基于线性规划的农作物种植策略

摘要

本文就华北山区的乡村农业在有限耕地资源下的最优种植策略问题，进行了深入分析与研究；根据题意和数据，总体采用线性规划模型的方式进行计算，并辅以蒙特卡洛法进行模型检验；采用了 python、scipy、SPSSPRO 等数学工具编程求解与绘图，计算得出了该问题每一种情况下的种植方案以及预期利润，并以此给出了相应合理化建议。

关于问题一，我们使用线性规划模型求解，并利用蒙特卡洛算法验证求解结果的正确性。首先，由于题目假定各农作物的预期销售量、种植成本、亩产量和销售价格相对 2023 保持稳定，因此在目标函数内不必考虑种植风险。同时为简化模型，假设所有年份的市场需求量均近似为前一年的产量。接着根据题目提示，抽象出本题所求问题的决策变量和目标函数，即通过改变各年各地块各植物的种植面积最大化生产经济效益（生产经济效益的计算在第一小问和第二小问有区别，即产量超出预期销量时情况一超出部分不计收益，情况二收益以原价的 50%计）。然后根据题目要求明确了地块面积限制、市场需求限制、豆类植物轮作要求、不连续重茬种植要求、种植地集中要求和单类作物面积不宜过小要求共六项约束条件。在求解过程中，我们先通过关联相关数据、转换数据格式、去除拆分并补全数据表等步骤进行数据预处理。接着运用 Python 的 scipy 和 pandas 模块模拟随机农作物售价，构建合理参数矩阵，表达目标函数和约束条件的代码逻辑，并通过 linprog 函数求得模型最优解。最后通过蒙特卡洛数学模型中大量随机样本的生成和分析，为线性规划模型的准确性和鲁棒性提供了验证。

关于问题二，本题的参数根据题目要求对各农作物的预期销售量、种植成本、亩产量和销售价格进行了随机化处理，参数矩阵随之改变。本题依然使用线性规划模型求解，并利用蒙特卡洛算法验证求解结果的正确性。且参考上题的情况二，假设当亩产量大于预期销售量时，超过部分按预期销售价格的 50%降价出售。预处理数据，目标函数及约束条件皆沿用问题一。

关于问题三，本题依旧使用线性回归模型进行求解。为研究各种农作物之间的可替代性和互补性，以及预期销售量与销售价格、种植成本等之间的相关性，本题模型根据题目提示，通过模拟进行计算求解。首先为研究可替代性和相关性，本题通过设置可替代性系数 α 和互补性系数 β ,利用对两者产量的定量分析研究两者之间的替代互补关系。本题的目标函数依然不变，引入用于模拟的作物组，并增加相应的约束条件。

本文的模型假设合理，充分考虑了多种现实因素，建模过程具有一定的创新性，求解结果准确且具有实用价值。研究结果不仅对本乡村具有指导意义，也为类似地区的农业规划提供了重要参考。

关键词：线性规划，蒙特卡洛，PYTHON，乡村经济可持续发展

一、问题重述

1.1 问题背景

位于华北山区的乡村，由于耕地资源的有限，种类的丰富以及地块的分散。为了合理利用耕地资源，优化农作物种植结构，促进乡村经济的可持续发展。因此，在制定种植方案时，需要综合考虑作物类型、种植布局、轮作要求和田间管理等限制因素，从而实现经济效益的最大化。

1.2 题目信息

· 农作物生产规律、各地块的种植适应性等限制条件
· 地块和农作物的种类以及信息
· 附件还提供了 2023 年农作物的具体种植情况以及销量产量等相关数据

1.3 待求解问题

问题一：分析数据，

（1）当超出预期销售量的部分滞销时，农作物最优的种植方案。

(2) 当超出预期销售量的部分半价销售时，农作物最优的种植方案。

问题二：

考虑小麦和玉米的销售量年增长率（5%-10%）以及其他作物销售量的 ± 5%变化。

考虑亩产量受气候影响的 ± 10%波动。

考虑种植成本年均增长 5%的趋势。

考虑粮食价格基本稳定，蔬菜价格年均增长 5%，食用菌价格年均下降 1%-5%。

提出农作物最优的种植方案。

问题三：

综合考虑作物之间的替代性、互补性，以及预期销售量、销售价格与种植成本之间的相关性。与问题 2 的结果相比较，分析差异。

二、问题分析

2.1 问题一的分析

问题一的核心是确定在销售量和市场情况相对稳定的假设下，如何最大化村庄在 2024-2030 年期间的农业种植收益。需要考虑两种情况：滞销导致浪费和滞销部分降价出售。最终目标是找到最优的种植方案，以避免或最小化滞销的影响，并确保最大的经济效益。首先从建立一个数学模型入手，将农作物的种植面积与预期销售量、种植成本、亩产量和销售价格关联起来。针对每种作物，分析其在不同耕地类型和大棚内的种植收益，结合销售限制条件，优化种植方案。同时，通过数学模型或其他优化算法确定不同作物的最佳种植比例，并针对两种不同的销售处理方式分别进行计算。

2.2 问题二的分析

问题二相较于问题一，增加了未来几年内的不确定性因素，如预期销售量、亩产量、种植成本和销售价格的波动。目标是在不确定性条件下，找到一种稳健的种植方案，以最大化经济效益，同时降低潜在的种植风险。可以采用情景分析等方法，考虑不同作物的价格、产量和销售量的波动情况。通过模拟生成不同的未来场景，评估在各种不确定条件下的种植收益，最终确定一种综合性的最优种植方案。还需要在模型中加入约束条件，如豆类作物轮作、种植地块分配等。

2.3 问题三的分析

问题三进一步考虑了现实中的作物替代性和互补性，以及各项经济变量之间的相关性。该问在问题二的基础上，通过更复杂的模型，找到更加优化的种植策略，从而进一步提升农业经济效益并降低风险。可以通过研究作物之间的替代和互补关系，以及经济变量之间的相关性。然后，进行优化求解。最终需要在充分考虑作物之间的相互关系和市场动态变化的前提下，确定一个能够最大化综合收益的种植方案，并将其与问题二的结果进行比较分析，评估改进效果。

三、模型假设

· 假设在 2024-2030 年间不会发生极端气候事件（如干旱、洪水、寒潮等）

· 假设农作物在生长过程中不受病虫害影响

· 假设农作物的市场需求和价格变化只遵循给定的趋势

· 假设每块土地的肥力保持不变，不考虑因持续耕作可能导致的土地退化或肥力提升

· 假设政府政策变化不会影响到乡村种植策略的改变

四、符号定义

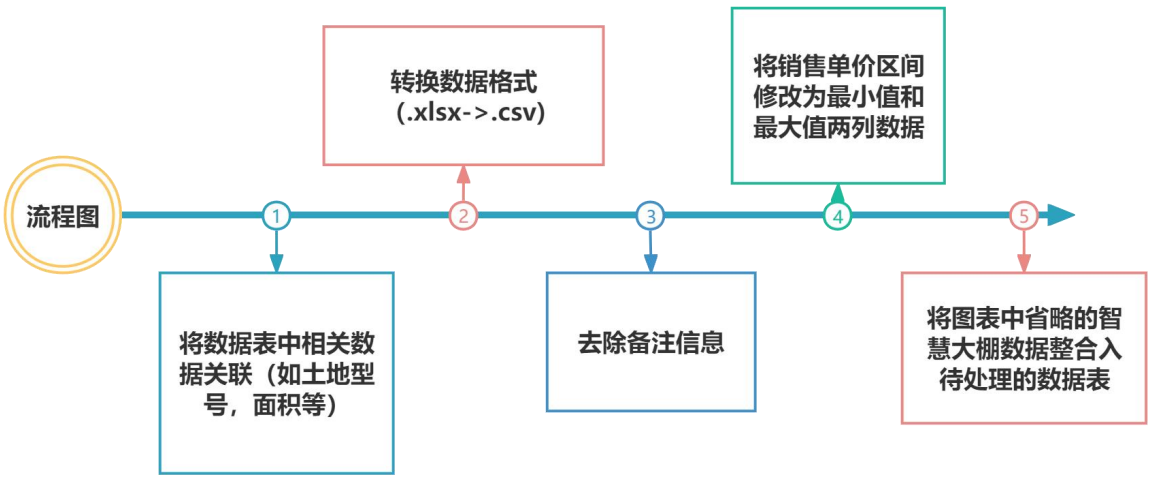
符号	含义
x_{ijt}	第 i 块地在第 t 季节种植第 j 种作物的面积（亩）
A_i	第 i 块地的总面积
Y_{jt}	第 j 种作物在第 t 季节的预期亩产量

P_{jt}	第 j 种作物在第 t 季节的销售价格
C_{jt}	第 j 种作物在第 t 季节的种植成本
D_{jt}	第 j 种作物在第 t 季节的市场需求量
T	轮作年份数
k_{ijt}	第 i 块地在第 t 季节种植的第 j 种作物是否为豆类（1为否，0为是）
N_j	第 j 种作物在每个季节的种植地块数量上限
M_{jt}	第 j 种作物在第 t 季节的最小种植面积

五、数据预处理

5.1 数据处理

将附件 2 给出的 2023 年的种植情况和销售数据作为模型的基础输入数据，预处理不同作物的亩产量、种植成本、预期销售量和价格等信息。由于附件给出的表格含有备注信息、省略的数据、合并的单元格和不同的附表等影响因素，不适合机器阅读。所以先将表格经由人工和机器进行处理以方便后续计算。



具体数据呈现如下：

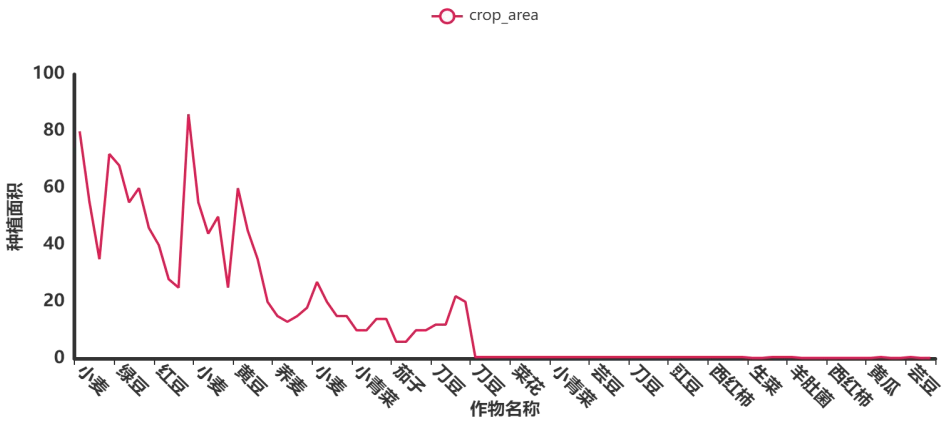
land_id	land_type	land_area
A1	平旱地	80.0
A2	平旱地	55.0
A3	平旱地	35.0
A4	平旱地	72.0
A5	平旱地	68.0
...		
F2	智慧大棚	0.6
F3	智慧大棚	0.6
F4	智慧大棚	0.6

land_id	crop_name	crop_area	season
A1	小麦	80.0	单季
A2	玉米	55.0	单季
A3	玉米	35.0	单季
A4	黄豆	72.0	单季
A5	绿豆	68.0	单季
...			
F4	芸豆	0.6	第一季
F4	芹菜	0.3	第二季
F4	菠菜	0.3	第二季

crop_name	land_type	season	yield	cost	price_min	price_max
黄豆	平旱地	单季	400.0	400.0	2.5	4.0
黑豆	平旱地	单季	500.0	400.0	6.5	8.5
红豆	平旱地	单季	400.0	350.0	7.5	9.0
绿豆	平旱地	单季	350.0	350.0	6.0	8.0
爬豆	平旱地	单季	415.0	350.0	6.0	7.5
...						
空心菜	智慧大棚	第二季	11000.0	5500.0	3.6	7.2
黄心菜	智慧大棚	第二季	5400.0	2750.0	4.8	6.0
芹菜	智慧大棚	第二季	6000.0	1200.0	3.8	5.8

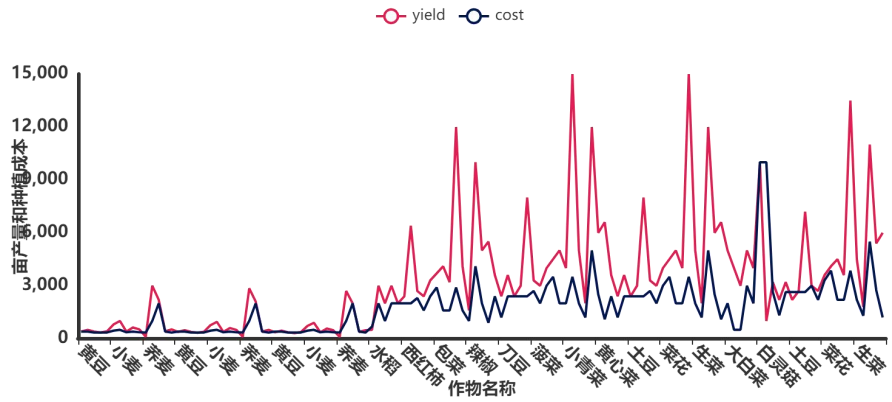
图一：关联相关数据　　图二：转换数据格式　　图三：去除，拆分或补全所需数据表

5.2 数据分析



图一：不同作物的种植面积

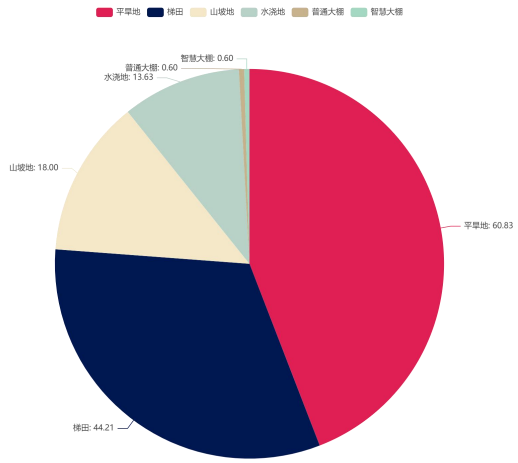
图一显示了 2023 年不同作物的种植面积分布情况，各作物在乡村地区的耕种面积占比表明了农民种植选择的多样性与区域差异。通过对这些数据的分析，可以发现主要作物种植面积的变化趋势以及其在乡村经济中的地位。



图二：不同作物类型亩产量和种植成本

图二展示了 2023 年各类作物的销售数据与亩产量之间的关系。分析这些数据，可以揭示出不同作物在市场中的竞争力及其经济效益。亩产量的高低直接影响了作物的市场价

格，而销售数据的波动则反映了农产品的需求情况。



图三：不同土地类型占比

图三则详细展示了乡村耕地资源的土地类型占比情况。不同类型的土地资源对作物种植的适应性和限制条件存在显著差异。耕地资源的合理分配不仅影响到单一作物的产量，还对整个乡村地区的农业生态环境产生深远影响。
因此，了解和掌握乡村地区的土地资源类型及其利用情况，对于制定合理的农业发展政策至关重要。

为方便对比和评价，使用 python 对 2023 年的种植数据进行计算（见附录），可以得出 2023 年种植利润如下表所示：

单价处理方式	2023 年的总利润
使用价格范围左端点作为单价	4894774.0
使用价格范围右端点作为单价	6957922.5
使用价格范围平均值作为单价	5926348.25
在价格范围内随机取值作为单价	5858119.16

六、模型建立与求解

6.1.1 问题一第一情况的模型建立

本题利用线性规划（LP）进行求解并采用蒙特卡洛模型（MCM）进行模型交叉验证。线性规划数学模型能够在给定约束条件下得到目标函数的精确最优解，从而得出优化资源的合理决策。同时，蒙特卡洛数学模型中大量随机样本的生成和分析，为线性规划模型的准确性和鲁棒性提供了验证。

由于题目假定各农作物的预期销售量、种植成本、亩产量和销售价格相对 2023 保持稳定，因此在目标函数内不必考虑种植风险。同时为简化模型，假设所有年份的市场需求

量均近似为前一年的产量。因此本题的目标函数唯一，即最大化生产的经济效益。约束条件共六项，即地块面积限制、市场需求限制、豆类植物轮作要求、不连续重茬种植要求、种植地集中要求和单类作物面积不宜过小要求。以下为本题目标函数与约束条件的具体罗列。

目标函数：

$$Z=\sum_{i}\sum_{j}\sum_{t}(P_{jt}\cdot Y_{jt}\cdot x_{ijt}-C_{jt}\cdot x_{ijt})$$

约束条件：

· 耕地面积限制：每块地的种植面积总和不能超过该地块的可用面积

$$\sum_{j}x_{ijt}\leq A_{i},\quad \forall i,\forall t$$

· 市场需求限制：各作物总产量不得大于市场需求

$$\sum_{i}\sum_{t}x_{ijt}\cdot Y_{jt}\leq D_{jt},\quad \forall j$$

·豆类作物三年内种植一次：每地块在三年内豆类植物的种植次数必须大于一

$$\sum_{t}^{t+2T}x_{ijt}\cdot k_{ijt}\geq 1,\quad \forall t,i$$

·不重茬：同一作物连续两季在同一土地上种植面积的和不得大于该地块面积

$$x_{ijt}+x_{ij,t}\leq A_{i},\quad \forall i,j,t$$

· 种植区域不分散：任何作物的种植地个数不得超过预定上限

$$\sum_{i}(x_{ijt}>0)\leq N_{j},\quad \forall j,t$$

· 单类作物面积不宜过小：任何实际种植的作物面积不应低于预设的下限

$$x_{ijt}\geq M_{jt}\cdot(x_{ijt}>0),\quad \forall i,j,t$$

6.1.2 问题一的求解

使用 python 对已经预处理过的数据：作物数据、地块数据、2023 年种植情况数据进行读取并使用 pandas 和 scipy 模块等进行分析和计算(代码见附录)可以得出如下结果：

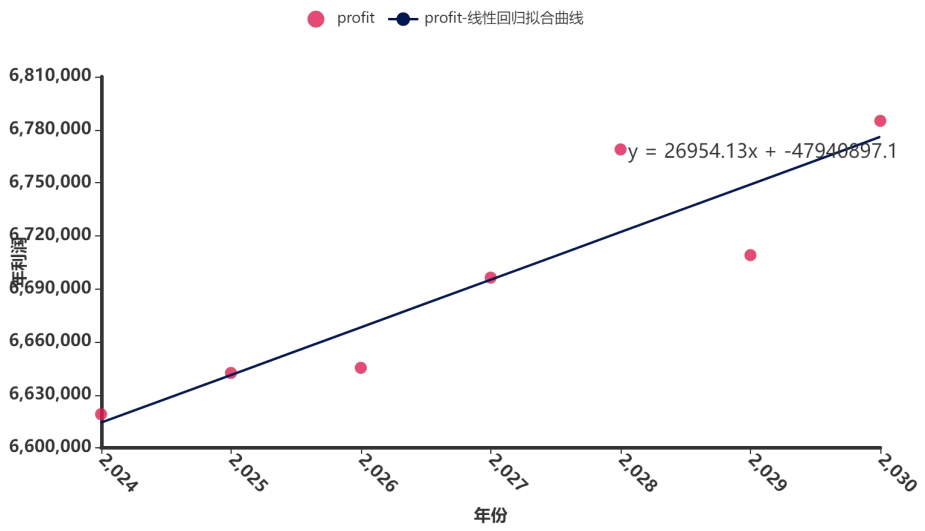
1. 2023 至 2030 年逐年的最优种植方案表（result1_1.xlsx）部分数据如下：

	地块名	黄豆	黑豆	红豆	绿豆	爬豆	小麦	玉米	谷子	高粱	黍子	荞麦	南瓜	红薯	莜麦	大麦
第一季	A1	0	0	0	0	0	80	0	0	0	0	0	0	0	0	0
	A2	0	0	0	0	0	0	20.01	0	0	23.66	11.32	0	0	0	0
	A3	0	35	0	0	0	0	0	0	0	0	0	0	0	0	0
	A4	0	0	0	0	0	72	0	0	0	0	0	0	0	0	0
	A5	0	0	0	0	0	0.083	0	0	0	0	0	0	0	7.423	15.63
	A6	0	0	0	0	0	0	0	0	37.31	0	0	0	0	0	0
	B1	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0
	B2	0	0	0	0	0	0	0	22.48	0	0	0	9.53	0	0	0
	B3	0	0	0	0	17.69	0	0	0	0	0	0	0	0	0	0
	B4	0	0	0	0	0	0	24.02	0	0	0	0	0	0	0	0
	B5	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0
	B6	0	0	0	14.29	0	0	71.71	0	0	0	0	0	0	0	0
	B7	0	0	0	0	0	0	0	55	0	0	0	0	0	0	0
	B8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	B9	0	0	0	0	0	0	0	35.77	0	0	0	0	14.23	0	0
	B10	0	0	0	0	0	0	0	25	0	0	0	0	0	0	0
	B11	9.38	0	50.62	0	0	0	0	0	0	0	0	0	0	0	0
	B12	45	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	B13	32.85	2.149	0	0	0	0	0	0	0	0	0	0	0	0	0

2. 2023 年至 2030 年逐年使用最优种植方案后计算所得的利润（profit1.1.xlsx）

年份	该年最优利润
2024	6619049.0711056
2025	6642370.3815447
2026	6645286.53734138
2027	6696306.789364
2028	6768820.41835522
2029	6709028.2294905
2030	6785004.42129809

3.将上表通过 SPSSPRO 软件绘制成散点图，并进行线性回归拟合，结果如下



2023 年至 2030 年逐年使用最优种植方案后计算所得的结果

将数据与 2023 年的利润数据相比较，其利润在 2024 年便有较大提升，并且逐年递增。利润与年份总体近似为正线性关系。

3. 采用蒙特卡洛法（见附录代码）对结果进行模型比较验证，结果显示线性规划模型的结果具有一定的合理性且更接近真实值。

6.2.1 问题一第二种情况模型的建立与求解

本题依然利用线性规划（LP）进行求解并采用蒙特卡洛模型（MCM）进行模型交叉验证。模型也假设不变。但第二种情况的目标函数与第一种情况的不同之处为，当作物的总产量超出相应的预期销售量，超出部分不浪费，而是按照 2023 年销售价格的 50%降价出售处理。因此目标函数的系数发生变化，但依然以最大化生产经济效益为目的。约束条件不变

目标函数：

$$Z=\sum_{i}\sum_{j}\sum_{t}\left(P_{jt}\cdot\min\{D_{jt},Y_{jt}\}+0.5\cdot P_{jt}\cdot\max\{D_{jt}-Y_{jt},\ 0\}-C_{jt}\cdot x_{ijt}\right)$$

约束条件：

（沿用问题一的第一种情况）

6.2.1 求解结果

在第一种情况代码的基础上，对目标函数中的参数进行了改动，加入了上述降价销售的部分，并对约束条件做出了相应的改动（代码见附录），并进行计算，可以得出如下结果：

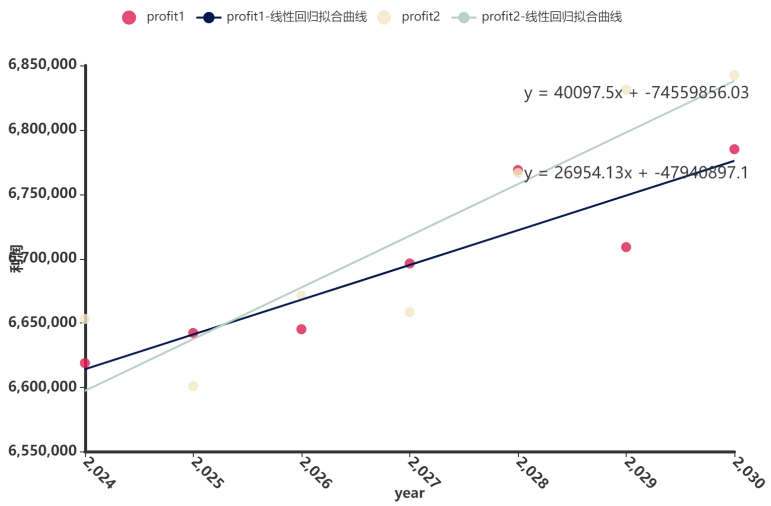
1. 在新的超售降价的条件下所得出的 2023 至 2030 年逐年的最优种植方案表（result1_2.xlsx）部分数据如下：

	地块名	黄豆	黑豆	红豆	绿豆	爬豆	小麦	玉米	谷子	高粱	黍子	荞麦	南瓜	红薯	莜麦	大麦
第一季	A1	0	0	0	57.37	0	22.63	0	0	0	0	0	0	0	0	0
	A2	0	0	0	0	0	55	0	0	0	0	0	0	0	0	0
	A3	0	0	0	0	18.99	0	3.262	0	0	0	0	0	12.75	0	0
	A4	0	7.805	0	0	0	26.37	37.83	0	0	0	0	0	0	0	0
	A5	0	0	0	0	0	0	37.56	30.44	0	0	0	0	0	0	0
	A6	0	0	4.539	0	0	0	0	8.93	0	19.69	13.18	8.669	0	0	0
	B1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	B2	0	0	0	0	0	0	29.71	0	0	0	0	0	0	0	16.29
	B3	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0
	B4	0	28	0	0	0	0	0	0	0	0	0	0	0	0	0
	B5	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0
	B6	0	0	0	0	0	49.18	0	0	0	0	0	0	0	0	0
	B7	0	0	0	0	0	0	0	30.35	0	0	0	0	0	24.65	0
	B8	0	0	0	0	0	44	0	0	0	0	0	0	0	0	0
	B9	49.82	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	B10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	B11	0	0	0	16.9	0	0	0	0	43.1	0	0	0	0	0	0
	B12	45	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	B13	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0

2. 在新条件下 2023 至 2030 年逐年使用最优种植方案后计算所得的利润

（profit1_2.xlsx）

3.将上述利润结果数据通过 SPSSPRO 软件绘制成散点图，结果显示：



6.3.1 问题二模型的建立与求解

本题依然利用线性规划（LP）进行求解并采用蒙特卡洛模型（MCM）进行模型交叉验证。 并假设当亩产量大于预期销售量时，超过部分按预期销售价格的 50%降价出售。相较与前一问，本题根据题目提示加入了变量的不确定性处理，对预期销售量、亩产量、种植成本和销售价格进行了随机化处理以适应要求。目标函数依然为最大化生产经济效益。约束条件沿用问题一。

变量的不确定性处理：

· 预期销售量：

小麦和玉米的年增长率介于 5%-10%之间

$$D_{jt}=D_{jt}\times(1+r_{jt}),r_{jt}\sim Uniform(0.05,0.10),for j\in 小麦,玉米$$

其他农作物的变化在± 10%之间：

$$Y_{jt}=Y_{j,2023}\times(1+r_{jt}),\quad r_{jt}\sim Uniform(-0.10,0.10)$$

· 亩产量：

每年变化在±10%之间：

$$Y_{jt}=Y_{j,2023}\times(1+r_{jt}),\quad r_{jt}\sim \mathrm{Uniform}(-0.10,0.10)$$

· 种植成本：

每年增长 5%左右：

$$C_{jt}=C_{j,2023}\times(1+0.05)^{t-2023}$$

· 销售价格：

粮食作物基本稳定：

$$P_{jt}=P_{j,2023},\ \text{for}\ j\in\text{粮食类作物}$$

蔬菜类作物平均每年增长 5%左右：

$$P_{jt}=P_{j,2023}\times(1+0.05)^{t-2023},\ \text{for}\ j\in\text{蔬菜类作物}$$

食用菌每年下降 1%-5%，其中羊肚菌每年下降 5%左右：

$$P_{jt}=P_{j,2023}\times(1+r_{jt}),\ r_{jt}\sim\text{Uniform}(-0.05,-0.01),\ \text{for}\ j\in\text{食用菌}$$

$$P_{jt}=P_{j,2023}\times(1-0.05)^{t-2023},\ \text{for}\ j=\text{羊肚菌}$$

目标函数：

$$Z=\sum_i\sum_j\sum_t\left(P_{jt}\cdot\min\{D_{jt},Y_{jt}\}+0.5\cdot P_{jt}\cdot\max\{D_{jt}-Y_{jt},\ 0\}-C_{jt}\cdot x_{ijt}\right)$$

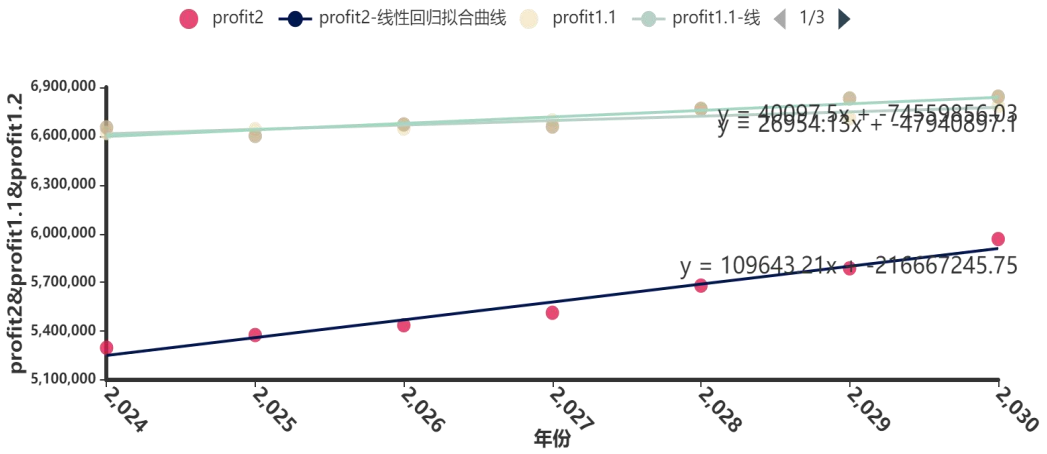
约束条件：（沿用题目一的约束条件）

6.3.2 求解结果

1. 2023 至 2030 年逐年的最优种植方案表（result.xlsx）2024 年部分数据如下：

	地块名	黄豆	黑豆	红豆	绿豆	爬豆	小麦	玉米	谷子	高粱	黍子	荞麦	南瓜	红薯	莜麦	大麦
第一季	A1	0	0	0	57.37	0	22.63	0	0	0	0	0	0	0	0	0
	A2	0	0	0	0	0	55	0	0	0	0	0	0	0	0	0
	A3	0	0	0	0	18.99	0	3.262	0	0	0	0	0	12.75	0	0
	A4	0	7.805	0	0	0	26.37	37.83	0	0	0	0	0	0	0	0
	A5	0	0	0	0	0	0	37.56	30.44	0	0	0	0	0	0	0
	A6	0	0	4.539	0	0	0	0	8.93	0	19.69	13.18	8.669	0	0	0
	B1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	B2	0	0	0	0	0	0	29.71	0	0	0	0	0	0	0	16.29
	B3	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0
	B4	0	28	0	0	0	0	0	0	0	0	0	0	0	0	0
	B5	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0
	B6	0	0	0	0	0	49.18	0	0	0	0	0	0	0	0	0
	B7	0	0	0	0	0	0	0	30.35	0	0	0	0	0	24.65	0
	B8	0	0	0	0	0	44	0	0	0	0	0	0	0	0	0
	B9	49.82	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	B10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	B11	0	0	0	16.9	0	0	0	0	43.1	0	0	0	0	0	0
	B12	45	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	B13	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0

2. 题目二各年的目标函数（即生产经济效益）与题目一的情况二所得结果的比较：



6.4.1 问题三的模型建立与求解

为研究各种农作物之间的可替代性和互补性，以及预期销售量与销售价格、种植成本等之间的相关性，本题模型根据题目提示，通过模拟进行计算求解。首先为研究可替代性和相关性，本题设置可替代性系数 $\alpha\{j,m\}$和互补性系数 $\beta\{j,m\}$，通过对两者产量的定量分析研究两者之间的替代互补关系。本题的目标函数依然不变，引入用于模拟的作物组，并增加相应的约束条件。

可替代性和互补性作物组模拟数据示例：

· 可替代性作物组：

如黄豆和黑豆，红豆和绿豆。

· 互补性作物组：

如玉米和小麦，土豆和白萝卜，西红柿和茄子。

目标函数：

$$Z=\sum_{i}\sum_{j}\sum_{t}\left(P_{jt}\cdot\min\{D_{jt},Y_{jt}\}+0.5\cdot P_{jt}\cdot\max\{D_{jt}-Y_{jt},\ 0\}-C_{jt}\cdot x_{ijt}\right)$$

约束条件：

· 可替代性约束：

限制被替代性作物的总种植面积不超过总面积的 10%

$$\alpha\{j,m\}\cdot x_{ijt}+x_{ijt}\quad\leq\quad 0.1A_{i},\qquad\forall j,m,i,t$$

· 互补性约束：

限制互补性作物的种植面积比例不超过总面积的 10%。

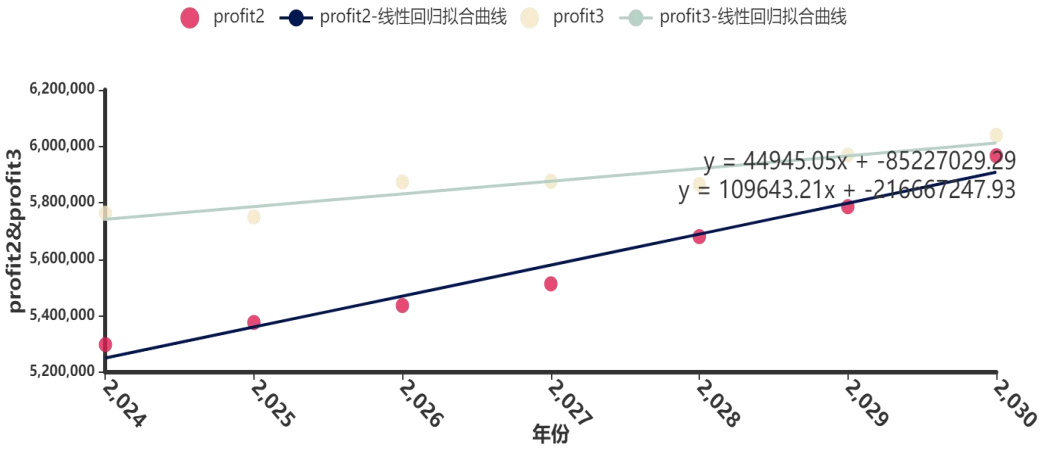
$$\beta\{j,m\}\cdot x_{ijt}\cdot x_{imt}\leq 0.1\gamma\qquad\forall j,m,i,t$$

6.4.2 求解结果

1.通过模拟得出，相互之间可替代性较优的作物有黄豆和黑豆，红豆和绿豆。具有较强互补性的作物有玉米和小麦，土豆和白萝卜，西红柿和茄子。

2.通过模拟分析预期销售量，销售价格和种植成本之间的关系，能够提前优化种植规划使的利润逐年上升且更稳定。

将上述利润结果数据通过 SPSSPRO 软件绘制成散点图结果显示：



七、模型的敏感性分析

在第二问模型的基础上，对模型算法进行敏感性分析，方式如下：

1. 将销售单价上下浮动 10%后计算 7 年利润并绘图（下图 1）
2. 将作物亩产量上下浮动 10%后计算 7 年利润并绘图（下图 2）
3. 将每亩成本上下浮动 10%后计算 7 年利润并绘图（下图 3）

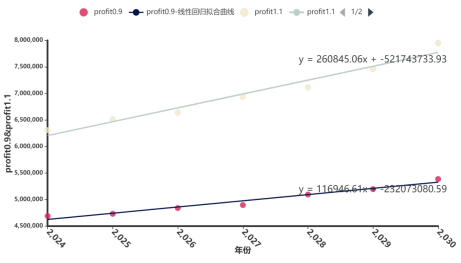


图 1：浮动销售单价

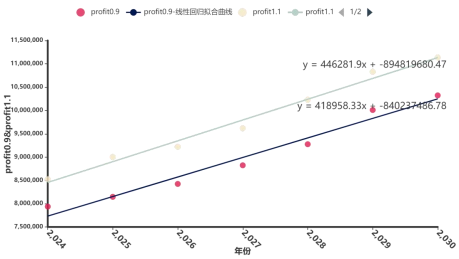


图 2：浮动亩产量

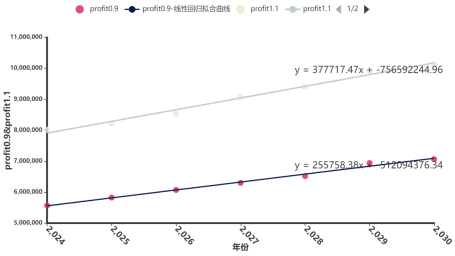


图 3：浮动每亩成本

结果显示，该模型对作物销售单价的变化最为敏感，因此在计算时应当格外注意。

八、模型的评价与优化

8.1 模型的优点

1. 约束条件充分且合理

模型充分考虑了不同类型耕地（如平旱地、梯田、山坡地和水浇地）的差异，对作物不能连续重茬种植进行了约束，这种划分不仅能够更准确地反映出每种地块的特性，便于因地制宜地制定种植策略，还符合农业实践中的土壤健康管理，避免了因重茬而导致的减产风险。

普通大棚和智慧大棚的引入，使得耕地资源得到了更高效的利用，尤其是智慧大棚能够种植两季作物，进一步提高了产出效率。豆类作物的轮作要求，有助于维持土壤肥力，减少化肥依赖，符合可持续农业的发展方向。

2. 可操作性强

在问题 3 中，通过模拟数据进行求解的方式，使得农户可以根据不同的市场预测情景调整种植策略。这种模拟方法为农户提供了应对市场变化的实践工具，有助于减少实际操作中的风险。

3.长期视角和可持续性

模型分析了 2024 至 2030 年期间的种植规划，体现了对农业生产的长期视角和规划能力。这有助于乡村在未来几年内实现更可持续的发展。通过因地制宜的种植方案优化有限耕地资源的利用效率，尤其是在不同地块的种植选择上体现了生态和经济的双重考虑。

4. 计算效率高

采用线性规划模型，与其他非线性规划模型相比，计算速度较快，可以更快的得出计算计算结果，便于快速的比较和分析。

8.2 模型的缺点

模型仍然具有一定的优化空间，具体可以从如下几个角度进行优化：

1.价格、产量和市场需求的动态变化

当前模型中，价格、产量和市场需求的变化虽在固定范围内，但是总体采用的是随机变化的方式，可以考虑通过采集更多历史数据进行分析等方式，结合天气、市场等多个因素来进行预测，从而让分析结果能更符合真实值

2.风险管理

当前模型并未引入风险管理等有关内容（如天气、病虫害等的问题）。可以在之后的优化中进行综合考虑，如限制某些高风险作物的种植面积等方式，使得种植方案更具有可行性。

附录

附录 A：支撑材料文件列表

文件名	文件类型	简介
code/calc_2023.py	python 代码文件	用于计算 2023 年的销售利润
code/data_pre_process.py	python 代码文件	用于附件数据的预处理
code/Q1.1py	python 代码文件	第一题情况 1 的求解代码
code/Q1.2py	python 代码文件	第一题情况 2 的求解代码
code/Q2.py	python 代码文件	第二题的求解代码
code/Q3.py	python 代码文件	第三题的求解代码
code/MCM.py	python 代码文件	蒙特卡洛模拟代码
code/results_template.xlsx	Excel 表格文件	种植方案的数据表格模板
data/processed_crop_data.csv	csv 表格文件	预处理的 2023 年产量和销售情况表
data/processed_land_data.csv	csv 表格文件	预处理的各种土地类型的统计表
data/processed_planting_data.csv	csv 表格文件	预处理的 2023 年种植情况表
results/1.1.xlsx	Excel 表格文件	第一题情况 1 的程序输出的种植方案
results/1.2.xlsx	Excel 表格文件	第一题情况 2 的程序输出的种植方案
results/results2.xlsx	Excel 表格文件	第二题的程序输出的种植方案
results/results3.xlsx	Excel 表格文件	第三题的程序输出的种植方案
results/profit1.1.xlsx	Excel 表格文件	第一题情况 1 的程序输出的利润表
results/profit1.2.xlsx	Excel 表格文件	第一题情况 2 的程序输出的利润表
results/profit2.xlsx	Excel 表格文件	第二题的程序输出的的利润表
results/profit3.xlsx	Excel 表格文件	第三题的程序输出的的利润表
final/result1_1.xlsx	Excel 表格文件	第一题情况 1 的最终种植方案
final/result1_2.xlsx	Excel 表格文件	第一题情况 2 的最终种植方案
final/result2.xlsx	Excel 表格文件	第二题的最终种植方案
code/sensitivity.py	python 代码文件	敏感性分析代码
results/sensitivity_*_*.xlsx	Excel 表格文件	共 6 个文件，敏感性分析计算结果
graph/1.png	PNG 图片文件	数据处理流程图

graph/2.jpeg	JPEG 图片文件	预处理数据示例截图
graph/3.png	PNG 图片文件	不同作物种植面积数据分析图
graph/4.png	PNG 图片文件	不同作物类型亩产量和成本数据分析图
graph/5.png	PNG 图片文件	不同土地类型占比分析图
graph/6.png	PNG 图片文件	问题一情况 1 最优种植方案表截图
graph/7.png	PNG 图片文件	问题一情况 1 最优种植方案的利润趋势图
graph/8.png	PNG 图片文件	问题一情况 2 最优种植方案表截图
graph/9.png	PNG 图片文件	问题一情况 2 最优种植方案的利润趋势图
graph/10.png	PNG 图片文件	问题二最优种植方案表截图
graph/11.png	PNG 图片文件	问题二最优种植方案的利润趋势图
graph/12.png	PNG 图片文件	问题三最优种植方案的利润趋势对比图
graph/13.png	PNG 图片文件	敏感性分析结果：浮动销售单价
graph/14.png	PNG 图片文件	敏感性分析结果：浮动亩产量
graph/15.png	PNG 图片文件	敏感性分析结果：浮动每亩成本

附录 B：数据预处理实现代码

```
import pandas as pd

# 读取 Excel 文件
land_df = pd.read_excel('data/data_land.xlsx')
crop_df = pd.read_excel('data/data_crop.xlsx')
planting_df = pd.read_excel('data/data_2023.xlsx')

# 检查缺失值
print("Land Data Missing Values:\n", land_df.isnull().sum())
print("Crop Data Missing Values:\n", crop_df.isnull().sum())
print("Planting Data Missing Values:\n", planting_df.isnull().sum())

# 数据清洗和转换
# 处理地块数据
land_df['land_area'] = land_df['land_area'].astype(float)

# 处理作物数据（价格范围转换为最低价格和最高价格）
crop_df[['yield', 'cost']] = crop_df[['yield', 'cost']].astype(float)
crop_df['price_min'] = crop_df['price_range'].apply(lambda x: float(x.split('-')[0]))
crop_df['price_max'] = crop_df['price_range'].apply(lambda x: float(x.split('-')[1]))
crop_df.drop(columns=['price_range'], inplace=True)

# 处理 2023 年种植数据
planting_df['crop_area'] = planting_df['crop_area'].astype(float)

# 保存数据
land_df.to_csv('data/processed_land_data.csv', index=False)
crop_df.to_csv('data/processed_crop_data.csv', index=False)
```

```
planting_df.to_csv('data/processed_planting_data.csv', index=False)

print("Data saved.")
```

附录 C：计算 2023 年利润实现代码

```
import pandas as pd
import random

planting_data = pd.read_csv('data/processed_planting_data.csv') # 2023 年的种植数据
# 读取之前处理好的数据
crop_data = pd.read_csv('data/processed_crop_data.csv')
land_data = pd.read_csv('data/processed_land_data.csv')

# 添加 land_type 到 planting_data
planting_data['land_type'] = planting_data['land_id'].apply(lambda x: land_data[land_data['land_id'] ==
x]['land_type'].values[0])

merged_data_2023 = pd.merge(planting_data, crop_data, on=['crop_name', 'season', 'land_type'])

merged_data_2023['profit'] = (merged_data_2023['price_min'] * merged_data_2023['yield'] -
merged_data_2023['cost']) * merged_data_2023['crop_area']

total_profit_2023 = merged_data_2023['profit'].sum()

print(f"2023 年的总利润（最低单价）为: {total_profit_2023} 元")

merged_data_2023['profit'] = (merged_data_2023['price_max'] * merged_data_2023['yield'] -
merged_data_2023['cost']) * merged_data_2023['crop_area']

total_profit_2023 = merged_data_2023['profit'].sum()

print(f"2023 年的总利润（最高单价）为: {total_profit_2023} 元")

# price 使用平均值
merged_data_2023['profit'] = ((merged_data_2023['price_min'] + merged_data_2023['price_max']) / 2 *
merged_data_2023['yield'] - merged_data_2023['cost']) * merged_data_2023['crop_area']

total_profit_2023 = merged_data_2023['profit'].sum()

print(f"2023 年的总利润（平均单价）为: {total_profit_2023} 元")

# price 使用随机值
merged_data_2023['random_price'] = merged_data_2023.apply(
    lambda row: random.uniform(row['price_min'], row['price_max']), axis=1
)
```

```
merged_data_2023['profit'] = (merged_data_2023['random_price'] * merged_data_2023['yield'] -
merged_data_2023['cost']) * merged_data_2023['crop_area']

total_profit_2023 = merged_data_2023['profit'].sum()

print(f"2023 年的总利润（随机单价）为: {total_profit_2023} 元")
```

附录 D：问题一第一问实现代码

```
import pandas as pd
from scipy.optimize import linprog
import random

planting_data = pd.read_csv('data/processed_planting_data.csv') # 2023 年的种植数据
yearly_data = {
    2023: planting_data
}

# 读取之前处理好的数据
crop_data = pd.read_csv('data/processed_crop_data.csv')
land_data = pd.read_csv('data/processed_land_data.csv')
# 豆类作物
legume_crops = ['黄豆', '黑豆', '红豆', '绿豆', '爬豆', '豇豆', '刀豆', '芸豆']

yearly_profit = {}

for current_year in range(2024, 2031):
    # 读取数据
    planting_data = yearly_data[current_year - 1] # 使用前一年的种植数据作为基础迭代

    # 添加 land_type 到 planting_data
    planting_data['land_type'] = planting_data['land_id'].apply(lambda x: land_data[land_data['land_id'] ==
x]['land_type'].values[0])

    # 合并 crop_data 和 planting_data
    merged_data = pd.merge(crop_data, planting_data, on=['crop_name', 'season', 'land_type'])

    print(f"开始计算{current_year}年的最优解")

    # 计算市场需求量（亩产量 * 去年种植面积）
    merged_data['market_demand'] = merged_data['yield'] * merged_data['crop_area']

    # 计算每种作物在每个季节的总市场需求量
    market_demand = merged_data.groupby(['crop_name', 'season'])['market_demand'].sum().reset_index()
    # 单季的作物改为第一季
```

```
    market_demand.loc[market_demand['season'] == '单季', 'season'] = '第一季'
    crop_data.loc[crop_data['season'] == '单季', 'season'] = '第一季'
    # print(market_demand)

    # 提取参数
    crops = crop_data['crop_name'].unique()
    lands = land_data['land_id'].unique()
    seasons = ["第一季", "第二季"]
    seasons = pd.Series(seasons)


    # 定义变量数量
    num_crops = len(crops)
    num_lands = len(lands)
    num_seasons = len(seasons)

    # 定义目标函数系数
    c = []
    for land in lands:
        for crop in crops:
            for season in seasons:
                crop_info = crop_data[(crop_data['crop_name'] == crop) & (crop_data['season'] == season) &
(crop_data['land_type'] == land_data[land_data['land_id'] == land]['land_type'].values[0])]

                if not crop_info.empty:
                    # price = crop_info['price_min'].values[0]
                    # price 使用最小值和最大值之间的随机数
                    price = random.uniform(crop_info['price_min'].values[0], crop_info['price_max'].values[0])
                    yield_per_acre = crop_info['yield'].values[0]
                    cost = crop_info['cost'].values[0]
                    c.append(-(price * yield_per_acre - cost))  # 因为 linprog 求最小值，所以这里加负号
                else:
                    c.append(0)
    # A b 存储约束条件
    A = []
    b = []

    # 地块面积限制
    for land in lands:
        for season in seasons:
            row = [0] * (num_crops * num_lands * num_seasons)
            for i, crop in enumerate(crops):
                index = lands.tolist().index(land) * num_crops * num_seasons + i * num_seasons +
seasons.tolist().index(season)
                row[index] = 1
```

```
                A.append(row)
                b.append(land_data[land_data['land_id'] == land]['land_area'].values[0])

    # 市场需求限制
    for crop in crops:
        for season in seasons:
            row = [0] * (num_crops * num_lands * num_seasons)
            for land in lands:
                index = lands.tolist().index(land) * num_crops * num_seasons + crops.tolist().index(crop) *
num_seasons + seasons.tolist().index(season)
                crop_info = crop_data[(crop_data['crop_name'] == crop) & (crop_data['season'] == season)]
                if not crop_info.empty:
                    yield_per_acre = crop_info['yield'].values[0]
                    row[index] = yield_per_acre
            A.append(row)
            demand_info = market_demand[(market_demand['crop_name'] == crop) & (market_demand['season']
== season)]
            if not demand_info.empty:
                b.append(demand_info['market_demand'].values[0])
            else:
                b.append(0)


    # 豆类轮作要求
    if current_year >= 2025:
        previous_years = [current_year - 1, current_year - 2]
        previous_data = pd.concat([yearly_data[y] for y in previous_years if y in yearly_data])

        for land in lands:
            legume_planted = previous_data[(previous_data['land_id'] == land) &
(previous_data['crop_name'].isin(legume_crops))]
            if legume_planted.empty:
                for crop in legume_crops:
                    row = [0] * (len(crops) * len(lands) * len(seasons))
                    for season in seasons:
                        index = list(lands).index(land) * len(crops) * len(seasons) + list(crops).index(crop) *
len(seasons) + list(seasons).index(season)
                        row[index] = -1
                    A.append(row)
                    b.append(0)



    # 求解线性规划问题
    res = linprog(c, A_ub=A, b_ub=b, method='highs')
```

```
        # 输出结果
        if res.success:
            print(f"{current_year}年最优解已找到")
            print(f"最优解：{ -res.fun}")

            yearly_profit[current_year] = -res.fun

            rows = []
            for i, land in enumerate(lands):
                for j, crop in enumerate(crops):
                    for k, season in enumerate(seasons):
                        index = i * num_crops * num_seasons + j * num_seasons + k
                        if res.x[index] > 0:
                            rows.append({
                                'land_id': land,
                                'crop_name': crop,
                                'crop_area': res.x[index],
                                'season': season
                            })

            current_year_data = pd.DataFrame(rows)
            print(f"{current_year}年的种植数据：")
            print(current_year_data.head())

            yearly_data[current_year] = current_year_data
        else:
            print(f"{current_year}年没有找到可行解")
            break


# 保存到 excel
template_df = pd.read_excel("code/results_template.xlsx")
# 0 - 53 行 为第一季
land_id_first = template_df["地块名"][0:54]
land_id_first = land_id_first + "_第一季"
# 54 - 81 行 为第二季
land_id_second = template_df["地块名"][54:82]
land_id_second = land_id_second + "_第二季"

template_cols = template_df.columns[2:]
with pd.ExcelWriter('results/1.1.xlsx') as writer:
    for year, data in yearly_data.items():
        if (year == 2023):
            continue
```

```
        df = pd.DataFrame(index=range(0, len(land_id_first) + len(land_id_second)), columns=template_cols)
        df["地块名"] = pd.concat([land_id_first, land_id_second], ignore_index=True)
        for i, row in data.iterrows():
            land_id = row['land_id']
            crop_name = row['crop_name']
            crop_area = row['crop_area']
            season = row['season']
            land_name_with_season = land_id + "_" + season
            index = df[df["地块名"] == land_name_with_season].index[0]
            df.at[index, crop_name] = crop_area

        # 空缺值填充为 0
        df = df.fillna(0)
        df.to_excel(writer, sheet_name=str(year), index=False)
        print(f"已保存{year}年的结果")


# 保存年度利润
yearly_profit_df = pd.DataFrame(yearly_profit.items(), columns=['year', 'profit'])
yearly_profit_df.to_excel('results/profit1.1.xlsx', index=False)

print("数据已保存")
```

附录 E：问题一第二问实现代码

```
import pandas as pd
from scipy.optimize import linprog
import random

planting_data = pd.read_csv('data/processed_planting_data.csv') # 2023 年的种植数据


yearly_data = {
    2023: planting_data
}

yearly_profit = {}

# 读取之前处理好的数据
crop_data = pd.read_csv('data/processed_crop_data.csv')
land_data = pd.read_csv('data/processed_land_data.csv')

# 豆类作物
```

```
legume_crops = ['黄豆', '黑豆', '红豆', '绿豆', '爬豆', '豇豆', '刀豆', '芸豆']

for current_year in range(2024, 2031):
    # 读取数据
    planting_data = yearly_data[current_year - 1] # 使用前一年的种植数据作为基础迭代

    # 添加 land_type 到 planting_data
    planting_data['land_type'] = planting_data['land_id'].apply(lambda x: land_data[land_data['land_id'] ==
x]['land_type'].values[0])

    # 合并 crop_data 和 planting_data
    merged_data = pd.merge(crop_data, planting_data, on=['crop_name', 'season', 'land_type'])

    print(f"开始计算{current_year}年的最优解")

    # 计算市场需求量（亩产量 * 去年种植面积）
    merged_data['market_demand'] = merged_data['yield'] * merged_data['crop_area']

    # 计算每种作物在每个季节的总市场需求量
    market_demand = merged_data.groupby(['crop_name', 'season'])['market_demand'].sum().reset_index()
    # 单季的作物改为第一季
    market_demand.loc[market_demand['season'] == '单季', 'season'] = '第一季'
    crop_data.loc[crop_data['season'] == '单季', 'season'] = '第一季'
    #print(market_demand)

    # 提取参数
    crops = crop_data['crop_name'].unique()
    lands = land_data['land_id'].unique()
    seasons = ["第一季", "第二季"]
    seasons = pd.Series(seasons)

    # 定义变量数量
    num_crops = len(crops)
    num_lands = len(lands)
    num_seasons = len(seasons)

    # 定义目标函数系数
    c = []
    c_discounted = []
    for land in lands:
        for crop in crops:
            for season in seasons:
                crop_info = crop_data[(crop_data['crop_name'] == crop) & (crop_data['season'] == season) &
(crop_data['land_type'] == land_data[land_data['land_id'] == land]['land_type'].values[0])]
```

```
                if not crop_info.empty:
                    # price = crop_info['price_min'].values[0]
                    # price 使用最小值和最大值之间的随机数
                    price = random.uniform(crop_info['price_min'].values[0], crop_info['price_max'].values[0])
                    yield_per_acre = crop_info['yield'].values[0]
                    cost = crop_info['cost'].values[0]
                    c.append(-(price * yield_per_acre - cost))  # 因为 linprog 求最小值，所以这里加负号
                    c_discounted.append(-(0.5 * price * yield_per_acre - cost))  # 50%降价
                else:
                    c.append(0)
                    c_discounted.append(0)
    # Combine original and discounted coefficients
    c.extend(c_discounted)

    # A b 存储约束条件
    A = []
    b = []

    # 地块面积限制
    for land in lands:
        for season in seasons:
            row = [0] * (2 * num_crops * num_lands * num_seasons)
            for i, crop in enumerate(crops):
                index = lands.tolist().index(land) * num_crops * num_seasons + i * num_seasons +
seasons.tolist().index(season)
                row[index] = 1
                row[index + num_crops * num_lands * num_seasons] = 1  # 超售部分
            A.append(row)
            b.append(land_data[land_data['land_id'] == land]['land_area'].values[0])

    # 市场需求限制
    for crop in crops:
        for season in seasons:
            row = [0] * (2 * num_crops * num_lands * num_seasons)
            for land in lands:
                index = lands.tolist().index(land) * num_crops * num_seasons + crops.tolist().index(crop) *
num_seasons + seasons.tolist().index(season)
                crop_info = crop_data[(crop_data['crop_name'] == crop) & (crop_data['season'] == season)]
                if not crop_info.empty:
                    yield_per_acre = crop_info['yield'].values[0]
                    row[index] = yield_per_acre
                    row[index + num_crops * num_lands * num_seasons] = yield_per_acre  # 超售部分
            A.append(row)
            demand_info = market_demand[(market_demand['crop_name'] == crop) & (market_demand['season']
== season)]
```

```
                if not demand_info.empty:
                    b.append(demand_info['market_demand'].values[0])
                else:
                    b.append(0)

        # 豆类轮作要求
        if current_year >= 2025:
            previous_years = [current_year - 1, current_year - 2]
            previous_data = pd.concat([yearly_data[y] for y in previous_years if y in yearly_data])

            for land in lands:
                legume_planted = previous_data[(previous_data['land_id'] == land) &
(previous_data['crop_name'].isin(legume_crops))]
                if legume_planted.empty:
                    for crop in legume_crops:
                        row = [0] * (2 * len(crops) * len(lands) * len(seasons))
                        for season in seasons:
                            index = list(lands).index(land) * len(crops) * len(seasons) + list(crops).index(crop) *
len(seasons) + list(seasons).index(season)
                            row[index] = -1
                            row[index + num_crops * num_lands * num_seasons] = -1   # 超售部分
                        A.append(row)
                        b.append(0)

        # 求解线性规划问题
        res = linprog(c, A_ub=A, b_ub=b, method='highs')

        # 输出结果
        if res.success:
            print(f"{current_year}年最优解已找到")
            print(f"最优解：{-res.fun}")
            yearly_profit[current_year] = -res.fun

            rows = []
            for i, land in enumerate(lands):
                for j, crop in enumerate(crops):
                    for k, season in enumerate(seasons):
                        index = i * num_crops * num_seasons + j * num_seasons + k
                        if res.x[index] > 0:
                            rows.append({
                                'land_id': land,
                                'crop_name': crop,
                                'crop_area': res.x[index],
                                'season': season
                            })
```

```
                    if res.x[index + num_crops * num_lands * num_seasons] > 0:
                        rows.append({
                            'land_id': land,
                            'crop_name': crop,
                            'crop_area': res.x[index + num_crops * num_lands * num_seasons],
                            'season': season,
                            'discounted': True
                        })

            current_year_data = pd.DataFrame(rows)
            print(f"{current_year}年的种植数据：")
            print(current_year_data.head())

            yearly_data[current_year] = current_year_data
        else:
            print(f"{current_year}年没有找到可行解")
            break

# 保存到 excel
template_df = pd.read_excel("code/results_template.xlsx")
# 0 - 53 行 为第一季
land_id_first = template_df["地块名"][0:54]
land_id_first = land_id_first + "_第一季"
# 54 - 81 行 为第二季
land_id_second = template_df["地块名"][54:82]
land_id_second = land_id_second + "_第二季"

template_cols = template_df.columns[2:]
with pd.ExcelWriter('results/1.2.xlsx') as writer:
    for year, data in yearly_data.items():
        if (year == 2023):
            continue

        df = pd.DataFrame(index=range(0, len(land_id_first) + len(land_id_second)), columns=template_cols)
        df["地块名"] = pd.concat([land_id_first, land_id_second], ignore_index=True)
        for i, row in data.iterrows():
            land_id = row['land_id']
            crop_name = row['crop_name']
            crop_area = row['crop_area']
            season = row['season']
            land_name_with_season = land_id + "_" + season
            index = df[df["地块名"] == land_name_with_season].index[0]
            df.at[index, crop_name] = crop_area
```

```
        # 空缺值填充为 0
        df = df.fillna(0)
        df.to_excel(writer, sheet_name=str(year), index=False)
        print(f"已保存{year}年的结果")


# 保存年度利润
yearly_profit_df = pd.DataFrame(yearly_profit.items(), columns=['year', 'profit'])
yearly_profit_df.to_excel('results/profit1.2.xlsx', index=False)

print("数据已保存")
```

附录 F：问题二实现代码

```
import pandas as pd
from scipy.optimize import linprog
import random

# 读取数据
planting_data = pd.read_csv('data/processed_planting_data.csv')  # 2023 年的种植数据
crop_data = pd.read_csv('data/processed_crop_data.csv')
land_data = pd.read_csv('data/processed_land_data.csv')

# 初始化数据
yearly_data = {2023: planting_data}
yearly_profit = {}

# 豆类作物
legume_crops = ['黄豆', '黑豆', '红豆', '绿豆', '爬豆', '豇豆', '刀豆', '芸豆']

# 定义作物类别
grain_crops = [
    "黄豆",
    "黑豆",
    "红豆",
    "绿豆",
    "爬豆",
    "小麦",
    "玉米",
    "谷子",
    "高粱",
    "黍子",
    "荞麦",
    "南瓜",
    "红薯",
```

```
    "莜麦",
    "大麦",
    "水稻"
]

vegetable_crops = [
    "豇豆",
    "刀豆",
    "芸豆",
    "土豆",
    "西红柿",
    "茄子",
    "菠菜",
    "青椒",
    "菜花",
    "包菜",
    "油麦菜",
    "小青菜",
    "黄瓜",
    "生菜",
    "辣椒",
    "空心菜",
    "黄心菜",
    "芹菜",
    "大白菜",
    "白萝卜",
    "红萝卜"
]

mushroom_crops = [
    "榆黄菇",
    "香菇",
    "白灵菇",
    "羊肚菌"
]

for current_year in range(2024, 2031):
    # 读取数据
    planting_data = yearly_data[current_year - 1]  # 使用前一年的种植数据作为基础迭代

    # 添加 land_type 到 planting_data
    planting_data['land_type'] = planting_data['land_id'].apply(lambda x: land_data[land_data['land_id'] ==
x]['land_type'].values[0])
```

```
    # 合并 crop_data 和 planting_data
    merged_data = pd.merge(crop_data, planting_data, on=['crop_name', 'season', 'land_type'])

    print(f"开始计算{current_year}年的最优解")

    # 计算市场需求量（亩产量 * 去年种植面积）
    merged_data['market_demand'] = merged_data['yield'] * merged_data['crop_area']

    # 计算每种作物在每个季节的总市场需求量
    market_demand = merged_data.groupby(['crop_name', 'season'])['market_demand'].sum().reset_index()
    # 单季的作物改为第一季
    market_demand.loc[market_demand['season'] == '单季', 'season'] = '第一季'
    crop_data.loc[crop_data['season'] == '单季', 'season'] = '第一季'

    # 提取参数
    crops = crop_data['crop_name'].unique()
    lands = land_data['land_id'].unique()
    seasons = ["第一季", "第二季"]
    seasons = pd.Series(seasons)

    # 定义变量数量
    num_crops = len(crops)
    num_lands = len(lands)
    num_seasons = len(seasons)

    # 定义目标函数系数
    c = []
    c_discounted = []
    for land in lands:
        for crop in crops:
            for season in seasons:
                crop_info = crop_data[(crop_data['crop_name'] == crop) & (crop_data['season'] == season) &
(crop_data['land_type'] == land_data[land_data['land_id'] == land]['land_type'].values[0])]

                if not crop_info.empty:
                    # 处理价格变化
                    if crop in grain_crops:
                        price = crop_info['price_min'].values[0]  # 粮食类作物价格稳定
                    elif crop in vegetable_crops:
                        price = crop_info['price_min'].values[0] * (1 + 0.05 * (current_year - 2023))  # 蔬菜类
作物价格每年增长 5%
                    elif crop in mushroom_crops:
                        if crop == '羊肚菌':
                            price = crop_info['price_min'].values[0] * (1 - 0.05 * (current_year - 2023))  # 羊
```

```
肚菌价格每年下降 5%
                        else:
                            price = crop_info['price_min'].values[0] * (1 - random.uniform(0.01, 0.05) *
(current_year - 2023))  # 其他食用菌价格每年下降 1%~5%
                        else:
                            price = random.uniform(crop_info['price_min'].values[0],
crop_info['price_max'].values[0])

                        # 处理亩产量变化
                        yield_per_acre = crop_info['yield'].values[0] * (1 + random.uniform(-0.1, 0.1))

                        # 处理种植成本变化
                        cost = crop_info['cost'].values[0] * (1 + 0.05 * (current_year - 2023))

                        c.append(-(price * yield_per_acre - cost))  # 因为 linprog 求最小值，所以这里加负号
                        c_discounted.append(-(0.5 * price * yield_per_acre - cost))  # 50%降价
                    else:
                        c.append(0)
                        c_discounted.append(0)
    # 降价后的目标函数系数
    c.extend(c_discounted)

    # A b 存储约束条件
    A = []
    b = []

    # 地块面积限制
    for land in lands:
        for season in seasons:
            row = [0] * (2 * num_crops * num_lands * num_seasons)
            for i, crop in enumerate(crops):
                index = lands.tolist().index(land) * num_crops * num_seasons + i * num_seasons +
seasons.tolist().index(season)
                row[index] = 1
                row[index + num_crops * num_lands * num_seasons] = 1  # 超售部分
            A.append(row)
            b.append(land_data[land_data['land_id'] == land]['land_area'].values[0])

    # 市场需求限制
    for crop in crops:
        for season in seasons:
            row = [0] * (2 * num_crops * num_lands * num_seasons)
            for land in lands:
                index = lands.tolist().index(land) * num_crops * num_seasons + crops.tolist().index(crop) *
num_seasons + seasons.tolist().index(season)
```

```
                crop_info = crop_data[(crop_data['crop_name'] == crop) & (crop_data['season'] == season)]
                if not crop_info.empty:
                    yield_per_acre = crop_info['yield'].values[0]
                    row[index] = yield_per_acre
                    row[index + num_crops * num_lands * num_seasons] = yield_per_acre  # 超售部分
            A.append(row)
            demand_info = market_demand[(market_demand['crop_name'] == crop) & (market_demand['season'] == season)]
            if not demand_info.empty:
                if crop in ["小麦", "玉米"]:
                    b.append(demand_info['market_demand'].values[0] * (1 + random.uniform(0.05, 0.1)))  # 小麦和玉米市场需求增长 5%~10%
                else:
                    b.append(demand_info['market_demand'].values[0] * (1 + random.uniform(-0.05, 0.05)))  # 其他作物市场需求变化±5%
            else:
                b.append(0)

    # 豆类轮作要求
    if current_year >= 2025:
        previous_years = [current_year - 1, current_year - 2]
        previous_data = pd.concat([yearly_data[y] for y in previous_years if y in yearly_data])

        for land in lands:
            legume_planted = previous_data[(previous_data['land_id'] == land) & (previous_data['crop_name'].isin(legume_crops))]
            if legume_planted.empty:
                for crop in legume_crops:
                    row = [0] * (2 * len(crops) * len(lands) * len(seasons))
                    for season in seasons:
                        index = list(lands).index(land) * len(crops) * len(seasons) + list(crops).index(crop) * len(seasons) + list(seasons).index(season)
                        row[index] = -1
                        row[index + num_crops * num_lands * num_seasons] = -1  # 超售部分
                    A.append(row)
                    b.append(0)

    # 求解线性规划问题
    res = linprog(c, A_ub=A, b_ub=b, method='highs')

    # 输出结果
    if res.success:
        print(f"{current_year}年最优解已找到")
        print(f"最优解： { -res.fun}")
        yearly_profit[current_year] = -res.fun
```

```
            rows = []
            for i, land in enumerate(lands):
                for j, crop in enumerate(crops):
                    for k, season in enumerate(seasons):
                        index = i * num_crops * num_seasons + j * num_seasons + k
                        if res.x[index] > 0:
                            rows.append({
                                'land_id': land,
                                'crop_name': crop,
                                'crop_area': res.x[index],
                                'season': season
                            })
                        if res.x[index + num_crops * num_lands * num_seasons] > 0:
                            rows.append({
                                'land_id': land,
                                'crop_name': crop,
                                'crop_area': res.x[index + num_crops * num_lands * num_seasons],
                                'season': season,
                                'discounted': True
                            })

            current_year_data = pd.DataFrame(rows)
            print(f"{current_year}年的种植数据：")
            print(current_year_data.head())

            yearly_data[current_year] = current_year_data
        else:
            print(f"{current_year}年没有找到可行解")
            break

# 保存到 excel
template_df = pd.read_excel("code/results_template.xlsx")
# 0 - 53 行 为第一季
land_id_first = template_df["地块名"][0:54]
land_id_first = land_id_first + "_第一季"
# 54 - 81 行 为第二季
land_id_second = template_df["地块名"][54:82]
land_id_second = land_id_second + "_第二季"

template_cols = template_df.columns[2:]
with pd.ExcelWriter('results/result2.xlsx') as writer:
    for year, data in yearly_data.items():
        if (year == 2023):
            continue
```

```
            df = pd.DataFrame(index=range(0, len(land_id_first) + len(land_id_second)), columns=template_cols)
            df["地块名"] = pd.concat([land_id_first, land_id_second], ignore_index=True)
            for i, row in data.iterrows():
                land_id = row['land_id']
                crop_name = row['crop_name']
                crop_area = row['crop_area']
                season = row['season']
                land_name_with_season = land_id + "_" + season
                index = df[df["地块名"] == land_name_with_season].index[0]
                df.at[index, crop_name] = crop_area

            # 空缺值填充为 0
            df = df.fillna(0)
            df.to_excel(writer, sheet_name=str(year), index=False)
            print(f"已保存{year}年的结果")

# 保存年度利润
yearly_profit_df = pd.DataFrame(yearly_profit.items(), columns=['year', 'profit'])
yearly_profit_df.to_excel('results/profit2.xlsx', index=False)

print("数据已保存")
```

附录 G：问题三实现代码

```
import pandas as pd
from scipy.optimize import linprog
import random

# 读取数据
planting_data = pd.read_csv('data/processed_planting_data.csv')  # 2023 年的种植数据
crop_data = pd.read_csv('data/processed_crop_data.csv')
land_data = pd.read_csv('data/processed_land_data.csv')

# 初始化数据
yearly_data = {2023: planting_data}
yearly_profit = {}

# 豆类作物
legume_crops = ['黄豆', '黑豆', '红豆', '绿豆', '爬豆', '豇豆', '刀豆', '芸豆']

# 定义作物类别
grain_crops = [
    "黄豆",
    "黑豆",
```

```
    "红豆",
    "绿豆",
    "爬豆",
    "小麦",
    "玉米",
    "谷子",
    "高粱",
    "黍子",
    "荞麦",
    "南瓜",
    "红薯",
    "莜麦",
    "大麦",
    "水稻"
]

vegetable_crops = [
    "豇豆",
    "刀豆",
    "芸豆",
    "土豆",
    "西红柿",
    "茄子",
    "菠菜",
    "青椒",
    "菜花",
    "包菜",
    "油麦菜",
    "小青菜",
    "黄瓜",
    "生菜",
    "辣椒",
    "空心菜",
    "黄心菜",
    "芹菜",
    "大白菜",
    "白萝卜",
    "红萝卜"
]

mushroom_crops = [
    "榆黄菇",
    "香菇",
    "白灵菇",
```

```
        "羊肚菌"
]

# 可替代性作物组
substitutable_crops = [
        ["黄豆", "黑豆"],
        ["红豆", "绿豆"]
]

# 互补性作物组
complementary_crops = [
        ("玉米", "小麦"),
        ("土豆", "白萝卜"),
        ("西红柿", "茄子")
]

for current_year in range(2024, 2031):
        # 读取数据
        planting_data = yearly_data[current_year - 1]  # 使用前一年的种植数据作为基础迭代

        # 添加 land_type 到 planting_data
        planting_data['land_type'] = planting_data['land_id'].apply(lambda x: land_data[land_data['land_id'] ==
x]['land_type'].values[0])

        # 合并 crop_data 和 planting_data
        merged_data = pd.merge(crop_data, planting_data, on=['crop_name', 'season', 'land_type'])

        print(f"开始计算{current_year}年的最优解")

        # 计算市场需求量（亩产量 * 去年种植面积）
        merged_data['market_demand'] = merged_data['yield'] * merged_data['crop_area']

        # 计算每种作物在每个季节的总市场需求量
        market_demand = merged_data.groupby(['crop_name', 'season'])['market_demand'].sum().reset_index()
        # 单季的作物改为第一季
        market_demand.loc[market_demand['season'] == '单季', 'season'] = '第一季'
        crop_data.loc[crop_data['season'] == '单季', 'season'] = '第一季'

        # 提取参数
        crops = crop_data['crop_name'].unique()
        lands = land_data['land_id'].unique()
        seasons = ["第一季", "第二季"]
        seasons = pd.Series(seasons)
```

```
    # 定义变量数量
    num_crops = len(crops)
    num_lands = len(lands)
    num_seasons = len(seasons)

    # 定义目标函数系数
    c = []
    c_discounted = []
    for land in lands:
        for crop in crops:
            for season in seasons:
                crop_info = crop_data[(crop_data['crop_name'] == crop) & (crop_data['season'] == season) &
(crop_data['land_type'] == land_data[land_data['land_id'] == land]['land_type'].values[0])]

                if not crop_info.empty:
                    # 处理价格变化
                    if crop in grain_crops:
                        price = crop_info['price_min'].values[0]  # 粮食类作物价格稳定
                    elif crop in vegetable_crops:
                        price = crop_info['price_min'].values[0] * (1 + 0.05 * (current_year - 2023))  # 蔬菜类
作物价格每年增长 5%
                    elif crop in mushroom_crops:
                        if crop == '羊肚菌':
                            price = crop_info['price_min'].values[0] * (1 - 0.05 * (current_year - 2023))  # 羊
肚菌价格每年下降 5%
                        else:
                            price = crop_info['price_min'].values[0] * (1 - random.uniform(0.01, 0.05) *
(current_year - 2023))  # 其他食用菌价格每年下降 1%~5%
                    else:
                        price = random.uniform(crop_info['price_min'].values[0],
crop_info['price_max'].values[0])

                    # 处理亩产量变化
                    yield_per_acre = crop_info['yield'].values[0] * (1 + random.uniform(-0.1, 0.1))

                    # 处理种植成本变化
                    cost = crop_info['cost'].values[0] * (1 + 0.05 * (current_year - 2023))

                    c.append(-(price * yield_per_acre - cost))  # 因为 linprog 求最小值，所以这里加负号
                    c_discounted.append(-(0.5 * price * yield_per_acre - cost))  # 50%降价
                else:
                    c.append(0)
                    c_discounted.append(0)
    # 降价后的目标函数系数
    c.extend(c_discounted)
```

```
    # A b 存储约束条件
    A = []
    b = []

    # 地块面积限制
    for land in lands:
        for season in seasons:
            row = [0] * (2 * num_crops * num_lands * num_seasons)
            for i, crop in enumerate(crops):
                index = lands.tolist().index(land) * num_crops * num_seasons + i * num_seasons +
seasons.tolist().index(season)
                row[index] = 1
                row[index + num_crops * num_lands * num_seasons] = 1  # 超售部分
            A.append(row)
            b.append(land_data[land_data['land_id'] == land]['land_area'].values[0])

    # 市场需求限制
    for crop in crops:
        for season in seasons:
            row = [0] * (2 * num_crops * num_lands * num_seasons)
            for land in lands:
                index = lands.tolist().index(land) * num_crops * num_seasons + crops.tolist().index(crop) *
num_seasons + seasons.tolist().index(season)
                crop_info = crop_data[(crop_data['crop_name'] == crop) & (crop_data['season'] == season)]
                if not crop_info.empty:
                    yield_per_acre = crop_info['yield'].values[0]
                    row[index] = yield_per_acre
                    row[index + num_crops * num_lands * num_seasons] = yield_per_acre  # 超售部分
            A.append(row)
            demand_info = market_demand[(market_demand['crop_name'] == crop) & (market_demand['season']
== season)]
            if not demand_info.empty:
                b.append(demand_info['market_demand'].values[0] * (1 + random.uniform(-0.05, 0.05)))  # 其
他作物市场需求变化±5%
            else:
                b.append(0)

    # 豆类轮作要求
    if current_year >= 2025:
        previous_years = [current_year - 1, current_year - 2]
        previous_data = pd.concat([yearly_data[y] for y in previous_years if y in yearly_data])

        for land in lands:
            legume_planted = previous_data[(previous_data['land_id'] == land) &
```

```
(previous_data['crop_name'].isin(legume_crops))]
                if legume_planted.empty:
                    for crop in legume_crops:
                        row = [0] * (2 * len(crops) * len(lands) * len(seasons))
                        for season in seasons:
                            index = list(lands).index(land) * len(crops) * len(seasons) + list(crops).index(crop) *
len(seasons) + list(seasons).index(season)
                            row[index] = -1
                            row[index + num_crops * num_lands * num_seasons] = -1  # 超售部分
                        A.append(row)
                        b.append(0)

    # 可替代性约束
    for group in substitutable_crops:
        for season in seasons:
            row = [0] * (2 * num_crops * num_lands * num_seasons)
            for crop in group:
                for land in lands:
                    index = lands.tolist().index(land) * num_crops * num_seasons + crops.tolist().index(crop) *
num_seasons + seasons.tolist().index(season)
                    row[index] = 1
                    row[index + num_crops * num_lands * num_seasons] = 1  # 超售部分
            A.append(row)
            b.append(land_data['land_area'].sum() * 0.5)  # 限制可替代性作物的总种植面积不超过总面积的
50%

    # 互补性约束
    for crop1, crop2 in complementary_crops:
        for season in seasons:
            row1 = [0] * (2 * num_crops * num_lands * num_seasons)
            row2 = [0] * (2 * num_crops * num_lands * num_seasons)
            for land in lands:
                index1 = lands.tolist().index(land) * num_crops * num_seasons + crops.tolist().index(crop1) *
num_seasons + seasons.tolist().index(season)
                index2 = lands.tolist().index(land) * num_crops * num_seasons + crops.tolist().index(crop2) *
num_seasons + seasons.tolist().index(season)
                row1[index1] = -1
                row1[index1 + num_crops * num_lands * num_seasons] = 1  # 超售部分
                row2[index2] = -1
                row2[index2 + num_crops * num_lands * num_seasons] = -1  # 超售部分
            A.append(row1)
            b.append(-land_data['land_area'].sum() * 0.3)  # 限制互补性作物的种植面积比例
            A.append(row2)
            b.append(-land_data['land_area'].sum() * 0.3)
```

```
    # 求解线性规划问题
    res = linprog(c, A_ub=A, b_ub=b, method='highs')

    # 输出结果
    if res.success:
        print(f"{current_year}年最优解已找到")
        print(f"最优解： { -res.fun}")
        yearly_profit[current_year] = -res.fun

        rows = []
        for i, land in enumerate(lands):
            for j, crop in enumerate(crops):
                for k, season in enumerate(seasons):
                    index = i * num_crops * num_seasons + j * num_seasons + k
                    if res.x[index] > 0:
                        rows.append({
                            'land_id': land,
                            'crop_name': crop,
                            'crop_area': res.x[index],
                            'season': season
                        })
                    if res.x[index + num_crops * num_lands * num_seasons] > 0:
                        rows.append({
                            'land_id': land,
                            'crop_name': crop,
                            'crop_area': res.x[index + num_crops * num_lands * num_seasons],
                            'season': season,
                            'discounted': True
                        })

        current_year_data = pd.DataFrame(rows)
        print(f"{current_year}年的种植数据：")
        print(current_year_data.head())

        yearly_data[current_year] = current_year_data
    else:
        print(f"{current_year}年没有找到可行解")
        break

# 保存到 excel
template_df = pd.read_excel("code/results_template.xlsx")
# 0 - 53 行 为第一季
land_id_first = template_df["地块名"][0:54]
land_id_first = land_id_first + "_第一季"
```

```
# 54 - 81 行 为第二季
land_id_second = template_df["地块名"][54:82]
land_id_second = land_id_second + "_第二季"

template_cols = template_df.columns[2:]
with pd.ExcelWriter('results/result3.xlsx') as writer:
    for year, data in yearly_data.items():
        if (year == 2023):
            continue

        df = pd.DataFrame(index=range(0, len(land_id_first) + len(land_id_second)), columns=template_cols)
        df["地块名"] = pd.concat([land_id_first, land_id_second], ignore_index=True)
        for i, row in data.iterrows():
            land_id = row['land_id']
            crop_name = row['crop_name']
            crop_area = row['crop_area']
            season = row['season']
            land_name_with_season = land_id + "_" + season
            index = df[df["地块名"] == land_name_with_season].index[0]
            df.at[index, crop_name] = crop_area

        # 空缺值填充为 0
        df = df.fillna(0)
        df.to_excel(writer, sheet_name=str(year), index=False)
        print(f"已保存{year}年的结果")

# 保存年度利润
yearly_profit_df = pd.DataFrame(yearly_profit.items(), columns=['year', 'profit'])
yearly_profit_df.to_excel('results/profit3.xlsx', index=False)

print("数据已保存")
```