蔬菜商品的补货和定价的优化模型

摘　要

本文针对现有的蔬菜类商超的补货和定价策略问题，通过蔬菜品类的历史销量与定价的关系，以及各蔬菜品类销量的占比关系，建立生鲜商超日最大收益的非线性优化模型。在此基础上，建立基于熵权法的多指标评价模型，结合品类内单品历史销量与定价的关系，对筛选得到的蔬菜单品建立各品类日最大收益的非线性优化模型。通过遗传算法对模型进行求解，得到最优的定价策略和补货总量。

本题的数据较多，在建立模型前首先对数据进行预处理，剔除异常值，并根据具体的分析需求选择对数据进行不同的预处理。如是否剔除打折数据和退货数据的选择上。

针对问题一，我们首先对蔬菜品类的销量分布进行**可视化分析**，得出不同品类的销量与时间、打折与否、价格的分布规律，并通过**相关性分析**得出不同品类间的比例关系和互补关系。然后根据**销量涨幅**的相关性，对同一品类内蔬菜单品的相关关系进行分析，得到品类内菜品的**替代单品**以及品类间单品的**互补关系**。

针对问题二，我们首先通过单元线性回归分析各品类销量和成本加成定价的关系，得出销量与成本加成定价成反比关系的结论。我们通过总收益经济学公式建立日收益与销量和定价的优化模型。由于**单元线性回归**的拟合结果不佳，R 方平均仅为 **0.05**。同时考虑到销量可能受到不同品类销量与定价的影响，因此我们选用**多元线性回归模型**对各品类的销量和成本加成定价进行回归分析，拟合结果良好，R 方平均为 **0.61**。同预测的日销售总量的分布规律一起作为约束条件，用遗传算法求解得商超日最大收益为 **1368.07** 元，六种蔬菜品类的补货量（千克）/定价（元/千克）依次为 **32.79/12.18、24.79/18.84、189.69/6.12、3.17/14.66、101.43/9.38、35.56/9.90。**

针对问题三，先通过线性回归分析各单品销量与定价的关系，剔除系数为正的单品，接着选取销量、利润率、打折次数、损耗率为评价指标，并通过熵权法得到四个指标的权重分别为 **0.61、0.20、0.11、0.08**。接着通过 **TOPSIS 法**进一步筛选得出 33 个蔬菜单品。并在第二问优化模型基础上建立各品类内的优化子模型。同样使用遗传算法对模型进行求解，得到各个单品最优的补货量和定价策略。

针对问题四，给出日进货量、更详细损耗数据、季节因素与天气数据、地理位置数据、促销活动与节假日数据、消费者用户画像、竞争关系这七项数据，能在一定程度上辅助制定补货及商品定价决策。

关键词：TOPSIS，非线性优化，多元线性回归，成本加成定价，遗传算法

一、 问题重述

由于蔬菜类商品的保鲜期大部分都很短，且其品相随时间推移越来越差，大部分蔬菜类商品当日未售完第二日也无法再次出售。因此生鲜商超将会根据往日各商品的供求情况进行每日补货。

由于蔬菜的品种、产地众多，交易时间紧迫，因此商家需在不知道具体单品和进货价格的情况下作出各蔬菜的补货决策。蔬菜定价通常采用“成本加成定价”法并对品相受损的商品进行打折销售。无论是需求层面还是供给层面时间都是一个重要因素：从需求层面来看蔬菜销售量往往与时间相关性很强；从供给层面来看蔬菜供给品种通常与季节相关。另外，考虑到销售空间的限制，确定一个最佳的销售组合极其重要。

问题 1 请分析蔬菜各品类销售量的分布规律以及不同品类商品销售量之间的相关关系；商品单品销售量的分布规律以及不同单品销售量之间可能存在的相关关系。

问题 2 已知生鲜商超将根据六种蔬菜品类制定补货计划，请分析各个蔬菜品类销售总量与成本加成定价之间的关系，并给出六个蔬菜品类 2023 年 7 月 1-7 日的每日补货总量和定价方案，从而使得商超总收益最高。

问题 3 考虑到对蔬菜类商品销售空间的限制，商超希望在商品品类的基础上进一步制定单品的补货方案。限制条件如下：单品总数在 27-33 个之间；各个单品至少要订购 2.5 千克；要满足市场对各个品类蔬菜商品的需求。根据 2023 年 6 月 24 日-30 日的可售蔬菜品种以及以上限制条件给出 7 月一日各蔬菜单品补货量以及定价方案。从而使利润最大化。

问题 4 商家还需收集哪些数据从而制定更加完善的补货及定价策略，请给出你的建议和理由。

二、 问题分析

2.1 对问题一的分析

问题一要求分析蔬菜各品类及单品销售量的分布规律及相互关系。这是一道可视化分析及相关性分析题。考虑使用箱线图、直方图、皮尔逊相关系数解决。首先选定销量分布的评价指标。第一个指标：时间，由于蔬菜类的销售量往往与时间存在一定的相关关系，从而首先对六种蔬菜品类以及各蔬菜单品销量随时间的分布规律进行可视化分析，观察其周期性以及分布特点。第二个指标：蔬菜的价格，从一般的经济学规律来讲，价格低时蔬菜销量将会上升，反之则下降。但

仍需结合具体数据进行分析。第三个指标：打折与否，通常打折与否也是影响销量分布的重要指标，但考虑到品质，时间等等因素对销量的综合影响，我们也不能主观断定打折则销量上升。同样需结合具体数据进行分析。

其次分析六种蔬菜品类之间的相互关系。首先考虑到第二问的进货决策我们对六种品类在总销售量中的占比进行处理分析：1.三年内各蔬菜品类销售额在总销售额中的占比。2.每个月各蔬菜品类销售额在每月总销售额中的占比。查阅资料得知不同蔬菜品类之间最常见的关系便是互补关系，如果两种商品互补，一种商品价格上涨，会使其需求量减少，同时，导致其互补商品需求量减少。由附件所提供数据可得，将要从各类蔬菜商品订购时间、价格涨幅率、销量涨幅率三个维度对六种蔬菜品类进行相关性分析，从而分析某些品类之间是否存在互补关系。

最后分析同品类不同单品之间的相互关系。查阅资料得知同品类不同单品之间最常见的关系为替代关系。简单来说便是当某种单品价格上涨时会导致本身需求量降低，同时将会导致其替代单品销量的上升。另外也可考虑单品间的互补关系。由附件所提供的数据可得，将要从各蔬菜单品的订购时间、价格涨幅率、销量涨幅率三个维度对同品类不同单品之间的替代关系以及互补关系进行相关性分析。下图 2.1-1 为问题一分析流程图。

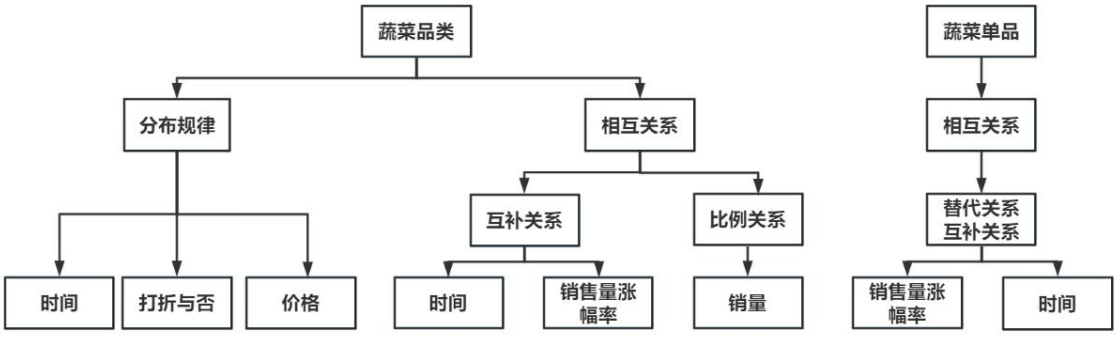


图 1 问题一分析流程图

2.2 对问题二的分析

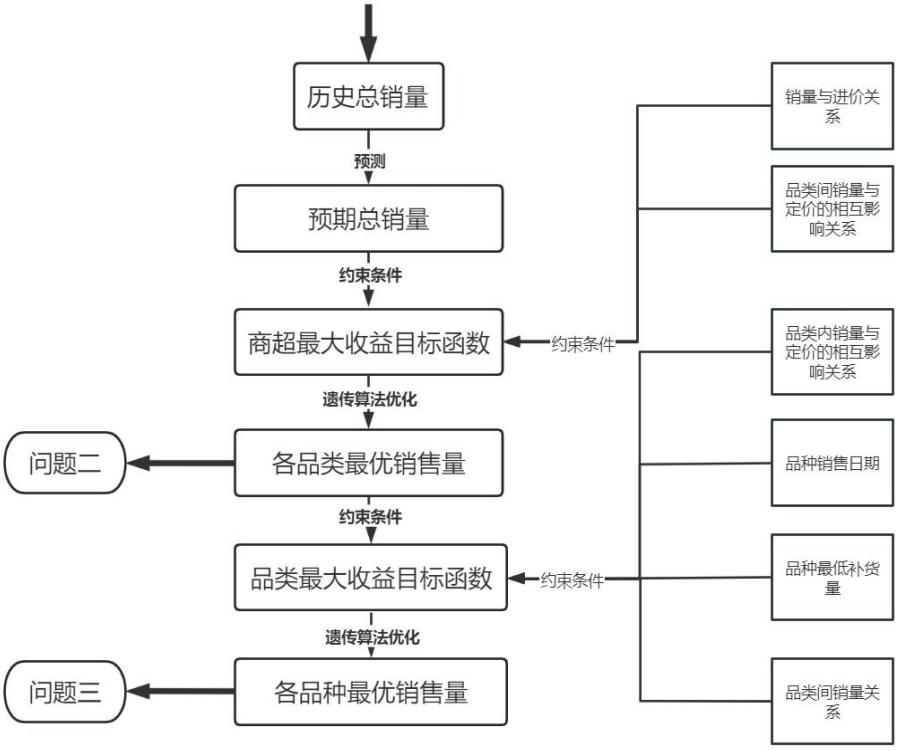
问题二要求分析给出各蔬菜品类补货总量和定价策略从而使商超收益最大。需在市场需求约束前提下，考虑定价与销售量之间的相互影响关系给出使得商超收益最大化的定价及补货策略。可将问题二划分为三部分：回归分析、预测问题、优化问题。

对于回归分析问题：本文根据历史数据对每一蔬菜品类的销量与该品类定价和其他品类定价和销量的多元线性关系进行拟合分析。首先对六个品类的销量和

定价进行异常值剔除，再确定每个品类的线性回归方程，并根据假设检验和残差检验观察模型的显著性和拟合度，并结合决定系数判断模型优劣。

对于预测问题：在进行数据预处理后，绘制进货价格以及补货总量时间序列图，观察数据特征。由于对 2023 年 7 月 1-7 日数据的预测属于短期预测，因此应优先采用短期预测模型。由于单日的补货价格和补货总量属于时间序列数据，因此可以采用 ARIMA、ARMA 等多种模型进行预测，并对预测结果进行检验从而选取最佳模型。

对于优化问题：本文的目标是找到最优各菜品的品类定价和总补货量，实现商超的最大总利润。但由于缺少商超历史补货量数据，本文用预测销量代替总补货量进行模型建立。因此对于本文要建立的总收益目标函数，决策变量为菜品各品类的定价和销量，约束条件为拟合后品类间和品类内部销量与定价的相互关系，以及预测后消费者对菜品品类的需求比例和超市销售总量的结果。



2.3 对问题三的分析

对于问题三，由于题目限定了可销售的品种数量，且 7 月 1 日可销售菜品数大于题目限定的单品数目范围，所以我们首先通过 TOPSIS 算法求解每个单品的重要程度，再去除单品中预测进货量小于最小陈列量 2.5 千克的单品，从而实现初步处理。最后根据剩余单品数是否位于规定范围，进行模型的建立和求解。若剩

余单品数位于规定范围，那么我们采用问题二中各个品类对总利润的销量最优值作为各个品类的最大利润优化模型中目标函数的约束条件。同时，品类内部各个单品的销量和定价的关系、品类间的销量比例关系都作为对目标函数的约束条件，从而在每个品类中求解出使该品类达到利润最大值的销量最优解，因为当各品类的利润达到最大时，也即满足了题目三要求的当日总利润最大。因此可以逐个得出各品类选取的品种类型，以及他们的定价和进货量。

2.4 、对问题四的分析

题目要求采集更多相关数据以更好地制定蔬菜商品的补货和定价决策。由于问题所给数据仅有每日进货价格，销售数量，销售价格以及损毁率。对销售成本的计算仅以以上数据得出。这样得出的成本计算模型太过理想化。因此本文建议采集与**成本相关**的数据，比如：进货数量、运输成本、人工成本、商品损耗具体数据。除成本相关数据外，**环境因素**对商超销售量以及利润也有一定的影响：例如商超地理位置、是否存在竞争商家、是否受到线上平台的冲击等。另外还需采集**市场需求**相关数据，例如：市场对各蔬菜类单品的需求量、消费者反馈与评价数据。同时还需考虑商品**供应链数据**，例如：当季蔬菜品类、供货商供货能力及供货质量等。

三、 模型假设

1.进货量假设：假设商家所进货数量均完全被销售，按照预期销售量和固定损耗率计算补货量，以消除数据集缺失补货量数据的影响。

2.蔬菜的损耗固定：本文使用平均损耗率来计算蔬菜的实际成本。

3.商超销售空间固定：商超的销售空间在一段时间内是恒定的。

4.销量外界影响假设：假设商超的补货量与进货量只与该商超的历史相关数据有关，不会受到其他商家折扣活动，定价等因素的影响。

四、 符号说明

符号	说明	单位
X_i	商品销售总量	千克
P_i	第 i 类品种蔬菜定价	元
Pro_i	第 i 类品种蔬菜利润	元
C_i	第 i 类品种蔬菜进价	元
PR_i	第 i 类品种蔬菜预期利润率	百分比

		（%）
X_i	第 i 类蔬菜销售数量	千克
Q_i	第 i 类蔬菜进货数量	千克

五、 问题一模型的建立与求解

5.1 数据预处理

对第一问涉及数据进行预处理：

（1）异常值处理，将数据集内的离群点进行滑动平均或是通过 3-西格玛检验进行剔除。

（2）缺失值填补，在进行时序分析时，某些数据集可能会存在缺失的状况从而导致时间点统计不连续，因此本文采用插值拟合的方法对缺失值进行填补。

（3）数据正向化标准化处理，保证数据处于零均值同方差的区间内，确保数据多个特征无量纲化。

（4）对于退货商品，本文将其对应的销售量数据剔除，并同时剔除所有退货商品。

5.2 各蔬菜品类销量的分布规律

5.2.1 各蔬菜品类销量随时间的分布规律

a.销售量逐日变化图

对各蔬菜品类每日销售量进行求和，退货商品不纳入销售量范围，利用 R-studio 对处理后数据进行可视化，绘制出各蔬菜品类销售量随时间变化图，如

图 5.2-1

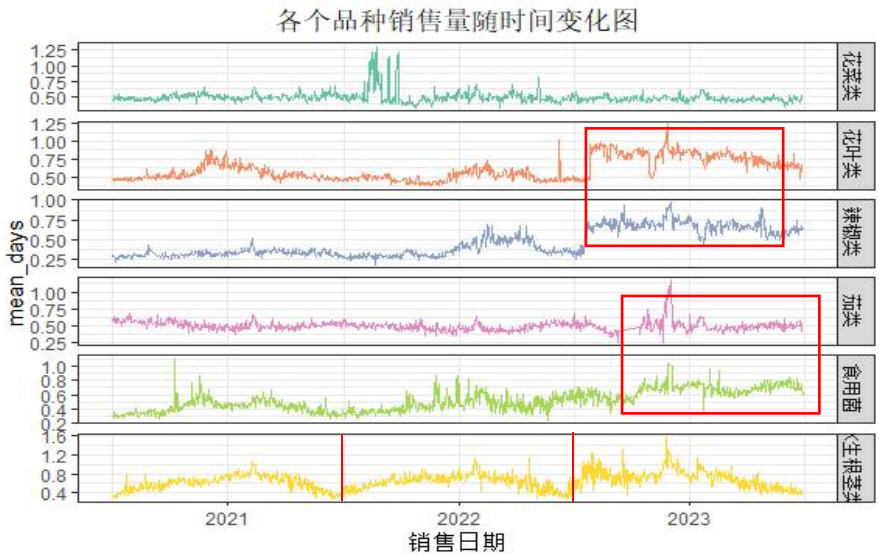


图 2 各蔬菜品类销售量三年内逐日变化图

水生根茎类分析:由折线图可知水生根茎类蔬菜销售量有明显的的季节性规律：当年 7 月到第二年 2 月销售量呈递增趋势，2 月至 7 月销售量呈递减趋势。由附件一数据可知水生根茎类蔬菜以莲藕为主要组成部分。自惠农网查阅资料可知莲藕一般在秋冬季节成熟，冬季上市。此时价格最低廉品质又最好。这正解释了为何水生根茎类蔬菜 2 月销售量最高。

花叶和辣椒类分析：花叶类与辣椒类蔬菜也没有明显季节性规律，但 2023 年销售量较 2021 年及 2022 年有较大增长。由中国经济网可知自 2022 年末疫情结束经济复苏。随着各地“扩内需、促消费”一系列举措出台，消费市场正呈现出融融暖意。这其中，线下餐饮市场的复苏最为旺盛。[1]

花菜类分析：花菜类蔬菜随时间分布较为平稳，但在 2021 年 7-9 月份销售量却远远高于平时。观察折线图可知在 2021 年 7-9 月份其它类别蔬菜都处于低谷时期，因此 2021 年 7-9 月份销售量的异常表现可能与商超进货决策或是随机突发情况有关。

b.销售量逐时变化图

从销售量逐日变化图我们仅能分析各蔬菜品类的季节性趋势，以及从宏观上分析造成分布规律特殊变化的因素，因此我们绘制一天内销售量的逐时变化图。逐时销售量取剔除异常值后数据对应时间销售总量的均值。可视化结果如图 5.2-2。

图 3 销售量逐小时变化趋势

发现在 21 点各品类销售量都有明显增长。但考虑到晚上蔬菜类商品销售量本该降低，因此考虑销售量的增长可能与打折销售。对一天内逐小时打折销售量统计情况进行可视化。如图 5.2-3 所示。

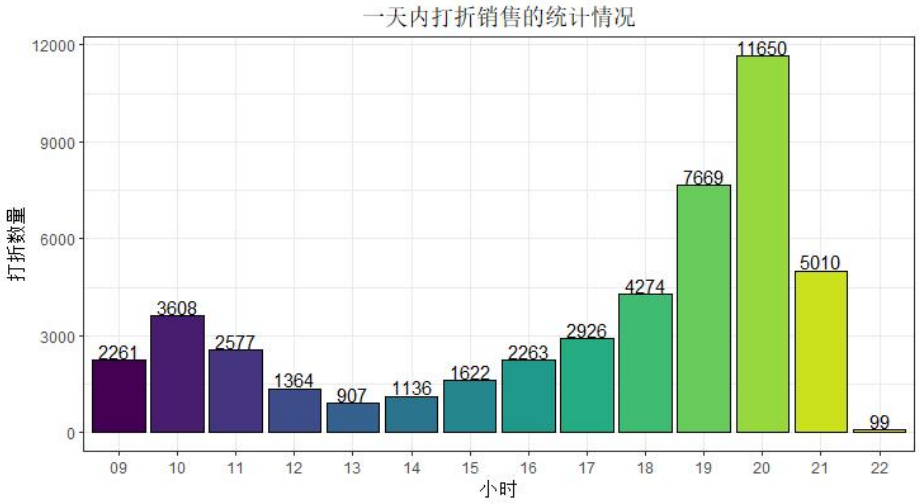


图 4 逐小时打折商品数量分布直方图

有关从两个可视化图中我们可看出销售量与打折数量有明显的正相关关系，这也可以解释为什么到了晚上各蔬菜品类的销售量反而都呈上升趋势。

5.2.2 销售量随价格变化分布规律

根据各品类单日总销售量与销售均价所绘制的散点图，如下图 5.2-4 所示。我们可知销售量与价格有一定的负相关关系。

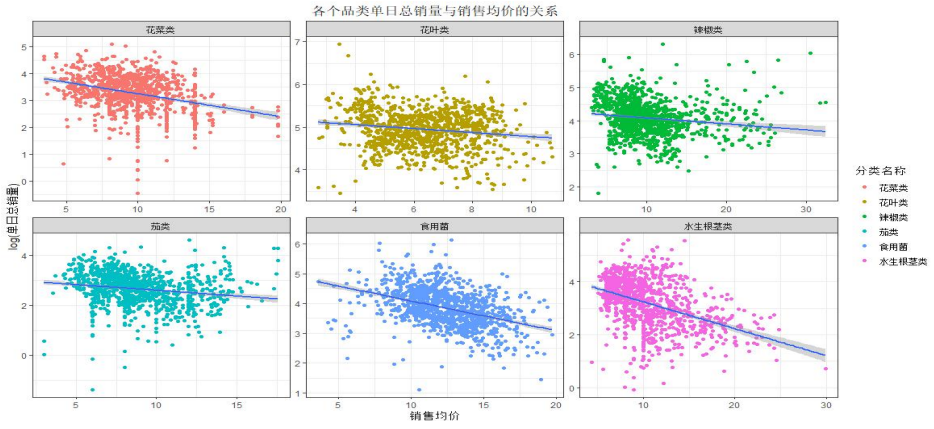


图 5 各个品类每日总销售量与销售均价的散点图

5.2.3 各蔬菜品类销量在打折与否条件下的分布规律

选取附件二将打折与否的销售量作出划分，再对异常值进行清洗。绘制销售量在打折与否条件下的箱型图。如下所示。

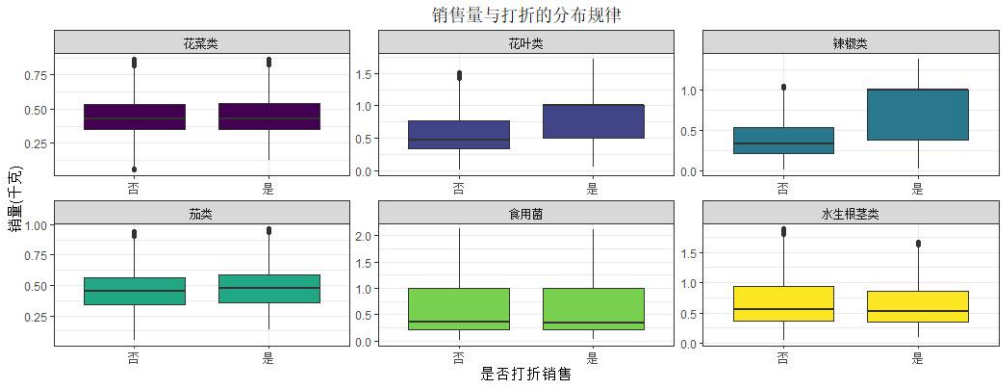


图 6 各蔬菜品类销量在打折与否条件下的箱型图

由上图可知花叶类以及辣椒类蔬菜销售量与打折与否的相关性更高且打折会导致销售量上升。其它类别在打折与不打折的条件下销售量总量都较为均衡。

但三年内的数据量过多我们可能很难看肉眼出销售量差距较小的数据，因此我们再对一天内打折销售的统计情况以及销售量在一天内的变化趋势进行综合分析，我们可以得出结论打折会导致销售量的上升。

5.3 各蔬菜品类之间的相互关系

5.3.1 比例关系

由于第二问要求在不知道具体品种只知道具体类别的情况下做出补货决策，因此我们对每个月各品类蔬菜比例关系进行可视化：

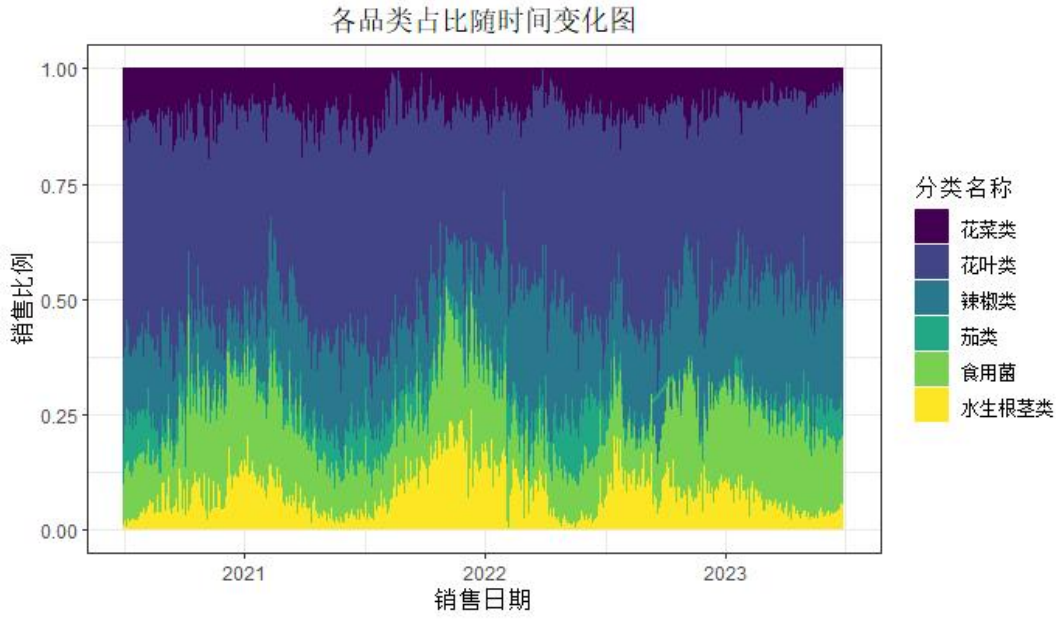


图 7 每月各蔬菜品类销量占比图

由图 8 分析可知，对于六种品类各自的占比，随年份具有明显周期性变化的有辣椒类、食用菌类和水生根茎类，而茄类、花菜类和花叶类无明显周期性。对于六种品类之间的占比关系，辣椒类、食用菌和水生根茎类的比例关系随时间推移维持稳定，花叶类和茄类受其他品类比例影响但大致稳定，而花菜类的占比相对独立并维持稳定。各蔬菜品类间的互补关系

将各蔬菜品类的每日的总销售量作为数据集的一个元素，构建 6 个数据集，利用

图 8 不同品类之间相关性热力图

spearman 相关系数法求解各品类销售量间的相关系数。输出结果如下图所示。

观察热力图可知花叶类与辣椒类蔬菜销售量以及食用菌类与辣椒类蔬菜相关性相对较高分别为 0.78、0.68。辣椒类蔬菜常常作为蔬菜辅助佐料进行烹饪。这也能解释为何互补商品中总有辣椒类蔬菜。

5.4 各单品间的相互关系

5.4.1 各单品间的替代关系

由于价格的涨幅会对销售量造成影响，二者成反比例关系。若单品之间存在替代关系，则一种单品的价格上涨将会使本单品的销售量下降，且使其替代单品销售量上升。从而我们可知单品销售量之间的涨幅率会存在一定的相关关系。且为线性相关。又因为各单品销售量为连续数据，且成正态分布。从而我们可利用皮尔逊相关系数确定变量之间的相关程度。

皮尔逊相关系数模型建立过程如下。

Step1 计算出每个单品每日的总销售额 TS。计算当日销售量相对前一日销售量的涨幅率，公式如下。

$$R_{1i}=\frac{TS_{1i}-TS_{1i-1}}{TS_{1i-1}},i=1,2,3.....n \tag{1}$$

从而得到六组数据$R_1:\{R_{11},R_{12}.....R_{1n}\}$，$R_2:\{R_{21},R_{22}.....R_{2n}\}$，……，$R_6:\{R_{61},R_{62}.....R_{6n}\}$

Step2 计算样本均值。

$$\overline{R_j}=\frac{\sum R_{ji}}{n},j=1,2,.....,6\ \ i=1,2,3.....n \tag{2}$$

Step3 计算样本协方差。

$$\mathrm{Cov}\ (R_m,R_n)=\frac{\sum_{i=1}^{n}(R_{mi}-\overline{R_m})(R_{ni}-\overline{R_n})}{n-1} \tag{3}$$

Step4 得出样本 Pearson 相关系数。

$$\text{样本 Pearson 相关系数:}\ r_{R_m,R_n}=\frac{\mathrm{Cov}\ (R_m,R_n)}{S_{R_m}S_{R_n}} \tag{4}$$

其中S_{R_m}是R_m样本的标准差。

将处理好的数据导入 Rstudio 从而得出各单品销售量涨幅率的相关系数。选取相关系数大于 0.85 的的单品对，由于其相关性较高且成正相关，即一种单品销售量的增长将导致另一种单品销售量的增长。从而它们可看做具有替代关系。下表将列出各品类中具有替代关系的单品。以下只给出部分数据，完整数据见支撑材料。

表 1 品类内各单品间的替代关系

品类名称	替代商品	
花叶类	木耳菜	白菜苔
	…	…
	娃娃菜	双沟白菜
辣椒类	小皱皮	紫尖椒
茄类	圆茄子	青茄子
	青茄子(1)	圆茄子(1)
食用菌	金针菇(盒)	海鲜菇(2)
	…	…
	海鲜菇(2)	金针菇(盒)

5.4.2 单品间的互补关系

与替代关系的求解流程类似，求出各单品的正相关系数大于 0.6 的商品组，以下给出部分互补商品，完整结果见支撑材料。

表 2 单品间的互补关系

单品名称	互补商品
红椒(1)	小米椒
小米椒	红椒(1)
…	…
紫茄子(1)	洪湖莲藕(脆藕)，青茄子(2)
金针菇(份)	竹叶菜(份)
野藕(2)	黑油菜，上海青(份)

六、 问题二模型的建立与求解

6.1 单元线性回归模型的建立

由于附件所给数据仅有销量、价格、品类这几方面数据，又基于成本加成定价法的约束，我们无需考虑运输成本、运营成本、市场竞争等因素，仅仅依赖于成本和利润的考量。因此固定品类之后销量看作只与价格相关。由于问题仅要求对各个品类单日总销售量及定价进行分析，因此本文不考虑不同品类之间相互影响。从而本文采用单变量回归对各蔬菜品类的销售总量与成本加成定价的关系进行拟合。

其次在绘制各品类蔬菜销量与定价之间的散点图，可以看出二者有一定的线性关系。因此本文采用单元线性回归对各蔬菜品类的销售总量与成本加成定价的关系进行拟合。模型建立如下。

Step1 对销售总量X_i数据进行 log 转化使其满足正态分布。

Step2 假设某蔬菜品类每日总销售量与定价之间的线性关系为

$$lnX_i = a_iP_i + b_i,\ j = 1,2,\ldots\ldots,6\ \ i = 1,2,\ldots\ldots,6 \tag{1}$$

Step3 构建代价函数

$$J(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(a_ix^{(i)} - y^{(i)}\right)^2 \tag{2}$$

Step4 求解代价函数最小值，从而确定a_i，再带回确定最优b_i。从而得出线性回归拟合曲线。

拟合结果如下图 9 所示。拟合曲线也在图中呈现。

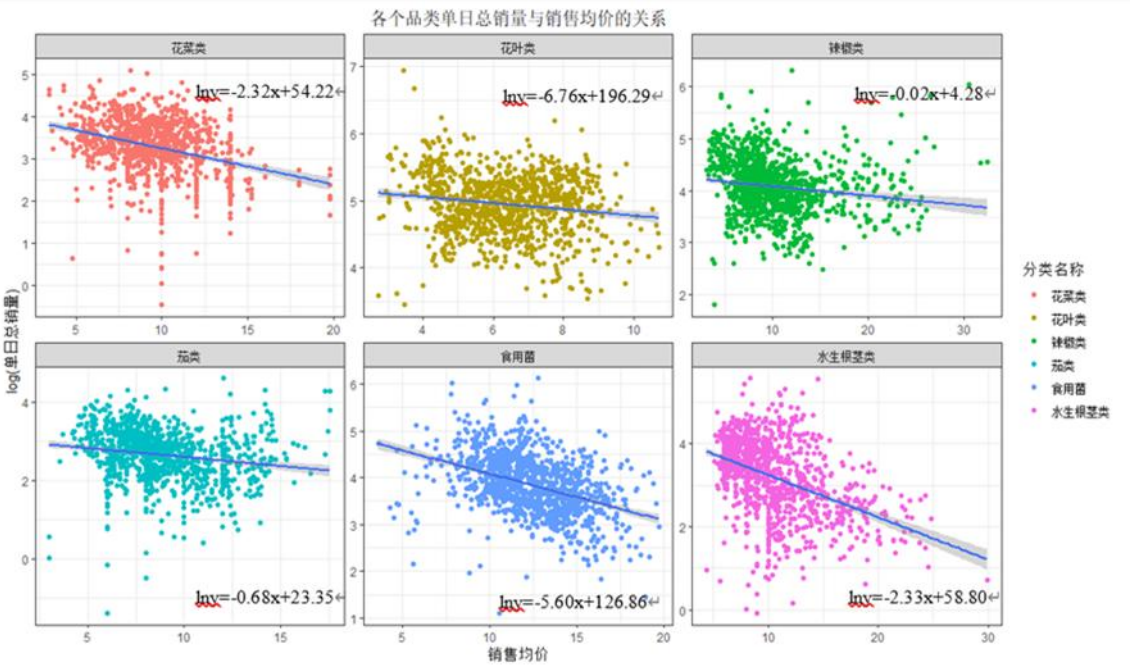


图 9 各品类单日总销售量与销售均价的关系

6.2 销售量与价格间的多元线性回归分析

对商超总利润进行优化模型的建立时，需考虑到一种蔬菜品类销量不仅与自身价格有关，还与其它蔬菜品类的销售量以及价格相关，若不考虑与其它蔬菜品类指标的相互影响则会导致约束条件过弱从而使得优化结果较差无法得到全局最优解。假设假设某蔬菜品类每日总销售量与定价之间的线性关系可如下式表示。

$$X_i = a_i P_i + \sum_{j \neq i} \left(b_j P_j + c_j X_j\right) + d_i \text{，} \ j = 1,2,\ldots\ldots,6 \ \ i = 1,2,\ldots\ldots,6 \tag{1}$$

利用多元线性回归分析对销售量及价格之间的线性关系进行拟合。X_1表示花菜类蔬菜单日销售总量，X_2表示花叶类蔬菜单日销售总量，X_3表示辣椒类蔬菜单日销售总量，X_4表示茄类蔬菜单日销售总量，X_5表示食用菌单日销售总量，X_6表示水生根茎类蔬菜单日销售总量。P_1表示花菜类蔬菜单日销售总量，P_2表示花叶类蔬菜单日销售总量，P_3表示辣椒类蔬菜单日销售总量，P_4表示茄类蔬菜单日销售总量，P_5表示食用菌单日销售总量，P_6表示水生根茎类蔬菜单日销售总量。

拟合结果如下表所示。具体结果见支撑材料。

花菜类	$X_1 = -2.48P_1 + 2.34P_2 + 0.07X_2 + \cdots + 0.14P_6 + 0.22X_6$
花叶类	$X_2 = -9.61P_2 + 2.61P_1 + 0.78X_1 + \cdots + -1.41P_6 + 0.28X_6$
辣椒类	$X_3 = -3.81P_3 + 2.48P_2 + 0.14X_2 + \cdots + 0.99P_6 + 0.26X_6$

茄类	$X_4 = -1.54P_4 + 0。04P_1 + 0.07X_1 + \cdots + 1.16P_6 + -0.05X_6$
食用菌	$X_5 = -5.78P_5 + 2.10P_1 + 0.00X_1 + \cdots \pm 1。06P_6 + 0.34X_6$
水生根茎	$X_6 = -1.34P_6 + 1.34P_1 + 0.04X_2 + \cdots + 1.43P_5 + 0.18X_5$

6.3 ARIMA 模型的建立

模型建立流程图如下所示

Step1 数据分析 通过对历史销售量以及历史进货价格的时间序列图的分析，我们发现两组数据均不是平稳白噪声序列，因此需进一步对数据进行转换。

Step2 数据转换 绘制原始各品类单日销量分布图，如下所示。课件每个品类单品销售分布都右偏。再进行 log 转换后数据呈相对标准的正态分布如下所示。

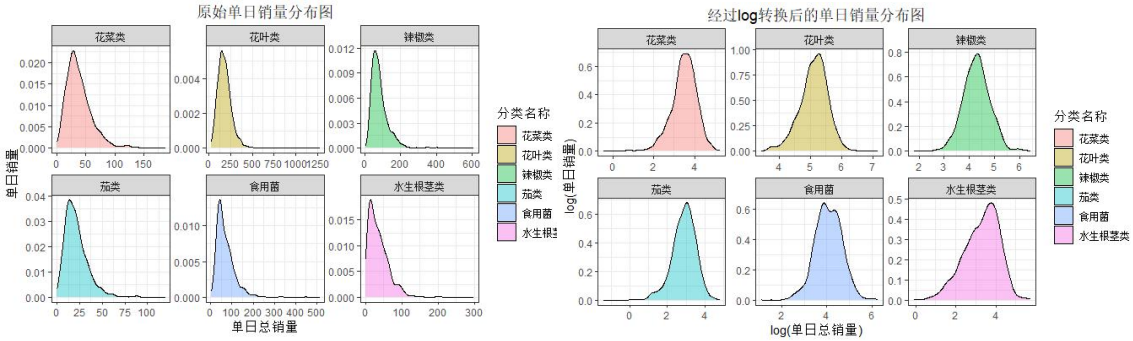


图 11 数据转换前后销量分布图

6.4 ARIMA 模型的求解

通过对模型的求解得出各品类蔬菜 1 日到 7 日的进价。

花菜	水生根茎类	花叶	食用菌	辣椒	茄类	日期
7.89	12.55	3.07	7.56	4.66	4.94	1
7.89	12.55	3.08	7.44	4.66	4.94	2
7.95	12.55	3.07	7.36	4.66	4.94	3
7.87	12.55	3.08	7.30	4.66	4.94	4
7.91	12.55	3.07	7.27	4.66	4.94	5
7.90	12.55	3.07	7.24	4.66	4.94	6
7.90	12.55	3.08	7.23	4.66	4.94	7

6.5 优化模型的建立

6.5.1 目标函数的确定：总利润的计算

第 i 类蔬菜的每日的总利润Pro_i，与第 i 类蔬菜的每日预期定价P_i、第 i 类蔬菜的每日预期销量X_i、第 i 类蔬菜的每日进价C_i以及进货量Q_i相关。PRO为每日总利润，从而目标函数的计算公式如下

$$PRO = \sum_{i=1}^{6} P_i * X_i - C_i * Q_i \; i = 1,2,\ldots\ldots,6 \tag{1}$$

第 i 类蔬菜的每日预期销量X_i作为决策变量进行优化。

6.5.2 约束条件的确定

a. 销量 X 与定价 P 的约束关系

销售量X_i不仅与自身定价有关还和其它品类蔬菜定价P_j以及销售量X_j相关。通过上述对各蔬菜品类的销售总量X_i与自身定价P_i、其它蔬菜品类定价P_j以及其它蔬菜品类销售量X_j的关系进行线性回归拟合分析，我们已知有以下数学关系。a_i 、a_j 、b_i为所求相关系数。

$$X_i = a_i P_i + \sum_{j \neq i} \left(b_j P_j + c_j X_j\right) + d_i \text{，} \; j = 1,2,\ldots\ldots,6 \;\; i = 1,2,\ldots\ldots,6 \tag{1}$$

b. 销售量X_i与进货量Q_i的约束关系

由于我们不知道商超每日进货量，无法拟合销售量与进货量之间的关系，因此我们将二者关系如下，其中d_i表示折损率，根据所提供信息我们可将折损率看成一个定值。关系表示如下

$$Q_i = \frac{X_i}{1 - d_i} \tag{2}$$

c. 成本加成定价法[1]

依据成本加成定价法可知定价P_i以及利润率PR_i之间的关系,关系表示如下。

$$P_i * X_i = Q_i * C_i * (1 + PR_i) \tag{3}$$

d.商超容量约束

销售量当然是越大越好，但考虑到商超的容量问题，销售量应控制在一定的数量范围之中。由于我们没有此商超规模的具体数据，我们没有更多的相关条件去判断商超容量，因此利用历史数据中的蔬菜品类总销售量对 2023 年 1-7 日蔬菜总销量进行预测，对于预测值给定 95%的置信区间。

$$\sum_{i=1}^{6} X_i^{min} \leq \sum_{i=1}^{6} X_i \quad \leq \sum_{i=1}^{6} X_i^{max} \tag{4}$$

e. 市场需求的约束

因为条件要求所制定的策略需要满足市场对各类蔬菜品类的需求，因此我们要将各蔬菜品类销量X_i在蔬菜销售总量中的占比r_i约束在一个合理区间内。由于我们没有更多相关条件去预测各蔬菜品类销售总量的占比，考虑到商超对市场需求量已有一定把握，本文选择将往年七月份各蔬菜品类销量X_i在蔬菜销售总量中的平均占比$\bar{r}_i$作为销售量占比r_i及约束条件如下式所示。

$$\begin{gathered} r_i = \frac{X_i}{\sum_{i=1}^{6} X_i} \\ r_i = \bar{r}_i \end{gathered} \tag{5}$$

6.5.3 确立优化模型

综合以上所有分析，我们得出优化模型，如下式所示。

$$\max_{X_i} \sum_{i=1}^{6} P_i * X_i - C_i * Q_i \;\; i = 1,2,\ldots\ldots,6$$

$$s.t. \begin{cases} X_1 = a_1 P_1 + \sum_{j\neq 1} \left(a_j P_j + b_j X_j\right) + c_1 \\ X_2 = a_2 P_2 + \sum_{j\neq 2} \left(a_j P_j + b_j X_j\right) + c_2 \\ \cdots \\ X_6 = a_1 P_6 + \sum_{j\neq 6} \left(a_j P_j + b_j X_j\right) + c_6 \\ \dfrac{X_i}{\sum_{i=1}^{6} X_i} = \bar{r}_i \\ \sum_{i=1}^{6} X_i^{min} \leq \sum_{i=1}^{6} X_i \leq \sum_{i=1}^{6} X_i^{max} \\ Q_i = \dfrac{X_i}{1 - d_i} \end{cases} \tag{1}$$

6.5.4 遗传算法对优化模型的求解

由于目标函数为非凸函数，同时是一个非线性规划问题，因此我们选择采用遗传算法对目标函数进行求解。

遗传算法求解过程

Step1 设定初始状态，参数以及编码，导入数据

Step2 初始化开始迭代进化

Step3 对可行解进行选择、重组、变异和合并

Step4 依据目标函数值的大小分配适应度值

Step5 计算当代最优个体序号

Step6 重复 3、4、5 三个步骤，直至进化到符合最优或达到最大遗传代数

Output ： 各蔬菜品类销售量及定价最优化结果

	花菜类	水生根茎类	花叶类	食用菌	辣椒类	茄子类	销售总量
最优补货量 kg	32.79	24.79	189.69	39.17	101.43	35.56	1368.07
最优定价 元/kg	12.18	18.84	6.12	14.66	9.38	9.90	

七、 问题三模型的建立与求解

7.1 品种数目的初步筛选

问题三中要求使用 2023 年 6 月 24-30 日的可售品种，经过数据整理一共有 49 个品种可供选择，首先对每个品种的定价和销量进行线性拟合，由于对于某一单品，其定价越高，销量越低，故将拟合后直线斜率大于零的单品剔除，最终得到 36 个品种。

接着，本文利用 TOPSIS 算法，根据品种的利润率、销量、打着次数、损耗率四个指标进行重要性评价，该算法求解流程如下：

TOPSIS 算法求解流程

Step 1 将数据导入 Python ，根据熵权法得到利润率、销量、打折次数、损耗率共四个指标的权重

Step 2 对四个指标数据进行指标属性同向化处理并构造归一化初始矩阵

Step 3 确定最优方案和最劣方案

Step 4 根据欧几里得距离计算各评价对象与最优方案、最劣方案的接近程度

Step 5 计算各评价对象的综合评价指数

Step 6 根据综合评价指数进行排序，给出评价结果

Output ：各供货商 TOPSIS 评价结果

进行 TOPSIS 算法对重要性排序后，我们得到 36 个品种的排序关系，之后取出重要性位于前 33 名的单品当作进行优化模型的变量。相关表格如下，仅给出前 18 项数据部分，完整数据见支撑材料。

表 3 重要性排名前 33 类单品

小米椒(份)	云南油麦菜(份)	芜湖青椒(1)	竹叶菜	小皱皮(份)	西兰花
螺丝椒(份)	娃娃菜	双孢菇(盒)	紫茄子(2)	苋菜	海鲜菇(包)
螺丝椒	净藕(1)	洪湖藕带	小青菜(1)	红薯尖	西峡花菇(1)

7.2 补货与定价方案约束条件的更新

为了能得到每个品种在 7 月 1 号补货量和定价的最优方案，我们引入了多种约束条件，令商超在当日可以获得最大收益。

由问题二和函数关系可知：在各品类取得某一销量时，商超收益最大，此时的销量被成为最优销量。因此，问题三的求解利用问题二中各品类在 7 月 1 号的最佳销量，作为品类内部各个品种的销量总和约束，并且这一约束也包含了问题二优化模型中不同品类间的销量和定价影响约束。在此基础上，我们建立第 i 个品类最大收益目标子模型，如下式所示：

$$\max_{X_{ij}} \sum_{j=1}^{n_i} P_{ij} * X_{ij} - C_{ij} * Q_{ij} \quad j = 1,2,\ldots\ldots,n_i \tag{1}$$

在该函数中，每一个单品的销量之和等于每一个品种的最优，因此，我们可以得到关于销量的非线性优化方程。该方程有以下约束条件：

a.销量的约束

我们将第二问中求得每个品种最优销量设为S_i^{max},得到单品销量的约束关系如下：

$$\sum_{i=1}^{6} \sum_{j=1}^{n_i} X_{ij} \quad = S_i^{max} \tag{2}$$

b.市场需求约束

第 i 类品类蔬菜销量总数为$\sum_{j=1}^{n_i} X_{ij}Y_{ij}$，所有单品销量总数为$\sum_{i=1}^{6}\sum_{j=1}^{ni} X_{ij}Y_{ij}$，第 i 类单品销量固定占比为$\bar{r}_i$。从而给出下列约束条件的计算方式。

$$\frac{\sum_{j=1}^{n_i} X_{ij}Y_{ij}}{\sum_{i=1}^{6}\sum_{j=1}^{ni} X_{ij}Y_{ij}} = \bar{r}_i \tag{3}$$

c.销量的约束

因此利用历史数据中的蔬菜总销量对 2023 年 1-7 日蔬菜总销量进行预测，对于预测值给定 95%的置信区间。从而得出总销量最大值约束以及最小值约束。

$$\sum_{i=1}^{6}\sum_{j=1}^{n_i} X_{ij}^{min} \leq \sum_{i=1}^{6}\sum_{j=1}^{n_i} X_{ij} \leq \sum_{i=1}^{6}\sum_{j=1}^{n_i} X_{ij}^{max} \tag{4}$$

d. 销量 X 与定价 P 的约束关系

由于单品数量过多，若考虑其他单品价格以及销量对所讨论单品销售量X_{ij}及定价P_{ij}产生的影响，则拟合维数过高，无法求解也无法对其进行合理解释。因此本文仅考虑单品自身销量及定价之间的约束关系。m_{ij}、n_{ij}为二者拟合系数。

$$X_{ij} = m_{ij}P_{ij} + n_{ij} \tag{5}$$

接下来分析更新的约束条件。

e.单品数量约束

根据题目条件可售单品总数需控制在 27-33 个，此条件对指示变量Y_{ij}进行了约束。如下式

$$27 \leq \sum_{i=1}^{6}\sum_{j=1}^{n_i} Y_{ij} \leq 33 \tag{6}$$

f.最小陈设量约束

由题意知，第 i 类蔬菜品类内第 j 类单品销售量X_{ij}

$$2.5 \leq X_{ij} \tag{7}$$

g.各品类占比约束

$$\frac{\sum_{j=1}^{n_i} X_{ij}Y_{ij}}{\sum_{i=1}^{6}\sum_{j=1}^{n_i} X_{ij}Y_{ij}} = \bar{r}_i \tag{8}$$

7.3 模型汇总

综合以上，我们可得出某一品类的最终优化模型。

$$max \sum_{i=1}^{6}\sum_{j=1}^{n_i} Y_{ij} * （P_{ij} * X_{ij} - C_{ij} * Q_{ij}）$$

$$s.t. \begin{cases} X_{ij} = m_{ij}P_{ij} + n_{ij} \\ \sum_{i=1}^{6}\sum_{j=1}^{n_i} X_{ij} = S_i^{max} \\ \dfrac{\sum_{j=1}^{n_i} X_{ij}Y_{ij}}{\sum_{i=1}^{6}\sum_{j=1}^{ni} X_{ij}Y_{ij}} = \bar{r}_i \\ 27 \le \sum_{i=1}^{6}\sum_{j=1}^{n_i} Y_{ij} \le 33 \\ X_{ij} \ge 2.5 \\ Q_i = \dfrac{X_i}{1-d_i} \\ \sum_{i=1}^{6}\sum_{j=1}^{n_i} X_{ij}^{min} \le \sum_{i=1}^{6}\sum_{j=1}^{n_i} X_{ij} \le \sum_{i=1}^{6}\sum_{j=1}^{n_i} X_{ij}^{max} \end{cases} \quad (1)$$

由于该模型为某一品类的优化模型，所以我们需要分别对六种蔬菜品类进行求解。

7.4 模拟退火算法对模型的求解

相关算法的步骤和求解过程可见问题二模型求解，由该模型得到的最优解即

	菠菜	菜心	大白菜	苋菜	小皱皮（份）	紫茄子（2）	白玉菇（袋）	芜湖青椒（1）	金针菇（1）
补货量	4.354	4.985	17.426	4.864	4.834	17.203	3.945	35.826	7.837
利润率（元/千克）	3.804	7.396	0.849	1.798	1.974	2.963	2.857	2.659	3.749
利润（元）	16.573	11.084	14.863	8.43	7.834	49.745	9.263	78.655	23.884
	云南生菜	黄白菜（2）	红薯尖	云南生菜（份）	西兰花	青茄子（1）	双孢菇（盒）	螺丝椒（份）	
补货量	14.54	7.489	5.389	13.953	19.78	3.65	7.947	7.629	
利润率（元/千克）	3.252	2.877	2.493	1.583	3.867	2.9	1.863	2.843	
利润（元）	45.962	21.953	12.43	23.345	64.174	8.63	14.825	17.239	
	甜白菜	云南油麦菜	竹叶菜	娃娃菜	净藕（1）	红椒（1）	金针菇（盒）	西峡花菇（1）	
补货量	4.281	9.4	6.613	14.862	20.629	3.815	27.845	2.578	
利润率（元/千克）	3.22	2.945	1.953	1.755	2.285	6.482	1.37	7.492	
利润（元）	13.318	25.39	12.963	25.185	46.285	20.272	37.847	19.478	
	上海青	小青菜（1）	奶白菜	云南油麦菜（份）	长线菇	螺丝椒	小米椒（份）	杏鲍菇（1）	
补货量	6.894	2.983	5.431	8.993	2.895	4.716	9.893	3.719	
利润率（元/千克）	3.055	2.481	1.963	1.75	4.184	3.867	2.855	4.129	
利润（元）	21.247	7.425	10.335	13.856	11.33	25.893	25.134	13.885	

为商超在 7 月 1 日获得最大收益的每一蔬菜品种的销量，由销量可得 7 月 1 日商家最优的定价策略和补货策略。

八、 问题四模型的建立与求解

题目要求采集更多相关数据以更好地制定蔬菜商品的补货和定价决策。查阅资料可知商超对生鲜商品补货与定价策略通常考虑采集以下数据：

进货量：有了进货量数据我则可以根据销售量数据了解商超库存水平，避免过度库存或库存紧张。由于本题没有提供每日进货数量，因而本文只能对每日进货数量以及每日销量，通过已知折损率构建简单的线性相关关系。这样建构的关系太过理想化，难以随现实情况的变化而改变。若有进货量数据则可以对进货量及销售量进行回归分析，得到更符合现实规律的相关关系，从而优化约束，进而优化已有模型。

更详细损耗数据：虽然有些蔬菜保鲜期较短，品相随时间推移而变差。但不同单品的保鲜周期互不相同，消费者对各单品品质损耗程度的接受度也有所不同。比如消费者对生菜等花叶类蔬菜品质损耗接受度很低，但对水生根茎类蔬菜品质损耗接受度较高。因此更详细损耗数据应包括单品保鲜周期，消费者对损耗接受程度。这样更能制定合理的补货策略

季节性因素与天气数据：天气和季节对蔬菜的需求和供给有较大影响，由第一问对蔬菜类商品的销售量时序图分析可知，蔬菜类商品通常都有季节性趋势，即在相应的季节某些商品需求量会上升，从而影响其它蔬菜类商品的需求。天气会影响消费者的消费欲望例如天气过于炎热时便会减少出门购物的可能性从而导致商品销量的降低。因此第二题的天气状况也因作为补货量决策的相关数据。

地理位置：商超的所处地点会对商品销量及销量稳定性产生较大影响，若商超处在较偏远地区则消费者群体较为稳定，从而可降低对其它变量对销售量的影响。

促销活动与节假日数据：促销活动与节假日通常都会对商品销量造成影响，节假日直接影响购物的人流量以及购物的时间安排。在节假日期间购物人流量将大大提高，从而导致商品销售量的增长。同样促销活动的举办也将对销售量产生影响，若有促销折扣率与销量的相关关系则可以对商品定价进行进一步优化。制定更加补货量及定价决策。

消费者用户画像：用户画像能更好了解消费者消费习惯以及消费类型，从而为客户提供更加适合的消费体验。例如针对不同年龄段、不同性别、不同消费观

念的人群进行分类。从而为各类人群提供相应的商品类型、定价策略、商品供应量。从而优化定价决策。

是否存在竞争商家及竞争商家定价策略：竞争商家势必会互相争抢客源，若对同种品质商品，竞争商家定价更低，那么本商超有两种价格制定规则，1.保持原价，利润不变但销售量将降低。2.降价，利润降低但销售量影响较小。无论何种定价策略都将对销售额产生影响，因此需不断获取数据动态调整定价方案。

九、 模型的检验与分析

十、 模型的评价、改进与推广

10.1 模型的优点

1.从多方面的指标去探究品类的销量分布

2.在预测时选用多种模型，综合考虑取得效果最佳的模型

3.在建立优化模型约束条件时考虑到不同品类间或者不同单品间销量和定价的影响。

4.通过选取不同的指标对决策变量进行多重筛选，降低优化模型求解复杂度

10.2 模型的缺点

1.在进行问题二优化求解时没有考虑打折与销量之间的关系，没有引入打折与否的约束因素。

2.使用的 TOPSIS 方法认为各个指标之间是互相独立的，忽略了指标之间的相互关系。

3.在进行多元线性回归时没有考虑销量和价格之间的交互效应，没有进一步提高拟合度。

10.3 模型的改进

1.在建立优化模型时可以加上对定价或者销量的正则化项，可能加快收敛。

2. 在建立优化模型时可以加入有关于打折的约束因素。

十一、 参考文献

[1]. 姜爽,沈烈志,金玉然.基于成本加成定价法的玉件产品定价模型研究[J].商场现代化,2007(13):66-67.时令蔬菜抢“鲜”上市 全国餐饮业呈现较强复苏势头

[2]. 周跃.基于遗传算法的机器人路径规划优化研究[J].电子元器件与信息技术,2023,7(06):65-68.DOI:10.19772/j.cnki.2096-4455.2023.6.017.

[3]. 卢静. 生鲜农产品的库存控制及动态定价研究[D].天津大学,2021.DOI:10.27356/d.cnki.gtjdu.2019.000182.

[4]. 曾敏敏. 基于时间情境A生鲜社区超市的动态定价策略研究[D].西南财经大学,2022.DOI:10.27412/d.cnki.gxncu.2021.002858.

十二、 附录

附录 2

介绍：问题一代码

```
    library(readxl)
library(tidyverse)

# 读入数据
df1 <- read_excel("C:/Users/Lenovo/Desktop/国赛数据/附件 1.xlsx")
df2 <- read_excel("C:/Users/Lenovo/Desktop/国赛数据/附件 2.xlsx")
df3 <- read_excel("C:/Users/Lenovo/Desktop/国赛数据/附件 3.xlsx")
df4 <- read_excel("C:/Users/Lenovo/Desktop/国赛数据/附件 4.xlsx")
df5 <- read_excel("C:/Users/Lenovo/Desktop/国赛数据/蔬菜品种整理.xlsx")


# 合并数据框
df_merge <- inner_join(df2,df1,by = "单品编码")
df3 <- df3%>%rename(销售日期 = 日期)
df_merge_all <- df_merge%>%inner_join(df3,by = c("单品编码","销售日期"))
write_xlsx(df_merge,"合并版.xlsx")
# 按照日期和商品编码进行分类，统计每个商品每天的销售量
df_merge_allgroups <- df_merge%>%filter(销售类型!="退货")%>%group_by(销售日期,单品名称)%>%summarise(`销售总量` = sum(`销量(千克)`))%>%ungroup()

#统计各个蔬菜品种的情况
df_mean_allgroups<-df_merge%>%filter(销售类型!="退货",是否打折销售!="是")%>%group_by(分类名称,销售日期)%>%filter(`销量(千克)` >= quantile(`销量(千克)`, 0.25) - 1.5 * IQR(`销量(千克)`) &

`销量(千克)` <= quantile(`销量(千克)`, 0.75) + 1.5 * IQR(`销量(千克)`))%>%summarise(mean_days = mean(`销量(千克)`))%>%ungroup()
df_dingjia <- df_merge%>%group_by(分类名称,销售日期)%>%summarise(销售均价 = mean(`销售单价(元/千克)`))%>%ungroup()


df_wide <- df_merge_allgroups%>%pivot_wider(names_from = `单品名称`,values_from = `销售总量`)
# 处理缺失值
df_wide_clean <- as.data.frame(apply(apply(df_wide,2,function(x) ifelse(is.na(x),0,x))[,-1],2,function(x) as.numeric(x)))
#根据涨幅计算相关性
```

```r
df_wide_diff <- df_wide_clean%>%mutate_all(~ .-lag(.,default = first(.)))
cor_per_items <- cor(df_wide_diff)

# 查找每个变量对应相关性最强的几个变量
find_top_pos_cor <- function(row, threshold = 0.6) {
  # 找出负相关系数大于阈值的索引
  pos_cor_indices <- which(row > threshold & row!=1)

  # 如果没有符合条件的负相关，则返回空向量
  if (length(pos_cor_indices) == 0) {
    return(NULL)
  }

  # 对符合条件的负相关系数进行排序
  sorted_pos_cor <- sort(row[pos_cor_indices], decreasing = TRUE)

  # 找出前三个负相关系数的索引
  top_pos_cor_indices <- pos_cor_indices[1:min(3,
length(pos_cor_indices))]

  # 找出前三个负相关系数的变量名称
  top_pos_cor_variables <- colnames(cor_per_items)[top_pos_cor_indices]

  return(top_pos_cor_variables)
}

# 创建一个空的数据框来存储结果
result <- data.frame(单品名称 = colnames(cor_per_items),
Top_Positive_Correlations = NA)

# 遍历相关性系数矩阵的每一行
for (i in 1:nrow(cor_per_items)) {
  # 找出每一行负相关最大的三个变量
  top_pos_cor_vars <- find_top_pos_cor(cor_per_items[i,])

  # 如果找到了符合条件的变量，则将其添加到结果数据框中
  if (!is.null(top_pos_cor_vars)) {
    result$Top_Positive_Correlations[i] <- paste(top_pos_cor_vars,
collapse = ", ")
  }
}

# 移除没有找到负相关的行
```

```
result <- result[!is.na(result$Top_Positive_Correlations), ]
colnames(result) <- c("单品名称","捆绑商品")
write_xlsx(result,"C:/Users/Lenovo/Desktop/国赛代码/各单品间的捆绑关
系.xlsx")


# 将结果整理成数据框
result_df <- data.frame("单品名称" = colnames(cor_per_items), first =
first_element,second = second_element,third = third_element)

#统计各个蔬菜品种的销量随时间变化的情况
ggplot(data = df_mean_allgroups,aes(x = `销售日期`,y = mean_days,color = `
分类名称`))+
  geom_line(size = 0.5)+
  facet_grid(`分类名称`~.,scales = "free")+
  labs(title = "各个品种销售量随时间变化图")+
  scale_color_brewer(palette = "Set2")+
  theme_bw()+
  theme(plot.title = element_text(hjust = 0.5))

#不同菜品的相互关系
#统计各个蔬菜品种销量之间的比例关系
#按月份比较各种菜品之间的比例关系
df_merge <- df_merge%>%mutate(month = lubridate::month(销售日期),year =
lubridate::year(销售日期),yearmonth = paste(year,month))
df_month <- df_merge%>%filter(销售类型 != "退货")%>%group_by(销售日期,分类
名称,month,yearmonth)%>%summarise(销量 = sum(`销量(千克)`))%>%
  ungroup()%>%group_by(yearmonth,分类名称,month)%>%summarise(总销量 =
sum(销量))%>%
  ungroup()%>%group_by(month,分类名称)%>%mutate(月均销量 = mean(总销量))
df_month <- df_month%>%select(month,分类名称,月均销
量)%>%distinct()%>%group_by(month)%>%mutate(总销量 = sum(月均销量),比例 =
月均销量/总销量)
ggplot(data = df_month,aes(x = factor(month),y = 比例,fill = 分类名称))+
  geom_bar(stat = "identity",color = "black",width = 0.7)+
  geom_text(aes(label = paste0(sprintf("%.0f", 比例*100), "%")),position =
position_stack(vjust = 0.5)) +  # 添加百分比标签
  theme_bw() +  # 移除背景和坐标轴
  labs(title = "每月各品类销量占比图")+
  scale_fill_brewer(palette = "Blues")+
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        plot.title = element_text(hjust = 0.5))
```

```r
#计算不同蔬菜品种之间的相关性系数(根据销量)
df_wide_group <- df_mean_allgroups%>%pivot_wider(names_from = `分类名称
`,values_from = mean_days)
df_wide_group <- as.data.frame(apply(df_wide_group,2,function(x)
as.numeric(ifelse(is.na(x),0,x)))[,-1])
cor_per_group <- cor(df_wide_group)
cor_per_group <- as.data.frame(as.table(cor_per_group))
# 绘制热图
ggplot(data = cor_per_group,aes(reorder(Var1,desc(Var1)),Var2,fill =
Freq))+
  geom_tile()+
  geom_text(aes(label = sprintf("%.2f",Freq)),color = "white")+
  theme_minimal()+
  xlab("")+ylab("")+
  labs(title = "根据销量计算不同品类之间的相关性")+
  theme(plot.title = element_text(hjust = 0.5))
##根据定价差价计算
df_wide_group <- df_dingjia%>%pivot_wider(names_from = `分类名称
`,values_from = 销售均价)
df_wide_group <- as.data.frame(apply(df_wide_group,2,function(x)
as.numeric(ifelse(is.na(x),0,x)))[,-1])
df_wide_group <- df_wide_group%>%mutate_all(~ .-lag(.,default = first(.)))
cor_per_group <- cor(df_wide_group[-1,])
cor_per_group <- as.data.frame(as.table(cor_per_group))
# 绘制热图
ggplot(data = cor_per_group,aes(reorder(Var1,desc(Var1)),Var2,fill =
Freq))+
  geom_tile()+
  geom_text(aes(label = sprintf("%.2f",Freq)),color = "white")+
  theme_minimal()+
  xlab("")+ylab("")+
  labs(title = "根据定价变化计算不同品类之间的相关性")+
  theme(plot.title = element_text(hjust = 0.5))

#分析打折商品中各个品类的占比
df_dazhe <- df_merge%>%filter(`是否打折销售`=="是")%>%mutate(总量 =
n())%>%group_by(分类名称)%>%summarise(每个品种的占比 = n()/总
量)%>%distinct()
ggplot(data = df_dazhe,aes(x= "",y = 每个品种的占比,fill = 分类名称))+
  geom_bar(stat = "identity",color = "black")+
  coord_polar(theta = "y")+
  scale_fill_viridis(discrete = TRUE)+
```

```
  theme_void()+
  geom_text(aes(label = paste(sprintf("%.2f",每个品种的占比),"%")),position = position_stack(vjust = 0.5),color = "white")+
  labs(title = "打折商品中各个种类的占比情况")+
  theme(plot.title = element_text(hjust = 0.5))
pie3D(df_dazhe$每个品种的占比,labels = paste(round(df_dazhe$每个品种的占比,2),"%",seq = "")
      ,explod = 0.1
      ,col = rainbow(length(df_dazhe$每个品种的占比))
      ,main = "打折商品中各个种类的占比情况")


#销售量和是否打折的关系图
filtered_data <- df_merge %>%filter(`销售类型` != "退货")%>%
  group_by(分类名称,是否打折销售) %>%
  filter(`销量(千克)` >= quantile(`销量(千克)`, 0.25) - 1.5 * IQR(`销量(千克)`) &
           `销量(千克)` <= quantile(`销量(千克)`, 0.75) + 1.5 * IQR(`销量(千克)`)) %>%
  ungroup()
ggplot(data = filtered_data,aes(x = `是否打折销售`,y = `销量(千克)`,fill = 分类名称))+
  geom_boxplot()+
  facet_wrap(.~分类名称,nrow = 2,ncol = 3,scales = "free")+
  theme_bw()+
  scale_fill_viridis(discrete = TRUE)+
  labs(title = "销售量与打折的分布规律")+
  theme(plot.title = element_text(hjust = 0.5))

#花叶类各种单品的销售量的时间序列图
df_huaye_danpin <- df_merge%>%filter(`分类名称` == "花叶类")%>%group_by(单品名称)%>%summarise(总销量 = sum(`销量(千克)`))%>%arrange(desc(总销量))%>%slice_head(n = 5)
df_huaye <- df_merge_all%>%filter(`分类名称` == "花叶类",单品名称 %in% df_huaye_danpin$单品名称)%>%mutate(yearmonth = paste(lubridate::year(销售日期),month))
df_huaye <- df_huaye%>%
  group_by(销售日期,单品名称)%>%mutate(`单日销售总量` = sum(`销量(千克)`))%>%ungroup()%>%
  group_by(yearmonth,单品名称)%>%summarise(`平均销量` = mean(单日销售总量))%>%ungroup()
ggplot(data = df_huaye,aes(x = yearmonth,y = `平均销量`,color = `单品名称`,group = 单品名称))+
```

```
  geom_line(size = 0.6)+
  scale_color_brewer(palette = "Set3")+
  scale_y_continuous(limits = c(0,80))+
  labs(title = "花叶类商品各个单品的销售量与时间的关系")+
  theme_bw()+
  theme(plot.title = element_text(hjust = 0.5))

#花菜类。。。。。
df_huacai <- df_merge%>%filter(`分类名称` == "花菜类")%>%group_by(销售日期,
单品名称)%>%summarise(`销售总量` = sum(`销量(千克)`))%>%ungroup()
ggplot(data = df_huacai,aes(x = `销售日期`,y = `销售总量`,color = `单品名
称`))+
  geom_line(size = 0.6)+
  scale_color_brewer(palette = "Set3")+
  scale_y_continuous(limits = c(0,80))+
  labs(title = "花菜类商品各个单品的销售量与时间的关系")+
  theme_bw()+
  theme(plot.title = element_text(hjust = 0.5))

#食用菌类
df_junlei <- df_merge%>%filter(`分类名称` == "食用菌")%>%group_by(销售日期,
单品名称)%>%summarise(`销售总量` = sum(`销量(千克)`))%>%ungroup()
ggplot(data = df_junlei,aes(x = `销售日期`,y = `销售总量`,color = `单品名
称`))+
  geom_line(size = 0.6)+
  scale_color_brewer(palette = "Set3")+
  scale_y_continuous(limits = c(0,80))+
  labs(title = "花菜类商品各个单品的销售量与时间的关系")+
  theme_bw()+
  theme(plot.title = element_text(hjust = 0.5))

##辣椒类
df_lajiao <- df_merge%>%filter(`分类名称` == "辣椒类")%>%group_by(销售日期,
单品名称)%>%summarise(`销售总量` = sum(`销量(千克)`))%>%ungroup()
ggplot(data = df_lajiao,aes(x = `销售日期`,y = `销售总量`,color = `单品名
称`))+
  geom_line(size = 0.6)+
  scale_color_brewer(palette = "Set3")+
  scale_y_continuous(limits = c(0,80))+
  labs(title = "花菜类商品各个单品的销售量与时间的关系")+
  theme_bw()+
  theme(plot.title = element_text(hjust = 0.5))
###茄类
```

```
df_qielei <- df_merge%>%filter(`分类名称` == "茄类")%>%filter(!`是否打折销售` == "是" & !`销售类型` == "退货")%>%group_by(销售日期,单品名称)%>%summarise(`销售总量` = sum(`销量(千克)`))%>%ungroup()
ggplot(data = df_qielei,aes(x = `销售日期`,y = `销售总量`,color = `单品名称`))+
  geom_line(size = 0.6)+
  scale_color_brewer(palette = "Set3")+
  scale_y_continuous(limits = c(0,80))+
  labs(title = "花菜类商品各个单品的销售量与时间的关系")+
  theme_bw()+
  theme(plot.title = element_text(hjust = 0.5))
### 水生根茎类
df_genjing <- df_merge%>%filter(`分类名称` == "水生根茎类")%>%filter(!`是否打折销售` == "是" & !`销售类型` == "退货")%>%group_by(销售日期,单品名称)%>%summarise(`销售总量` = sum(`销量(千克)`))%>%ungroup()
ggplot(data = df_genjing,aes(x = `销售日期`,y = `销售总量`,color = `单品名称`))+
  geom_line(size = 0.6)+
  scale_color_brewer(palette = "Set3")+
  scale_y_continuous(limits = c(0,80))+
  labs(title = "花菜类商品各个单品的销售量与时间的关系")+
  theme_bw()+
  theme(plot.title = element_text(hjust = 0.5))


#同类别内，不同单品之间的相互关系分析
#根据销量的涨幅，而非销量本身
#计算花叶类单品之间的相互关系
df_huaye <-  df_merge%>%filter(`分类名称` == "花叶类" & !(`销售类型` == "退货") & !(`是否打折销售` == "是"))%>%group_by(year,month,单品名称)%>%filter(`销量(千克)` >= quantile(`销量(千克)`, 0.25) - 1.5 * IQR(`销量(千克)`) &

`销量(千克)` <= quantile(`销量(千克)`, 0.75) + 1.5 * IQR(`销量(千克)`))%>%summarise(`销售总量` = mean(`销量(千克)`))%>%ungroup()
df_huaye <- df_huaye%>%pivot_wider(names_from = 单品名称,values_from = 销售总量)
date <- df_huaye[,c(1,2)]
df_huaye <- apply(df_huaye[,-c(1,2)],2,function(x)
as.numeric(ifelse(is.na(x),0,x)))
df_huaye <- cbind(date,df_huaye)
df_huaye_diff <- df_huaye%>%mutate_at(vars(-c(year,month)),
~ .-lag(.,default = first(.)))
```

```r
cor_huaye <- cor(df_huaye_diff[,-c(1,2)])

#找出除了自己以外最大的值
# 使用 apply 函数，对每一行应用一个自定义函数来找到最小的负数
min_negative_values <- apply(cor_huaye, 1, function(row) {
  # 使用 which 函数找到负数的索引
  negative_indices <- which(row < 0)

  # 如果存在负数，返回最小的负数，否则返回 NA
  if (length(negative_indices) > 0) {
    return(min(row[negative_indices]))
  } else {
    return(NA)
  }
})

# 打印结果
min_negative_values <- as.data.frame(min_negative_values)

# 定义一个函数，用于找出每一行负相关最大的三个变量
find_top_neg_cor <- function(row, threshold = 0.7) {
  # 找出负相关系数大于阈值的索引
  neg_cor_indices <- which(row < -threshold)

  # 如果没有符合条件的负相关，则返回空向量
  if (length(neg_cor_indices) == 0) {
    return(NULL)
  }

  # 对符合条件的负相关系数进行排序
  sorted_neg_cor <- sort(row[neg_cor_indices], decreasing = TRUE)

  # 找出前三个负相关系数的索引
  top_neg_cor_indices <- neg_cor_indices[1:min(3,
length(neg_cor_indices))]

  # 找出前三个负相关系数的变量名称
  top_neg_cor_variables <- colnames(cor_huaye)[top_neg_cor_indices]

  return(top_neg_cor_variables)
}

# 创建一个空的数据框来存储结果
```

```r
result_huaye <- data.frame(单品名称 = colnames(cor_huaye),
Top_Negative_Correlations = NA)

# 遍历相关性系数矩阵的每一行
for (i in 1:nrow(cor_huaye)) {
  # 找出每一行负相关最大的三个变量
  top_neg_cor_vars <- find_top_neg_cor(cor_huaye[i,])

  # 如果找到了符合条件的变量，则将其添加到结果数据框中
  if (!is.null(top_neg_cor_vars)) {
    result_huaye$Top_Negative_Correlations[i] <- paste(top_neg_cor_vars,
collapse = ", ")
  }
}

# 移除没有找到负相关的行
result_huaye <-
result_huaye[!is.na(result_huaye$Top_Negative_Correlations), ]
colnames(result_huaye) <- c("单品名称","互补商品")
write_xlsx(result_huaye,"C:/Users/Lenovo/Desktop/国赛代码/花叶类单品间的
替代关系.xlsx")



#计算花菜类单品之间的相互关系
df_huacai <-  df_merge%>%filter(`分类名称` == "花菜类" & !(`销售类型` == "
退货") & !(`是否打折销售` == "是"))%>%group_by(year,month,单品名
称)%>%filter(`销量(千克)` >= quantile(`销量(千克)`, 0.25) - 1.5 * IQR(`销
量(千克)`) &

`销量(千克)` <= quantile(`销量(千克)`, 0.75) + 1.5 * IQR(`销量(千
克)`))%>%summarise(`销售总量` = mean(`销量(千克)`))%>%ungroup()
df_huacai <- df_huacai%>%pivot_wider(names_from = 单品名称,values_from =
销售总量)
date <- df_huacai[,c(1,2)]
df_huacai <- apply(df_huacai[,-c(1,2)],2,function(x)
as.numeric(ifelse(is.na(x),0,x)))
df_huacai <- cbind(date,df_huacai)
df_huacai_diff <- df_huacai%>%mutate_at(vars(-c(year,month)),
~ .-lag(.,default = first(.)))
cor_huacai <- cor(df_huacai_diff[,-c(1,2)])

#找出除了自己以外最大的值
```

```r
# 使用apply 函数，对每一行应用一个自定义函数来找到最小的负数
min_negative_values <- apply(cor_huacai, 1, function(row) {
  # 使用which 函数找到负数的索引
  negative_indices <- which(row < 0)

  # 如果存在负数，返回最小的负数，否则返回NA
  if (length(negative_indices) > 0) {
    return(min(row[negative_indices]))
  } else {
    return(NA)
  }
})

# 打印结果
min_negative_values <- as.data.frame(min_negative_values)

# 创建一个空的数据框来存储结果
result_huacai <- data.frame(单品名称 = colnames(cor_huacai),
Top_Negative_Correlations = NA)

# 遍历相关性系数矩阵的每一行
for (i in 1:nrow(cor_huacai)) {
  # 找出每一行负相关最大的三个变量
  top_neg_cor_vars <- find_top_neg_cor(cor_huacai[i,])

  # 如果找到了符合条件的变量，则将其添加到结果数据框中
  if (!is.null(top_neg_cor_vars)) {
    result_huacai$Top_Negative_Correlations[i] <-
paste(top_neg_cor_vars, collapse = ", ")
  }
}

# 移除没有找到负相关的行
result_huacai <-
result_huacai[!is.na(result_huacai$Top_Negative_Correlations), ]
#发现花菜类单品之间并没有可以平替的商品



##计算茄类菜品的关系
df_qielei <-  df_merge%>%filter(`分类名称` == "茄类" & !(`销售类型` == "退
货") & !(`是否打折销售` == "是"))%>%group_by(year,month,单品名
称)%>%filter(`销量(千克)` >= quantile(`销量(千克)`, 0.25) - 1.5 * IQR(`销
```

```
量(千克)`) &

`销量(千克)` <= quantile(`销量(千克)`, 0.75) + 1.5 * IQR(`销量(千
克)`))%>%summarise(`销售总量` = mean(`销量(千克)`))%>%ungroup()
df_qielei <- df_qielei%>%pivot_wider(names_from = 单品名称,values_from =
销售总量)
date <- df_qielei[,c(1,2)]
df_qielei <- apply(df_qielei[,-c(1,2)],2,function(x)
as.numeric(ifelse(is.na(x),0,x)))
df_qielei <- cbind(date,df_qielei)
df_qielei_diff <- df_qielei%>%mutate_at(vars(-c(year,month)),
~ .-lag(.,default = first(.)))
cor_qielei <- cor(df_qielei_diff[,-c(1,2)])

#找出除了自己以外最大的值
min_negative_values <- apply(cor_qielei, 1, function(row) {
  # 使用 which 函数找到负数的索引
  negative_indices <- which(row < 0)

  # 如果存在负数，返回最小的负数，否则返回 NA
  if (length(negative_indices) > 0) {
    return(min(row[negative_indices]))
  } else {
    return(NA)
  }
})

# 打印结果
min_negative_values <- as.data.frame(min_negative_values)

# 创建一个空的数据框来存储结果
result_qielei <- data.frame(单品名称 = colnames(cor_qielei),
Top_Negative_Correlations = NA)

# 遍历相关性系数矩阵的每一行
for (i in 1:nrow(cor_qielei)) {
  # 找出每一行负相关最大的三个变量
  top_neg_cor_vars <- find_top_neg_cor(cor_qielei[i,])

  # 如果找到了符合条件的变量，则将其添加到结果数据框中
  if (!is.null(top_neg_cor_vars)) {
    result_qielei$Top_Negative_Correlations[i] <-
paste(top_neg_cor_vars, collapse = ", ")
```

```
  }
}

# 移除没有找到负相关的行
result_qielei <-
result_qielei[!is.na(result_qielei$Top_Negative_Correlations), ]
colnames(result_qielei) <- c("单品名称","互补商品")
write_xlsx(result_qielei,"C:/Users/Lenovo/Desktop/国赛代码/茄类单品间的替代关系.xlsx")


##计算辣椒类单品之间的替代关系
df_lajiao <-  df_merge%>%filter(`分类名称` == "辣椒类" & !(`销售类型` == "退货") & 是否打折销售!="是")%>%group_by(year,month,单品名称)%>%filter(`销量(千克)` >= quantile(`销量(千克)`, 0.25) - 1.5 * IQR(`销量(千克)`) &

`销量(千克)` <= quantile(`销量(千克)`, 0.75) + 1.5 * IQR(`销量(千克)`))%>%summarise(`销售总量` = mean(`销量(千克)`))%>%ungroup()
df_lajiao <- df_lajiao%>%pivot_wider(names_from = 单品名称,values_from = 销售总量)
date <- df_lajiao[,c(1,2)]
df_lajiao <- apply(df_lajiao[,-c(1,2)],2,function(x)
as.numeric(ifelse(is.na(x),0,x)))
df_lajiao <- cbind(date,df_lajiao)
df_lajiao_diff <- df_lajiao%>%mutate_at(vars(-c(year,month)),
~ .-lag(.,default = first(.)))
cor_lajiao <- cor(df_lajiao_diff[,-c(1,2)])

#找出除了自己以外最大的值
min_negative_values <- apply(cor_lajiao, 1, function(row) {
  # 使用 which 函数找到负数的索引
  negative_indices <- which(row < 0)

  # 如果存在负数，返回最小的负数，否则返回 NA
  if (length(negative_indices) > 0) {
    return(min(row[negative_indices]))
  } else {
    return(NA)
  }
})

# 打印结果
min_negative_values <- as.data.frame(min_negative_values)
```

```r
# 创建一个空的数据框来存储结果
result_lajiao <- data.frame(单品名称 = colnames(cor_lajiao),
Top_Negative_Correlations = NA)

# 遍历相关性系数矩阵的每一行
for (i in 1:nrow(cor_lajiao)) {
  # 找出每一行负相关最大的三个变量
  top_neg_cor_vars <- find_top_neg_cor(cor_lajiao[i,])

  # 如果找到了符合条件的变量，则将其添加到结果数据框中
  if (!is.null(top_neg_cor_vars)) {
    result_lajiao$Top_Negative_Correlations[i] <-
paste(top_neg_cor_vars, collapse = ", ")
  }
}

# 移除没有找到负相关的行
result_lajiao <-
result_lajiao[!is.na(result_lajiao$Top_Negative_Correlations), ]
colnames(result_lajiao) <- c("单品名称","互补商品")
write_xlsx(result_lajiao,"C:/Users/Lenovo/Desktop/国赛代码/辣椒类单品间的
替代关系.xlsx")

#### 计算食用菌
df_junlei <-  df_merge%>%filter(`分类名称` == "食用菌" & !(`销售类型` == "
退货") & !(`是否打折销售` == "是"))%>%group_by(year,month,单品名
称)%>%filter(`销量(千克)` >= quantile(`销量(千克)`, 0.25) - 1.5 * IQR(`销
量(千克)`) &

`销量(千克)` <= quantile(`销量(千克)`, 0.75) + 1.5 * IQR(`销量(千
克)`))%>%summarise(`销售总量` = mean(`销量(千克)`))%>%ungroup()
df_junlei <- df_junlei%>%pivot_wider(names_from = 单品名称,values_from =
销售总量)
date <- df_junlei[,c(1,2)]
df_junlei <- apply(df_junlei[,-c(1,2)],2,function(x)
as.numeric(ifelse(is.na(x),0,x)))
df_junlei <- cbind(date,df_junlei)
df_junlei_diff <- df_junlei%>%mutate_at(vars(-c(year,month)),
~ .-lag(.,default = first(.)))
cor_junlei <- cor(df_junlei_diff[,-c(1,2)])

#找出除了自己以外最大的值
```

```r
min_negative_values <- apply(cor_junlei, 1, function(row) {
  # 使用 which 函数找到负数的索引
  negative_indices <- which(row < 0)

  # 如果存在负数，返回最小的负数，否则返回 NA
  if (length(negative_indices) > 0) {
    return(min(row[negative_indices]))
  } else {
    return(NA)
  }
})

# 打印结果
min_negative_values <- as.data.frame(min_negative_values)

# 创建一个空的数据框来存储结果
result_junlei <- data.frame(单品名称 = colnames(cor_junlei),
Top_Negative_Correlations = NA)

# 遍历相关性系数矩阵的每一行
for (i in 1:nrow(cor_junlei)) {
  # 找出每一行负相关最大的三个变量
  top_neg_cor_vars <- find_top_neg_cor(cor_junlei[i,])

  # 如果找到了符合条件的变量，则将其添加到结果数据框中
  if (!is.null(top_neg_cor_vars)) {
    result_junlei$Top_Negative_Correlations[i] <-
paste(top_neg_cor_vars, collapse = ", ")
  }
}

# 移除没有找到负相关的行
result_junlei <-
result_junlei[!is.na(result_junlei$Top_Negative_Correlations), ]
colnames(result_junlei) <- c("单品名称","互补商品")
write_xlsx(result_junlei,"C:/Users/Lenovo/Desktop/国赛代码/食用菌单品间的
替代关系.xlsx")

#### 水声根茎类
df_genjing <-  df_merge%>%filter(`分类名称` == "水生根茎类" & !(`销售类型`
== "退货") & !(`是否打折销售` == "是"))%>%group_by(year,month,单品名
称)%>%filter(`销量(千克)` >= quantile(`销量(千克)`, 0.25) - 1.5 * IQR(`销
量(千克)`) &
```

```
`销量(千克)` <= quantile(`销量(千克)`, 0.75) + 1.5 * IQR(`销量(千
克)`))%>%summarise(`销售总量` = mean(`销量(千克)`))%>%ungroup()
df_genjing <- df_genjing%>%pivot_wider(names_from = 单品名称,values_from =
销售总量)
date <- df_genjing[,c(1,2)]
df_genjing <- apply(df_genjing[,-c(1,2)],2,function(x)
as.numeric(ifelse(is.na(x),0,x)))
df_genjing <- cbind(date,df_genjing)
df_genjing_diff <- df_genjing%>%mutate_at(vars(-c(year,month)),
~ .-lag(.,default = first(.)))
cor_genjing <- cor(df_genjing_diff[,-c(1,2)])

#找出除了自己以外最大的值
min_negative_values <- apply(cor_genjing, 1, function(row) {
  # 使用 which 函数找到负数的索引
  negative_indices <- which(row < 0)

  # 如果存在负数，返回最小的负数，否则返回 NA
  if (length(negative_indices) > 0) {
    return(min(row[negative_indices]))
  } else {
    return(NA)
  }
})

# 打印结果
min_negative_values <- as.data.frame(min_negative_values)

# 创建一个空的数据框来存储结果
result_genjing <- data.frame(单品名称 = colnames(cor_genjing),
Top_Negative_Correlations = NA)

# 遍历相关性系数矩阵的每一行
for (i in 1:nrow(cor_genjing)) {
  # 找出每一行负相关最大的三个变量
  top_neg_cor_vars <- find_top_neg_cor(cor_genjing[i,])

  # 如果找到了符合条件的变量，则将其添加到结果数据框中
  if (!is.null(top_neg_cor_vars)) {
    result_genjing$Top_Negative_Correlations[i] <-
paste(top_neg_cor_vars, collapse = ", ")
  }
```

```
}

# 移除没有找到负相关的行
result_genjing <-
result_genjing[!is.na(result_genjing$Top_Negative_Correlations), ]
#### 根茎类也没有可以替代的商品


##销售量在一天内的变化规律
df_time <- df_merge%>%filter(`销售类型`!= "退货")%>%mutate(小时 =
str_sub(扫码销售时间,1,2))%>%group_by(小时,分类名称)%>%summarise(销量均值
= mean(`销量(千克)`))
ggplot(data = df_time,aes(x = 小时,y = 销量均值,color = 分类名称,group = 分
类名称))+
  geom_line(size = .9)+
  theme_bw()+
  labs(title = "销售量在一天内的变化趋势")+
  theme(plot.title = element_text(hjust = 0.5))

### 打折数量在一天内的变化规律
df_dazhe <- df_merge%>%filter(`是否打折销售`=="是")%>%mutate(小时 =
str_sub(扫码销售时间,1,2))%>%group_by(小时)%>%summarise(打折数量 = n())
ggplot(data = df_dazhe,aes(x =小时,y = 打折数量,fill = 小时))+
  geom_bar(stat = "identity",color = "black")+
  #scale_fill_viridis(guide = "none",discrete = TRUE)+
  theme_bw()+
  labs(title = "一天内打折销售的统计情况")+
  theme(plot.title = element_text(hjust = 0.5))+
      geom_text(aes(label = 打折数量),vjust = -0.1)
```

附录 3

介绍：问题 2 代码

```
    #分析商超每天各个蔬菜品类的销售总量关系
start <- as.POSIXct("2023-6-1")
```

```r
end <- as.POSIXct("2023-6-30")
df_percent_perdays <- df_merge%>%group_by(销售日期,分类名称)%>%summarise(销售总量 = sum(`销量(千克)`))%>%
  ungroup()%>%group_by(销售日期)%>%mutate(单日销售总量 = sum(`销售总量`),比例 = 销售总量/单日销售总量)%>%
  ungroup()%>%filter(`单日销售总量` >= quantile(`单日销售总量`, 0.25) - 1.5 * IQR(`单日销售总量`) & `单日销售总量` <= quantile(`单日销售总量`, 0.75) + 1.5 * IQR(`单日销售总量`))%>%
  filter(销售日期 >=start & 销售日期 <=end)
#画出各个蔬菜品类销售比例随时间的变化图
ggplot(data = df_percent_perdays,aes(x = 销售日期,y = 销售总量,fill = 分类名称))+
  geom_bar(stat = "identity",position = "fill",color = NA,size = .2)+
  scale_fill_viridis(discrete = TRUE)+
  labs(title = "各品类占比随时间变化图")+
  ylab("销售比例")+
  theme_bw()+
  theme(plot.title = element_text(hjust = 0.5))
#观察可以发现比例关系是具有季节性的

### 画出单日销售总量随时间变化图，观察单日销售量的变化趋势
ggplot(data = df_percent_perdays,aes(x = 销售日期,y = 单日销售总量))+
  geom_line()

### 对单日销售量进行季节性分解
library(zoo)
#生成时序对象
df_salecount <- df_percent_perdays%>%select(销售日期,单日销售总量)%>%distinct()%>%ungroup()
salecount.ts <- ts(zoo(df_salecount$单日销售总量,order.by = df_salecount$销售日期))
#时间序列图上面已经画过了,可以看到有一定的季节性趋势，我们通过季节性分解来查看
library(forecast)
df_salecount.stl <- stl(salecount.ts,s.window = "periodic")
plot(df_salecount.stl)
seasonplot(salecount.ts)
### 进行指数平滑预测，分别使用一次，二次，三次去查看效果
fit1 <- ets(salecount.ts,model = "ZNN")
plot(forecast(fit1,7),xlab = "year",ylim = c(0,1000))
accuracy(fit1)#效果并不理想
### 使用二次
fit2 <- ets(salecount.ts,model = "ZZN")
plot(forecast(fit2,7),xlab = "year",ylim = c(0,1000))
```

```
### 使用三次
fit3 <- ets(salecount.ts,model = "ZZZ")
plot(forecast(fit3,7),xlab = "year",ylim = c(0,1000))

### 综合对比发现单日总销量应该使用一次指数平滑预测，不具有趋势性和季节性
forecast(fit3,7)


#### 使用ARIMA模型进行预测
#先检验时序数据的方法是否为常数
Box.test(salecount.ts,type = "Ljung-Box")
##发现方差并不是常数，因此需要进行变换
#salecount.ts <- log(salecount.ts)#是否进行对数变化呢?

ndiffs(salecount.ts)
dsalecount <- diff(salecount.ts)
plot(dsalecount)#这个图可以重新画一下
##发现进行一阶差分后数据变得平稳
##进行adf检验
library(tseries)
adf.test(dsalecount)
#序列满足平稳性要求
### 进一步选择模型
Acf(dsalecount)
Pacf(dsalecount)
fit_arima <- auto.arima(salecount.ts)
###p为4,d为1，q为3
### 进行模型的检验
qqnorm(fit_arima$residuals)
qqline(fit_arima$residuals)
Box.test(fit_arima$residuals,type = "Ljung-Box")
###检验基本通过，因此我们采用这个模型
totalsalepredict <- as.data.frame(forecast(fit_arima,7))##可以把这个图画出来
rownames(totalsalepredict) <-
seq(as.Date("2023-7-1"),as.Date("2023-7-7"),by = "days")
totalsalepredict <- as.data.frame(apply(totalsalepredict,2,exp))
write_xlsx(totalsalepredict,"C:/Users/Lenovo/Desktop/国赛代码/2023年7月
1-2023年7月7日预测的单日销售总量.xlsx")

#### 分析各个品类总销量与定价的关系

#花叶类的拟合
data_huaye <- df_merge_all%>%filter(销售类型 != "退货" & `分类名称` == "花叶
```

```
类" & 是否打折销售 !="是")%>%
  group_by(销售日期,分类名称)%>%summarise(单日总销量 = sum(`销量(千克)`),销
售均价 = mean(`销售单价(元/千克)`))%>%
  filter(`单日总销量` >= quantile(`单日总销量`, 0.25) - 1.5 * IQR(`单日总销量
`) & `单日总销量` <= quantile(`单日总销量`, 0.75) + 1.5 * IQR(`单日总销量`))%>%
  filter(`销售均价` >= quantile(`销售均价`, 0.25) - 1.5 * IQR(`销售均价`) & `
销售均价` <= quantile(`销售均价`, 0.75) + 1.5 * IQR(`销售均价`))
fit_huaye <- lm(单日总销量~ 销售均价,data = data_huaye)
p1 <- ggplot(data = data_huaye,aes(x = 销售均价,y = 单日总销量))+
  geom_point()+
  geom_smooth(formula = y~x)+
  geom_abline(slope = coef(fit_huaye)[2],intercept =
coef(fit_huaye)[1],color = "red",size = .9)+
  theme_bw()+
  labs(tile = "花叶类单日总销量与销售价格的关系")

#花菜类的拟合
data_huacai <- df_merge_all%>%filter(销售类型 != "退货" & `分类名称` == "花
菜类",是否打折销售 !="是")%>%
  group_by(销售日期,分类名称)%>%summarise(单日总销量 = sum(`销量(千克)`),销
售均价 = mean(`销售单价(元/千克)`))%>%
  filter(`单日总销量` >= quantile(`单日总销量`, 0.25) - 1.5 * IQR(`单日总销量
`) & `单日总销量` <= quantile(`单日总销量`, 0.75) + 1.5 * IQR(`单日总销量`))%>%
  filter(`销售均价` >= quantile(`销售均价`, 0.25) - 1.5 * IQR(`销售均价`) & `
销售均价` <= quantile(`销售均价`, 0.75) + 1.5 * IQR(`销售均价`))
fit_huacai <- lm(单日总销量~ 销售均价,data = data_huacai)
p2 <- ggplot(data = data_huacai,aes(x = 销售均价,y = 单日总销量))+
  geom_point()+
  geom_smooth(formula = y~x)+
  geom_abline(slope = coef(fit_huacai)[2],intercept =
coef(fit_huacai)[1],color = "red",size = .9)+
  theme_bw()+
  labs(tile = "花叶类单日总销量与销售价格的关系")

#辣椒类的拟合
data_lajiao <- df_merge_all%>%filter(销售类型 != "退货" & `分类名称` == "辣
椒类",是否打折销售 !="是")%>%
  group_by(销售日期,分类名称)%>%summarise(单日总销量 = sum(`销量(千克)`),销
售均价 = mean(`销售单价(元/千克)`))%>%
  filter(`单日总销量` >= quantile(`单日总销量`, 0.25) - 1.5 * IQR(`单日总销量
`) & `单日总销量` <= quantile(`单日总销量`, 0.75) + 1.5 * IQR(`单日总销量`))%>%
  filter(`销售均价` >= quantile(`销售均价`, 0.25) - 1.5 * IQR(`销售均价`) & `
销售均价` <= quantile(`销售均价`, 0.75) + 1.5 * IQR(`销售均价`))
```

```r
fit_lajiao <- lm(单日总销量~ 销售均价,data = data_lajiao)
p3 <- ggplot(data = data_lajiao,aes(x = 销售均价,y = 单日总销量))+
  geom_point()+
  geom_smooth(formula = y~x)+
  geom_abline(slope = coef(fit_lajiao)[2],intercept =
coef(fit_lajiao)[1],color = "red",size = .9)+
  theme_bw()+
  labs(tile = "花叶类单日总销量与销售价格的关系")

#食用菌类的拟合
data_junlei <- df_merge_all%>%filter(销售类型 != "退货" & `分类名称` == "食
用菌",是否打折销售 !="是")%>%
  group_by(销售日期,分类名称)%>%summarise(单日总销量 = sum(`销量(千克)`),销
售均价 = mean(`销售单价(元/千克)`))%>%
  filter(`单日总销量` >= quantile(`单日总销量`, 0.25) - 1.5 * IQR(`单日总销量
`) & `单日总销量` <= quantile(`单日总销量`, 0.75) + 1.5 * IQR(`单日总销量`))%>%
  filter(`销售均价` >= quantile(`销售均价`, 0.25) - 1.5 * IQR(`销售均价`) & `
销售均价` <= quantile(`销售均价`, 0.75) + 1.5 * IQR(`销售均价`))
fit_junlei <- lm(单日总销量~ 销售均价,data = data_junlei)
p4 <- ggplot(data = data_junlei,aes(x = 销售均价,y = 单日总销量))+
  geom_point()+
  geom_abline(slope = coef(fit_junlei)[2],intercept =
coef(fit_junlei)[1],color = "red",size = .9)+
  theme_bw()+
  labs(tile = "花叶类单日总销量与销售价格的关系")

#水生根茎类
data_genjing <- df_merge_all%>%filter(销售类型 != "退货" & `分类名称` == "
水生根茎类",是否打折销售 !="是")%>%
  group_by(销售日期,分类名称)%>%summarise(单日总销量 = sum(`销量(千克)`),销
售均价 = mean(`销售单价(元/千克)`))%>%
  filter(`单日总销量` >= quantile(`单日总销量`, 0.25) - 1.5 * IQR(`单日总销量
`) & `单日总销量` <= quantile(`单日总销量`, 0.75) + 1.5 * IQR(`单日总销量`))%>%
  filter(`销售均价` >= quantile(`销售均价`, 0.25) - 1.5 * IQR(`销售均价`) & `
销售均价` <= quantile(`销售均价`, 0.75) + 1.5 * IQR(`销售均价`))
fit_genjing <- lm(单日总销量~ 销售均价,data = data_genjing)
p5 <- ggplot(data = data_genjing,aes(x = 销售均价,y = 单日总销量))+
  geom_point()+
  geom_abline(slope = coef(fit_genjing)[2],intercept =
coef(fit_genjing)[1],color = "red",size = .9)+
  theme_bw()
  labs(tile = "花叶类单日总销量与销售价格的关系")
```

```r
#茄类
data_qielei <- df_merge_all%>%filter(销售类型 != "退货" & `分类名称` == "茄类",是否打折销售 !="是")%>%
    group_by(销售日期,分类名称)%>%summarise(单日总销量 = sum(`销量(千克)`),销售均价 = mean(`销售单价(元/千克)`))%>%
    filter(`单日总销量` >= quantile(`单日总销量`, 0.25) - 1.5 * IQR(`单日总销量`) & `单日总销量` <= quantile(`单日总销量`, 0.75) + 1.5 * IQR(`单日总销量`))%>%
    filter(`销售均价` >= quantile(`销售均价`, 0.25) - 1.5 * IQR(`销售均价`) & `销售均价` <= quantile(`销售均价`, 0.75) + 1.5 * IQR(`销售均价`))
fit_qielei <- lm(单日总销量~ 销售均价,data = data_qielei)
p6 <- ggplot(data = data_qielei,aes(x = 销售均价,y = 单日总销量))+
  geom_point()+
  geom_abline(slope = coef(fit_qielei)[2],intercept = coef(fit_qielei)[1],color = "red",size = .9)+
  theme_bw()+
  labs(tile = "花叶类单日总销量与销售价格的关系")


#按照日期合并数据
data_merge <- data_huaye%>%inner_join(data_huacai,by = "销售日期",suffix = c(".花叶",".花菜"))
data_merge <- data_merge%>%inner_join(data_lajiao,by = "销售日期")
data_merge <- data_merge%>%inner_join(data_junlei,by = "销售日期",suffix = c(".辣椒",".食用菌"))
data_merge <- data_merge%>%inner_join(data_genjing,by = "销售日期")
data_merge <- data_merge%>%inner_join(data_qielei,by = "销售日期",suffix = c(".水生根茎","茄类"))#%>%select(-c(1,2,5,8,11,14,17))

#尝试使用线性回归去拟合
data_merge <- data_merge%>%ungroup()%>%select(-c(1,2,5,8,11,14,17))
fit_huaye_2 <- lm(单日总销量.花叶~.,data = data_merge)
fit_huacai_2 <- lm(单日总销量.花菜~.,data = data_merge)
fit_lajiao_2 <- lm(单日总销量.辣椒~.,data = data_merge)
fit_junlei_2 <- lm(单日总销量.食用菌~.,data = data_merge)
fit_genjing_2 <- lm(单日总销量.水生根茎~.,data = data_merge)
fit_qielei_2 <- lm(单日总销量.茄类~.,data = data_merge)

### 花叶定价预测
huayedingjia.ts <- ts(zoo(data_huaye$销售均价,order.by = data_huaye$销售日期))
fit_arima_huaye <- auto.arima(huayedingjia.ts)
result_huaye <- as.data.frame(forecast(fit_arima_huaye,7))
```

```
### 花菜定价预测
huacaidingjia.ts <- ts(zoo(data_huacai$销售均价,order.by = data_huacai$销售日期))
fit_arima_huacai <- auto.arima(huacaidingjia.ts)
result_huacai <- as.data.frame(forecast(fit_arima_huacai,7))

### 辣椒定价预测
lajiaodingjia.ts <- ts(zoo(data_lajiao$销售均价,order.by = data_lajiao$销售日期))
fit_arima_lajiao <- auto.arima(lajiaodingjia.ts)
result_lajiao <- as.data.frame(forecast(fit_arima_lajiao,7))

### 食用菌定价预测
junleidingjia.ts <- ts(zoo(data_junlei$销售均价,order.by = data_junlei$销售日期))
fit_arima_junlei <- auto.arima(junleidingjia.ts)
result_junlei <- as.data.frame(forecast(fit_arima_junlei,7))

### 水生根茎定价预测
genjingdingjia.ts <- ts(zoo(data_genjing$销售均价,order.by = data_genjing$销售日期))
fit_arima_genjing <- auto.arima(genjingdingjia.ts)
result_genjing <- as.data.frame(forecast(fit_arima_genjing,7))

### 茄类定价预测
qieleidingjia.ts <- ts(zoo(data_qielei$销售均价,order.by = data_qielei$销售日期))
fit_arima_qielei <- auto.arima(qieleidingjia.ts)
result_qielei <- as.data.frame(forecast(fit_arima_qielei,7))

result <-
rbind(result_huaye,result_huacai,result_lajiao,result_junlei,result_genjing,result_qielei)
write_xlsx(result,"C:/Users/Lenovo/Desktop/国赛代码/2023年7月1日-7月7日各品类的预测销售单价.xlsx")


##汇总图
data<- df_merge_all%>%filter(销售类型 != "退货")%>%
  group_by(销售日期,分类名称)%>%summarise(单日总销量 = sum(`销量(千克)`),销售均价 = mean(`销售单价(元/千克)`))%>%
  filter(`单日总销量` >= quantile(`单日总销量`, 0.25) - 1.5 * IQR(`单日总销量`) & `单日总销量` <= quantile(`单日总销量`, 0.75) + 1.5 * IQR(`单日总销量`))%>%
```

```
  filter(`销售均价` >= quantile(`销售均价`, 0.25) - 1.5 * IQR(`销售均价`) & `
销售均价` <= quantile(`销售均价`, 0.75) + 1.5 * IQR(`销售均价`))

#画出每个品类单日销售总量的分布图
ggplot(data,aes(x = 单日总销量,fill = 分类名称))+
  geom_density(alpha = 0.4)+
  facet_wrap(.~分类名称,nrow = 2,ncol = 3,scales = "free")+
  labs(title = "原始单日销量分布图")+
  ylab("单日销量")+
  theme_bw()+
  theme(plot.title = element_text(hjust = .5))

#画出 log 转换后的
ggplot(data,aes(x = log(单日总销量),fill = 分类名称))+
  geom_density(alpha = 0.4)+
  facet_wrap(.~分类名称,nrow = 2,ncol = 3,scales = "free")+
  labs(title = "经过 log 转换后的单日销量分布图")+
  ylab("log(单日销量)")+
  theme_bw()+
  theme(plot.title = element_text(hjust = .5))


#拟合曲线图
ggplot(data,aes(x = 销售均价,y = log(单日总销量)))+
  geom_point(aes(color = 分类名称))+
  geom_smooth(method = "lm",formula = y~x)+
  theme_bw()+
  facet_wrap(.~分类名称,nrow = 2,ncol = 3,scales = "free")+
  labs(title = "各个品类单日总销量与销售均价的关系")+
  theme(plot.title = element_text(hjust = 0.5))

result<-data.frame("品类" = c("花叶类","花菜类","辣椒类","食用菌","水生根茎类
","茄类"),
                   "斜率" =
c(coef(fit_huaye)[2],coef(fit_huacai)[2],coef(fit_lajiao)[2],coef(fit_jun
lei)[2],coef(fit_genjing)[2],coef(fit_qielei)[2]),
                   "截距" =
c(coef(fit_huaye)[1],coef(fit_huacai)[1],coef(fit_lajiao)[1],coef(fit_jun
lei)[1],coef(fit_genjing)[1],coef(fit_qielei)[1]))

write_xlsx(result,"C:/Users/Lenovo/Desktop/国赛数据/各个单品的拟合结
果.xlsx")
```

```
###保存拟合结果
write_xlsx(as.data.frame(coef(fit_huaye_2))%>%rownames_to_column(var = "变量"),"C:/Users/Lenovo/Desktop/国赛代码/花叶类多元线性回归拟合结果.xlsx")
write_xlsx(as.data.frame(coef(fit_huacai_2))%>%rownames_to_column(var = "变量"),"C:/Users/Lenovo/Desktop/国赛代码/花菜类多元线性回归拟合结果.xlsx")
write_xlsx(as.data.frame(coef(fit_lajiao_2))%>%rownames_to_column(var = "变量"),"C:/Users/Lenovo/Desktop/国赛代码/辣椒类多元线性回归拟合结果.xlsx")
write_xlsx(as.data.frame(coef(fit_junlei_2))%>%rownames_to_column(var = "变量"),"C:/Users/Lenovo/Desktop/国赛代码/食用菌类多元线性回归拟合结果.xlsx")
write_xlsx(as.data.frame(coef(fit_genjing_2))%>%rownames_to_column(var = "变量"),"C:/Users/Lenovo/Desktop/国赛代码/水生根茎类多元线性回归拟合结果.xlsx")
write_xlsx(as.data.frame(coef(fit_qielei_2))%>%rownames_to_column(var = "变量"),"C:/Users/Lenovo/Desktop/国赛代码/茄类多元线性回归拟合结果.xlsx")


#花叶类批发价预测
df_jinjia <- df_merge_all%>%filter(`分类名称` == "花叶类")%>%group_by(销售日期)%>%filter(`批发价格(元/千克)` >= quantile(`批发价格(元/千克)`, 0.25) - 1.5 * IQR(`批发价格(元/千克)`) &

`批发价格(元/千克)` <= quantile(`批发价格(元/千克)`, 0.75) + 1.5 * IQR(`批发价格(元/千克)`))%>%summarise(进价均值 = mean(`批发价格(元/千克)`))
#时间序列图
ggplot(data = df_jinjia,aes(x = 销售日期,y = 进价均值))+
  geom_line()
#构建模型预测进价
jinjia.ts <- ts(zoo(df_jinjia$进价均值,order.by = df_jinjia$销售日期))
ndiffs(jinjia.ts)
djinjia <- diff(jinjia.ts)
plot(djinjia)#这个图可以重新画一下
##发现进行一阶差分后数据变得平稳
##进行 adf 检验
library(tseries)
adf.test(djinjia)
#序列满足平稳性要求
fit_arima <- auto.arima(jinjia.ts)
###检验基本通过，因此我们采用这个模型
result <- as.data.frame(forecast(fit_arima,7))##可以把这个图画出来
rownames(result) <- seq(as.Date("2023-7-1"),as.Date("2023-7-7"),by = "days")
write_xlsx(result,"C:/Users/Lenovo/Desktop/国赛代码/2023 年 7 月 1-2023 年 7 月
```

```
7日预测的花叶类批发价格.xlsx")

#花菜类批发价格预测
df_jinjia <- df_merge_all%>%filter(`分类名称` == "花菜类")%>%group_by(销售日期)%>%filter(`批发价格(元/千克)` >= quantile(`批发价格(元/千克)`, 0.25) - 1.5 * IQR(`批发价格(元/千克)`) &

`批发价格(元/千克)` <= quantile(`批发价格(元/千克)`, 0.75) + 1.5 * IQR(`批发价格(元/千克)`))%>%summarise(进价均值 = mean(`批发价格(元/千克)`))
#时间序列图
ggplot(data = df_jinjia,aes(x = 销售日期,y = 进价均值))+
  geom_line()
#构建模型预测进价
jinjia.ts <- ts(zoo(df_jinjia$进价均值,order.by = df_jinjia$销售日期))
ndiffs(jinjia.ts)
djinjia <- diff(jinjia.ts)
plot(djinjia)#这个图可以重新画一下
##发现进行一阶差分后数据变得平稳
##进行adf检验
library(tseries)
adf.test(djinjia)
#序列满足平稳性要求
fit_arima <- auto.arima(jinjia.ts)
###检验基本通过，因此我们采用这个模型
result <- as.data.frame(forecast(fit_arima,7))##可以把这个图画出来
rownames(result) <- seq(as.Date("2023-7-1"),as.Date("2023-7-7"),by = "days")
write_xlsx(result,"C:/Users/Lenovo/Desktop/国赛代码/2023年7月1-2023年7月7日预测的花菜类批发价格.xlsx")

#辣椒类批发价格预测
df_jinjia <- df_merge_all%>%filter(`分类名称` == "辣椒类")%>%group_by(销售日期)%>%filter(`批发价格(元/千克)` >= quantile(`批发价格(元/千克)`, 0.25) - 1.5 * IQR(`批发价格(元/千克)`) &

`批发价格(元/千克)` <= quantile(`批发价格(元/千克)`, 0.75) + 1.5 * IQR(`批发价格(元/千克)`))%>%summarise(进价均值 = mean(`批发价格(元/千克)`))
#时间序列图
ggplot(data = df_jinjia,aes(x = 销售日期,y = 进价均值))+
  geom_line()
#构建模型预测进价
jinjia.ts <- ts(zoo(df_jinjia$进价均值,order.by = df_jinjia$销售日期))
ndiffs(jinjia.ts)
```

```
djinjia <- diff(jinjia.ts)
plot(djinjia)#这个图可以重新画一下
##发现进行一阶差分后数据变得平稳
##进行 adf 检验
library(tseries)
adf.test(djinjia)
#序列满足平稳性要求
fit_arima <- auto.arima(jinjia.ts)
###检验基本通过，因此我们采用这个模型
result <- as.data.frame(forecast(fit_arima,7))##可以把这个图画出来
rownames(result) <- seq(as.Date("2023-7-1"),as.Date("2023-7-7"),by =
"days")
write_xlsx(result,"C:/Users/Lenovo/Desktop/国赛代码/2023 年 7 月 1-2023 年 7 月
7 日预测的辣椒类批发价格.xlsx")

#水生根茎类
df_jinjia <- df_merge_all%>%filter(`分类名称` == "水生根茎类")%>%group_by(销
售日期)%>%filter(`批发价格(元/千克)` >= quantile(`批发价格(元/千克)`, 0.25) -
1.5 * IQR(`批发价格(元/千克)`) &

`批发价格(元/千克)` <= quantile(`批发价格(元/千克)`, 0.75) + 1.5 * IQR(`批发
价格(元/千克)`))%>%summarise(进价均值 = mean(`批发价格(元/千克)`))
#时间序列图
ggplot(data = df_jinjia,aes(x = 销售日期,y = 进价均值))+
  geom_line()
#构建模型预测进价
jinjia.ts <- ts(zoo(df_jinjia$进价均值,order.by = df_jinjia$销售日期))
ndiffs(jinjia.ts)
djinjia <- diff(jinjia.ts)
plot(djinjia)#这个图可以重新画一下
##发现进行一阶差分后数据变得平稳
##进行 adf 检验
library(tseries)
adf.test(djinjia)
#序列满足平稳性要求
fit_arima <- auto.arima(jinjia.ts)
###检验基本通过，因此我们采用这个模型
result <- as.data.frame(forecast(fit_arima,7))##可以把这个图画出来
rownames(result) <- seq(as.Date("2023-7-1"),as.Date("2023-7-7"),by =
"days")
write_xlsx(result,"C:/Users/Lenovo/Desktop/国赛代码/2023 年 7 月 1-2023 年 7 月
7 日预测的水生根茎类批发价格.xlsx")
```

```
#茄类
df_jinjia <- df_merge_all%>%filter(`分类名称` == "茄类")%>%group_by(销售日
期)%>%filter(`批发价格(元/千克)` >= quantile(`批发价格(元/千克)`, 0.25) - 1.5
* IQR(`批发价格(元/千克)`) &
                                                                              `
批发价格(元/千克)` <= quantile(`批发价格(元/千克)`, 0.75) + 1.5 * IQR(`批发价
格(元/千克)`))%>%summarise(进价均值 = mean(`批发价格(元/千克)`))
#时间序列图
ggplot(data = df_jinjia,aes(x = 销售日期,y = 进价均值))+
  geom_line()
#构建模型预测进价
jinjia.ts <- ts(zoo(df_jinjia$进价均值,order.by = df_jinjia$销售日期))
ndiffs(jinjia.ts)
djinjia <- diff(jinjia.ts)
plot(djinjia)#这个图可以重新画一下
##发现进行一阶差分后数据变得平稳
##进行 adf 检验
library(tseries)
adf.test(djinjia)
#序列满足平稳性要求
fit_arima <- auto.arima(jinjia.ts)
###检验基本通过，因此我们采用这个模型
result <- as.data.frame(forecast(fit_arima,7))##可以把这个图画出来
rownames(result) <- seq(as.Date("2023-7-1"),as.Date("2023-7-7"),by =
"days")
write_xlsx(result,"C:/Users/Lenovo/Desktop/国赛代码/2023 年 7 月 1-2023 年 7 月
7 日预测的茄类批发价格.xlsx")


#食用菌类
df_jinjia <- df_merge_all%>%filter(`分类名称` == "食用菌")%>%group_by(销售日
期)%>%filter(`批发价格(元/千克)` >= quantile(`批发价格(元/千克)`, 0.25) - 1.5
* IQR(`批发价格(元/千克)`) &
                                                                            `批
发价格(元/千克)` <= quantile(`批发价格(元/千克)`, 0.75) + 1.5 * IQR(`批发价格
(元/千克)`))%>%summarise(进价均值 = mean(`批发价格(元/千克)`))
#时间序列图
ggplot(data = df_jinjia,aes(x = 销售日期,y = 进价均值))+
  geom_line()
#构建模型预测进价
jinjia.ts <- ts(zoo(df_jinjia$进价均值,order.by = df_jinjia$销售日期))
ndiffs(jinjia.ts)
djinjia <- diff(jinjia.ts)
```

```r
plot(djinjia)#这个图可以重新画一下
##发现进行一阶差分后数据变得平稳
##进行 adf 检验
library(tseries)
adf.test(djinjia)
#序列满足平稳性要求
fit_arima <- auto.arima(jinjia.ts)
###检验基本通过，因此我们采用这个模型
result <- as.data.frame(forecast(fit_arima,7))##可以把这个图画出来
rownames(result) <- seq(as.Date("2023-7-1"),as.Date("2023-7-7"),by =
"days")
    write_xlsx(result,"C:/Users/Lenovo/Desktop/国赛代码/2023 年 7 月 1-2023 年
7 月 7 日预测的食用菌批发价格.xlsx")
```

附录 3

介绍：问题三代码

```r
    ##筛选出这几天的可售商品
start <- as.POSIXct("2023-6-24")
end <- as.POSIXct("2023-6-30")
df_keshoupinzhong <- df_merge_all%>%filter(销售日期 >= start & 销售日期 <=
end)%>%select(单品名称,分类名称)%>%distinct()
write_xlsx(df_keshoupinzhong,"C:/Users/Lenovo/Desktop/国赛代码/2023 年 6 月
24 日-2023 年 6 月 30 日预测的可售商品.xlsx")

### 找出这几天可售商品的销售额占比
df_lirun_percent <- df_merge_all%>%filter(销售日期 >= start & 销售日期 <=
end,`是否打折销售`!="是")%>%group_by(单品名称,分类名称)%>%summarise(利润率 =
mean((`销售单价(元/千克)` - `批发价格(元/千克)`)/`批发价格(元/千克)`))
df_lirun_percent <- df_lirun_percent%>%group_by(分类名称)%>%mutate(总利润 =
sum(利润率),比例 = 利润率/总利润)
write_xlsx(df_lirun_percent,"C:/Users/Lenovo/Desktop/国赛代码/2023 年 6 月 24
日-2023 年 6 月 30 日可售单品单品利润占总利润的比例.xlsx")

df_xiaoliang <- df_merge_all%>%filter(销售日期 >= start & 销售日期 <=
end)%>%group_by(单品名称,分类名称)%>%summarise(平均每日销量 = sum(`销量(千
```

```
克)`)/7)
df_dazhecishu <- df_merge_all%>%filter(销售日期 >= start & 销售日期 <=
end)%>%group_by(单品名称,分类名称)%>%mutate(是否打折 = as.numeric(ifelse(是
否打折销售 == "是",1,0)))%>%summarise(平均每日打折次数 = sum(是否打折)/7)
df_lirun <- df_merge_all%>%filter(销售日期 >= start & 销售日期 <=
end)%>%group_by(单品名称,分类名称)%>%summarise(利润率 = mean((`销售单价(元/
千克)` - `批发价格(元/千克)`)/`批发价格(元/千克)`))
df_sunhao <- df_merge_all%>%filter(销售日期 >= start & 销售日期 <=
end)%>%inner_join(df4,by = "单品名称")%>%select(单品名称,分类名称,`损耗率
(%)`)%>%distinct()

df_all <- df_xiaoliang%>%inner_join(df_dazhecishu,by = c("单品名称","分类名
称"))%>%inner_join(df_lirun,by = c("单品名称","分类名称
"))%>%inner_join(df_sunhao,by = c("单品名称","分类名称"))


func <- function(name){
  data <- df_merge_all%>%filter(销售类型 != "退货" & `单品名称` == name)%>%
    group_by(销售日期)%>%summarise(单日总销量 = sum(`销量(千克)`),销售均价 =
mean(`销售单价(元/千克)`))%>%
    filter(`单日总销量` >= quantile(`单日总销量`, 0.25) - 1.5 * IQR(`单日总销
量`) & `单日总销量` <= quantile(`单日总销量`, 0.75) + 1.5 * IQR(`单日总销量`))%>%
    filter(`销售均价` >= quantile(`销售均价`, 0.25) - 1.5 * IQR(`销售均价`) &
`销售均价` <= quantile(`销售均价`, 0.75) + 1.5 * IQR(`销售均价`))
}

names <- c(df_all$单品名称)
result <- data.frame()
result$`单品名称` <- NULL
result$`截距` <- NULL
result$`斜率` <- NULL

###计算每个单品销量与定价的线性回归系数
for(name in names){
  data <- func(name)
  fit <- lm(单日总销量~销售均价,data)
  temp <- data.frame("单品名称" = name,"截距" = coef(fit)[1],"斜率" =
coef(fit)[2])
  result <- rbind(result,temp)
}

### 筛选出系数为负值的蔬菜，在对蔬菜进行
result_filtered <- result%>%filter(斜率 < 0)
```

```r
names_filtered <- result_filtered$单品名称

df_all <- df_all%>%filter(单品名称 %in% names_filtered)

##归一化处理函数
z_value <- function(x){
  x / sqrt(sum(x^2))
}

###对数据进行归一化处理
df_name <- df_all[,c(1,2)]
df_all_scaled <- apply(df_all[,-c(1,2)],2,z_value)
df_all_scaled <- cbind(df_name,df_all_scaled)


### 利用熵权法求出每个指标的权重
positive_scaled<-function(x){
  (x - min(x))/(max(x)-min(x))
}
negative_scaled<-function(x){
  (max(x) - x)/(max(x) - min(x))
}

df_scaled_positive <- apply(df_all[,-c(1,2,4,6)],2,positive_scaled)
df_scaled_negative <- apply(df_all[,-c(1,2,3,5)],2,negative_scaled)
df_scaled_all <- cbind(df_name,df_scaled_positive,df_scaled_negative)

first1 <- function(data)
{
  x <- c(data)
  for(i in 1:length(data))
    x[i] = data[i]/sum(data[])
  return(x)
}


data1 <- apply(df_scaled_all[,-c(1,2)],2,first1)

first2 <- function(data)
{
  x <- c(data)
  for(i in 1:length(data)){
    if(data[i] == 0){
```

```
      x[i] = 0
    }else{
      x[i] = data[i] * log(data[i])
    }
  }
  return(x)
}

#计算信息熵
data2 <- apply(data1,2,first2)
k <- 1/log(length(data2[,1]))
d <- -k * colSums(data2)
d <- 1-d
w <- d/sum(d)
### 求出最后的权重
print(w)

### 根据权重重新对原来标准化后的数进行变换
df_all_scaled$平均每日销量 <- 0.61*df_all_scaled$平均每日销量
df_all_scaled$利润率 <- 0.20*df_all_scaled$利润率
df_all_scaled$平均每日打折次数 <- 0.11*df_all_scaled$平均每日打折次数
df_all_scaled$`损耗率(%)` <- 0.08*df_all_scaled$`损耗率(%)`

### 求出每列的最大值
z_max <- apply(df_all_scaled[,-c(1,2)],2,max)
z_min <- apply(df_all_scaled[,-c(1,2)],2,min)

### 计算出距离
dist <-function(x, std){
  res <- c()
  for ( i in 1 : nrow(x)) {
    res[i] = sqrt(sum((x[i,-c(1,2)]-std)^2))
  }

  return(res)
}

#最优距离
du <- dist(df_all_scaled,z_max)
#最劣距离
dn <- dist(df_all_scaled,z_min)

### 根据结果筛选出最终的33个单品
```

```r
df_all_final <- df_all_scaled%>%add_column(du = du,dn = dn)%>%mutate(ci = dn/(du+dn))%>%arrange(-ci)
df_fillterd <- df_all_final[1:33,]
write_xlsx(df_fillterd,"C:/Users/Lenovo/Desktop/国赛代码/根据熵权法加 Topsis 评价选出的 7 月 1 日销售的 33 个单品.xlsx")
result_filtered <- result_filtered%>%filter(单品名称 %in% df_fillterd$单品名称)
write_xlsx(result_filtered,"C:/Users/Lenovo/Desktop/国赛代码/2023 年 7 月 1 日筛选单品线性回归拟合结果.xlsx")

### 花叶类发现这几个产品无法去衡量相关性
names <- c(df_all%>%filter(分类名称 == "花叶类",单品名称 %in% result_filtered$单品名称)%>%select(单品名称))
data_huaye <-func(names$单品名称[1])
###计算每个单品销量与定价的线性回归系数
for(name in names$单品名称[-1]){
  tmp <- func(name)
  data_huaye <- data_huaye%>%full_join(tmp,by = "销售日期")
}
## 处理缺失值
data_huaye<- as.data.frame(apply(data_huaye[,-1],2,function(x)
as.numeric(ifelse(is.na(x),0,x))))
cor_huaye <- cor(data_huaye)
heatmap(cor_huaye)
### 拟合曲线
func2 <- function(data,results){
  dependent_columns <- seq(1, ncol(data), by = 2)

  # 获取自变量列（除了因变量列之外的所有列）
  #independent_columns <- setdiff(1:ncol(data), dependent_columns)

  # 使用循环或 lapply 函数遍历每个因变量列
  for (dep_col in dependent_columns) {
    # 选择因变量和自变量列
    y <- data[, dep_col]
    x <- data[, -dep_col]

    # 执行多元线性回归
    model <- lm(y ~ ., data = x)

    # 将回归结果存储在列表中
    results[[dep_col]] <- model
  }
```

```
  results
}
result_huaye <- list()
result_huaye <- func2(data_huaye,result_huaye)

### 花菜类
names <- c(df_all%>%filter(分类名称 == "花菜类",单品名称 %in%
result_filtered$单品名称)%>%select(单品名称))
data_huacai <-func(names$单品名称[1])
###计算每个单品销量与定价的线性回归系数
for(name in names$单品名称[-1]){
  tmp <- func(name)
  data_huacai <- data_huacai%>%full_join(tmp,by = "销售日期")
}
### 拟合花菜类品类内的结果
data_huacai<- as.data.frame(apply(data_huacai[,-1],2,function(x)
as.numeric(ifelse(is.na(x),0,x))))
result_huacai <- list()
result_huacai <- func2(data_huacai,result_huacai)

### 水生根茎类
names <- c(df_all%>%filter(分类名称 == "水生根茎类",单品名称 %in%
result_filtered$单品名称)%>%select(单品名称))
data_genjing <-func(names$单品名称[1])
###计算每个单品销量与定价的线性回归系数
for(name in names$单品名称[-1]){
  tmp <- func(name)
  data_genjing <- data_genjing%>%full_join(tmp,by = "销售日期")
}
###拟合结果
data_genjing<- as.data.frame(apply(data_genjing[,-1],2,function(x)
as.numeric(ifelse(is.na(x),0,x))))
result_genjing <- list()
result_genjing <- func2(data_genjing,result_genjing)

### 辣椒类
names <- c(df_all%>%filter(分类名称 == "辣椒类",单品名称 %in%
result_filtered$单品名称)%>%select(单品名称))
data_lajiao <-func(names$单品名称[1])
###计算每个单品销量与定价的线性回归系数
for(name in names$单品名称[-1]){
  tmp <- func(name)
  data_lajiao <- data_lajiao%>%full_join(tmp,by = "销售日期")
```

```
}
###拟合结果
data_lajiao<- as.data.frame(apply(data_lajiao[,-1],2,function(x)
as.numeric(ifelse(is.na(x),0,x))))
result_lajiao <- list()
result_lajiao <- func2(data_lajiao,result_lajiao)

### 食用菌类
names <- c(df_all%>%filter(分类名称 == "食用菌",单品名称 %in%
result_filtered$单品名称)%>%select(单品名称))
data_junlei <-func(names$单品名称[1])
###计算每个单品销量与定价的线性回归系数
for(name in names$单品名称[-1]){
  tmp <- func(name)
  data_junlei <- data_junlei%>%full_join(tmp,by = "销售日期")
}
###拟合结果
data_junlei<- as.data.frame(apply(data_junlei[,-1],2,function(x)
as.numeric(ifelse(is.na(x),0,x))))
result_junlei <- list()
result_junlei <- func2(data_junlei,result_junlei)

### 茄类
names <- c(df_all%>%filter(分类名称 == "茄类",单品名称 %in%
result_filtered$单品名称)%>%select(单品名称))
data_qielei <-func(names$单品名称[1])
###计算每个单品销量与定价的线性回归系数
for(name in names$单品名称[-1]){
  tmp <- func(name)
  data_qielei <- data_qielei%>%full_join(tmp,by = "销售日期")
}
###拟合结果
data_qielei<- as.data.frame(apply(data_qielei[,-1],2,function(x)
as.numeric(ifelse(is.na(x),0,x))))
result_qielei <- list()
result_qielei <- func2(data_qielei,result_qielei)


# 假设你已经有多个多元线性回归模型，并将它们添加到 regression_models 列表中

# 打开一个文本文件以写入模型公式
file <- file("C:/Users/Lenovo/Desktop/国赛代码/茄类各单品的多元线性回归方
程.txt", "w")
```

```
# 遍历每个回归模型
for (i in seq(1,length(result_qielei),by = 2)) {
  # 获取回归模型的详细信息
  model_summary <- summary(result_qielei[[i]])

  # 提取回归系数
  coefficients <- coef(result_qielei[[i]])

  # 构建模型公式
  formula <- as.formula(paste("Y ~", paste(names(coefficients), "*",
coefficients, collapse = " + ")))
  # 将回归公式写入文件
  writeLines(paste("Model ", i, " Formula: ", formula, "\n"), con = file)
}

# 关闭文件
    close(file)
```

附录 5

介绍：优化求解算法

```
    import numpy as np
import math
from scipy.optimize import minimize
import matplotlib.pyplot as plt

log = math.log
# 1.花菜类 2.水生根茎类 3.花叶类 4.食用菌 5.辣椒类 6.茄子类
c = [7.89, 12.55, 3.07, 7.56, 4.66, 4.94]
d = [0.1551, 0.1365, 0.1283, 0.0945, 0.0924, 0.0668]

n1 = 0.05
n2 = 0.06
n3 = 0.42
```

```
n4 = 0.15
n5 = 0.26
n6 = 0.06

max_value = 546.4
min_value = 250
mean = 362.82

# 定义目标函数
def objective_function(x):
    obj_value = -(x[0]* x[6] - c[0]* (x[0]/(1 - d[0])) +
                  x[1]* x[7] - c[1]* (x[1]/(1 - d[1])) +
                  x[2]* x[8] - c[2]* (x[2]/(1 - d[2])) +
                  x[3]* x[9] - c[3]* (x[3]/(1 - d[3])) +
                  x[4]* x[10] - c[4]* (x[4]/(1 - d[4])) +
                  x[5]* x[11] - c[5]* (x[5]/(1 - d[5])))
    return obj_value

# 定义不等式约束条件
def inequality_constraint1(x):
    return max_value - np.sum(x[0:6])
def inequality_constraint2(x):
    return np.sum(x[0:6]) - min_value
def inequality_constraint3(x):
    return 13.03 - x[6]
def inequality_constraint4(x):
    return x[6] - 10
def inequality_constraint5(x):
    return 18.84 - x[7]
def inequality_constraint6(x):
    return x[7] - 11.74
def inequality_constraint7(x):
    return 6.12 - x[8]
def inequality_constraint8(x):
    return x[8] - 3.85
def inequality_constraint9(x):
    return 14.73 - x[9]
def inequality_constraint10(x):
    return x[9] - 8.90
def inequality_constraint11(x):
    return 9.38 - x[10]
def inequality_constraint12(x):
    return x[10] - 5.07
```

```
def inequality_constraint13(x):
    return 9.90 - x[11]
def inequality_constraint14(x):
    return x[11] - 6.82

def equality_constraint1(x):
    return (n1-1)*x[0]+n1*x[1]+n1*x[2]+n1*x[3]+n1*x[4]+n1*x[5]
def equality_constraint2(x):
    # 这里是一个示例等式约束条件
    return n2*x[0]+(n2-1)*x[1]+n2*x[2]+n2*x[3]+n2*x[4]+n2*x[5]
def equality_constraint3(x):
    # 这里是一个示例等式约束条件
    return n3*x[0]+n3*x[1]+(n3-1)*x[2]+n3*x[3]+n3*x[4]+n3*x[5]
def equality_constraint4(x):
    # 这里是一个示例等式约束条件
    return n4*x[0]+n4*x[1]+n4*x[2]+(n4-1)*x[3]+4*x[4]+n4*x[5]
def equality_constraint5(x):
    # 这里是一个示例等式约束条件
    return n5*x[0]+n5*x[1]+n5*x[2]+n5*x[3]+(n5-1)*x[4]+n5*x[5]
def equality_constraint6(x):
    # 这里是一个示例等式约束条件
    return n6*x[0]+n6*x[1]+n6*x[2]+n6*x[3]+n6*x[4]+(n6-1)*x[5]

### 多元线性回归约束
# 1.花菜类 2.水生根茎类 3.花叶类 4.食用菌 5.辣椒类 6.茄子类
def equality_constraint7(x):
    return 0.22*x[1]+ 0.07*x[2] + 0*x[3] + 0.07*x[4]+ 0.22*x[5]+ \
           -2.48*x[6]+ 0.15*x[7]+ 2.34*x[8]+ -0.41*x[9]+0.27*x[10]+
-1.4*x[11]+ 28.08 -x[0]
def equality_constraint8(x):
    # 这里是一个示例等式约束条件
    return 0.36 * x[0] + 0.04 * x[2] + 0.18* x[3] + 0.14 * x[4] + -0.22 * x[5]
+ \
           1.35 * x[6] + -1.34 * x[7] + -2.04 * x[8] + 1.43 * x[9] + 1.02 *
x[10] + 1.00 * x[11] + -26.30 - x[1]
def equality_constraint9(x):
    # 这里是一个示例等式约束条件
    return 0.79 * x[0] + 0.28 * x[1] + 0.56 * x[3] + 0.49 * x[4] + 0.25 * x[5]
+ \
           2.61 * x[6] + -1.41 * x[7] + -9.62 * x[8] + 7.6 * x[9] + -1.13 *
x[10] + -7.68 * x[11] + 93.07 - x[2]
def equality_constraint10(x):
    # 这里是一个示例等式约束条件
```

```
    return 0 * x[0] + 0.34 * x[1] + 0.15 * x[2] + 0.35 * x[4] + -0.01 * x[5]
+ \
           -0.02 * x[6] + -1.06 * x[7] + 2.10 * x[8] + -5.78 * x[9] + 2.62 *
x[10] + -2.53 * x[11] + 64.06 - x[3]
def equality_constraint11(x):
    # 这里是一个示例等式约束条件
    return 0.23 * x[0] + 0.26 * x[1] + 0.14 * x[2] + 0.37 * x[3] + 1.07 * x[5]
+ \
           -0.39 * x[6] + 0.99 * x[7] + -2.49 * x[8] + -1.26 * x[9] + -3.81
* x[10] + 9.05 * x[11] + -25.70 - x[4]
def equality_constraint12(x):
    # 这里是一个示例等式约束条件
    return 0.07 * x[0] + -0.05 * x[1] + 0.01 * x[2] + 0 * x[3] + 0.12 * x[4]
+ \
           0.04 * x[6] + 1.16 * x[7] + 2.44 * x[8] + 1.15 * x[9] + 0.64 * x[10]
+ -1.54 * x[11] + -25.28 - x[5]


def non_negative_constraint(x):
    return x  # 所有变量必须为非负数


# 创建一系列均匀分布的mean 值
num_points = 1000
mean_values = np.linspace(min_value, max_value, num_points)

# 用于保存每个mean 值对应的最小值
max_values = []
max_value = -1
max_result = None

for mean in mean_values:
    # 初始猜测值n
    initial_guess = [mean * n1, mean * n2, mean * n3, mean * n4, mean * n5,
mean * n6,11.52,15.29,4.99,11.82,7.23,8.36]
    # 1.花菜类 2.水生根茎类 3.花叶类 4.食用菌 5.辣椒类 6.茄子类
    # 定义约束条件列表
    constraints = [{'type': 'ineq', 'fun': inequality_constraint1},
                   {'type': 'ineq', 'fun': inequality_constraint2},
                   {'type': 'ineq', 'fun': inequality_constraint3},
                   {'type': 'ineq', 'fun': inequality_constraint4},
                   {'type': 'ineq', 'fun': inequality_constraint5},
                   {'type': 'ineq', 'fun': inequality_constraint6},
```

```
                {'type': 'ineq', 'fun': inequality_constraint7},
                {'type': 'ineq', 'fun': inequality_constraint8},
                {'type': 'ineq', 'fun': inequality_constraint9},
                {'type': 'ineq', 'fun': inequality_constraint10},
                {'type': 'ineq', 'fun': inequality_constraint11},
                {'type': 'ineq', 'fun': inequality_constraint12},
                {'type': 'ineq', 'fun': inequality_constraint13},
                {'type': 'ineq', 'fun': inequality_constraint14},
                {'type': 'eq', 'fun': equality_constraint7},
                {'type': 'eq', 'fun': equality_constraint8},
                {'type': 'eq', 'fun': equality_constraint9},
                {'type': 'eq', 'fun': equality_constraint10},
                {'type': 'eq', 'fun': equality_constraint11},
                {'type': 'eq', 'fun': equality_constraint12},
                {'type': 'ineq', 'fun': non_negative_constraint}]

    result = minimize(objective_function, initial_guess,
constraints=constraints)
    print("总利润最大值：", -result.fun)
    print("进货量最优解：", "花菜类：", result.x[0]/(1-d[0]), "水生根茎类:",
result.x[1]/(1-d[1]), "花叶类", result.x[2]/(1-d[2]), "食用菌
",result.x[3]/(1-d[3]), "辣椒类", result.x[4]/(1-d[4]), "茄子类",
result.x[5]/(1-d[5]))
    print("销售量定价：", "花菜类：", result.x[6], "水生根茎类:", result.x[7],
"花叶类", result.x[8], "食用菌",result.x[9], "辣椒类", result.x[10], "茄子类
", result.x[11])
    print("比例：","花菜类：",result.x[0]/sum(result.x[0:6]),"水生根茎类:",
result.x[1]/sum(result.x[0:6]), "花叶类", result.x[2]/sum(result.x[0:6]), "
食用菌",result.x[3]/sum(result.x[0:6]), "辣椒类",
result.x[4]/sum(result.x[0:6]), "茄子类", result.x[5]/sum(result.x[0:6]))
    print(sum(result.x[0:6]))
    max_values.append(-result.fun)
    if -result.fun > max_value:
        max_value = -result.fun
        max_result = result


#绘制最小值与mean 的关系图
plt.rcParams['font.sans-serif'] = ['SimHei']  # 设置中文字体为黑体或其他支持
的中文字体
plt.rcParams['axes.unicode_minus'] = False  # 用于正常显示负号（-）
plt.plot(mean_values[1:], max_values[1:])
plt.xlabel('初始值')
```

```
plt.ylabel('利润最大值')
plt.title('利用遗传算法得到的利润最大值与不同初始值的关系')
plt.grid(True)
plt.savefig("C:\\Users\\Lenovo\\Desktop\\国赛代码\\利用遗传算法得到的利润最大值与不同初始值的关系.png")
#plt.close()

#等式固定比例约束条件
'''def equality_constraint1(x):

    return (n1-1)*x[0]+n1*x[1]+n1*x[2]+n1*x[3]+n1*x[4]+n1*x[5]

def equality_constraint2(x):
    # 这里是一个示例等式约束条件
    return n2*x[0]+(n2-1)*x[1]+n2*x[2]+n2*x[3]+n2*x[4]+n2*x[5]
def equality_constraint3(x):
    # 这里是一个示例等式约束条件
    return n3*x[0]+n3*x[1]+(n3-1)*x[2]+n3*x[3]+n3*x[4]+n3*x[5]
def equality_constraint4(x):
    # 这里是一个示例等式约束条件
    return n4*x[0]+n4*x[1]+n4*x[2]+(n4-1)*x[3]+4*x[4]+n4*x[5]
def equality_constraint5(x):
    # 这里是一个示例等式约束条件
    return n5*x[0]+n5*x[1]+n5*x[2]+n5*x[3]+(n5-1)*x[4]+n5*x[5]
def equality_constraint6(x):
    # 这里是一个示例等式约束条件
    return n6*x[0]+n6*x[1]+n6*x[2]+n6*x[3]+n6*x[4]+(n6-1)*x[5]
                {'type': 'eq', 'fun': equality_constraint1},
            {'type': 'eq', 'fun': equality_constraint2},
            {'type': 'eq', 'fun': equality_constraint3},
            {'type': 'eq', 'fun': equality_constraint4},
            {'type': 'eq', 'fun': equality_constraint5},
                {'type': 'eq', 'fun': equality_constraint6},
```