基于线性规划模型的最优种植规划方案问题研究

摘要

本文针对农作物的种植策略问题，通过建立线性规划模型，实现对农作物最优种植规划方案的求解。该模型不仅考虑了农作物种植过程中的基本要素，如亩产量、种植成本、销售价格等核心参数，还结合了农作物未来市场销售的预期变化，综合分析了可能影响种植决策的多重因素。

针对问题一，考虑到各种农作物未来的预期销售量、种植成本、亩产量和销售价格的十分稳定，没有相应的不确定因素，销售情况以及作物滞销对最优规划方案产生的影响。建立整数线性规划模型，实现制定应对农作物本身浮动情况的合理最优种植方案。

针对问题二，考虑到各种农作物的种植和销售情况会根据实际的市场或是气候等外界因素的影响发生一定的波动。建立整数线性规划模型，利用随机数浮动，蒙特卡洛方法，求解出可以完善应对农作物由于外界因素导致的影响的种植方案，使得最优种植方案更加的稳定和灵活。

针对问题三，考虑到在现实生活中，各种农作物之间可能存在一定的互替性和互补性，不同农作物的市场销售也存在相互依存的情况。建立整数线性规划模型，利用随机数浮动，蒙特卡洛方法，处理时间序列数据进行预测。求解出能够更加灵敏的顺应市场变化的规划方案，在没有重大自然灾害或是极端变化影响的情况下，这是一个能够提供直观易懂的决策信息的最优种植方案。

总之，本片文章主要在建立根据不同外界因素，制定出适合当前地区经济效益最高的种植方案，通过三个问题的提出，层层递进的方式优化充分考虑到了多种约束条件，注重可持续发展与最优经济效益之间的平衡，对不同地区都有有较强的适应性和灵活性，确保在不确定的环境中仍能提升经济效益，最大化了作物间的协同效应，提高长期收益的稳定性。

关键词：农作物优化规划，不确定性应对，经济效益最大化，最优种植方案，市场销量预测

目录

一、 问题重述

1.1 问题背景：

在乡村经济可持续发展的大背景下，不合理的耕地资源规划常常带来经济上的损失以及浪费，在有多种耕地资源的地区，能够适宜多种作物的生长，导致种植方案规划难度大大提升，这为农业劳动者带来很大困难。某华北山区农村共有 4 种、1201 亩，共计 34 块耕地，耕地种类、大小、温度都会对收成造成影响，同一耕地要对作物进行轮作且三年内至少种植一次豆类作物，如何减少种植风险带来的影响，带来更高的收益成为种植策略中关键的问题。

1.2 问题重述：

问题 1： 对耕地种植方案进行优化规划，寻找种植作物利润最大化的方案，减少滞销作物，侧重点是对于将合理计划各个耕地的种植情况，使得收益最大化，并满足所有约束条件。

问题 2 ：综合评估种植过程中的各种风险，减少这些因素带来的影响，规划出使收益最大化的种植方案。

问题 3 ：综合考虑作物的可替代性与互补性，评估预期销售量与实际成本、利润等对方案的影响，与问题 2 结果进行比较 。

二、 问题分析

2.1 问题一的分析

在作物种植规划中，可以通过数学模型来寻找作物种植的最优方案，我们考虑了耕作中会出现的影响因素以及作物滞销产生的影响，对问题进行求解。

通过对题目进行分析，我们在求解过程中需要对模型加上以下约束条件：

1） 同一种作物不能连续种植在同一块耕地上
2） 三年内每块耕地上至少种植一次豆类作物
3） 单种作物的种植面积应该至少为耕地种植面积的 10%
4） 作物只能在相应的季节种植在相应的土地上
5） 单个作物的总产量应该在预期销售量的 90%-110%之间
6） 作物在同一季节同一耕地上的种植面积不能超过此土地最大面积
7） 每个耕地在每个季最多种植三种作物

综合以上问题并且考虑到约束条件，可以通过建立综合农业规划模型对问题进行求解。

2.2 问题二的分析

对作物的最佳种植方案进行研究时，各种农作物的销售量亩产量等也是规划最大利益时的重要影响因素，通过随机变量来模拟实际情况的不确定性，减少潜在的种植风险，结合不同的的作物特点以及销售量情况，制定最优方案，增加收益的稳定性。

基于题目，对此问提出以下约束条件：

1） 同一种作物不能连续种植在同一块耕地上
2） 三年内每块耕地上至少种植一次豆类作物
3） 单种作物的种植面积应该至少为耕地种植面积的 10%
4） 作物只能在相应的季节种植在相应的土地上
5） 单个作物的总产量应该在预期销售量的 90%-110%之间
6） 作物在同一季节同一耕地上的种植面积不能超过此土地最大面积
7） 每个耕地在每个季最多种植三种作物

综合数据情况以及约束条件，可以通过建立一个逐级农业规划模型对问题进行方案优化。

2.3 问题三的分析

问题二的条件均处在理想环境中，在问题三中，我们进一步考虑到现实生活中的生物因素以及市场因素，更加全面的对种植方案进行优化，使其更具有可行性

基于问题三，共有以下 10 条约束条件

1） 同一种作物不能连续种植在同一块耕地上
2） 三年内每块耕地上至少种植一次豆类作物
3） 单种作物的种植面积应该至少为耕地种植面积的 10%
4） 作物只能在相应的季节种植在相应的土地上
5） 单个作物的总产量应该在预期销售量的 90%-110%之间
6） 作物在同一季节同一耕地上的种植面积不能超过此土地最大面积
7） 每个耕地在每个季最多种植三种作物
8） 耕地种植两种互补性作物会使产量提高一个比例，需要调整目标函数
9） 耕地种植两种互斥性作物会使产量降低一个比例，需要调整目标函数
10） 生成随机波动，确保销量有一定的波动性

综合数据情况以及约束条件，可以通过建立一个逐级作物生产与市场预测模型对问题进行方案优化。

三、 模型假设

（1） 假设每一年种植后土地营养没有任何流失。
（2） 假设每种耕地的类型，数量，面积永远不会发生变化。
（3） 假设种植的作物的总品类永远不会发生变化。
（4） 假设同一种农作物对土地类型的要求不会跟随季节，天气，温度，湿度变化。
（5） 假设农作物种植过程当中，不会受到突发重大灾害，虫病害对农作物产量造成影响。
（6） 假设种植过程当中，人工成本始终不变。
（7） 假设同样的农作物的产量会有相同的上限，市场的需求也是有限制的。

（8） 假设所有农作物的价格不会因为外界因素，例如人为炒作等发生改变。

四、 数据图形可视化分析

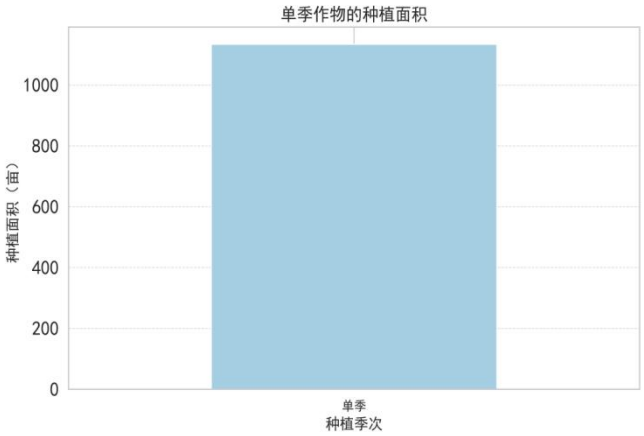

图 1 -单季作物种植面积

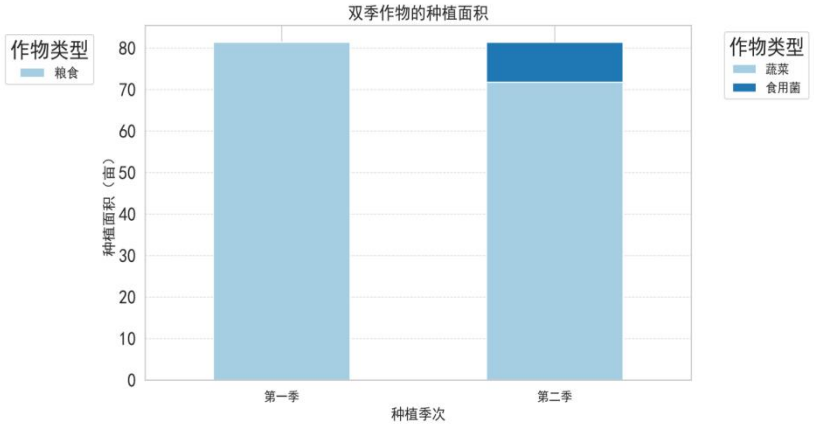

图 2 -双季作物的种植面积

该图表显示总量超过 1100 亩耕地进行单季种植，作物类型主要为粮食作物，说明该地区在单季耕地主要依赖粮食种植获得经济收益。大约有 80 亩耕地进行双季种植，第一季主要种植蔬菜，第二季蔬菜的种植面积减少大约 10 亩，用于种植食用菌。粮食作物集中在单季耕地，季次变化和资源消耗影响其经济效益，而蔬菜因种植周期灵活，适应更多气候条件，成为双季耕地的主要作物。

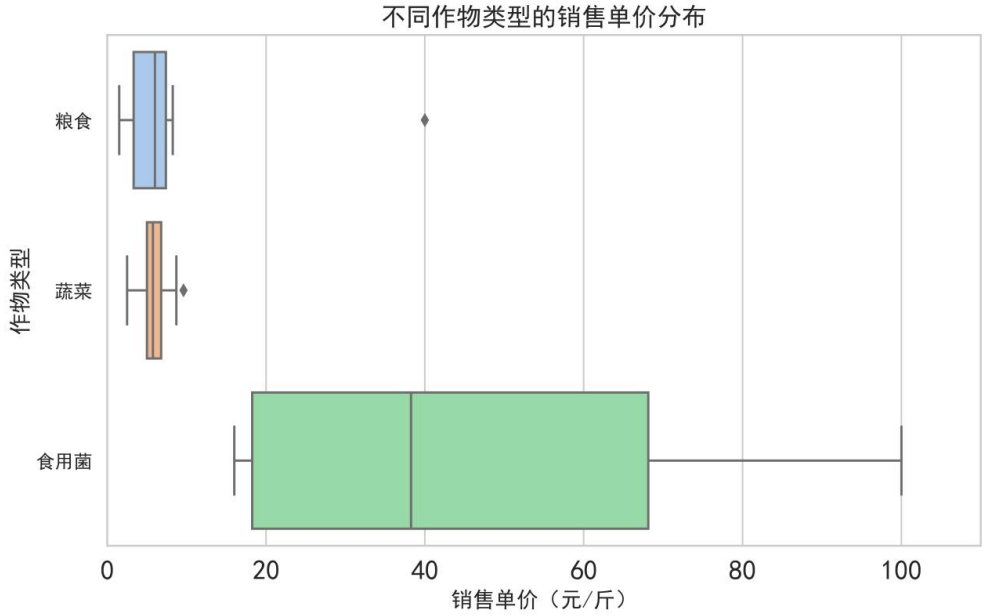

图 3 -不同作物类型的销售单价分布

该图展示了不同作物的价格分布。粮食销售单价主要集中在 2-10 元/斤，波动幅度较小，价格相对稳定。蔬菜的单价范围略小，集中在 3-8 元/斤，有少量异常点，

表明部分蔬菜价格高于平均。食用菌的价格波动最显著，分布在 18-68 元/斤，最高接近 100 元/斤，表明其价格受市场需求和品种差异的影响较大。这说明食用菌价格波动性较强，相较于粮食和蔬菜，稳定性较低。

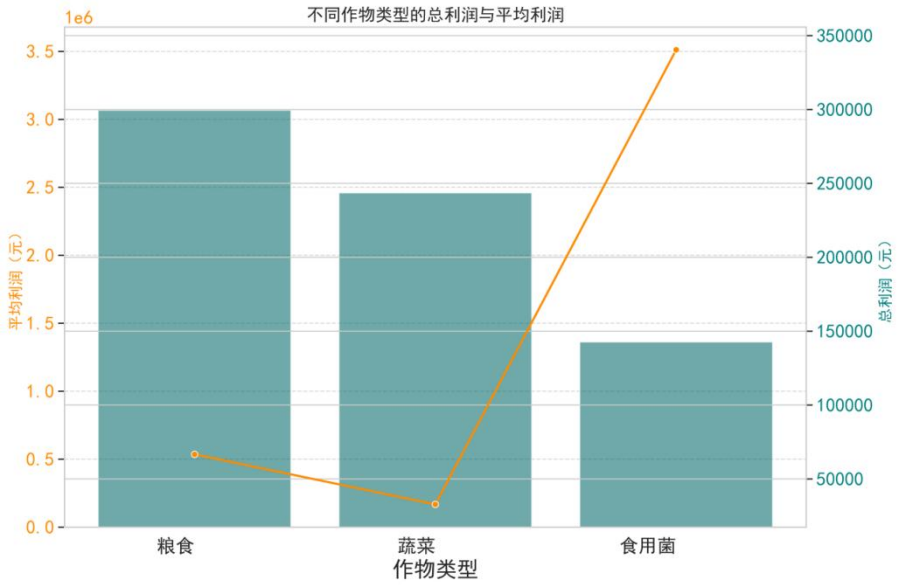

图 4 -不同作物类型的总利润与平均利润

这幅图表展示了各个种类作物总利润和平均利润，通过将折线图和柱状图重叠表现了总利润和平均利润的关系。粮食和蔬菜的平均利润相对来说都较低，但是粮食和蔬菜的总利润达到了 30 万元和 24 万元，说明这两种作物虽然平均利润较低，但是需求量较大。食用菌的平均利润比粮食和蔬菜高出接近 6 倍，但总利润较低，说明市场需求量并不是特别多。

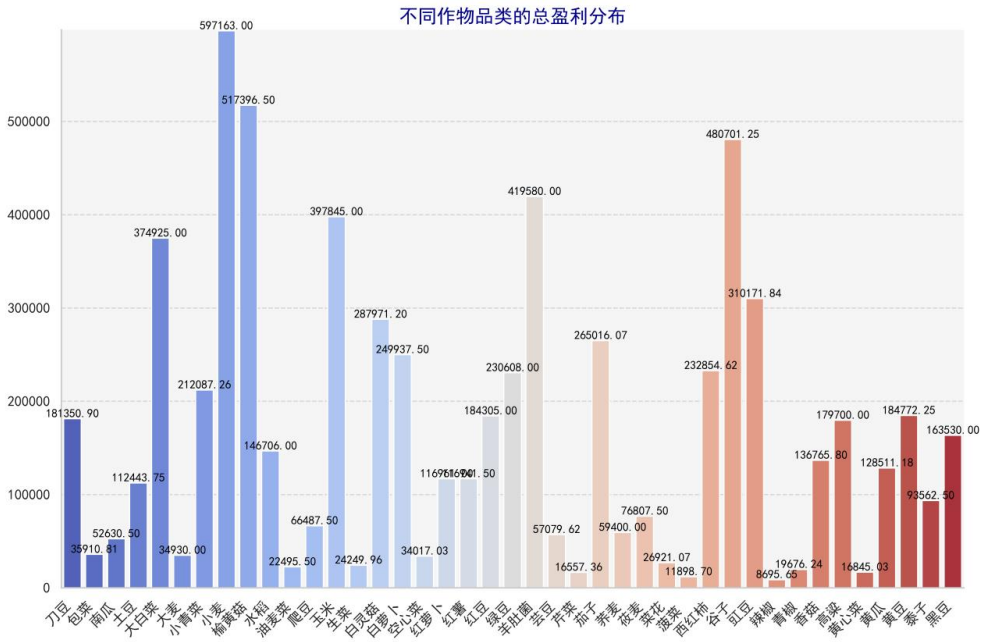

图 5 -不同作物品类的总盈利分布

该图表对 2023 年在此耕地种植的 41 种作物的总盈利进行分析。由图中数据可以得知，2023 年盈利最高的三种作物分别是：小麦，榆黄菇、谷子，总盈利分别为：597163.00 元、517396.50 元、480701.25 元。盈利最低的三种作物分别是：辣椒、菠菜、芹菜，总盈利分别是 8695.65 元、11898.70 元、16557.36 元。

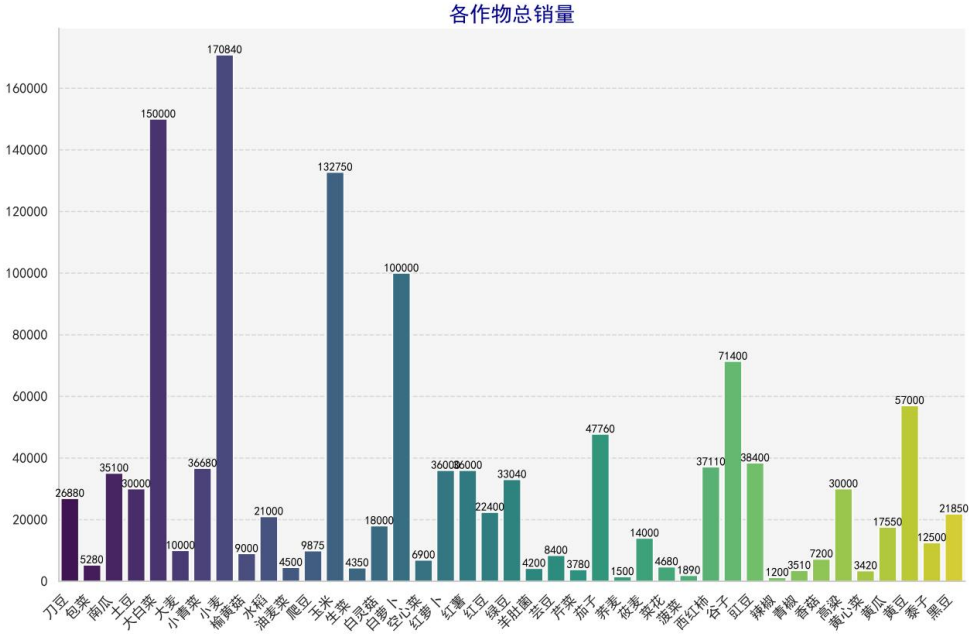

图 6 -各作物总销量

该图表对 2023 年在此耕地种植的 41 种作物的总销量进行分析。由图中数据可以得知，2023 年销量最高的三种作物分别是：小麦，大白菜、玉米，总销量分别为：170840 斤、150000 斤、132750 斤。销量最低的三种作物分别是：辣椒、菠菜、荞麦，总销量分别 1200 斤，1500 斤、1890 斤。

综合上述两张图表，2023 年小麦销量多，盈利高，可以适当增加种植比例，辣椒、菠菜销量少，利润低，可以适当减少种植比例。

五、 符号说明

符号	说明
p_a	第 a 种作物以斤为单位售卖的价格（元）
c_a	第 a 种作物以亩为单位种植的成本（元）
y_a	第 a 种作物以亩为单位的产量（斤）
d_a	第 a 种作物的预计销量（斤）
m_a	第 a 种作物每块地最小种植面积
p_{sa}	第 a 种作物第 s 季的价格
c_{sa}	第 a 种作物第 s 季的亩产量
y_{sa}	第 a 种作物第 s 季的种植成本
α	0.6
μ	0.3
σ	0.1
I	作物种类
A	耕地种类
S	种植季数
Y	种植年份

六、 模型的建立与求解

6.1 问题一模型的建立与求解

6.1.1 综合农业规划模型的建立

为了应对现代农业发展中的复杂性与多样化需求，基于对问题的分析和数据的预处理提出了一种适用于 2024-2030 年农业综合规划的优化模型。该模型的主要目标是在多重约束条件下，通过对资源的合理分配与优化对土地和作物的管理，实现农业总收益的最大化。该模型的设计不仅考虑了各种农业作物的耕作特点和资源的有限性，还考虑到了可持续发展等因素，确保其在经济效益与生态效益之间取得平衡实现经济利益最大化。

定义决策变量为$x_{i,a,s}$, 即每个种植季 s 时在土地a上种植的作物 i 的种植面积。

模型建立的目的是为了分别使 2023-2024 两年利润最大化，即分别求出 2023-2024 年的总收入减去总成本的最大值。

针对第一问，我们分别对两种情况建立不同模型并求解，

情况（1）作物滞销，造成浪费。

此时，目标函数为：

$$max \sum_{i=1}^{I} \sum_{a=1}^{A} \sum_{s=1}^{S} p_1 \min\{y_{ias}, d_i\} - c_i x_{ias}$$

情况（2）超出部分以 50%价格出售，

此时，目标函数为：

$max\sum_{i=1}^{I}\sum_{a=1}^{A}\sum_{s=1}^{S}x_{ias}\min\{y_i x_{ias}, d_i\}+0.5p_i\max\{0, y_i x_{ias}-d_i\}-c_i x_{ias}$ 共有七条约束条件：

1） 每三年内豆类在该耕地上的种植面积不小于该耕地的种植面积

$$\sum_{s\in S}\sum_{y\in\{y_0,y_0+3,\ldots\}}\sum_{i\in 豆类作物}x_{c,f,s}\geq 面积(a)，\forall a\in A$$

2） 同一种作物不能连续种植在同一块耕地上

$$z_{i，\ a,first}+z_{i,a,second}=0, 若(i,a,first)或(i,a,second)在上一年被种植$$

3） 作物在同一季节同一耕地上的种植面积不能超过此土地最大面积

$$\sum_{i\in I}x_{i,a,s}\leq 面积(a),\forall a\in A,\forall s\in S$$

4） 单个作物的总产量应该在预期销售量的 90%-110%之间

$$0.9\times 预期销售(i)\leq\sum_{a\in A}\sum_{s\in S}x_{i,a,s}\times 预期销售(i),\forall i\in I$$

5） 作物只可被种植在相应的耕地上,例：

- 水稻不能在第一季种植在水浇地上（i = 16）：

$$x_{16,a,first}=0$$

- 普通大鹏第二季种植的作物类型仅为食用菌

$$x_{i,a,second}=0,\forall i\notin\{38.39,40,41\}$$

6） 单种作物种植面积应该至少为耕地种植面积的 10%

$$x_{i,a,s}\geq 0.1\times 面积(a)\times z_{i,a,s}$$

$$x_{i,a,s}\leq 面积(a)\times z_{i,a,s}$$

$$\forall i\in I,\forall a\in A,\forall s\in S$$

7） 每个耕地在每个季最多种植三种作物

$$\sum_{i\in I}z_{ias}\leq 3,\forall a\in A,\forall s\in S$$

6.1.2 综合农业规划模型的因素影响

农作物的种植面积：不同耕作物的种植面积是该模型中的主要影响因素，将直接

决定总产量的多与少以及最终的收益。

单季种植耕地和双季种植耕地：同一块耕地一年种植次数将会决定收成次数，从而影响最终的收益效果。

市场与需求约束：农产品的市场需求将会影响农作物价格的稳定性，从而影响耕地对最优种植的更作物品种选择。

6.1.3 综合农业规划模型求解

在本研究中，我们利用 Python 编程语言和 PuLP 库来实现和求解所建立的多维农业规划优化模型。PuLP 是一个强大的线性规划工具，它提供了一个直观的接口来构建和求解优化问题，包括整数规划和混合整数规划问题。我们的模型就属于混合整数线性规划问题的范畴，因为它包含了二进制决策变量(表示是否种植某种作物)和连续变量(如产量和收益)。

首先，我们通过创建一个 LpProblem 对象来初始化优化问题，将其命名为"Crop_Planning"，并设置优化目标为最大化。这个设置反映了我们的核心目标：在给定约束条件下最大化农业收益。

接下来，我们定义了决策变量。在代码中，决策变量被定义为一个三维字典 x, 其键为(I, A, Y, S)，分别代表作物编号、地块编号、年份和季度。同时设置一个二进制变量，即只能取 0 或 1,表示在特定地块的特定时期是否种植某种作物。这种定义方式允许我们灵活地表示复杂的种植决策。

目标函数的构建是通过 PuLP 的 1pSum 函数完成的。这个函数计算了所有决策变量的加权和，其中权重是每种作物在每块地的预期收益。具体来说，收益计算考虑了作物的销售价格、产量、种植成本和土地面积。此外，还添加了 excess 变量来对产量超出预期的部分做出惩罚。

约束条件的添加是模型构建的关键部分。我们添加了多种约束条件，包括土地利用限制、轮作要求、产量控制等。这些约束条件确保了得到的解决方案是切实可行的，符合实际农业生产的各种限制。例如，我们限制了每块地每季只能种植一种作物，确保了平旱地、梯田和山坡地每年只能种一季，水浇地可以种一季水稻或两季蔬菜，普通大棚每年种两季(第一季蔬菜，第二季食用菌),智慧大棚每年种两季蔬菜。此外，我们还添加了豆类作物的轮作要求，确保每三年至少种植一次豆类作物，这反映了可持续农业实践的考虑，以及作物最小种植面积的要求也保证了种植分散度的统一。

在完成模型的构建后，我们通过调用 model.solve()方法对优化问题进行求解。PuLP 会自动选择最适合当前问题的求解器，如 CBC（Coin-or branch and cut）或 GLPK（GNU Linear Programming Kit），以确保高效求解。选择的求解器能够处理不同规模和复杂度的优化问题，因此，具体的求解时间将取决于问题的规模、变量的数量以及约束条件的复杂性。求解过程可能需要一定时间，尤其是在大规模农业规划问题中。

一旦求解完成，我们便能够通过访问每个决策变量的 value 属性，提取出最优的种植方案。这个方案明确指出了在每个时段内每块地应种植的具体作物类型，从而确保在所有约束条件下实现农业收益的最大化。这一最优方案不仅优化了作物分布，还为农户提供了精确的种植策略，以提高土地资源的利用效率和整体生产效益。

6.2 问题二模型的建立与求解

6.2.1 逐级农业规划模型的建立

由于问题二在问题一的基础上增加了不确定因素，为了实现从 2024 年到 2030 年的长期总收益最大化。该模型基于问题一的理想模型，按照年份逐年调整合适的种植

方案，其中每一年的种植方案是基于前一年的优化结果做出的。通过将大的优化环境进一步依次划分为 7 个较小的优化子环境，模型能够在逐级分化的时间和空间范围内提高精准度，使得收益逐年提升，最终经济效益最大化。这样可以更好地应对年度间的市场波动、气候变化等不确定因素，及时更新合适的种植方案。

定义决策变量为$x_{i,a,s}$，即每个种植季 s 时在土地a上种植的作物 i 的种植面积

模型建立的目的是为了分别在减少种植风险的基础上，使 2023-2024 两年利润最大化，即同样需要分别求出 2023-2024 年的总收入减去总成本的最大值

此时，目标函数为：

$$max\sum_{s=1}^{2}\sum_{a=1}^{54}\sum_{i=1}^{18}(P_{sa}Y_{sa}-C_{sa})x_{isa}$$

共有七条约束条件：

1.每三年内豆类在该耕地上的种植面积不小于该耕地的种植面积

$$\sum_{s\in S}\sum_{a\in A}\sum_{i\in 豆类作物}x_{c,f,s}\geq 面积(a)，\ \forall a\in A$$

2.同一种作物不能连续种植在同一块耕地上

$$z_{i，\ a,first}+z_{i,a,second}=0,若(i,a,first)或(i,a,second)在上一年被种植$$

3.作物在同一季节同一耕地上的种植面积不能超过此土地最大面积

$$\sum_{i\in I}x_{i,a,s}\leq 面积(a),\forall a\in A,\forall s\in S$$

4.单个作物的总产量应该在预期销售量的 90%-110%之间

$$0.9\times 预期销售(i)\leq\sum_{a\in A}\sum_{s\in S}x_{i,a,s}\times 预期销售(i),\forall i\in I$$

5.作物只可被种植在相应的耕地上,例：

- 水稻不能在第一季种植在水浇地上（i = 16）：

$$x_{16,a,first}=0$$

- 普通大鹏第二季种植的作物类型仅为食用菌

$$x_{i,a,second}=0,\forall i\notin\{38.39,40,41\}$$

6.单种作物种植面积应该至少为耕地种植面积的 10%

$$x_{i,a,s}\geq 0.1\times 面积(a)\times z_{i,a,s}$$

$$x_{i,a,s}\leq 面积(a)\times z_{i,a,s}$$

$$\forall i\in I,\forall a\in A,\forall s\in S$$

7.每个耕地在每个季最多种植三种作物

$$\sum_{i\in I} z_{ias} \leq 3$$

6.2.2 逐级农业规划模型的因素影响

市场需求的影响：市场需求的多少可能会对销售量产生相应的增加或是减少，从而影响每种作物的收益，最终对第二年的方案制定产生影响。

气候环境的因素影响：收到气候环境温度的影响，农作物的产量也会或多或少的发生改变，从而影响最终的经济效益改变第二年的种植方案。

种植成本的影响：不同的作物在不同的环境因素的影响下所造成的损耗是不一样的，所导致的将会是种植成本的升高或是降低，最终影响后续最优种植方案的制定。

6.2.3 逐级农业规划模型求解

在基于问题一的基础上，本研究进一步引入了更多不稳定因素的考量，特别是针对预期销售量、亩产量、种植成本和销售价格等变量的波动性，为了更好地应对这些随机情景的变化，我们采用了蒙特卡洛模拟方法。通过这种方法，我们将时间维度按年度划分成若干个较小的优化场景。对每个场景下的不同随机情况进行模拟和统计分析，从而得出一个在各种不确定性条件下的最优种植方案。

为了实现这一目标，我们仍然使用了 Python 编程语言和 PuLP 库进行建模和求解。首先，我们通过创建一个名为 "Crop_Planning_year" 的 LpProblem 对象来初始化优化问题，并将其目标设定为最大化农业总收益。与问题一中的模型不同的是，我们在这里修改了连续种植的约束条件，特别是针对重茬种植问题，引入了一个新的变量 `last_year`。该变量确保某块地在某一年所种植的作物不能与前一年种植的作物相同，以防止因连续种植导致的产量下降等问题。

在模型构建完成后，我们使用 `model.solve()` 方法来进行求解。PuLP 会根据问题的特点自动选择最合适的求解器，如 CBC（Coin-or branch and cut）或 GLPK（GNU Linear Programming Kit），以便高效地处理这个优化问题。由于该问题涉及多个不确定性因素和复杂的约束条件，求解过程所需的时间取决于模型的规模、变量数量以及问题的复杂度。

求解完成后，我们可以通过查看决策变量的 `value` 属性来获得最优种植方案。该方案能够明确指出在作物产量和市场波动的情况下，每年在每块土地上应选择种植的作物类型，以在各种不确定条件下实现农业收益的最大化。这不仅增强了模型应对市场变化和环境波动的能力，也为决策者提供了更为灵活和可靠的种植决策依据。

6.3 问题三模型的建立与求解

6.3.1 逐级作物生产与市场预测优化模型的建立

基于问题二的逐级农业规划模型中，我们将时间维度扩展至三年前，通过对前三年的预测销量数据进行分析，构建了一个简单的时间序列数据集。利用该数据集，结合权重调整和随机数浮动的方式，模型可以对当年度的作物销量进行预测。此外，模型还整合了不同作物间的互斥性与互补性，从而在时间和空间维度上协同优化种植策略，达到优化种植的目的。

定义决策变量为$x_{i,a,s}$，即每个种植季 s 时在土地a上种植的作物面积

此时，目标函数为：

$$max\sum_{s=1}^{2}\sum_{a=1}^{54}\sum_{i=1}^{18}(P_{sa}Y_{sa}-C_{sa})x_{isa}$$

共有九条约束条件：

1） 每三年内豆类在该耕地上的种植面积不小于该耕地的种植面积

$$\sum_{s\in S}\sum_{a\in A}\sum_{i\in 豆类作物}x_{c,f,s}\geq 面积(a)，\ \forall a\in A$$

2） 同一种作物不能连续种植在同一块耕地上

$$z_{i，\ a,first}+z_{i,a,second}=0, 若(i,a,first)或(i,a,second)在上一年被种植$$

3） 作物在同一季节同一耕地上的种植面积不能超过此土地最大面积

$$\sum_{i\in I}x_{i,a,s}\leq 面积(a),\forall a\in A,\forall s\in S$$

4） 单个作物的总产量应该在预期销售量的 90%-110%之间

$$0.9\times 预期销售(i)\leq\sum_{a\in A}\sum_{s\in S}x_{i,a,s}\times 预期销售(i),\forall i\in I$$

5） 作物只可被种植在相应的耕地上,例：

- 水稻不能在第一季种植在水浇地上（i = 16）：

$$x_{16,a,first}=0$$

- 普通大鹏第二季种植的作物类型仅为食用菌

$$x_{i,a,second}=0,\forall i\notin\{38.39,40,41\}$$

6） 单种作物种植面积应该至少为耕地种植面积的 10%

$$x_{i,a,s}\geq 0.1\times 面积(a)\times z_{i,a,s}$$

$$x_{i,a,s}\leq 面积(a)\times z_{i,a,s}$$

$$\forall i\in I,\forall a\in A,\forall s\in S$$

7） 每个耕地在每个季最多种植三种作物

$$\sum_{i\in I}z_{ias}\leq 3$$

8） 耕地种植两种互补性作物会使产量提高一个比例，需要调整目标函数

调整项=$\sum_{a,s}\beta_{i1,i2}\,.\,x_{i1,a,s}.\,x_{i2,a,s},\beta_{i1,i2}>1$

9)耕地种植两种互斥性作物会使产量降低一个比例，需要调整目标函数

调整项=$\sum_{a,s}\theta_{i1,i2}\cdot x_{i1,a,s}\cdot x_{i2,a,s}\,,\theta_{i1,i2}<1$

10)根据历史数据生成随机波动，确保销量有一定的波动性

公式：$S_t=\ \alpha\cdot S_{t-1}+\mu\cdot S_{t-2}+\sigma\cdot S_{t-3}$

随机波动=$\eta\sim N$（φ，α）

最小销量约束：预测销量的 90%：

$$S_{min}=\ 0.9\cdot S_t^{noise}$$

最小销量约束：预测销量的 110%：

$$S_{max}=\ 1.1\cdot S_t^{noise}$$

实际生产约束量：$S_{min}\le P_{total}\le S_{max}$

实际生产量公式：$P_{total}=\ \sum_{a,s}x[crop_{id},a,s]\cdot y_{per\_mu}$

预测销量公式：$S_t^{noise}=$（$\alpha\cdot S_{t-1}+\mu\cdot S_{t-2}+\sigma\cdot S_{t-3}$）·（$1+\eta$）

销量约束：$0.9\cdot S_t^{noise}\le P_{total}\le 1.1\cdot S_t^{noise}$

6.3.2 逐级作物生产与市场预测优化模型的因素影响

作物选择的影响：不同的作物之间需要考虑到互斥性和互补性对作物产量的影响，互斥的作物可能会导致其产量降低，导致经济效益降低；相反，互补性可能会导致其产量升高，盈利增加。目标模型考虑的不仅仅是单一作物的收益，还包括作物组合的效益最大化。

前期种植情况的影响：模型通过逐年更新决策变量，根据前期数据和预测结果，动态调整种植方案，以实现最优生产效率。模型利用历史数据、市场波动和天气预报等外界因素综合分析，使决策变量能够灵活应对不确定性。

时间与空间的影响：逐级优化模型在时间维度上通过多阶段预测逐步优化，每一个时间节点都根据前期数据更新决策。空间维度则通过作物轮作、区域规划等方法，优化资源配置和土地使用。从而使得农作物的经济效益最大化。

6.3.3 逐级作物生产与市场预测优化模型求解

对比问题一和问题二，本研究中，模型对于空间维度和时间维度上的复杂度还有不稳定因素进行了更多的考量。在现实生活中，不同作物如果种植位置相邻会对彼此产生一些或好或坏的影响。 这样的互补性和互斥性会对作物的产量造成较大的影响。在模型中，我们首先基于蒙特卡洛模拟的方法，在一定范围中产生随机数作为两种特定作物互相种植的互补或互斥影响（对产量），这里，我们查阅了相关资料和网站，

得到了特定的互补或互斥作物搭档，并得到一个大致的影响程度范围。同时，从时间维度上，我们也对销量进行了预测，从模型的第四年开始，我们会基于前三年的销量对市场进行一个简单的评估，我们给前三年的销售量赋予不同的权重，再用随机数浮动的方式得到本年度的预测销量，该销量考虑到了前三年销量的变化，对于前三年销量的不稳定或是大幅改变做出应对，平衡了时间维度上，市场对于销量所带来的不稳定影响。这里我们同样使用了 Python 编程语言和 PuLP 库来实现和求解，我们通过创建一个 LpProblem 对象来初始化优化问题，将其命名为

"Crop_Planning_year_space 并设置优化目标为最大化。相较于模型二，我们在 last_year 的基础上，加入 year_before_last 和 three_years_ago 来对于预测销量做出约束，进而平衡市场对销量产生的波动影响。同时加入了互斥性和互补性两种约束条件，如果同一土地的作物符合我们特定的作物组合，那么他们的产量也会受到相对应的影响。在完成模型的构建后，我们调用 model.solve() 方法来求解这个优化问题。 PuLP 会自动选择适当的求解器(如 CBC 、GLPK 等)来解决这个问题。求解过程会花 费一定的时间，这取决于问题的规模和复杂度。求解完成后，我们就可以通过访问决策变量的 value 属性来获取最优的种植方案，这个方案告诉我们在每年市场和产量波动且农作物各自会相互影响的情况下，我们应该在每块地种植什么 作物，以实现最大的总体收益。

6.4 最优种植模型的改进

为应对现代农业的多样化需求，最优种植模型经历了逐步改进，逐渐提升决策的精度和效果，最终目标是实现经济效益的最大化。模型从初期的综合规划逐步发展为更加复杂的逐级优化和市场预测分析。

1. 综合农业规划模型

最初的模型专注于 2024-2030 年的农业规划，通过合理分配资源和优化土地及作物管理，最大化农业收益。模型在设计时考虑了作物种植特点、资源限制和可持续发展等因素，合理分配资源，力求在经济和生态效益之间取得平衡，确保农业系统的长期经济效益。

2. 逐级农业规划模型

在综合农业规划模型的基础上，逐级农业规划模型引入了逐年调整的优化机制，每年的种植方案基于前一年的优化结果。通过将大范围的优化环境分解为 7 个层层递进的优化子环境，模型能够更精确地考虑到如何应对市场波动、气候变化等不确定因素，确保种植方案的灵活性与持续化以逐步提升年度收益。

3. 逐级作物生产与市场预测模型

在逐级农业规划模型的基础上，逐级作物生产与市场预测模型进一步扩展了时间维度，扩大到利用前三年的销量数据构建时间序列，结合权重调整和随机浮动预测当前年的作物销量。另外也进一步扩展了空间维度，通过考虑作物的互斥性与互补性，模型在

时间和空间维度上优化作物组合，在应对复杂的市场要求和环境变化是具有灵活性和整体效益维持。

6.5 最优种植方案结果分析

6.5.1 问题一结果分析

对于该模型我们进行了灵敏度分析，主要分为三步，参数扰动分析、单因素分析和多因素分析。

1.参数扰动分析

在分数扰动分析中，我们通过对关键参数（作物的亩产量，成本，平均价格等）进行调整，来查看模型的反应。我们可以对每亩产量、每亩种植成本进行±10%的扰动，将平均价格替换为最高价格和最低价格，观察目标函数和种植决策变量的变化。从结果来看，模型受到的影响较大，出现了方案大幅变化的情况，因此，接下来我们会对单个因素进行分析。

2.单因素分析

我们对作物的亩产量，成本，平均价格各自进行了单独的调整，从我们的分析结果来看，亩产量种植成本固定百分比的扰动对模型的影响较为有限，总体浮动在 5%左右，但是平均价格的变化对模型的影响非常大，会极大的影响决策，导致方案的大幅更改。

3.多因素分析

为了确定我们的猜想，我们将亩产量，成本，平均价格两两组成组合进行两个因素调整的分析，从结果来看，亩产量和成本组合的结果与初始结果相近，但是包含了平均价格的组合却出现了很大变动，这进一步验证了市场和价格对于模型的较大干预。

综上所述，该模型对于作物本身的浮动稳定性较好，可以很好的应对农作物情况的变化，但是对于市场和价格变化却应变有限，出现了不稳定的情况，但是总体来看，在市场较为平静的情况下，它可以给农民提供一个合理且稳定的最优化种植方案。

6.5.2 问题二结果分析

对于该模型，我们同样采用灵敏度分析。

1. 参数扰动分析

在分数扰动分析中，我们通过对关键参数（作物的亩产量，成本，平均价格等）进行调整，来查看模型的反应。我们可以对每亩产量、每亩种植成本进行±10%的扰动，将平均价格替换为最高价格和最低价格，观察目标函数和种植决策变量的变化。从结果来看，模型受到的影响同样较大，出现了方案前后大相径庭的情况，因此，接下来我们会对单个因素进行分析。

2. 单因素分析

我们对作物的亩产量，成本，平均价格各自进行了单独的调整，从我们的分析结果来看，亩产量固定百分比的扰动对模型的影响几乎可以忽略不计，总体浮动在 1%以内，种植成本的固定百分比的扰动对模型的影响很小，总体浮动在 3%左右，但是平均价格的变化对模型的影响依旧非常大，会极大的影响决策，导致方案的大幅更改。

3. 多因素分析

我们将亩产量，成本，平均价格两两组成组合进行两个因素调整的分析，从

结果来看，亩产量和成本组合的结果与初始结果相差无几，但是包含了平均价格的组合依旧出现了很大变动，但对比来看，亩产量与价格的组合变动比种植成本和价格的组合更小，由此可以看出，模型对于亩产量的精确预测帮助模型对于价格的不稳定波动进行平衡。

综上所述，该模型对于作物本身的浮动稳定性非常好，可以完善的应对农作物情况的变化，但是对于市场和价格变化却依旧应变有限，出现了不稳定的情况，但是因为对于作物本身浮动稳定性的进一步提升，价格波动的影响也受到了平衡。这个模型可以表现稳定且良好的去应对农作物产量和成本的常规浮动，但是对于市场的变化能力仍然有限。因此，在市场较为稳定的情况下，该模型会成为决定种植策略时的有力帮手。

6.5.3 问题三结果分析

对于该模型，我们同样采用灵敏度分析。

1. 参数扰动分析

在分数扰动分析中，我们通过对关键参数（作物的亩产量，成本，平均价格等）进行调整，来查看模型的反应。我们可以对每亩产量、每亩种植成本进行±10%的扰动，将平均价格替换为最高价格和最低价格，观察目标函数和种植决策变量的变化。从结果来看，该模型受到的影响没有前两个大，方案仍是在基础方案上进行改动，没有出现方案完全不同大幅变化的情况。

2. 单因素分析

我们对作物的亩产量，成本，平均价格各自进行了单独的调整，因为基于上一个模型进行改进，所以该模型对于亩产量的浮动应对非常好，总体浮动在 1%以内。该模型对于种植成本的变化应对能力也有所提升，浮动在 2%以内，表现良好。对于价格变动，在价格剧烈浮动（从最高价到最低价时）出现了非常大的决策变化，最优化方案大幅改变，但是在较大但不极端的价格变化情况下，该模型可以做出较好的预测和应对。大致的浮动在 15%左右，若是在 20%以内的价格浮动，模型的方案变化大致在 8%以内，表现相较于前两者，大幅提高。

3. 多因素分析

三种组合对于出现的变化都有一定程度的提升，其中亩产量与种植成本的应对仍是最优秀的。

综上所述，该模型因为加入了时间序列预测销量和农作物互补性和互斥性等时间空间维度的约束，对于市场的变化更加灵敏也能更好的做出决策，且对于作物本身的变化应对仍然优秀。总的来说，在没有突发重大灾害或是市场极端变化的情况下，这是一个非常优秀且鲁棒性很高的模型，可以为最优化种植方案的决策者提供直观、易懂的决策支持信息。

七、 模型的优缺点与改进

7.1 模型的优点

1. 综合性与可扩展性强
该模型以综合农业规划为基础，旨在适应 2024-2030 年农业发展的多元化需求。它充分考虑了多种约束条件，如资源配置、市场需求、气候变化及作物特性等，提供了全方位的农业规划解决方案。同时，模型的设计注重可持续发展与经济效益之间的平衡，使其具有较强的适应性和扩展性，能够灵活应用于不同区域及规模的农业生产场景。
2. 逐级优化机制提高决策精度
该模型通过逐年调整种植方案，并将大规模优化环境拆分为多个子环境，使种植规划更加细化和精准。该逐步优化机制能够根据每年的具体情况、前期决策结果以及市场变化，动态调整种植策略，确保在不确定的环境中仍能提升经济效益。这种分阶段的优化方式不仅增强了模型应对不确定性的能力，还提高了长期收益的稳定性。
3. 融合时间序列预测与互补性优化
该模型扩展了时间维度，通过分析前三年的销量数据并应用时间序列预测，对未来年度的作物需求和市场波动进行准确预测。此外，模型考虑了作物间的互斥性与互补性，优化了空间上的作物组合，从而最大化了作物间的协同效应。这种组合优化不仅提升了整体生产效益，还增强了模型在不同生产环境下的适用性，以提高经济效益。

7.2 模型的缺点

1. 依赖历史数据的局限性
该模型依赖于历史市场数据和气候信息进行预测，但在面对突发的市场波动或极端气候变化时，预测的准确性可能受到限制。虽然模型通过逐级优化和时间序列预测减少了部分不确定性，但在应对突发因素时仍可能反应滞后，导致决策效果不佳。
2. 复杂性与计算成本较高
模型的逐级优化设计和多维度因素考虑，使其具备很强的适应性和灵活性，但也导致了计算复杂性和数据需求的增加。由于需要处理多个年度、多作物和多维度的数据，模型的求解过程可能耗时较长，计算资源的消耗较大，尤其是在处理大规模数据时，可能面临计算效率问题。
3. 忽视了更精细的市场动态与政策因素
尽管模型结合了市场需求预测，但对更加复杂和微观的市场变化（如消费者偏好、农产品国际贸易波动）缺乏深入分析。此外，农业政策变化（如补贴、税收）对种植决策的潜在影响在模型中考虑不足。这些因素可能对未来农作物收益产生重大影响，但当前模型未完全纳入。

7.3 模型的改进

1. 引入机器学习和人工智能技术
为克服历史数据依赖的局限性，模型可以引入先进的机器学习算法，如深度学习或强化学习，提升对市场波动和极端气候事件的预测能力。通过自适应学习机制，模型能够持续更新和调整预测方法，从而更有效地应对市场的不确定性。

2. 提高计算效率与简化模型结构

为解决模型计算复杂性高的问题，可以引入优化算法或使用并行计算技术，以加速求解过程并降低计算成本。采用优化算法不仅提升了计算效率，还能在处理大规模数据时提供更快的解决方案。同时，简化模型结构和减少冗余变量也有助于提高计算精度和效率。

3. 结合天气预报与气象大数据

为了更好地应对气候环境的不确定性，模型可以尝试结合实时天气预报和气象大数据，动态调整种植方案。这将帮助提前应对气候变化的潜在环境因素影响，确保作物产量稳定，降低风险，并提高对应农作物的经济效益。

八、 参考文献

[1] 张玉娥,白育铭,孙小兵,等.吕梁市豆类产业发展现状与思考[J].中国农技推广,2024,40(05):19-21.
[2] 高忠伦,付洪波,邵慧鑫,等.兵团农业结构调整下不同农作物种植成本收益的调查分析[J].全国商情(理论研究),2011,(08):26-27.DOI:10.16834/j.cnki.issn1009-5292.2011.08.011.
[3] 姜德华.农业结构最优化问题探讨[J].经济地理,1982,(03):171-175.
[4] 刘张勇.河北省花生生产经济效益分析[D].西南大学,2011.
[5] 康晓风,王乃斌,杨小唤.粮食种植面积提取方法的发展与现状[J].资源科学,2002,(05):8-12.
[6] 陈秀芬.山东棉花种植成本收益分析[D].山东农业大学,2014.

附录

Question1-code

```
    def create_model(crops, fields, planting_2023, stats_2023, years, seasons, expected_sales,costs, case):
        model = pulp.LpProblem("Crop_Planning", pulp.LpMaximize)


        print("Crops:", crops['id'].tolist())
        print("Stats crop_id:", stats_2023['crop_id'].tolist())


        # 创建决策变量
        # 作物 id
        x = pulp.LpVariable.dicts("plant",
                    ((int(c), str(f), int(y), str(s))
                     for c in crops['id']
                     for f in fields['name']
                     for y in years
                     for s in seasons),
                    lowBound=0, cat='Continuous')


        excess = pulp.LpVariable.dicts("excess",
                    ((int(c), int(y)) for c in crops['id'] for y in years),
                    lowBound=0, cat='Continuous')


        #二进制变量表示是否种植
        z = pulp.LpVariable.dicts("is_planted",
                    ((int(c), str(f), int(y), str(s))
                     for c in crops['id']
                     for f in fields['name']
                     for y in years
                     for s in seasons),
                    cat='Binary')


        # 目标函数
        if case == 1:
            model += pulp.lpSum(x[int(c),str(f),int(y),str(s)] * stats_2023[stats_2023['crop_id'] ==
c]['yield_per_mu'].iloc[0] *
                                stats_2023[stats_2023['crop_id'] == c]['avg_price'].iloc[0] -
                                x[int(c),str(f),int(y),str(s)] * costs[costs['crop_id'] ==
c]['cost_per_mu'].iloc[0]
                                for c in crops['id'] for f in fields['name'] for y in years for s in seasons
                                if not stats_2023[stats_2023['crop_id'] == c].empty)
        elif case == 2:
            model += pulp.lpSum(x[int(c),str(f),int(y),str(s)] * stats_2023[stats_2023['crop_id'] ==
```

```
c]['yield_per_mu'].iloc[0] *
                              stats_2023[stats_2023['crop_id'] == c]['avg_price'].iloc[0] -
                              x[int(c),str(f),int(y),str(s)] * costs[costs['crop_id'] ==
c]['cost_per_mu'].iloc[0] +
                              excess[int(c),int(y)] * stats_2023[stats_2023['crop_id'] == c]['avg_price'].iloc[0]
* 0.5
                              for c in crops['id'] for f in fields['name'] for y in years for s in seasons
                              if not stats_2023[stats_2023['crop_id'] == c].empty)

        # 约束条件
        # 1. 土地面积约束
        for f in fields['name']:
            for y in years:
                for s in seasons:
                    model += pulp.lpSum(x[int(c),str(f),int(y),str(s)] for c in crops['id']) <= fields[fields['name']
== f]['area'].iloc[0]

        # 2. 作物轮作约束
        for y in years[1:]:  # 从第二年开始
            for f in fields['name']:
                for c in crops['id']:
                    model += x[int(c),str(f),int(y),'第一季'] + x[int(c),str(f),int(y),'第二季'] + \
                            x[int(c),str(f),int(y)-1,'第一季'] + x[int(c),str(f),int(y)-1,'第二季'] <= \
                            fields.loc[fields['name'] == f, 'area'].values[0]

        # 3. 豆类种植约束
        bean_crops = crops[crops['name'].str.contains('豆类', na=False)]['id']
        for f in fields['name']:
            for y in range(0, len(years), 3):
                model += pulp.lpSum(x[int(c),str(f),int(y),str(s)]
                                    for s in seasons
                                    for c in bean_crops) >= \
                    fields.loc[fields['name'] == f, 'area'].values[0]

        # 4. 产量约束
        for c in crops['id']:
            for y in years:
                total_production = pulp.lpSum(x[int(c),str(f),int(y),str(s)] * stats_2023[stats_2023['crop_id'] ==
c]['yield_per_mu'].iloc[0]
                                              for f in fields['name'] for s in seasons)
                pred_sales = expected_sales[c]
                model += total_production - pred_sales <= excess[int(c),int(y)]

        # 5. 最小种植面积的约束
```

```
        for c in crops['id']:
            for f in fields['name']:
                for y in years:
                    for s in seasons:
                        area_min = fields[fields['name'] == f]['area'].iloc[0] * 0.1
                        model += x[int(c), str(f), int(y), str(s)] >= area_min * z[int(c), str(f), int(y), str(s)]  #
确保种植时达到最小面积
                        model += x[int(c), str(f), int(y), str(s)] <= fields[fields['name'] == f]['area'].iloc[0]
* z[int(c), str(f), int(y), str(s)]  # 确保未选择种植时面积为0


        # 6. 作物种植约束:
        for y in years:
            for s in seasons:
                for f in fields['name']:
                    for c in crops['id']:
                        land_type = fields.loc[fields['name'] == f, 'type'].values[0]
                        field_name = f'{land_type}({s})'
                        if crops.loc[crops['id'] == c, field_name].iloc[0] == 0:
                            model += x[int(c),str(f),int(y),str(s)] == 0
        # 7. 季节性约束:
        for y in years:
            for s in seasons:
                for f in fields['name']:
                    for c in crops['id']:
                        land_type = fields.loc[fields['name'] == f, 'type'].values[0]
                        if land_type in ['平旱地', '梯田', '山坡地']:  #都是单季作物
                            if s == '第二季':
                                for c in crops['id']:
                                    model += x[int(c),str(f),int(y),str(s)] == 0
                        elif land_type == '水浇地':
                            if s == '第一季':
                                model += x[16,str(f),int(y),str(s)] == 0  # 水稻
                            else:
                                for c in crops['id']:
                                    if c not in [35, 36, 37]:  # 大白菜、白萝卜和红萝卜
                                        model += x[int(c),str(f),int(y),str(s)] == 0
                        elif land_type == '普通大棚':
                            if s == '第二季':
                                for c in crops['id']:
                                    if c not in [38, 39, 40, 41]:  # 食用菌的编号
                                        model += x[int(c),str(f),int(y),str(s)] == 0
                        elif land_type == '智慧大棚':
                            for crop in [35, 36, 37]:  # 大白菜、白萝卜和红萝卜不能种在智慧大棚
                                model += x[int(c),str(f),int(y),str(s)] == 0
```

```
        # 8. 分散度约束
        for y in years:
            for s in seasons:
                for f in fields['name']:
                    model += pulp.lpSum(z[int(c), str(f), int(y), str(s)] for crop in crops['id']) <= 3  # 每季最多种 3 种作物
        return model
```

Question2-code

```
    def update_planting_plan(year, crops, fields, planting_2023, stats_2023, years, seasons, expected_sales, costs, last_year):
        model = pulp.LpProblem("Crop_Planning_year", pulp.LpMaximize)

        print("Crops:", crops['id'].tolist())
        print("Stats crop_id:", stats_2023['crop_id'].tolist())

        # 创建决策变量
        # 作物 id
        x = pulp.LpVariable.dicts("plant",
                    ((int(c), str(f), str(s))
                     for c in crops['id']
                     for f in fields['name']
                     for s in seasons),
                    lowBound=0, cat='Continuous')

        #二进制变量表示是否种植
        z = pulp.LpVariable.dicts("is_planted",
                    ((int(c), str(f), str(s))
                     for c in crops['id']
                     for f in fields['name']
                     for s in seasons),
                    cat='Binary')



        # 目标函数
        model += pulp.lpSum(x[int(c),str(f),str(s)] * stats_2023[stats_2023['crop_id'] == c]['yield_per_mu'].iloc[0] *
                            stats_2023[stats_2023['crop_id'] == c]['avg_price'].iloc[0] -
                            x[int(c),str(f),str(s)] * costs[costs['crop_id'] == c]['cost_per_mu'].iloc[0]
                            for c in crops['id'] for f in fields['name'] for y in years for s in seasons
                            if not stats_2023[stats_2023['crop_id'] == c].empty)
        # 约束条件
        # 1. 土地面积约束
        for f in fields['name']:
```

```
            for s in seasons:
                    model += pulp.lpSum(x[int(c),str(f),str(s)] for c in crops['id']) <= fields[fields['name'] ==
f]['area'].iloc[0]



        # 2. 作物轮作约束
        if last_year:
            last_year_vars = set((v.name.split('_')[1], v.name.split('_')[2], v.name.split('_')[3])
                                 for v in last_year.variables() if v.varValue > 0)
            for land in fields['name']:
                for crop in crops['id']:
                    if (crop, land, 'first') in last_year_vars or (crop, land, 'second') in last_year_vars:
                        model += z[land, crop,'first'] == 0
                        model += z[land, crop,'second'] == 0


        # 3. 豆类种植约束
        bean_crops = crops[crops['name'].str.contains('豆类', na=False)]['id']
        if (year-2023)%3 == 0:
            for f in fields['name']:
                for y in range(0, len(years), 3):
                    model += pulp.lpSum(x[int(c),str(f),int(y),str(s)]
                                        for s in seasons
                                        for c in bean_crops) == \
                             fields.loc[fields['name'] >= f, 'area'].values[0]


        # 4. 产量约束
        for c in crops['id']:
            expected_sales_min = expected_sales[c] * 0.90
            expected_sales_max = expected_sales[c] * 1.10
            total_production = pulp.lpSum(x[int(c),str(f), str(s)] * stats_2023[stats_2023['crop_id'] ==
c]['yield_per_mu'].iloc[0]
                                          for f in fields['name'] for s in seasons)
            pred_sales = expected_sales[c]
            #model += total_production  >= expected_sales_min
            model += total_production  <= expected_sales_max


        # 5. 最小种植面积的约束
        for c in crops['id']:
            for f in fields['name']:
                for s in seasons:
                    area_min = (fields[fields['name'] == f]['area'].iloc[0]) * 0.1
                    model += x[int(c), str(f), str(s)] >= area_min * z[int(c), str(f), str(s)]  # 确保种植时达到最
小面积

                    model += x[int(c), str(f), str(s)] <= (fields[fields['name'] == f]['area'].iloc[0]) * z[int(c),
```

```
str(f), str(s)]  # 确保未选择种植时面积为 0


        # 6. 作物种植约束：
        for s in seasons:
            for f in fields['name']:
                for c in crops['id']:
                    land_type = fields.loc[fields['name'] == f, 'type'].values[0]
                    field_name = f'{land_type}({s})'
                    if crops.loc[crops['id'] == c, field_name].iloc[0] == 0:
                        model += x[int(c),str(f),str(s)] == 0
        # 7. 季节性约束：
        for s in seasons:
            for f in fields['name']:
                for c in crops['id']:
                    land_type = fields.loc[fields['name'] == f, 'type'].values[0]
                    if land_type in ['平旱地', '梯田', '山坡地']:  #都是单季作物
                        if s == '第二季':
                            for c in crops['id']:
                                model += x[int(c),str(f),str(s)] == 0
                    elif land_type == '水浇地':
                        if s == '第一季':
                            model += x[16,str(f),str(s)] == 0  # 水稻
                        else:
                            for c in crops['id']:
                                if c not in [35, 36, 37]:  # 大白菜、白萝卜和红萝卜
                                    model += x[int(c),str(f),str(s)] == 0
                    elif land_type == '普通大棚':
                        if s == '第二季':
                            for c in crops['id']:
                                if c not in [38, 39, 40, 41]:  # 食用菌的编号
                                    model += x[int(c),str(f),str(s)] == 0
                    elif land_type == '智慧大棚':
                        for crop in [35, 36, 37]:  # 大白菜、白萝卜和红萝卜不能种在智慧大棚
                            model += x[int(c),str(f),str(s)] == 0
        # 8. 分散度约束
        for y in years:
            for s in seasons:
                for f in fields['name']:
                    model += pulp.lpSum(z[int(c), str(f), str(s)] for crop in crops['id']) <= 3  # 每季最多种 3 种作
物




        for c in crops['id']:
            for f in fields['name']:
```

```
            for s in seasons:
                model += x[int(c), str(f), str(s)] <= fields[fields['name'] == f]['area'].iloc[0]


    return model
```

Question3-code

```
def update_planting_plan(year, crops, fields, planting_2023, stats_2023, years, seasons, expected_sales, costs,
last_year, year_before_last, three_years_ago):
    model = pulp.LpProblem("Crop_Planning", pulp.LpMaximize)


    print("Crops:", crops['id'].tolist())
    print("Stats crop_id:", stats_2023['crop_id'].tolist())


    # 创建决策变量
    # 作物 id
    x = pulp.LpVariable.dicts("plant",
                ((int(c), str(f), str(s))
                 for c in crops['id']
                 for f in fields['name']
                 for s in seasons),
                lowBound=0, cat='Continuous')


    #二进制变量表示是否种植
    z = pulp.LpVariable.dicts("is_planted",
                ((int(c), str(f), str(s))
                 for c in crops['id']
                 for f in fields['name']
                 for s in seasons),
                cat='Binary')



    # 目标函数
    model += pulp.lpSum(x[int(c),str(f),str(s)] * stats_2023[stats_2023['crop_id'] ==
c]['yield_per_mu'].iloc[0] *
                    stats_2023[stats_2023['crop_id'] == c]['avg_price'].iloc[0] -
                    x[int(c),str(f),str(s)] * costs[costs['crop_id'] == c]['cost_per_mu'].iloc[0]
                    for c in crops['id'] for f in fields['name'] for y in years for s in seasons
                    if not stats_2023[stats_2023['crop_id'] == c].empty)
    # 约束条件
    # 1. 土地面积约束
    for f in fields['name']:
            for s in seasons:
                model += pulp.lpSum(x[int(c),str(f),str(s)] for c in crops['id']) <= fields[fields['name'] ==
f]['area'].iloc[0]
```

```
        # 2. 作物轮作约束
        if last_year:
            last_year_vars = set((v.name.split('_')[1], v.name.split('_')[2], v.name.split('_')[3])
                                 for v in last_year.variables() if v.varValue > 0)
            for land in fields['name']:
                for crop in crops['id']:
                    if (crop, land, 'first') in last_year_vars or (crop, land, 'second') in last_year_vars:
                        model += z[land, crop,'first'] + z[land, crop,'second'] == 0

        # 3. 豆类种植约束
        bean_crops = crops[crops['name'].str.contains('豆类', na=False)]['id']
        if (year-2023)%3 == 0:
            for f in fields['name']:
                for y in range(0, len(years), 3):
                    model += pulp.lpSum(x[int(c),str(f),int(y),str(s)]
                                        for s in seasons
                                        for c in bean_crops) == \
                             fields.loc[fields['name'] == f, 'area'].values[0]

        # 4. 产量约束
        for c in crops['id']:
            expected_sales_min = expected_sales[c] * 0.90
            expected_sales_max = expected_sales[c] * 1.10
            total_production = pulp.lpSum(x[int(c),str(f), str(s)] * stats_2023[stats_2023['crop_id'] ==
c]['yield_per_mu'].iloc[0]
                                          for f in fields['name'] for s in seasons)
            pred_sales = expected_sales[c]
            model += total_production  >= expected_sales_min
            model += total_production  <= expected_sales_max

        # 5. 最小种植面积的约束
        for c in crops['id']:
            for f in fields['name']:
                for s in seasons:
                    area_min = fields[fields['name'] == f]['area'].iloc[0] * 0.1
                    model += x[int(c), str(f), str(s)] >= area_min * z[int(c), str(f), str(s)]  # 确保种植时达到最
小面积
                    model += x[int(c), str(f), str(s)] <= fields[fields['name'] == f]['area'].iloc[0] * z[int(c),
str(f), str(s)]  # 确保未选择种植时面积为 0

        # 6. 作物种植约束:
        for s in seasons:
            for f in fields['name']:
```

```
                for c in crops['id']:
                    land_type = fields.loc[fields['name'] == f, 'type'].values[0]
                    field_name = f'{land_type}({s})'
                    if crops.loc[crops['id'] == c, field_name].iloc[0] == 0:
                        model += x[int(c),str(f),str(s)] == 0
        # 7. 季节性约束：
        for s in seasons:
            for f in fields['name']:
                for c in crops['id']:
                    land_type = fields.loc[fields['name'] == f, 'type'].values[0]
                    if land_type in ['平旱地', '梯田', '山坡地']:  #都是单季作物
                        if s == '第二季':
                            for c in crops['id']:
                                model += x[int(c),str(f),str(s)] == 0
                    elif land_type == '水浇地':
                        if s == '第一季':
                            model += x[16,str(f),str(s)] == 0  # 水稻
                        else:
                            for c in crops['id']:
                                if c not in [35, 36, 37]:  # 大白菜、白萝卜和红萝卜
                                    model += x[int(c),str(f),str(s)] == 0
                    elif land_type == '普通大棚':
                        if s == '第二季':
                            for c in crops['id']:
                                if c not in [38, 39, 40, 41]:  # 食用菌的编号
                                    model += x[int(c),str(f),str(s)] == 0
                    elif land_type == '智慧大棚':
                        for crop in [35, 36, 37]:  # 大白菜、白萝卜和红萝卜不能种在智慧大棚
                            model += x[int(c),str(f),str(s)] == 0


        # 8. 分散度约束
        for y in years:
            for s in seasons:
                for f in fields['name']:
                    model += pulp.lpSum(z[int(c), str(f), int(y), str(s)] for crop in crops['id']) <= 3  # 每季最
多种 3 种作物


        # 9. 加入互补性作物组合的约束
        complementary_crops = [
            (1, 7, 1.10),  # 黄豆(1) + 玉米(7) => 产量增加 10%
            (5, 6, 1.12),  # 豇豆(5) + 小麦(6) => 产量增加 12%
            (2, 5, 1.15),  # 南瓜(2) + 豇豆(5) => 产量增加 15%
            (2, 1, 1.15),  # 南瓜(2) + 黄豆(1) => 产量增加 15%
            (7, 5, 1.20),  # 玉米(7) + 豇豆(5) => 产量增加 20%
```

```
            (8, 3, 1.10),  # 白萝卜(8) + 红萝卜(3) => 产量增加 10%
            # 可以根据实际情况继续添加其他互补性作物组合
        ]
        for crop1, crop2, multiplier in complementary_crops:
            for f in fields['name']:
                for s in seasons:
                # 如果两种作物都种植在同一块地，应用产量的互补性提升
                    model += x[crop1, f, s] * x[crop2, f, s] * multiplier
        # 10. 加入互损性作物组合的约束
        conflicting_crops = [
            (2, 3, 0.85),  # 玉米(2) + 高粱(3) => 产量减少 15%
            (4, 6, 0.90),  # 大白菜(4) + 辣椒(6) => 产量减少 10%
            (8, 9, 0.80),  # 白萝卜(8) + 西红柿(9) => 产量减少 20%
            (1, 6, 0.85),  # 黄豆(1) + 辣椒(6) => 产量减少 15%
            (7, 3, 0.75),  # 玉米(7) + 高粱(3) => 产量减少 25%
            # 可以根据实际情况继续添加其他互损性作物组合
        ]
        for crop1, crop2, penalty in conflicting_crops:
            for f in fields['name']:
                for s in seasons:
                    # 如果两种作物都种植在同一块地，应用产量的互损性惩罚
                    model += x[crop1, f, s] * x[crop2, f, s] * penalty


        # 11. 预测销量约束
        for crop_id in crops['id']:
            if last_year is not None and year_before_last is not None and three_years_ago is not None:
                last_year_sales = stats_2023[stats_2023['crop_id'] == crop_id]['sales_last_year'].iloc[0]
                year_before_last_sales = stats_2023[stats_2023['crop_id'] ==
crop_id]['sales_year_before_last'].iloc[0]
                three_years_ago_sales = stats_2023[stats_2023['crop_id'] ==
crop_id]['sales_three_years_ago'].iloc[0]
            # 使用上述公式预测今年的销量
                predicted_sales = predict_sales(last_year_sales, year_before_last_sales, three_years_ago_sales)
            # 设置销量的约束，确保种植计划在该预测范围内
                expected_sales_min = predicted_sales * 0.90  # 允许 10%的最低浮动
                expected_sales_max = predicted_sales * 1.10  # 允许 10%的最高浮动


                total_production = pulp.lpSum(x[int(crop_id), str(f), str(s)] * stats_2023[stats_2023['crop_id'] ==
crop_id]['yield_per_mu'].iloc[0]
                                              for f in fields['name'] for s in seasons)
                model += total_production >= expected_sales_min
                model += total_production <= expected_sales_max


            return model
```

Question1 - result
详见 result1_1.xlsx, result1_2.xlsx
Question2 - result
详见 result2.xlsx