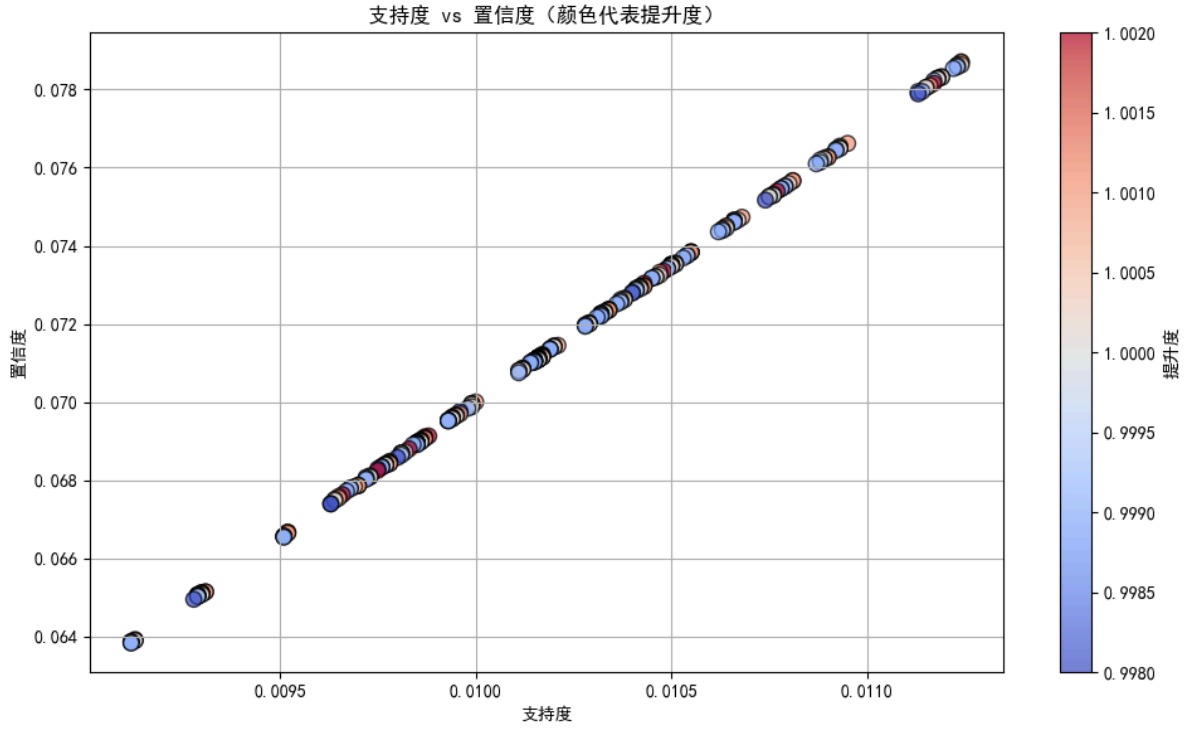
[294 rows x 6 columns]
指标计算完成，用时：0.03 秒

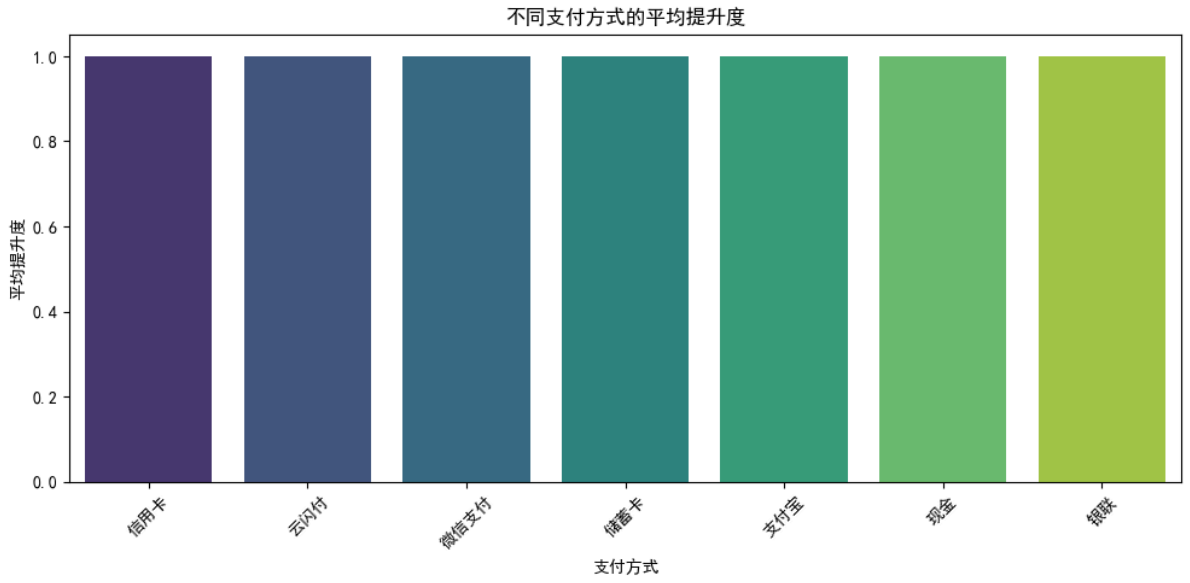
全部处理完成，结果保存到 支付方式_商品类别_关联规则.csv
总耗时：7850.42 秒

已经按照置信度进行排序，所有规则的置信度、支持度几乎一致，**没有发现**符合支持度>0.01并且置信度>0.5的规则。符合数据均匀分布的特性。

5.可视化分析

对于结果csv进行可视化如下。





通过可视化，可以有如下结论：

1. 支持度和置信度基本线性相关，表示**所有支付方式使用差异不大。**
2. 提升都大约都为1，说明“支付方式”和“购买种类”是**几乎独立**的。

这也符合”互评作业1“种所发现的数据均匀、独立分布特性。

任务三：时间序列模式挖掘

这一任务需要挖掘不同支付方式与商品类别之间的关联规则。

同任务二，我们发现每年只有12个月、每周只有7天，在这种情况下，再结合数据中共计42种商品类别，因此可以构建一个7*42或者12*42的dict，用于存储每种事务出现的次数，进而只需要分析dict就可以计算支持度、置信度、提升度。

1.加载数据

同上。加载很快，每个文件只需要大约5s。

2.提取时间和种类

可以知道，原本文件使用string保存datetime，因此可以用dateutil.parser.parse处理这个string，获取时间。

```
date = parse(str(row['purchase_date']))

month = date.month
quarter = (month - 1) // 3 + 1
weekday = date.weekday()
```

然后提取种类list，同上。

由于逐行分析时候对字符串进行了处理，相对而言较为缓慢，因此时间开销为上面任务二的一倍，每个文件要17min左右。

```
处理文件: part-00015.parquet
读取用时: 5.15 秒, 共 8437500 条记录
100%|██████████| 8437500/8437500 [17:25<00:00, 8067.38it/s]
处理有效行数: 8437500, 分析用时: 1045.89 秒
```

3.构建事务

同样使用dict构建统计量。

```python
# 季节性、月度、周统计容器
month_counts = defaultdict(Counter)      # {month: {category: count}}
quarter_counts = defaultdict(Counter)    # {quarter: {category: count}}
weekday_counts = defaultdict(Counter)    # {weekday: {category: count}}

# 时序模式原始序列（按月合并后的聚合）
monthly_sequence = defaultdict(int)      # {(month, category): count}
```

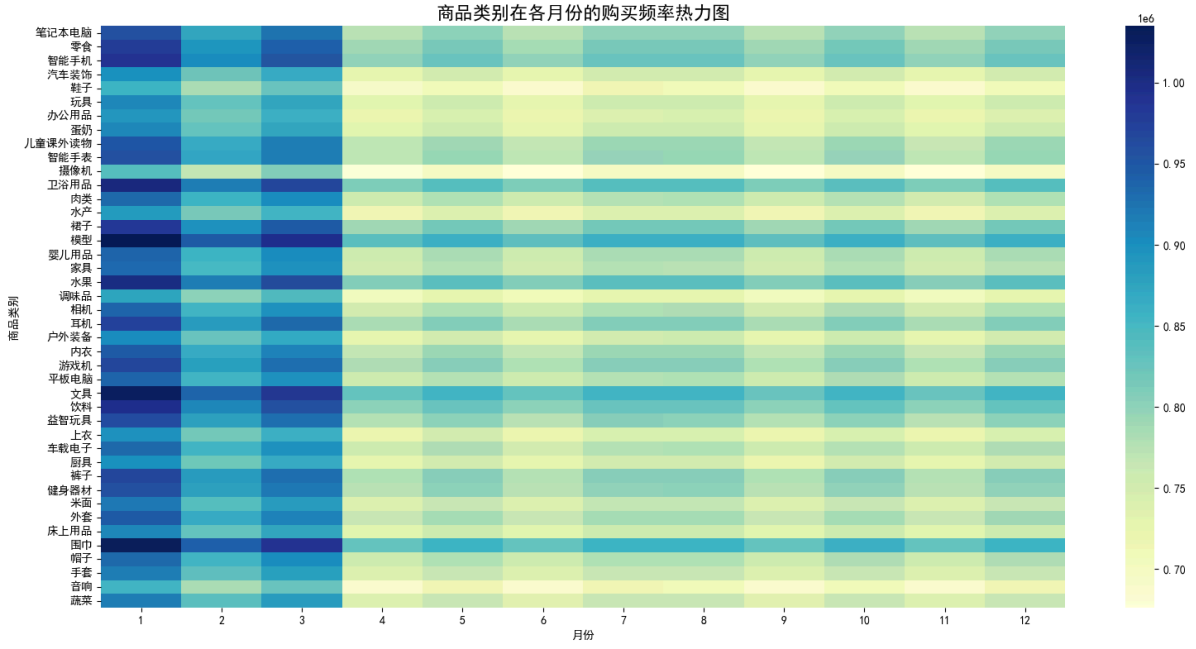
4.时间序列挖掘

商品和月份的分布不需要挖掘。

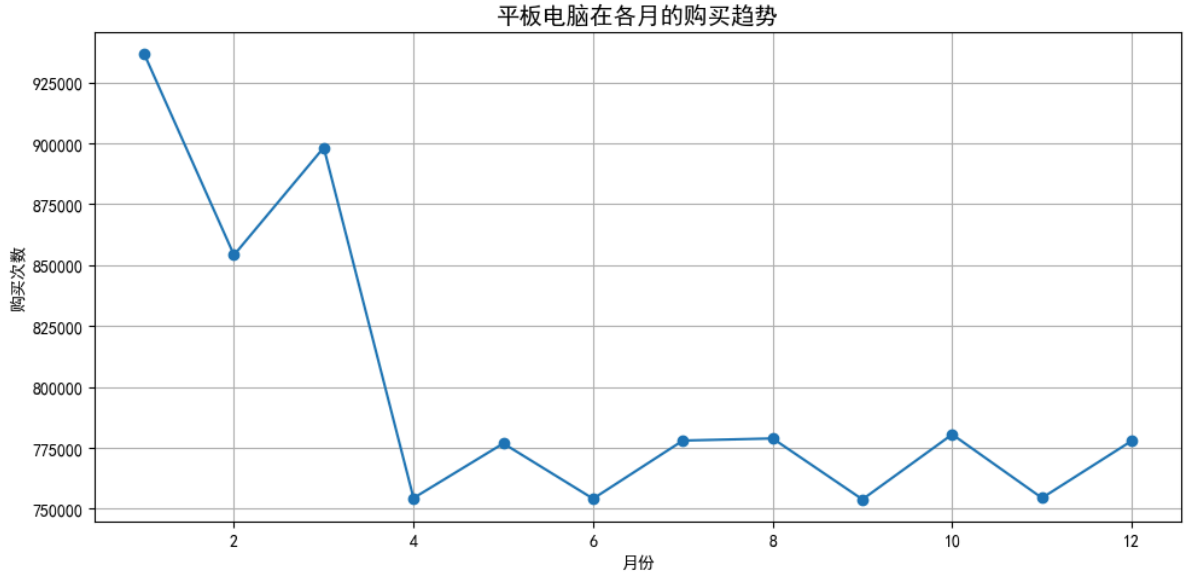
对于月度转移，我们使用每个月的商品到下个月过程中所有42种商品的变化作为转移量，12月转移到1月，进行统计，保存到csv。

5.可视化分析

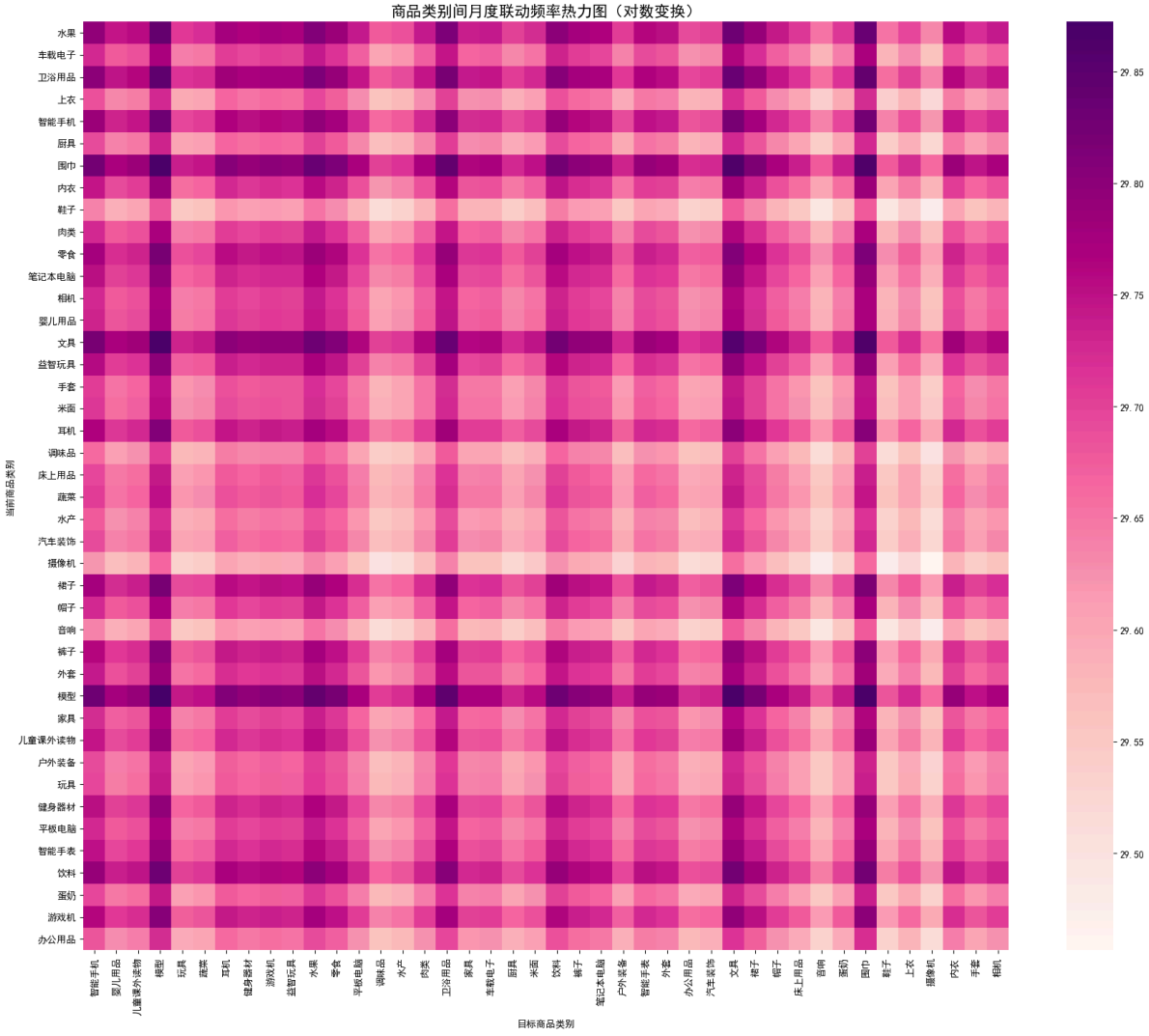
商品-月度热力图如下。可以看出1月购买量最多，1-3月（第一季度）购买最多，各种商品购买基本均匀分布。



以平板电脑为例，观测折线图。同样可以看到1月最多，1-3月（第一季度）购买最多。



商品-月度转移矩阵如下。可以看出变化很小，几乎没有什么特别的时间倾向，符合互评作业1中均匀分布结论。



任务四：退款模式分析

这一任务需要挖掘与"已退款"或"部分退款"状态相关的商品类别组合。

同上，退款模式只有两种（已退款、部分退款），再结合数据中共计42种商品类别，如果仅考虑两种商品的组合，可以构建一个42*41/2=861的dict，用于存储每种事务出现的次数，进而只需要分析dict就可以计算支持度、置信度、提升度。

1.加载数据

同上。每个文件用时约5s。

2.提取退款模式和种类

先设定退款参数，对于逐行处理，提取退款模式和种类list。

```
PAYMENT_STATUSES = ['已退款', '部分退款']

statues = row['payment_status']
cats = [c for c in row['categories'] if c in CATEGORIES]
```

由于逐行分析时间开销为与任务二的一致，每个文件要8min左右。

```
处理文件: part-00008.parquet
读取文件用时: 4.84 秒
100%|██████████| 8437500/8437500 [08:20<00:00, 16871.26it/s]
总行数: 6750945, 退款行数: 4500296, 用时: 500.11 秒
```

所有文件处理完成，总交易数: 107997513, 退款交易数: 71997798

3.构建事务

同上，使用统计量进行构建。

```
# 统计变量
refund_combos_counts = defaultdict(int) # {(cate, cate): num}
all_combos_counts = defaultdict(int)
refund_rows = 0
total_rows = 0
```

4.关联规则挖掘

要求找出支持度 ≥ 0.005、置信度 ≥ 0.4的规则，使用公式挖掘。

```
support = refund_count / total_rows
confidence = refund_count / refund_rows
lift = confidence / (total_count / total_rows)
```

结果保存为csv。

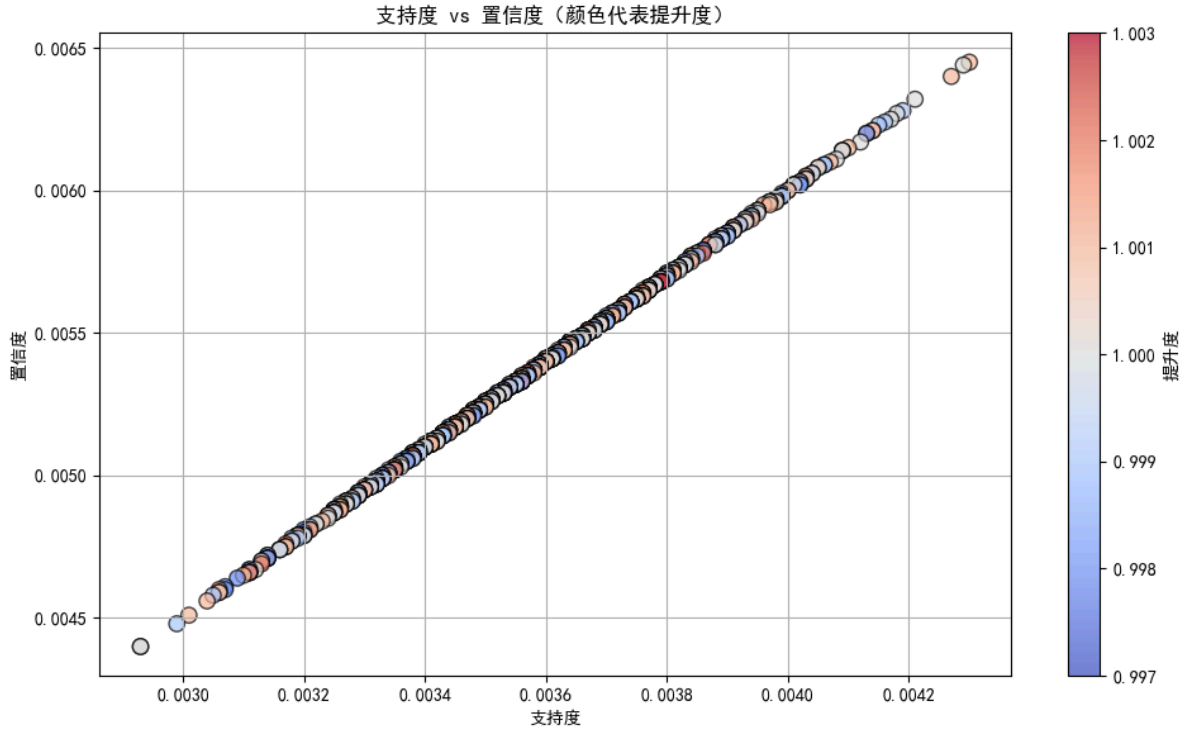
```
results.append({
    "商品组合": " & ".join(combo),
    "退款次数": refund_count,
    "出现总次数": total_count,
    "支持度": round(support, 5),
    "置信度": round(confidence, 5),
    "提升度": round(lift, 3)
})
```

结果中没有挖掘到符合要求的规则，按照置信度top10如下。符合数据均匀分布。不同二元组合的退款慈湖从30W次到40W次不等，但是对于事务全集来说，还是差的有些远了，并不足以支撑挖掘到有效规则。

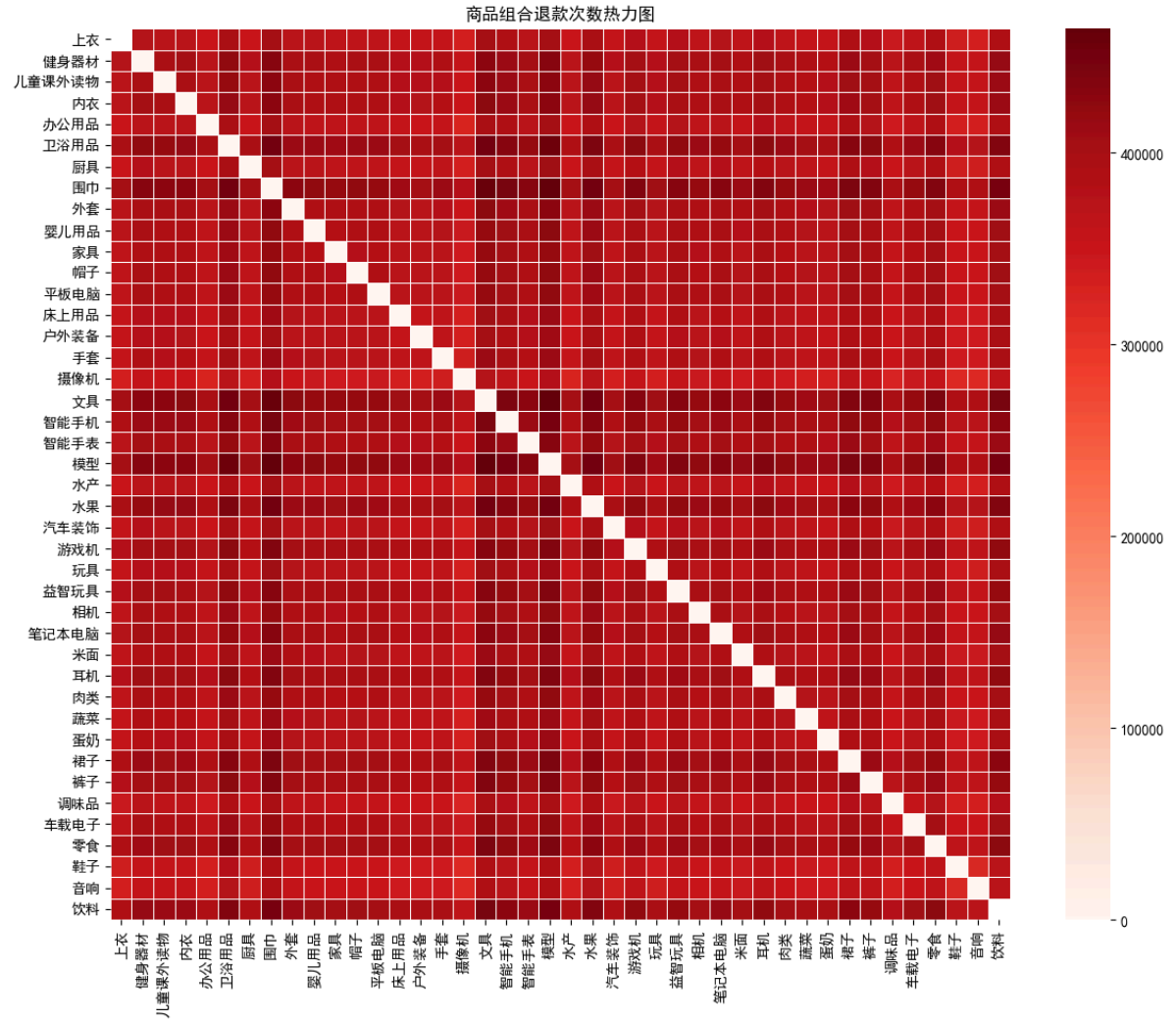
商品组合	退款次数	出现总次数	支持度	置信度	提升度
围巾 & 模型	464682	696485	0.0043	0.00645	1.001
文具 & 模型	463321	694972	0.00429	0.00644	1.000
围巾 & 文具	461135	691209	0.00427	0.00640	1.001
卫浴用品 & 模型	454996	682691	0.00421	0.00632	1.000
卫浴用品 & 围巾	452169	679059	0.00419	0.00628	0.999
模型 & 水果	451561	677258	0.00418	0.00627	1.000
围巾 & 水果	450320	675621	0.00417	0.00625	1.000
卫浴用品 & 文具	449598	675260	0.00416	0.00624	0.999
文具 & 水果	448476	673366	0.00415	0.00623	0.999
模型 & 饮料	447327	671102	0.00414	0.00621	1.000

5.可视化分析

首先按照退款次数和支持度进行绘制散点图。无法挖掘有效信息。支持度和置信度线性相关并且所有种类没有什么太大的差别，说明而这几乎是独立的。商品组合和退款没有明显的相关关系。



再看热力图。由于没有统计弹种商品，所以对角线为空白。



结论

各种模式挖掘都没有挖掘到同时符合支持度和置信度要求的规则，所有频繁项集的支持度、置信度、提升度几乎一致，数据几乎符合均匀分布，没有太多有用的业务价值。