疫苗生产问题的数据分析

摘要

新冠肺炎在全球范围内快速传播，给各国的经济社会带来沉重打击，给全世界的人民带来深重灾难。为更好地应对新冠肺炎，新冠肺炎疫苗的研发与生产成了当务之急。而疫苗生产公司的生产方案是其中的重要一环，我们将从多个角度来研究分析此问题。

针对问题一，需要对题目所给的每箱疫苗在各个工位上的生产时间做均值、最值、方差和分布等统计分析。首先对数据进行预处理，我们用 *Dixon* 检测法对数据进行离群点分析，并在此基础上对每种疫苗在各个工位上的生产时间进行了概率分布检测，发现数据均服从正态分布。

针对问题二，需要以各工位生产每箱疫苗的平均时间为依据，对疫苗的整体生产顺序做出规划，使生产十箱疫苗的总时间最短。我们枚举了疫苗所有可能的生产顺序，并计算在每种顺序下的总时间，取其最短时间所对应顺序作为答案。我们的生产方案为：4-5-10-7-8-1-2-3-6-9，总时间为 184.6549 分钟。

针对问题三，需要使疫苗交货总时间比问题二缩短 5%，计算此时的最大概率，并获取缩短时间比例与最大概率之间的数学关系。由于在实际生产中，无法预先安排出最佳的方案。因此我们希望找出生产总时间的数学期望最小的生产顺序。我们用大量数据测试每一种生产顺序，用测试得到的所有样本所对应的生产总时间的平均数代替该生产顺序的生产总时间的数学期望，找出其最小值对应的生产安排，作为我们答案。我们的生产方案为：4-10-7-8-2-6-1-3-5-9，平均生产时间为 204.8139 分钟。在该情况下求得总时间缩短 5%的概率为5.03635×10^{-23}。

针对问题四，需要在可靠性为 90%要求下，分配生产任务，计算完成生产任务的最短时间。在可靠性为 90%的情况下，生产时间的波动范围为$\mu - 1.645\sigma \le t \le \mu + 1.645\sigma$，然后在$t = \mu - 1.645\sigma$的情况下计算生产所需的最小天数。我们计算出完成生产任务所需的最短时间为 173 天。

针对问题五，需要以疫苗公司最大销售额为目标，制定疫苗的生产计划。首先，我们将各个工位生产时间不超过 100 天与各种疫苗存在订单量的上限作为条件，以最大销售额为目标，进行线性规划，获取生产方案初稿。由于线性规划忽略了实际生产的细节，我们调整生产方案初稿，使其满足生产安排与边界条件，获得最大销售额下的生产计划——YM1 疫苗 0 箱、YM2 疫苗 500 箱、YM3 疫苗 600 箱、YM4 疫苗 0 箱 YM5 疫苗 1200 箱、YM6 疫苗 1546 箱、YM7 疫苗 491 箱、YM8 疫苗 800 箱、YM9 疫苗 600 箱、YM10 疫苗 900 箱，最大销售额 31213800 美元。

关键词： 动态规划 线性规划 多目标规划

一、问题重述

1.1 问题背景

新冠肺炎快速传播，全球的疫情感染率显著上升，给全世界的人民带来了巨大的影响，阻碍了世界经济的整体发展。为了更好地应对新冠肺炎，寻求一条发展之路。世界各国都行动了起来，研发新冠疫苗，一起为抵御新冠疫情所带来的风险贡献自己的一份力量。

某疫苗生产公司，其疫苗的生产一共需要经过四个工位操作的生产流程，并且每个流程一仅能生产 100 剂疫苗，这 100 剂疫苗在生产过程中，会装进同一加工箱按照固定的工位加工顺序进行处理和加工。同时为了防止在生产多种类型的疫苗过程中，发生由于混乱而导致的包装错误。公司规定在每个工位不能同时生产不同类型的疫苗，且禁止疫苗插队生产。插队生产概念指的是当某种类型的疫苗进入第一个工位后，其顺序就不能发生变化，当前一种类型疫苗离开上一个工位后，下一种类型的疫苗才能进入该工位进行生产处理。

疫苗生产公司为了安全，对它们所生产的 10 种生产类型的疫苗在每个工位上都进行了 50 次的模拟生产，由于生产过程中会受到受到生产设备、纯化等因素的限制，每个类型每次的生产时间存在波动。同时这 10 种类型的疫苗也存在各自的生产任务指标的制约。

1.2 问题提出

问题一：根据问题一的要求，我们需要对题目所给的每箱疫苗在各个工位上的生产时间做均值、最值、方差等统计分析，为疫苗的生产人员了解各工位生产疫苗的能力水平提供参考。

问题二：为了能够尽快地对 10 种类型的疫苗进行检测，题目要求我们以各工位生产每箱疫苗的平均时间为依据，对疫苗的整体生产顺序做出规划，建立合适的数学模型，计算生产总时间，确保能在最短的时间内，实现疫苗生产到疫苗检测的交付工作。

问题三：实际生产中，每个工位生产每类疫苗的时间是随机的。题目要求我们建立适当的模型，使疫苗交货总时间比问题二缩短 5%。明确各个疫苗的生产顺序，以最大的概率完成此任务为目标，求解缩短时间比例与最大概率之间的数学关系。

问题四：题目给出了 10 种类型疫苗的生产任务，并限制了每日的工作时限。同时还要求了同种疫苗必须在全部生产完成才能生产另外类型的疫苗。我们需要在可靠性为90%要求下，建立合适的数学模型，安排生产任务，计算完成生产任务的最短时间。

问题五：问题五以疫苗公司最大销售额为目标，允许疫苗公司在规定时间内选择部分疫苗进行生产，且各生产任务可以拆分，但每个工位一天只能工作 16 小事，要求我们建立合适的数学模型来制定疫苗的生产计划。

二、模型假设

1. 假设工人换班、更换疫苗生产品种的衔接过程种造成的时间浪费可以忽略；
2. 假设疫苗的合格率为 100%；
3. 假设疫苗的生产互不相干，其所需生产时间的概率分布独立。

三、符号说明

符号	符号意义
r_{22}	表示检验较大值是否为离群点的统计量
r_{22}'	表示检验较小值是否为离群点的统计量
α	表示检出水平
n	表示样本容量
x_i	表示来自母体的样本，i表示样本个数
$\bar{x}$	表示n个样本的平均值
T_{min}	表示最短生产总时间
$et(i,j,k)$	表示从第一个疫苗进入生产流程开始，到第j个进入流程的疫苗在第i个工位上结束后的时间间隔长度，k表示全排列的方式，是一个状态变量
$v(j)$	表示第j个进入流程的疫苗它本身的疫苗类型编号
$t[v(j),i]$	表示编号为$v(j)$类型的疫苗在i工位上的平均生产时间
P	表示目标概率
r	表示缩短时间的比例
μ	表示某种排列方式的生产总时间均值
$p(t)$	表示该生产顺序方式下的概率密度函数
μ_{min}	表示全排列生产顺序情况下的所有生产总时间均值的最小值
t	表示完成生产任务的总工期
t_i	表示完成第i个类型的疫苗的工期
B	表示该疫苗分配产量与类型的情况下，疫苗公司的销售额
P_i	表示第i种疫苗的单价
L_i	表示第i种疫苗的订单上限额度
t_{ij}	表示第i种疫苗第j个工位阶段生产所需要的时间
T_{mn}	表示第m天第n个工位生产疫苗所花费的总时间

注：未列出符号及重复的符号以出现处为准

四、模型的建立与求解

4.1 问题一

4.1.1 问题一的分析

为了使疫苗生产公司的管理者更直接地了解到各个工位生产不同类型疫苗的能力。我们通过题目附件中给出的各工位生产不同类型疫苗的生产时间集合，利用统计分析的方法，计算生产时间的均值、方差和最值。利用夏皮洛-威尔克检验依次对所有生产时间进行正态分布等概率分布类型的验证，寻求满足各工位各类型疫苗生产时间的概率分布类型。得到所有工位所有类型疫苗的生产时间的统计分析数据，为疫苗生产提供参考。

4.1.2 问题一的解答

对于问题一的解答，我们需要先清楚统计分析的检验方法。根据问题的分析，明确概率分布的检验方法之后，对附件 1 中的所有工位所有类型疫苗的生产时间集合做验证，利用求解值作为依据，给予疫苗生产参考意见。

4.1.2.1 数据预处理

由于离群点对于附件 1 中提供的小样本数据影响非常显著，因此我们需要对附件 1 中的所有工位所有类型疫苗的生产时间做离群点检测，来剔除极端值对于统计分析的影响。

1)离群点检测

通过查阅文献，我们采用 *Dixon* 检测法剔除小样本中的异常数据。此方法的原理是通过离群值与临近值的差值与极值的比值这个统计量来判断是否存在异常值。其符合我们样本数量个数的统计量为：

$$r_{22}=\frac{x_n-x_{n-2}}{x_n-x_3}\ \text{和}\ r_{22}{}'=\frac{x_3-x_1}{x_{n-2}-x_1} \tag{1}$$

其中x_n是样本中的数据。r_{22}是检验较大值是否为离群点的统计量，$r_{22}{}'$是检验较小值是否为离群点的统计量。

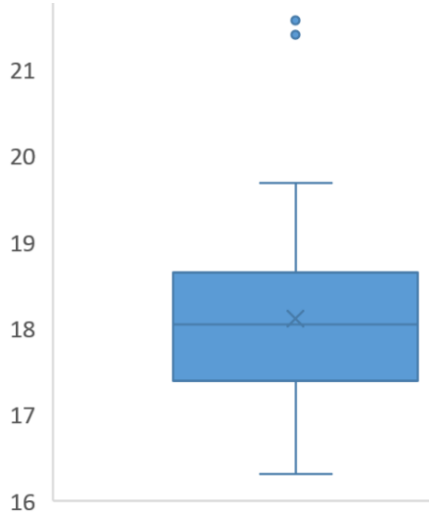

图 1 数据预处理箱型图

我们通过统计量与 *Dixon* 检验临界值表$D(\alpha,n)$进行比较，表中α表示检出水平，n表示样本容量。

用此比较来判定样本中的某数据是否为离群值。若该值是离群值，则我们将其中数据样本中剔除，用一个数据来替代它，消除离群值对于样本统计分析的影响。

4.1.2.2 理论准备

下面我们对计算过程中需要用到的概率分布检验的专用名词做出解释：

1)夏皮洛-威尔克检验

夏皮洛-威尔逊检验是一种检验数据是否服从正态分布的方法，此方法的目的在于判断工位上所有类型疫苗的生产时间是否属于正态分布。

其检验的原假设如下

H_0: x_i $(i = 1,2,3 \ldots)$ 是来自一个正态分布总体的样本，统计量是：

$$W = \frac{(\sum_{i=1}^{n} a_i x_i)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{2}$$

其中x_i表示的来自母体的样本，i表示样本个数，$\bar{x}$是n个样本的平均值。

4.1.2.3 计算求解

根据我们的问题分析和题目附件中给出的各工位各类型疫苗生产时间数据，结合离群点检测数据预处理方法和夏皮洛-威尔克检验对数据进行处理。

Step1.求解计算未经过数据预处理后的所有工位上的所有类型的疫苗生产时间的均值$\bar{x}$；

Step2.利用 *Dixon* 检测法方法对所有工位所有类型疫苗生产时间进行离群点检测的数据预处理，离群点检测具有能够从样本数据中剔除对我们附件中小样本数据的影响，能够更加准确地表示我们的统计分析数据；

Step3.用原疫苗生产时间中的均值$\bar{x}$来代替离群值检测中筛选出来的离群值，削弱此类数据对于统计分析结果的影响，对处理完的生产时间数据进行均值$\overline{x'}$、方差Var、最值x_M的求解，得到这三个结果数据；

Step4.对步骤三数据预处理后的数据进行夏皮洛-威尔克检验，判断预处理后的数据是否满足正态分布。若满足，则该工位上该类型疫苗的生产时间数据的概率分布满足正态分布；若不满足检验，则我们需要对其进行进一步讨论来判断该工位该类疫苗的概率分布类型。

我们通过预处理、检验和求解，发现所有工位所有类型疫苗的生产时间均满足正态分布，因此不需要对不满足的夏皮洛-威尔克检验的情况进行进一步分类讨论。

4.1.2.4 结果呈现

根据题目要求，通过附件 1 中的给出的所有工位所有类型的疫苗的生产时间，我们得到它们的均值、最值、方差和概率分布特性。所有的生产时间的概率分布均满足正态分布。(具体数据详见附录 1)

4.2 问题二

4.2.1 问题二的分析

根据问题二的题目要求，为了尽可能地缩短时间成本，我们首先对疫苗生产顺序进行规划，对顺序进行全排列列举，寻找所有疫苗类型的排列情况。随后以总生产时间最短为目标函数，每个疫苗在每个工位上结束生产的时刻为变量，将疫苗生产不允许插队和疫苗生产进入工位后顺序不变作为约束条件，进行量化处理。

在模型求解过程中，根据动态规划的方法，列出状态转移方程。最终以最后一个疫苗离开生产流程为生产循环结束的判定条件，当生产过程结束时，变量即为该生产顺序排列情况下的生产总时间。通过计算出全排列所有排列顺序可能情况下的生产总时间，比较所有的生产总时间变量，确定最终的目标函数，最短总生产时间。

4.2.2 模型的准备与建立

4.2.2.1 理论准备

下面我们将对模型中所需要用到的一些专用名词做出解释

1)生产任务全排列

排列指的是从n个元素中取出m个元素，并按照一定的顺序将它们排列起来。而全排列指的就是将所有的当$m=n$时，所有可能的$n!$种排列方式。由于附件中的数据量较小，因此我们采用全排列的方式。目的在于找出所有解，明确极值，寻找最短的生产时间。

4.2.2.2 模型的建立

我们通过对问题进行分析，并结合已知条件得到模型的目标函数，利用相关公式求解生产任务全排列情况下的生产顺序规划方案，细化模型，为之后的求解过程奠定理论基础。

1)目标函数的确定

我们将问题中的生产总时间阶段化处理，定量化的表达从第一个疫苗进入生产流程开始到各个阶段所需要的时间。为了保证生产总时间最短，我们要使最后一个疫苗离开生产流程的时间最短，建立目标函数：

$$T_{min}=\min\left(et(i,j,k)\right),i=4,j=10 \tag{3}$$

其中$et(i,j,k)$表示从第一个疫苗进入生产流程开始，到第j个进入流程的疫苗在第i个工位上结束后的时间间隔长度，k表示全排列的方式，是一个状态变量。

并且变量i、j、k满足条件$i=1,\dots,4$，$j=1,\dots,10$。k一共有 10！种取值情况。

在这里使得生产总时间 T 最小，我们就需要找到最优排列情况下的$et(4,10,k)$。

2)约束条件的限制

根据题目的要求，我们知道疫苗生产过程中不允许出现插队的情况，即进入第一个工位生产的疫苗顺序确定后，就一定要保持这个顺序不变。同时在某工位生产的疫苗若需要进入下一个工位进行生产，其必须满足下一个工位原本生产的疫苗已经完成生产进入下一个生产环节，且自身在该工位的生产已经结束。因此我们对约束条件进行定量化处理和表示：

$$\begin{cases} t[v(j-1),i+1]\geq t[v(j),i] \\ et(j,i,k)-et(j-1,i,k)\geq t[v(j),i] \end{cases} \tag{4}$$

其中$v(j)$表示第j个进入流程的疫苗它本身的疫苗类型编号，$t[v(j),i]$为编号为$v(j)$类型的疫苗在i工位上的平均生产时间。

其中对于插队的限制条件也在这两个工位顺序生产限制因素中体现出来，因此不需要再对插队进行定量化的限制因素表现。

3)生产任务全排列的应用

通过所有疫苗的类型数量，求解出所有可能的生产顺序的全排列方式，计算每种生产顺序排列情况下的$et(4,10,k)$，也就是生产总时间 T。最终通过对比所有生产顺序排列情况下的$et(4,10,k)$，求出最终结果，此结果即为生产总时间的最小值T_{min}。

4.2.3 模型的解答与分析

我们通过应用数学模型，明确目标函数，结合约束条件和生产任务全排列的可能对模型进行求解。

4.2.3.1 模型的求解

我们根据问题分析，题目给出的每个类型的疫苗在各个工位上的平均生产时间，我

们可以得到我们所运用到的各个参数的变化规律，具体如下说明：

1)每种类型的疫苗在工位上生产过程中满足状态转移方程：

$$et(i,j,k) = t[v(j),i] + \max\{et(i,j-1,k), et(i-1,j,k)\} \quad (5)$$

该式子表示从第一个进入生产流程的疫苗开始到第j个进入生产流程的疫苗，在第i个工位上结束生产的时间间隔，应该等于其自身在第i个工位上的生产时间和$et(i,j-1,k), et(i-1,j,k)$式子的最大值。

后面的最大值公式中体现了约束条件，即该疫苗能够进入生产流程，应该满足上一个进入该流程的疫苗已经进入下一个工位流程，且自身在该工位的生产已经结束，而$et(i,j-1,k), et(i-1,j,k)$恰好就体现了约束条件的限制。

2) $et(i,j,k)$在同一种生产顺序排列模式下，有固定的计算规律和生产判别停止条件，即从：

$$et(1,1,k) \rightarrow et(4,1,k) \rightarrow \cdots \rightarrow et(4,10,k) \quad (6)$$

当排列形式固定的情况下，当$et(i,j,k)$转化为$et(4,10,k)$时，生产过程结束，不再继续进行计算。此结果$et(4,10,k)$即为该生产顺序排列情况下的生产总时间 T。

我们利用动态规划的方式，具体求解模型的解，即生产总时间的最小值，步骤过程和示例图如下：

Step1.通过生产顺序全排列寻找所有的生产排列方式，一一对其的生产总时间 T 结果进行求解，最终选取一个最小的生产总时间T_{min}；

Step2.针对每个不同的排列方式，计算生产总时间 T，初始变量为$et(1,1,k)$，即生产顺序第一个的疫苗进入整个生产流程；

Step3.利用约束条件，来判断已经在工位生产流程中的疫苗是否能够进入下一个工位生产阶段；还未进入工位生产流程的疫苗是否能够能进入；

Step4.若通过判断可以确认该类型疫苗可以进入下一个工位生产阶段或还未进入生产流程的疫苗，满足条件约束条件，能够进入生产环节，则运用状态转移方程，求解每个疫苗下一阶段的生产时刻数据，保留所有疫苗的初始进入 CJ1 工位生产流程的时刻和最后离开 CJ4 工位生产的时刻；

Step5.按照步骤 3 和步骤 4 一直循环求解，根据变量$et(i,j,k)$的变化规律，通过它的值来判断在该生产排列顺序情况下，整个生产过程是否已经结束。当变量为$et(4,10,k)$时刻，整个生产过程结束，并且变量$et(4,10,k)$的值即为该生产排列顺序情况下的生产总时间 T；

Step6.重复步骤 1，继续求解其它生产过程排列顺序情况下的最终生产总时间 T，对比所有排列情况下的生产总时间 T，进行对比和比较求解出生产总时间的最小值T_{min}和每个类型疫苗进入生产流程和离开生产流程的时刻。

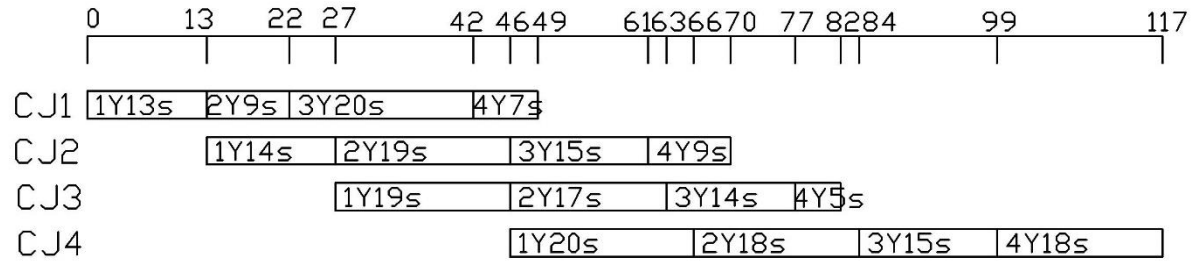

图 2 数据预处理箱型图

4.2.3.2 模型的结果与分析

1).结果展示

根据题目要求，通过附件给出的各个工位各个类型生产的平均时间和前文模型中的公式，用 Dev C++软件计算出最短的生产时间以及各个型号疫苗进入和离开生产流程的时刻，结果呈现如下所示：

表 1 结果呈现图

加工顺序	进入 CJ1 时刻*(s)*	离开 CJ4 时刻*(s)*
4	0.0000	41.9896
5	7.9887	55.9020
10	16.7587	71.9544
7	29.7409	91.0420
8	40.9011	107.8734
1	56.9212	127.8863
2	70.2052	146.8287
3	80.0761	161.9451
6	100.1345	175.7052
9	119.2086	184.6549

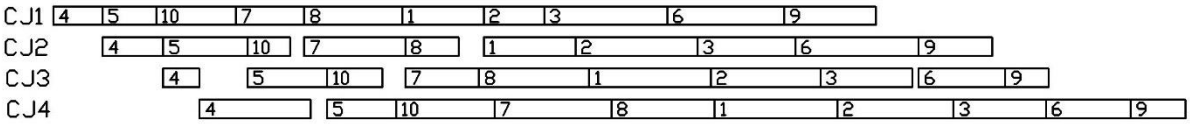

图 3 生产顺序结果示意图

2).结果分析

最终通过动态规划的方法，我们求解出了总的最短生产时间的疫苗类型排列顺序，为 4-5-10-7-8-1-2-3-6-9，此结果下，最终的时间为为 184.6549*s*。

这一种生产形式下，能够更大程度的降低时间成本。我们同时计算了另一种，逐一进入生产流程方式，即上一种类型疫苗生产完毕后，下一种类型疫苗再进入情况下的生产总时间。发现后者结果为 643.8466*s*。两个结果相差较大。

因此我们可以知道，当一个疫苗进入下一个工位流程后，另一个疫苗进入该工位生产，寻找合理地排序方式，是能够最大可能节约时间成本，减少生产总时间的最有效的方法。

4.3 问题三

4.3.1 问题三的分析

在第二题中，我们已经算出在所有疫苗的生产时间都已确定的情况下，最短的生产总时间。而第三问要求我们在第二问的基础上再将生产总时间缩短 5%。然而，在实际生产的过程中，疫苗的每一阶段的生产时间并不确定，而是服从某一参数确定的正态分布。因此，实际交货时间也将服从某一未知分布。要求的交货时间的改变，在交货时间内完成的概率也随之改变。在本题中我们将通过合理的安排生产方案，最小化生产时间，并在此基础上找出交货时间和完成的概率的关系。

通过对第二题的结论进行分析，我们已经得知，对于不同的某种疫苗单阶段生产时间，最优的生产顺序也是不同的。值得注意的是，在实际生产中，我们无法预先知道每种疫苗每一阶段的生产时间，也就无法预先安排出最佳的方案。因此，我们希望找出生产总时间的数学期望最小的那一种生产顺序。

由于影响生产总时间的因素数量多且关系复杂，我们将用大量数据测试每一种生产顺序，用测试得到的所有样本所对应的生产总时间的平均数代替该生产顺序的生产总时间的数学期望。

4.3.2 模型的准备与建立

4.3.2.1 理论准备

根据题目要求，以最大概率完成任务为目标，确定生产顺序和缩短时间比例与最大概率的关系，建立数学模型。并对模型求解中需要运用到的专用名词做下说明，以便之后求解过程中直接运用。

1)生产顺序全排列

疫苗生产顺序全排列指的是，针对所有种类的疫苗，考虑它们在工位生产流程中所有的顺序安排情况。为了确定题目中要求的生产顺序，我们对先疫苗生产顺序全排列，考虑此情形下的最短生产时间，并选择合理的疫苗生产顺序后，方便对概率分布进行求解。

2)K-S 检验(克尔莫克洛夫-斯米洛夫检验)

克尔莫克洛夫-斯米洛夫检验方法是一种非参数统计检验方法，是一种检验样本数据是否满足正态分布的方式。

K-S test 的原假设H_0：总体的x具有F的分布，构造检验统计量：

$$Z = \sqrt{n}\max(|F_n(x_{i-1}) - F(x_i)|, |F_n(x_i) - F(x_i)|) \tag{7}$$

我们需要用到该检验方法，对大样本的随机数据进行概率分布形式的求解，为之后求解最终的数学关系提供基础信息。

3)随机化生成数据

考虑实际生产中，每个工位生产每种疫苗所需的时间具有随机性。因此，对每种生产顺序安排情况下，随机生成满足其自身所在工位所属疫苗类型的概率分布规律的大量数据组。每组数据中共有 40 个数据，包含了所有工位所有类型疫苗的随机生成数据。为模型的求解提供数据支持。

4.3.2.2 模型的建立

我们通过对问题进行分析，并结合已知数据与条件，得到最终函数关系的概率分布，利用相关公式求解出函数关系的数学表达式，细化模型，为之后的求解过程奠定理论基础。

1)缩短的时间比例与最大概率之间的关系确定

通过题目的已知要求和数据，我们通过寻找以最优生产顺序与最大概率完成任务的概率分布的情况，利用概率分布求解出缩短时间比例与最大概率之间的数学关系，具体关系式如下：

$$P = \int_{-\infty}^{\mu_{min}(1-r)} p(t)dt \tag{8}$$

其中P表示目标概率，μ_{min}表示第二题结果，r是缩短时间的比例，$p(t)$表示该生产顺序方式下的概率密度函数。

2)最短生产总时间的均值求解

在生产顺序全排列情形中，针对每一种排列方式，我们都随机化生成了大量数据组。

针对每一种数据组，我们可以求解出满足该数据组和该排列方式的最短生产总时间T_{min}。针对该排列方式，我们可以求解出很多个数据组的最短生产总时间，依靠多组T_{min}，求解出该排列方式的生产总时间均值μ。求出全排列生产顺序情况下的所有生产总时间均值μ，寻找它的最小值μ_{min}和该值下的生产顺序排列方式。

3)生产总时间的计算

对于最短的生产总时间的均值求解中，关于最短生产总时间计算，我们考虑运用问题二的模型，直接求解出每种排列方式和每种随机生成的数据组情形下的生产最短总时间。以方便在最短生产总时间的均值求解过程中可以直接调用生产最短总时间在各种情形下的数据。

4.3.3 模型的解答与分析

我们通过应用数学模型，明确最终的求解目标，结合随机生成的数据样本和题目的条件要求，对模型进行求解。

4.3.3.1 模型的求解

在模型的求解过程中，我们需要明确求解的步骤和流程。首先需要明确目标解，其次通过全排列情况下的随机数据，求解最小生产总时间的均值，然后利用假设检验计算选择出的排列方式的概率分布，最终通过概率密度函数和题目要求的总时间缩小值，求解出最终缩短时间比例与最大概率之间的关系表达式。

Step1.为了确定最终的疫苗生产顺序，我们考虑采用全排列所有生产顺序的方式，来进一步明确生产顺序；

Step2.充分考虑到生产过程中的实际性，我们对于每一种生产顺序情形下的随机生成大量的数据组，每组数据中包含 40 个工位和疫苗类型生产时间的随机数据，且这些数据均满足各自的概率分布；

Step3.求解每一种生产顺序，每一种随机数据组类型的生产最短时间T_{min}，并以此求解每一种生产顺序中所有数据组的生产最短时间，最终计算出每种生产顺序的生产总时间均值μ；

Step4.寻找所有生产顺序可能情况下的最短生产总时间μ_{min}，因为最短生产总时间是能够以最大概率完成这个任务为目标的基本要求；

Step5.利用理论准备中的 K-S 检验方法对步骤 4 中求解出来的最短总时间μ_{min}，对应生产顺序情况下的所有随机数据组中的生产最短总时间T_{min}，做概率分布的检验。找在这个生产顺序情形下，所有随机数据组中求解出的最短总时间T_{min}数值，满足哪种概率分布形式和该分布形式下的具体概率密度函数；

Step6.利用步骤 5 中求解出来的概率分布形式和概率密度函数，带入模型中缩短时间比例与最大概率的数学关系，求解出这两者在题目条件背景情况下的具体关系。这种概率密度函数对应的生产顺序，即为以最大概率完成这个任务目标条件下的最优生产顺序。

4.3.3.2 模型的结果与分析

1).结果展示

根据题目要求，用 Dev C++软件计算出缩短时间比例与最大概率之间的数学关系，以及此结果对应生产顺序下的最终疫苗生产顺序。

经过检验法和最终的结果验证，我们可以知道最终生产最短总时间的概率分布满足正态分布的关系。我们用 Q-Q 图和直方图的呈现来说明此分布满足正态分布的结论：

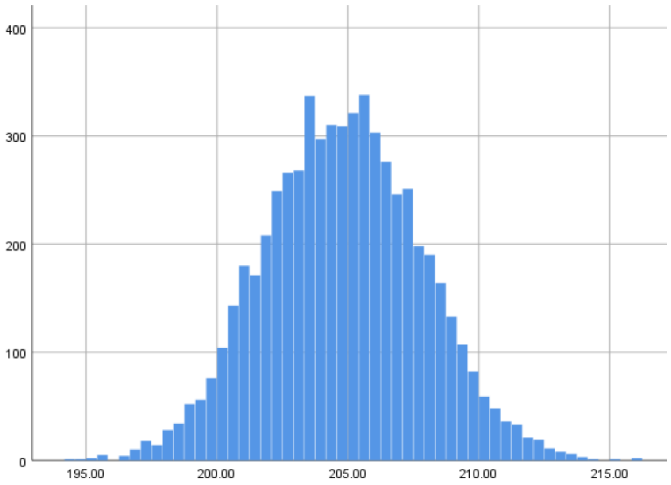

图 4 直方结果示意图

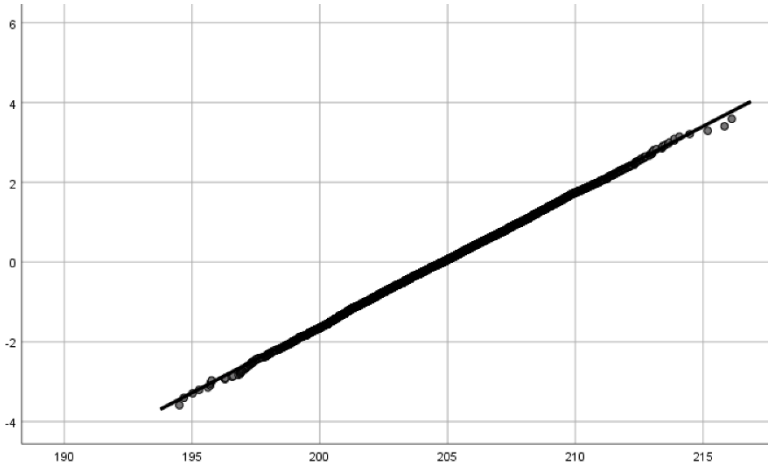

图 5 Q-Q 结果示意

数学关系表达式如下：

$$P = \int_{-\infty}^{\mu_{min}(1-r)} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \tag{9}$$

其中μ_{min}为 184.6549，σ为 8.975，μ为 204.8135。

当缩短时间比例r为 5%时，最大概率为5.03635×10^{-23}

最终的疫苗生产顺序为 4-10-7-8-2-6-1-3-5-9。

2).结果分析

从上文的结果中可以发现，在最优生产方案下，要想缩短 5%的时间，其概率只有为5.03635×10^{-23}，在现实生活中几乎不可能发生。

分析可以发现，第二题的最优生产总时间之所以较短，是因为在安排方案时预知了每一工位的每一个生产时间。而在实际生产中，不可避免的会发生和计划不一致的情况，也就导致了时间上的浪费。

事实上，即使将限制条件放宽到总时间增加 5%，其最大概率也不过只有 0.13%。只有将时间限制放宽到总时间增加 15%时，其概率才能达到 99.4%，满足工业生产的要求。

4.4 问题四

4.4.1 问题四的分析

根据问题四的题干要求，需要在可靠性是90%的条件下对附件2的数据安排生产方案，求出完成任务的最小天数。由于一些外部原因，工厂的每个工位每天生产时间不能超过16小时，且每种类型疫苗全部生产完成之后才能生产另外类型的疫苗。

我们建立了可靠性理论模型，在可靠性为90%的情况下，确定每种疫苗在每个工位的具体生产时间范围，并以此为基础计算生产每种疫苗达到订单要求所需的最少时间。最后，我们安排十种疫苗的生产方案，得出完成任务所需的最小天数。

4.4.2 问题四的解答

4.4.2.1 理论准备

我们根据题目的要求，对模型求解过程中需要用到的一些专有词汇做出了解释和说明：

1)可靠性

可靠性是产品质量的一种特性，其指产品在规定的条件和规定的时间情况下，完成规定功能的能力，也就是完成规定任务的可能性。在本题中，产品的规定条件和规定时间是工位每天生产的时间为16小时和每种类型疫苗的生产任务不能拆分，只有同类型疫苗生产完成才能生产下一种类型的疫苗。在这两个条件背景下，以可靠性90%来安排生产任务，即以完成任务的可能性为90%的前提进行生产任务安排。

2)正态分布下的先验概率

正态分布下的先验概率指的就是在一个满足正态分布的事件的前提下，根据以往经验预测与该事件相关的概率。在本题中，我们以问题一中的概率分布结果为基础，可以得到所有工位所有种类疫苗的生产时间均满足正态分布的结论。因此事件在满足正态分布情况下的先验概率，则其误差量将会在期望值的左右对称，即该情况下的正态分布具有对称性。

4.4.2.2 模型的建立与求解

我们通过对问题进行分析，并结合已知条件得到确定模型的解，利用理论准备中的概念，细化模型，为之后的求解过程奠定理论基础。

1)工期确定

根据题目要求，我们以附件2给出的10种类型疫苗的生产任务和完成生产任务的可靠性为90%的前提，来计算至少需要多少天才能完成所有类型疫苗的生产任务。建立函数关系如下：

$$t = \left[\sum_{i=1}^{10} t_i\right] \tag{10}$$

其中t表示完成生产任务的总工期，t_i表示完成第i个类型的疫苗的工期，其中每类疫苗的工期可以不是整数，但所有类型疫苗的工期累加之后的总工期需要向上取整，得到具体的天数。

2)求解步骤

Step1.通过问题一中的计算结果，我们可以得到所有工位所有类型疫苗的生产时间的标准差σ和期望μ；

Step2.利用第一步中求解出来的标准差σ和期望μ的结果，并结合理论准备中的正态分布下的先验概率的分布形式，绘制出该情况下正态分布的图像，误差量在期望值的左右对称；

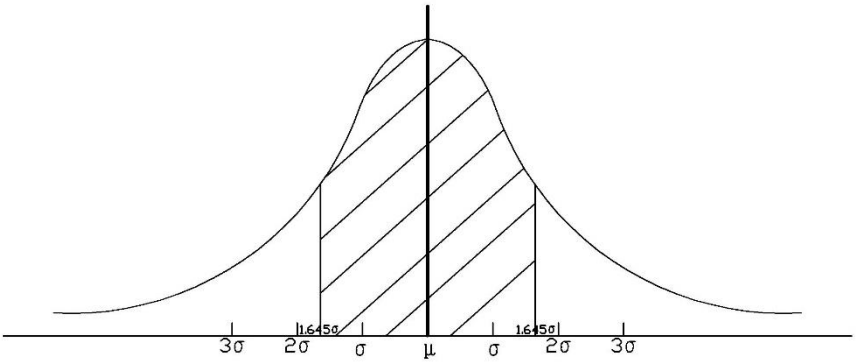

图 6 正态分布下先验概率结果示意图

Step3.以标准差σ、期望μ和可靠性为基础，通过查表得到单边显著性水平在 0.05 情况下的分位数，最终的计算结果是 1.645；

Step4.得到单边显著性水平在 0.05 情况下的分位数后，逐个计算每个种类疫苗的工期t_i，最终利用工期的公式，得到在可靠性为 90%前提下，安排生产任务，完成生产任务的最小天数t。

4.4.2.3 模型的解答

我们通过应用数学模型，结合题目所给的 10 种类型疫苗的生产任务和所有工位所有类型疫苗的生产时间的数据对模型进行求解。得到各类型疫苗的生产时间和总的工期长度结果如下图所示：

表 2 结果呈现图

疫苗类型	生产时间 *(min)*	疫苗类型	生产时间 *(min)*
1	18113.7076	6	29622.5738
2	9084.5382	7	32017.9142
3	11178.5646	8	13423.6051
4	16419.4460	9	8069.5593
5	14408.6627	10	13721.7620
总生产时间 *(min)*	166060.3336	生产工期 (天)	173

4.5 问题五

4.5.1 问题五的分析

问题五要求我们在给定每种疫苗订单上限，以及每种疫苗单价的情况下，通过合理的安排生产计划，使在 100 天内生产的疫苗的销售额最大。值得注意的是，在本题的疫苗生产中，只要求每种疫苗按序生产，不要求同种疫苗的每一阶段都为连续的，可以对疫苗生产进行适当的拆分。

生产每箱疫苗所需的时间服从概率分布，但是该时间无法事先确定，但由于疫苗的总产量较大，根据大数定律，我们可以用某种疫苗某一阶段每一箱所需的平均生产时间代替该种疫苗该阶段每一箱的生产时间。

我们利用线性规划方式并结合题目要求的约束性条件，求解计划天数和订单数量限

制下的生产计划分配中的疫苗类型和具体数量。再利用多目标规划模型对疫苗生产顺序分配后，对生产过程的生产总时间进行计算验证。若该分配情况下生产总时间满足题给要求的天数，则适当提高疫苗数量与种类，选取收益较大的疫苗类型，在满足计划天数内最大可能的提高疫苗销售额；若该分配不满足计划天数的要求，则适当降低疫苗种类和数量，削减收益较小的疫苗数量，在保证计划天数的时间限制范围内，尽可能地增大销售额。最后可以得到最终的生产计划。

4.5.2 问题五的解答

4.5.2.1 理论准备

接下来我们对模型中需要用到的专用名词做出解释：

1)切比雪夫大数定律

大数定律是指在用来描述稳定性的极限定理。在随机试验中，随着实验次数的增加，随机事件的频率会趋于一个常数，逐渐稳定，也就是这个随机事件的概率。

在切比雪夫大数定律中，假定有$X_1, X_2, \dots X_n$两两把相关的随机变量，且它们的方差和期望是σ_i^2, μ_i，若存在常数M，使得$\sigma_i^2 \le M$，则存在：

$$\frac{1}{n}(X_1 + \cdots + X_n) \xrightarrow{P} \frac{1}{n}(\mu_1 + \cdots + \mu_n) \tag{11}$$

当n趋向于正无穷时，我们可以用均值来体现它的属性。

因此我们利用大数定律的结论结合题目中所有工位所有类型疫苗的生产时间进行分析，用所有疫苗的平均生产时间$\overline{x'}$来带入模型进行计算，既简化了模型，又不会对模型的性质产生影响，能够高效地结合模型计算。

2)线性规划

线性规划其本质在于通过合理地利用优先的条件和材料，做出最优的决策，提供科学的依据。因此在本题中，我们考虑先通过线性规划和约束条件来求解出疫苗类型以及分配的具体产量的初始值。

如图 3 所示，在生产各类型疫苗的过程中，存在着四各工位同时工作的时间。因此我们我们的目的在于通过线性规划求出四个工位同时工作时刻的最佳分配方案，再通过对于首尾四个工位不在工作的时间段进行调整，得到销售额。

3)多目标规划

多目标规划通过约束条件，明确目标的优先顺序，求解出尽可能满足第一目标的解，若多个解满足第一目标的解值相同，则退而求其次，选择最大程度满足第二目标的解，以此类推。我们考虑通过多目标规划来计算产量不一样的疫苗种类分配情况下的生产顺序问题，确定四个工位的先后目标顺序，求出最大销售额条件下的疫苗种类及其产量的分配方案。

4.5.2.2 模型的建立

我们根据题目要求和问题的分析，通过已知条件确定目标函数，并通过线性规划相关公式求解分配的各类型疫苗及其产量，利用已知模型求得生产顺序，并适当补充、调整使结果满足限制条件，为模型的解答奠定理论基础。

1)目标函数的确定

我们根据题目生产任务可以拆分且工位生产时间不能超过 16 小时的限制要求，建立了以最大销售额为目标的目标函数，目标函数的具体参数如下：

$$\max B = \sum_{i=1}^{10} P_i x_i \tag{12}$$

其中x_i表示第i种疫苗的分配产量，P_i表示第i种疫苗的单价，B表示该疫苗分配产量与类型的情况下，疫苗公司的销售额。

2)约束条件的确定

根据题目的要求和附件中给出的疫苗生产任务的订单信息，我们可以知道对于目标函数而言，存在了许多约束条件。因此为了方便模型的求解，我们对约束条件进行了定量化的表示和处理，如下：

$$\begin{cases} x_i \le L_i \\ \sum_{i=1}^{10} x_i t_{ij} \le 100 \\ T_{mn} \le 16 \end{cases} \tag{13}$$

其中L_i表示第i种疫苗的订单上限额度，t_{ij}表示第i种疫苗第j个工位阶段生产所需要的时间，T_{mn}表示第m天第n个工位生产疫苗所花费的总时间。

这三个约束条件分别表达了，疫苗类型的生产限度不能超过订单的需求量，公司生产疫苗的计划天数为 100 天以及每个工位每天的生产时间不能超过 16 小时，这三种含义。

3)多目标规划的生产顺序计算

在计算出疫苗分配类型与产量后，在此基础上对生产疫苗的具体分配产量方案进行计算。利用多目标规划的方案，对所有分配方案进行生产顺序上的计算，明确 CJ4 工位剩余生产时间最小为第一目标，CJ3 工位为第二目标，CJ2 工位为第三目标，CJ1 工位的剩余生产时间恒为 0。因为，CJ1 工位可以不间断地补充疫苗进行生产。求解这种疫苗分配类型与产量情况下的最优生产顺序，尽可能地缩短生产总时间，提高生产效率，增大销售额。

4)最终解的调整

通过在多目标规划生产顺序最优模型的求解下，得到该疫苗分配类型与产量情况下的生产时间，运用约束条件中的计划生产天数限制，来调整疫苗最终的种类及其相应分配的产量。

4.5.2.3 模型的求解

我们通过对问题进行分析，并结合已知条件得到模型的目标函数，利用约束条件种的相关限制性定量化公式，结合线性规划的方法，求解出一个满足最大销售额的疫苗类型及生产数量的分配方案。

Step1.根据题目要求和附件给出的限制因素，我们将其转化为目标函数以及相对应的量化后的约束条件，利用线性规划的方法，求解出一个疫苗类型及数量的初始分配方案；

Step2.运用多目标规划方式，对第一步中求出的所有疫苗类型及产量做生产顺序的分配计算。计算出在不考虑工位之间生产顺序关系和计划天数条件的情况下，生产所有疫苗产量后各工位的剩余时间。以模型中明确的工位目标优先度为依据。寻找在该种疫苗类型及产量的合理配置下，实现总的生产时间最短的生产顺序方案，能够在要求的计划天数内生产出更多的疫苗，销售额更高；

Step3.将第二步中利用多目标规划方式计算出的总的最短生产时间与约束条件中的计划生产天数进行对比，判断是否满足计划天数的要求。

a)若最短生产时间小于计划天数，则可以在剩余的时间内尽可能地调整生产方案，安排销售额与生产时间比例效益较高，单位时间内收益较大的疫苗进行生产，使得最终销售额最大；

b)若最终的总最短生产时间大于计划天数，则应当适当地调整生产方案，减小部分疫苗种类的生产数量，选择性地降低单位生产时间内收益较低的疫苗的数量，使得最终销售额尽可能大；

Step4.对第三步结束之后，最终得到的生产顺序和疫苗生产类型及其响应的产量进行验证计算，保证最短总生产时间能够满足计划天数的要求，并确保最终的疫苗安排生产方案能够实现总销售额最大化。

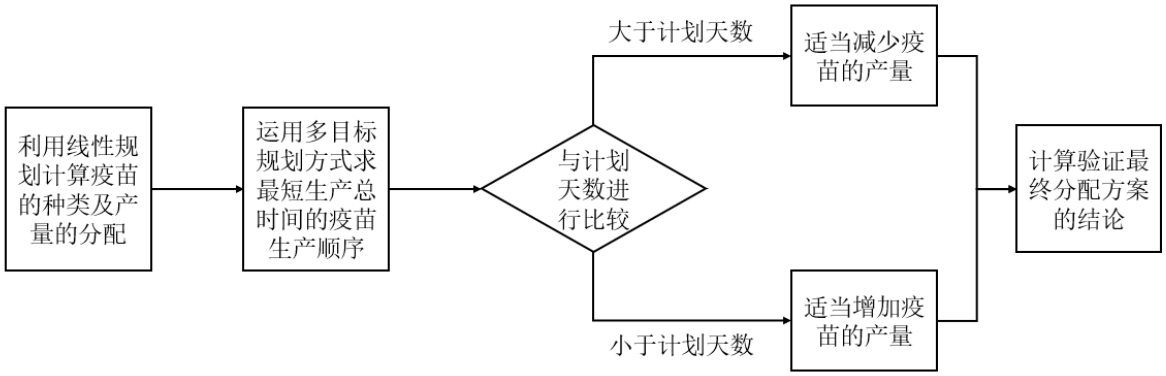

图 7 正态分布下先验概率结果示意图

4.5.2.4 模型的结果与分析

1).结果展示

根据题目要求，通过问题一和附件中的数据，利用疫苗产品订单需求以及所有工位所有类型疫苗的平均生产时间信息，用 Dev C++软件计算出总销售额最大化情况下的生产安排方案，以验证模型的正确性。

表 3 结果呈现表

疫苗类型	分配数量(剂)	销售额(美元)
YM1	0	0
YM2	50000	2700000
YM3	60000	3000000
YM4	0	0
YM5	120000	5040000
YM6	154600	6957000
YM7	49100	2356800
YM8	80000	4080000
YM9	60000	2760000
YM10	90000	4320000
销售总金额(美元)		31213800

2).结果分析

我们考虑到分配过程与多目标规划中的目标优先级的关系，做出了如下所示的图表来进一步说明我们的结论，表中每一箱疫苗中含有 100 剂的数量。

表 4 分配过程呈现表

疫苗类型	第一工位每箱生产时间(min)	第二工位每箱生产时间(min)	第三工位每箱生产时间(min)	第四工位每箱生产时间(min)	每箱疫苗单价(美元/箱)	单价与第一目标的收益率	生产箱数
1	13.2840	14.9621	19.8460	20.0129	4200	209.8646	0
2	9.8709	19.9075	17.9282	18.9424	5400	285.0748	500
3	20.0584	15.9726	14.9704	15.1164	5000	330.7666	600
4	7.9887	9.9366	5.9359	18.1284	4300	237.1969	0
5	8.7701	13.7220	13.0052	11.2495	4200	373.3499	1200
6	19.0741	20.0944	14.1485	13.7601	4500	327.0325	1546
7	11.1601	16.4961	12.0137	19.0876	4800	251.4722	492
8	16.0201	8.7289	17.9794	16.8314	5100	303.0051	800
9	15.0146	12.0351	7.0419	8.9497	4600	513.9837	600
10	12.9822	7.0110	9.0492	16.0524	4800	299.0207	900
时间	134.2232	138.8663	131.9184	158.1308	/		

我们通过比对图表中四个工位的总生产时间，可以看出 CJ4 工位生产疫苗的时间最紧张，其是多目标规划中的第一目标。因此，我们通过比对各类疫苗单价与 CJ4 工位的时间的比值，即收益率，计算各类型疫苗的收益率。收益率越高，则选取该种类疫苗所获得的销售额越大。通过目标优先度、生产任务上限和收益率的比对以及模型中最后部分的适当调整、补充生产疫苗，我们最后就可以得到最终的各类型疫苗的生产剂数。

五、模型的综合评价和推广

5.1 模型的综合评价

5.1.1 模型的优点

本文计算所得答案皆为全局最优解，考虑周全；

本文所建模型既符合题目要求，亦符合现实情况，对实际生产具有一定知道作用；

本文使用的代码精简，运行效率高，可以较快获得模型的解。

5.1.2 模型的缺点分析

本文未考虑工人换班时的衔接所造成的时间浪费，过于理想，和实际情况有所偏离；

本文未考虑生产出的次品对总时间的影响。

5.2 模型的推广

该题目所探讨、研究的问题与模型具有较强的普适性。我们的模型在实际流水线中能够规划出时间最短的生产方案，对提高生产效率有一点的指导价值。

问题五的模型能以最大销售额为目标，求解出最优的生产方案，对于提高企业效益有重要作用。

整个数学模型和代码计算过程体系完整、严密，逻辑性强，对于物流运输的研究具有重要的作用和参考价值，满足当今社会的发展趋势，能够为工业生产的高效发展和进步贡献一定的力量。

六、参考文献

[1]王深,吕连宏,张保留,王斯一,吴静,付加锋,罗宏.基于多目标模型的中国低成本碳达峰碳中和路径研究[J/OL].环境科学研究

[2]谭昕玥,许忠好.应用曲线积分求解条件概率密度函数[J].高等数学研究,2021,24(04):45-48.

附录

运行环境：
Python 环境：Latest Python 3 Release - Python 3.9.5 (anaconda notebook)
C++ 版本：TDM-GCC 4.9.2 64-bit Release
操作系统: Microsoft Windows 10 家庭中文版 Version 10.0 (Build 17134)
Java 版本: Java 1.7.0_60-b19 with Oracle Corporation Java HotSpot(TM) 64-Bit Server VM mixed mode
Lingo 版本: LINDO API Version 13.0 Copyright (c) 2000-2020 Id: lindo.h 3254 2020-07-20 14:50:57Z svn

附录 1：KS 检验.py
此代码用于进行 KS 检验。

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import numpy as np
import pandas as pd
data = pd.read_excel('data2.xlsx', index_col=1, header=0, parse_dates=True)
data.drop(data.columns[0], axis=1, inplace=True)


# In[2]:


data_np=np.array(data)
data_np


#  已经转为 np 类型了。

# In[10]:


import copy
import scipy.stats as stats
from scipy.stats import ks_2samp


# In[12]:
```

```
ans1=np.zeros(120).reshape(3,40)
for i in range(40):
        slct=data_np[:,i]
        slct=np.sort(slct)
        M=np.max(slct)
        m=np.min(slct)
        seq=np.arange(m,M,(M-m)/50)
        print(ks_2samp(slct,seq) )



# In[26]:


ans2=pd.DataFrame(ans1)
ans2.to_excel('EVEN.xlsx')


# In[ ]:
```

附录 2：SW 检验.py
此代码用于进行 SW 检验。

```
#!/usr/bin/env python
# coding: utf-8

# In[53]:


import numpy as np
import pandas as pd
data = pd.read_excel('data2.xlsx', index_col=1, header=0, parse_dates=True)
data.drop(data.columns[0], axis=1, inplace=True)


# In[54]:


data_np=np.array(data)
data_np.shape
```

```
# 已经转为 np 类型了。

# In[55]:


import copy
import scipy.stats as stats


# In[60]:


ans1=np.zeros(120).reshape(3,40)
for i in range(40):
        slct=data_np[i,:]
    sta,pvalue=stats.shapiro(slct)
        ans1[0,i]=sta
        ans1[1,i]=pvalue
        if (pvalue>=0.05):
                ans1[2,i]=1
        else:
                ans1[2,i]=0


# In[62]:


ans2=pd.DataFrame(ans1)
ans2.to_excel('Gauss_dealed.xlsx')


# In[58]:


print(stats.shapiro(slct))
```

附录 3：t3_data_generate.py
此代码用于生成第三题中的随机化数据。

```
#!/usr/bin/env python
```

```
# coding: utf-8

# In[1]:


import numpy as np
import pandas as pd
import copy
import scipy.stats as stats


# In[4]:


data = pd.read_excel('t3_in.xlsx', index_col=0, header=0, parse_dates=True)


# In[6]:


data_np=np.array(data)
data_np.shape


# In[16]:


import math


# In[15]:


import random


# In[31]:


tar=np.arange(1,100001,1).reshape(100000,1)


# In[32]:
```

```
for i in range(40):
      tmp=np.random.normal(loc=data_np[i,0],scale=math.sqrt(   data_np[i,1]   ),size
=(100000,1) )
      tar=np.concatenate ([tar,tmp],axis=1)


# In[33]:


ans2=pd.DataFrame(tar)
ans2.to_excel('t3_data_generate.xlsx')


# In[ ]:
```

附录 4：t2.cpp
用于第二题的计算。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int INF=0x3f3f3f3f;
const long long mod=100000007;
const double e=2.718281828459045;
const double pi=3.1415926535;
#define CK cout<<"OK\n";
double t[11][5];
double minT;      //记录最短时间
vector<int>v;           //记录当前的序列
vector<int>minV;      //记录最小时间所对应的序列
bool u[11];  //是否可以用：可用 1，已用 0
int tmp;

void init(){
      minV.clear();
      v.clear();
      minT=45000;
      for(int i=0;i<=10;i++)u[i]=1;
      tmp=0;
}
```

```
double test(){
    double et[5][11];
    //et[i][j]表示第 i 阶段  第 j 种疫苗的结束时间。
    for(int j=0;j<=10;j++)et[0][j]=0;
    for(int i=1;i<=4;i++)et[i][0]=0;
    for(int i=1;i<=4;i++){
        for(int j=1;j<=10;j++){
            et[i][j]=t[ v[j-1] ][i] + max( et[i][j-1] , et[i-1][j] );
        }
    }
    return et[4][10];
}

void OUT(vector<int>tmpv){
    for(int i=0;i<v.size();i++){
        printf("%d ",tmpv[i]);
    }
    printf("\n");
}

void DO(){
    if(v.size()==10){
        double currentT=test();
        if(currentT<minT){
            minT=currentT;
            minV=v;
        }
        tmp++;
        //OUT(v);
        //tmp 为测试用数据

        return;
    }
    for(int i=1;i<=10;i++){
        if(u[i]){
            u[i]=0;
            v.push_back(i);
            DO();
            v.pop_back();
            u[i]=1;
        }
    }
}
```

```
int main(){
	init();
	freopen("t2_in.txt","r",stdin);
	freopen("t2_out.txt","w",stdout);
	for(int i=1;i<=10;i++){
		for(int j=1;j<=4;j++)scanf("%lf",&t[i][j]);
	}
	DO();
	printf("%.4lf\n",minT);
	for(int i=1;i<=10;i++){
		printf("%d\t",minV[i-1]);
	}
	return 0;
}
```

附录 5：t2_输出结果.cpp
用于显示第二题的结果。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int INF=0x3f3f3f3f;
const long long mod=100000007;
const double e=2.718281828459045;
const double pi=3.1415926535;
#define CK cout<<"OK\n";
double t[11][5];
double minT;	//记录最短时间
vector<int>v;		//记录当前的序列
vector<int>minV;	//记录最小时间所对应的序列
bool u[11];	//是否可以用：可用 1，已用 0
int tmp;

void init(){
	minV.clear();
	v.clear();
	minT=45000;
	for(int i=0;i<=10;i++)u[i]=1;
	tmp=0;
}

double test(){
	double et[5][11];
```

```
    //et[i][j]表示第 i 阶段  第 j 种疫苗的结束时间。
    for(int j=0;j<=10;j++)et[0][j]=0;
    for(int i=1;i<=4;i++)et[i][0]=0;
    for(int i=1;i<=4;i++){
        for(int j=1;j<=10;j++){
            et[i][j]=t[ v[j-1] ][i] + max( et[i][j-1] , et[i-1][j] );
        }
    }
    return et[4][10];
}

void OUT(vector<int>tmpv){
    for(int i=0;i<v.size();i++){
        printf("%d ",tmpv[i]);
    }
    printf("\n");
}

void DO(){
    if(v.size()==10){
        double currentT=test();
        if(currentT<minT){
            minT=currentT;
            minV=v;
        }
        tmp++;
        //OUT(v);
        //tmp 为测试用数据

        return;
    }
    for(int i=1;i<=10;i++){
        if(u[i]){
            u[i]=0;
            v.push_back(i);
            DO();
            v.pop_back();
            u[i]=1;
        }
    }
}

int main(){
    init();
```

```
    freopen("t2_in.txt","r",stdin);
    freopen("t2_out_result.txt","w",stdout);
    for(int i=1;i<=10;i++){
        for(int j=1;j<=4;j++)scanf("%lf",&t[i][j]);
    }
    DO();
    printf("总时间:\n%.4lf\n 顺序：\n",minT);
    for(int i=1;i<=10;i++){
        printf("%d\t",minV[i-1]);
    }
    printf("\n");
    printf("编号\t 开始\t 结束\n");
    double Et[5][11];
    //Et[i][j]表示第 i 阶段  第 j 种疫苗的结束时间。
    for(int j=0;j<=10;j++)Et[0][j]=0;
    for(int i=1;i<=4;i++)Et[i][0]=0;
    for(int i=1;i<=4;i++){
        for(int j=1;j<=10;j++){
            Et[i][j]=t[ minV[j-1] ][i] + max( Et[i][j-1] , Et[i-1][j] );
        }
    }
    for(int j=0;j<10;j++){
        printf("%d\t%.4lf\t%.4lf\n",  minV[j]  ,  Et[1][j+1]-t[  minV[j]  ][1]  ,
Et[4][j+1] );
    }

    return 0;
}
```

附录 6：t3_data1.cpp
用于从第三题生成的数剧中取出一部分并计算。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int INF=0x3f3f3f3f;
const long long mod=100000007;
const double e=2.718281828459045;
const double pi=3.1415926535;
#define CK cout<<"OK\n";
double t[11][5];
double minT;     //记录最短时间
vector<int>v;          //记录当前的序列
vector<int>minV;     //记录最小时间所对应的序列
```

```
bool u[11];  //是否可以用：可用 1，已用 0
int tmp;

void init(){
    minV.clear();
    v.clear();
    minT=45000;
    for(int i=0;i<=10;i++)u[i]=1;
    tmp=0;
}

double test(){
    double et[5][11];
    //et[i][j]表示第 i 阶段  第 j 种疫苗的结束时间。
    for(int j=0;j<=10;j++)et[0][j]=0;
    for(int i=1;i<=4;i++)et[i][0]=0;
    for(int i=1;i<=4;i++){
        for(int j=1;j<=10;j++){
            et[i][j]=t[ v[j-1] ][i] + max( et[i][j-1] , et[i-1][j] );
        }
    }
    return et[4][10];
}

void OUT(vector<int>tmpv){
    for(int i=0;i<v.size();i++){
        printf("%d ",tmpv[i]);
    }
    printf("\n");
}

void DO(){
    if(v.size()==10){
        double currentT=test();
        if(currentT<minT){
            minT=currentT;
            minV=v;
        }
        tmp++;
        //OUT(v);
        //tmp 为测试用数据

        return;
    }
```

```
    for(int i=1;i<=10;i++){
        if(u[i]){
            u[i]=0;
            v.push_back(i);
            DO();
            v.pop_back();
            u[i]=1;
        }
    }
}

int main(){
    freopen("t3_data1_in.txt","r",stdin);
    freopen("t3_data1_out.txt","w",stdout);
    int SS;
    scanf("%d",&SS);
    for(int I=0;I<100;I++){
        init();
        for(int i=1;i<=10;i++)for(int j=1;j<=4;j++)scanf("%lf",&t[i][j]);
        DO();
        printf("%.4lf\n",minT);
    }
    return 0;
}
```

附录 7：t3_data1_analyse.cpp
用于第三题分析

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int INF=0x3f3f3f3f;
const long long mod=100000007;
const double e=2.718281828459045;
const double pi=3.1415926535;
#define CK cout<<"OK\n";
int a[200];
int main(){
    int N=0;
    freopen("t3_data1_out.txt","r",stdin);
    freopen("t3_data1_analyse.txt","w",stdout);
    double t;
    memset(a,0,sizeof(a));
    while(~scanf("%lf",&t)){
```

```
            if(t<177)t=177;
            if(t>191)t=191;
            int s=t;
        //  cout<<s<<endl;
            a[s]++;
        }
        for(int i=177;i<=191;i++){
            printf("%d\n",a[i]);
        }
        return 0;
}
```

附录 8：t4.cpp
用于第四题的计算。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int INF=0x3f3f3f3f;
const long long mod=100000007;
const double e=2.718281828459045;
const double pi=3.1415926535;
#define CK cout<<"OK\n";
double sum[10];       //该种疫苗所需时间
double total;       //  总时间
double t[10][4];
double quan[10];

void init(){
    memset(sum,0,sizeof(sum));
    memset(t,0,sizeof(t));
    memset(quan,0,sizeof(quan));
    total=0;

}

int main(){
    init();
    freopen("t4_in.txt","r",stdin);
    freopen("t4_out2.txt","w",stdout);
    for(int i=0;i<10;i++)scanf("%lf",&quan[i]);
    for(int i=0;i<10;i++){
        for(int j=0;j<4;j++)scanf("%lf",&t[i][j]);
        sum[i]=max(
```

```
                max(
                    t[i][3]+t[i][2]+t[i][1]+quan[i]*t[i][0],
                    t[i][3]+t[i][2]+quan[i]*t[i][1]+t[i][0]
                ),
                max(
                    t[i][3]+quan[i]*t[i][2]+t[i][1]+t[i][0],
                    quan[i]*t[i][3]+t[i][2]+t[i][1]+t[i][0]
                )
            );
            total+=sum[i];
            printf("%.4lf\n",sum[i]);
        }
        printf("共  %.4lf    分钟\n",total);
        return 0;
}
```

附录 9：t5.cpp
用于编辑第五题的 lingo 程序。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int INF=0x3f3f3f3f;
const long long mod=100000007;
const double e=2.718281828459045;
const double pi=3.1415926535;
#define CK cout<<"OK\n";
int main(){
    freopen("t5_in.txt","r",stdin);
    freopen("t5_out.txt","w",stdout);
    double a[11][5];
    for(int i=1;i<=10;i++){
        for(int j=1;j<=4;j++)scanf("%lf",&a[i][j]);
    }
    for(int j=1;j<=4;j++){
        for(int i=1;i<=10;i++){
            printf("+%.4lfx%d",a[i][j],i);
        }
        printf("<=96000\n");
    }
    return 0;
}
```

附录 10：t5_2.cpp
用于编辑 lingo 的输出（第五题）。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int INF=0x3f3f3f3f;
const long long mod=100000007;
const double e=2.718281828459045;
const double pi=3.1415926535;
#define CK cout<<"OK\n";
double q[11];
double sum[11];
int main(){
    freopen("t5_2_in.txt","r",stdin);
    freopen("t5_2_out.txt","w",stdout);
    double a[11][5];
    for(int i=1;i<=10;i++)scanf("%lf",&q[i]);
    for(int i=1;i<=10;i++){
        for(int j=1;j<=4;j++)scanf("%lf",&a[i][j]);
    }
    for(int j=1;j<=4;j++){
        for(int i=1;i<=10;i++){
            sum[j]+=q[i]*a[i][j];
        }
        printf("第 %d 个工位剩余时间： %.4lf\n",j,96000-sum[j]);
    }
    return 0;
}
```

附录 11：t5_3_结果分析.cpp
用于第五题结果分析。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int INF=0x3f3f3f3f;
const long long mod=100000007;
const double e=2.718281828459045;
const double pi=3.1415926535;
#define CK cout<<"OK\n";
double q[11];
double sum[11];
int main(){
    freopen("t5_2_in.txt","r",stdin);
    freopen("t5_3_out.txt","w",stdout);
```

```
    double a[11][5];
    for(int i=1;i<=10;i++)scanf("%lf",&q[i]);
    for(int i=1;i<=10;i++){
        for(int j=1;j<=4;j++)scanf("%lf",&a[i][j]);
    }
    for(int j=1;j<=4;j++){
        for(int i=1;i<=10;i++){
            printf("%.4lf\t",a[i][j]);
        }

        printf("\n");
    }
    return 0;
}
```

附录 12：t5_4.cpp

用于在第五题中计算剩余时间，以便调整方案。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int INF=0x3f3f3f3f;
const long long mod=100000007;
const double e=2.718281828459045;
const double pi=3.1415926535;
#define CK cout<<"OK\n";
double q[11];
double sum[11];
int main(){
    freopen("t5_4_in.txt","r",stdin);
    freopen("t5_4_out.txt","w",stdout);
    double a[11][5];
    for(int i=1;i<=10;i++)scanf("%lf",&q[i]);
    for(int i=1;i<=10;i++){
        for(int j=1;j<=4;j++)scanf("%lf",&a[i][j]);
    }
    for(int j=1;j<=4;j++){
        for(int i=1;i<=10;i++){
            sum[j]+=q[i]*a[i][j];
        }
        printf("第 %d 个工位剩余时间： %.4lf\n",j,96000-sum[j]);
    }
    return 0;
}
```

附录 13：离群点处理.cpp
用于预处理。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
int main(){
	double a[50];
	double var=0,ave=0;
	freopen("input1.txt","r",stdin);
	freopen("output1.txt","w",stdout);

	for(int i=0;i<50;i++){
		scanf("%lf",&a[i]);
	}
	scanf("%lf%lf",&ave,&var);
	for(int i=0;i<50;i++){
		double dis=2*pow(var,0.5);
		if( abs(a[i]-ave)>dis )printf("%.4lf\t",ave);
		else printf("%.4lf\t",a[i]);
	}
	return 0;
}
```

附录 14：转换格式.cpp
用于转换附件中的数据格式。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
int main(){
	freopen("Trans_in.txt","r",stdin);
	freopen("Trans_out.txt","w",stdout);
	double a[55];
	for(int J=0;J<40;J++){
		for(int i=0;i<50;i++){
			cin>>a[i];
			printf("%.4lf\t",a[i]);
		}
		printf("\n");
	}
	return 0;
}
```

附录 15：t3_2.cpp
用于第三题数据处理。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int INF=0x3f3f3f3f;
const long long mod=100000007;
const double e=2.718281828459045;
const double pi=3.1415926535;
#define CK cout<<"OK\n";
int ind[50][11];
double ave[50];
double raw[11][5];
double aft[11][5];
double et[11][5];       //结束时间

double test(){
    //et[i][j]表示第 i 阶段  第 j 种疫苗的结束时间。
    for(int i=0;i<=10;i++)et[i][0]=0;
    for(int j=1;j<=4;j++)et[0][j]=0;
    for(int i=1;i<=10;i++){
        for(int j=1;j<=4;j++){
            et[i][j]=aft[i][j] + max( et[i][j-1] , et[i-1][j] );
        }
    }
    return et[10][4];
}

int main(){
    freopen("t3_2_in.txt","r",stdin);
    freopen("t3_2_out.txt","w",stdout);
    int r,N;
    scanf("%d",&r);
    for(int i=0;i<r;i++){
        for(int j=1;j<=10;j++)scanf("%d",&ind[i][j]);
        ave[i]=0;
    }
    //CK
    scanf("%d",&N);
    for(int I=0;I<N;I++){//对于每一组数据
        for(int i=1;i<=10;i++){
            for(int j=1;j<=4;j++){
                scanf("%lf",&raw[i][j]);
```

```
                //    printf("%.1lf\t",raw[i][j]);
                }
            //    printf("\n");
            }
//          CK
            for(int k=0;k<r;k++){       //测试每一种排列
                for(int i=1;i<=10;i++){
            //          cout<<ind[k][i]<<endl;
                    for(int j=1;j<=4;j++){
                        aft[ ind[k][i] ][j]=raw[ i ][j];

                    }

                }
                //CK
                double ans=test();
            //    cout<<ans;
                ave[k]+=ans/N;
            }
        }
        double mina=1000;
        int min_ind=0;
        for(int k=0;k<r;k++){
            printf("%.4lf\n",ave[k]);
            if(mina>ave[k]){
                mina=ave[k];
                min_ind=k;
            }
        }

        printf("\nmin:%.4lf\n",mina);
        for(int i=1;i<=10;i++){
            printf("%d\t",ind[min_ind][i]);
        }

        return 0;
}
```

附录 16：t3_3.cpp
用于测试第三题结果。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
```

```
const int INF=0x3f3f3f3f;
const long long mod=100000007;
const double e=2.718281828459045;
const double pi=3.1415926535;
#define CK cout<<"OK\n";
int ind[11];
double ans[6050];
double raw[11][5];
double aft[11][5];
double et[11][5];       //结束时间

double test(){
    //et[i][j]表示第 i 阶段  第 j 种疫苗的结束时间。
    for(int i=0;i<=10;i++)et[i][0]=0;
    for(int j=1;j<=4;j++)et[0][j]=0;
    for(int i=1;i<=10;i++){
        for(int j=1;j<=4;j++){
            et[i][j]=aft[i][j] + max( et[i][j-1] , et[i-1][j] );
        }
    }
    return et[10][4];
}

int main(){
    memset(ans,0,sizeof(ans));
    freopen("t3_3_in.txt","r",stdin);
    freopen("t3_3_out.txt","w",stdout);
    int r,N;
    for(int j=1;j<=10;j++)scanf("%d",&ind[j]);

    //CK
    scanf("%d",&N);
    for(int I=0;I<N;I++){//对于每一组数据
        for(int i=1;i<=10;i++){
            for(int j=1;j<=4;j++){
                scanf("%lf",&raw[i][j]);
            }
        }
        for(int i=1;i<=10;i++){
            for(int j=1;j<=4;j++){
                aft[ ind[i] ][j]=raw[ i ][j];
            }
        }
        ans[I]=test();
```

```
        printf("%.4lf\n",ans[I]);
    }


    return 0;
}
```

附录 17：t3_data1_check_result.cpp
用于检验第三题。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int INF=0x3f3f3f3f;
const long long mod=100000007;
const double e=2.718281828459045;
const double pi=3.1415926535;
#define CK cout<<"OK\n";
double t[11][5];
double minT;     //记录最短时间
vector<int>v;          //记录当前的序列
vector<int>minV;     //记录最小时间所对应的序列
bool u[11];  //是否可以用：可用 1，已用 0
int tmp;

void init(){
    minV.clear();
    v.clear();
    minT=45000;
    for(int i=0;i<=10;i++)u[i]=1;
    tmp=0;
}

double test(){
    double et[5][11];
    //et[i][j]表示第 i 阶段  第 j 种疫苗的结束时间。
    for(int j=0;j<=10;j++)et[0][j]=0;
    for(int i=1;i<=4;i++)et[i][0]=0;
    for(int i=1;i<=4;i++){
        for(int j=1;j<=10;j++){
            et[i][j]=t[ v[j-1] ][i] + max( et[i][j-1] , et[i-1][j] );
        }
    }
    return et[4][10];
```

```
}

void OUT(vector<int>tmpv){
	for(int i=0;i<tmpv.size();i++){
		printf("%d ",tmpv[i]);
	}
	printf("\n");
}

void DO(){
	if(v.size()==10){
		double currentT=test();
		if(currentT<minT){
			minT=currentT;
			minV=v;
		}
		tmp++;
		//OUT(v);
		//tmp 为测试用数据

		return;
	}
	for(int i=1;i<=10;i++){
		if(u[i]){
			u[i]=0;
			v.push_back(i);
			DO();
			v.pop_back();
			u[i]=1;
		}
	}
}

int main(){
	freopen("t3_data1_in.txt","r",stdin);
	freopen("t3_data1_out_check_result.txt","w",stdout);
	int SS;
	scanf("%d",&SS);
	for(int I=0;I<30;I++){
		init();
		for(int i=1;i<=10;i++)for(int j=1;j<=4;j++)scanf("%lf",&t[i][j]);
		DO();
		printf("%.4lf\n",minT);
		OUT(minV);
```

```
    }
    return 0;
}
```

附录 18：预处理后的数据

疫苗类型 The vaccine type	生产工位 Production location	1	2	3	4	5	6	7	8
YM1	CJ1	13.5377	11.6923	11.6501	12.795	13.6715	14.0347	13.8884	14.4384
YM1	CJ2	14.1363	16.5326	13.9109	15.0859	14.3844	13.5977	16.4193	15.6966
YM1	CJ3	20.8404	19.3997	17.8616	20.124	22.908	19.7275	19.6462	20.0335
YM1	CJ4	20.7801	19.8002	19.5594	20.4613	18.002	20.6775	17.9575	21.5586
YM2	CJ1	10.1832	10.5152	9.468	8.8258	8.9358	9.5554	10.3919	9.6794
YM2	CJ2	19.2658	19.7635	21.0001	18.3298	20.3271	19.0556	20.9111	20.2398
YM2	CJ3	18.0799	17.3088	18.8979	17.4954	16.9851	18.3999	17.3709	17.4393
YM2	CJ4	19.881	18.3955	18.5323	19.3086	17.5306	18.0953	21.4245	20.8779
YM3	CJ1	20.2696	20.1097	21.1741	20.8186	19.507	20.1093	20.5265	18.673
YM3	CJ2	15.9372	14.8813	15.3593	15.2764	15.6269	16.4128	16.1769	15.802
YM3	CJ3	14.9201	15.44	13.8593	15.5413	13.7167	15.0355	15.2458	13.6571
YM3	CJ4	14.5619	14.4684	14.586	15.6489	16.0289	13.7629	15.4136	14.1812
YM4	CJ1	8.4759	7.4903	9.0341	8.2445	8.9239	7.5836	8.0645	7.0605
YM4	CJ2	8.5173	8.3727	12.0243	10.3179	10.6474	11.0635	9.2422	9.5907
YM4	CJ3	6.3817	6.138	5.7168	6.1559	6.0829	5.897	5.9654	5.7147
YM4	CJ4	19.46	17.4073	18.4207	18.9704	16.8067	16.4495	19.0533	20.7891
YM5	CJ1	9.5411	9.412	9.8535	9.4772	8.5901	8.4565	9.9758	9.5409
YM5	CJ2	12.5807	11.7967	12.8777	12.7706	13.5902	12.5331	14.257	12.3067
YM5	CJ3	13.4364	11.9632	13.1807	12.2255	14.7985	13.7653	12.2397	13.2147
YM5	CJ4	10.7494	12.7447	10.6988	11.8836	10.4862	11.2877	9.4434	11.551
YM6	CJ1	19.6737	18.2219	18.6481	20.0393	19.0964	19.8368	20.3025	19.2268
YM6	CJ2	21.1665	19.6182	20.0675	20.1946	21.4089	19.1024	20.5275	20.353
YM6	CJ3	13.9583	14.2053	12.3636	13.4632	15.3094	14.7304	14.5745	14.7932
YM6	CJ4	11.9716	14.617	13.5023	13.5178	14.2087	13.8839	14.553	13.7211
YM7	CJ1	12.3419	10.3813	11.6329	11.9493	12.6894	11.3847	10.2103	9.9248
YM7	CJ2	16.321	16.1944	17.771	17.5826	14.6801	15.9724	16.1568	18.1257
YM7	CJ3	11.5951	11.6962	12.397	12.5684	11.7808	12.1173	12.4583	11.7545
YM7	CJ4	18.4259	19.7495	19.7768	19.0049	18.8198	19.1478	19.1789	18.5797
YM8	CJ1	14.4856	16.4488	16.8598	17.1458	17.4254	14.0555	16.0115	17.0458
YM8	CJ2	8.6982	8.4936	8.935	8.2935	8.3998	8.2069	8.7126	8.8275
YM8	CJ3	16.3152	17.4721	19.6865	18.968	19.3769	17.3752	17.1167	17.7175
YM8	CJ4	17.4274	17.0188	17.8254	17.0939	18.1929	17.523	17.3792	17.3276
YM9	CJ1	12.8254	15.2918	15.8352	15.4948	14.6064	15.7402	14.746	13.0461

YM9	CJ2	12.7973	12.1674	13.1654	11.4488	10.9958	10.3646	10.4562	13.3162
YM9	CJ3	6.6825	6.9448	6.5514	7.4019	6.6919	6.8599	7.3434	6.3337
YM9	CJ4	9.113	8.9394	9.6333	9.4329	8.9738	8.8978	8.4983	8.354
YM10	CJ1	12.7739	12.748	12.9133	13.0979	13.138	12.4625	13.5046	13.3015
YM10	CJ2	6.71	7.5018	7.4571	7.1321	7.1059	6.3132	6.5245	7.136
YM10	CJ3	9.7264	8.6719	9.3712	9.1924	8.7475	9.4649	9.8025	9.2004
YM10	CJ4	15.5279	15.6108	15.6592	16.2751	16.3192	16.5168	15.8662	17.4872

疫苗类型 The vaccine type	生产工位 Production location	9	10	11	12	13	14	15	16
YM1	CJ1	12.8978	12.9699	14.8339	12.5664	16.0349	12.8759	11.7925	13.7269
YM1	CJ2	13.852	15.1873	15.0774	14.2303	15.0326	13.5084	15.7481	13.5776
YM1	CJ3	20.0229	19.0208	19.112	20.49	19.1604	21.4367	20.8252	21.0984
YM1	CJ4	19.7072	21.2391	19.97	18.9282	18.7281	18.1143	16.5052	19.8046
YM2	CJ1	8.9333	8.4349	8.9702	10.2614	11.6821	9.8078	11.6035	9.8441
YM2	CJ2	19.9755	19.9292	19.9692	22.0237	18.3358	20.4716	21.0826	18.6782
YM2	CJ3	16.6019	19.5763	17.0515	18.4494	17.8681	16.7294	17.5289	17.07
YM2	CJ4	18.4417	18.7901	19.3232	19.1034	18.8751	18.7661	19.1922	18.7117
YM3	CJ1	20.8123	21.5163	20.4943	21.1287	20.1269	19.7074	19.8193	21.814
YM3	CJ2	15.3831	15.3914	13.978	15.3736	17.8089	15.4067	16.8155	15.013
YM3	CJ3	15.8998	15.6293	15.8985	15.1017	13.9067	15.3893	12.671	17.2272
YM3	CJ4	14.3102	15.884	14.5913	15.9726	14.5617	14.6399	16.458	12.8065
YM4	CJ1	7.0095	8.1102	9.4122	7.9971	8.2916	8.8084	8.2669	9.2247
YM4	CJ2	10.078	8.6147	9.9562	10.1663	7.6405	10.138	9.5744	11.1569
YM4	CJ3	5.8856	6.0061	6.0244	6.1112	6.0119	6.0879	6.0697	5.8207
YM4	CJ4	16.5755	18.4011	20.05	17.5302	18.4007	17.4314	18.647	18.1716
YM5	CJ1	10.3845	5.927	7.4591	9.4055	7.147	8.9287	8.2887	8.0881
YM5	CJ2	13.5853	13.3751	13.2706	13.4288	14.3062	13.729	we	12.3742
YM5	CJ3	14.6742	13.3891	12.4956	12.1429	14.2665	11.6067	12.8831	13.7783
YM5	CJ4	10.5771	12.5106	10.8101	9.8395	10.3013	11.4359	11.7964	11.0032
YM6	CJ1	19.1383	17.8723	18.3309	17.9364	19.2695	19.9109	18.1695	21.5383
YM6	CJ2	19.8834	21.1933	21.2128	20.4289	19.8129	20.2798	19.4659	19.2077
YM6	CJ3	13.6914	13.9817	13.3845	15.1929	14.0173	13.698	12.9553	14.4908
YM6	CJ4	14.4155	13.5958	13.5507	14.6127	13.8933	14.6474	13.3814	13.37
YM7	CJ1	11.5405	12.2901	10.0116	11.9345	9.541	11.7174	12.2823	11.6964
YM7	CJ2	15.9246	14.9944	17.6234	15.5851	16.2213	18.7292	15.7262	16.9239
YM7	CJ3	11.3804	12.6096	12.5279	11.9913	11.5172	10.794	11.1202	12.1743
YM7	CJ4	18.7081	18.1852	18.8048	18.4292	16.7402	19.4369	18.7922	18.6377
YM8	CJ1	16.7742	15.8897	17.0262	16.0006	16.0669	16.1812	15.1061	17.5239
YM8	CJ2	9.038	8.0488	8.6015	8.8807	8.7028	8.742	8.9903	8.8041

YM8	CJ3	16.7219	18.1085	18.4131	18.7231	17.6136	18.2701	16.5918	17.0664
YM8	CJ4	17.4411	16.7789	17.181	16.574	16.7707	16.0488	17.9536	17.6768
YM9	CJ1	16.0437	13.7432	15.1446	14.3854	16.0945	13.7842	12.8237	15.3097
YM9	CJ2	11.1456	11.7702	11.5575	11.5211	11.0913	11.9126	11.3673	10.0932
YM9	CJ3	7.0878	7.1803	7.2164	7.248	7.0986	6.2319	6.9971	7.648
YM9	CJ4	8.9571	8.9731	8.55	9.4651	8.9002	9.393	8.8977	9.286
YM10	CJ1	13.3347	13.4041	13.3917	12.388	13.3082	12.9244	13.2299	12.9524
YM10	CJ2	7.3321	7.4947	6.7193	7.7555	7.5538	8.5793	7.3066	7.4354
YM10	CJ3	8.5679	9.3034	9.6899	8.9375	9.5718	9.1628	8.762	9.451
YM10	CJ4	15.9467	17.0119	15.6644	16.5498	15.8699	16.9276	16.1857	16.2099

疫苗类型 The vaccine type	生产工位 Production location	17	18	19	20	21	22	23	24
YM1	CJ1	11.8529	13.3252	12.7586	12.8351	10.7412	13.3426	13.7254	14.4897
YM1	CJ2	15.2916	15.8351	15.1049	14.9175	13.7859	15.3714	15.5525	14.2577
YM1	CJ3	19.1764	18.6663	19.738	18.8436	20.1001	20.7394	21.3546	18.0391
YM1	CJ4	20.6825	18.3235	19.6736	22.2905	19.9478	22.0271	18.3198	18.7018
YM2	CJ1	8.7493	10.0125	10.9337	9.9155	10.9492	9.0585	9.1243	9.7259
YM2	CJ2	20.5946	19.3096	18.0512	17.5137	20.2323	17.7416	19.41	18.7872
YM2	CJ3	16.7962	20.1778	17.7449	17.5191	18.4115	18.1006	17.8528	17.6174
YM2	CJ4	19.9594	19.9407	18.6886	17.3011	18.2159	19.5632	20.479	17.943
YM3	CJ1	19.7397	18.559	20.5455	19.9674	18.5169	19.71	19.3432	19.4592
YM3	CJ2	15.6925	16.3277	16.2748	15.2629	15.0179	16.2495	14.9201	16.4013
YM3	CJ3	15.07	13.9678	14.6999	14.787	15.1837	17.7873	14.5664	15.7512
YM3	CJ4	14.4229	15.5197	14.3333	15.1803	15.9835	14.4777	17.0034	15.7059
YM4	CJ1	8.6003	7.9625	6.827	8.7871	8.0226	8.9199	7.2223	8.213
YM4	CJ2	9.436	9.8391	11.3244	10.3105	10.9608	10.3763	9.49	9.2893
YM4	CJ3	5.7583	5.7971	5.95	6.1706	6.2094	5.7759	5.9177	5.9821
YM4	CJ4	17.2511	18.7276	18.7174	18.9297	18.1205	18.8864	18.0951	18.81
YM5	CJ1	8.8431	7.7374	8.9373	9.6263	8.7969	8.6362	8.7927	8.0617
YM5	CJ2	13.0258	13.5506	15.9122	12.8313	15.1473	14.214	12.8277	13.1
YM5	CJ3	12.3064	15.0108	13.125	11.844	13.1021	12.8301	12.7488	12.6138
YM5	CJ4	12.9151	11.2942	9.9469	11.164	9.9671	13.3774	11.8328	11.8967
YM6	CJ1	20.4099	20.0989	17.0929	19.0782	18.5997	19.553	16.4356	18.7603
YM6	CJ2	20.8542	20.7173	20.5531	21.6321	20.4855	19.7009	20.2917	20.0512
YM6	CJ3	14.2818	13.1016	14.4567	14.4608	15.3142	13.1972	14.8284	15.8136
YM6	CJ4	12.9235	14.2452	13.3452	13.2742	14.236	14.2894	13.3122	12.9656
YM7	CJ1	12.423	10.909	9.5551	12.3412	12.8179	12.0559	10.4183	13.2878
YM7	CJ2	15.6995	16.054	15.5268	16.434	17.0624	16.3585	18.7304	16.3036
YM7	CJ3	13.2816	10.2193	11.2798	12.7823	10.993	12.5066	11.7685	12.4331

YM7	CJ4	19.9276	18.8461	19.4584	18.4658	18.9495	19.4942	18.4356	20.1301
YM8	CJ1	15.061	15.049	16.2657	16.2775	15.2419	15.2438	14.3606	16.0543
YM8	CJ2	8.6296	8.8275	8.5406	8.8275	9.1803	8.758	8.7781	8.4575
YM8	CJ3	18.0825	18.1144	17.4154	17.6597	18.5017	17.1501	17.4949	17.3656
YM8	CJ4	14.7471	18.5081	16.7952	16.4648	17.6575	14.966	17.2623	18.1726
YM9	CJ1	16.8918	15.5654	15.8022	13.964	15.669	14.9864	15.3212	16.5908
YM9	CJ2	10.4482	11.2035	12.3573	11.1729	11.3709	10.5873	12.2406	11.6522
YM9	CJ3	7.0781	6.8336	6.954	6.374	6.7764	6.9978	7.6244	7.2502
YM9	CJ4	8.5309	9.8984	8.3148	8.5994	8.6044	8.9272	9.5176	8.9554
YM10	CJ1	13.0255	12.2419	13.3416	13.2895	13.6186	13.0593	13.432	13.4464
YM10	CJ2	6.5446	6.4157	6.6489	6.6556	6.2221	6.4314	7.4102	7.6133
YM10	CJ3	9.6746	9.3695	8.8117	9.8172	9.0476	8.7348	8.5426	9.6482
YM10	CJ4	16.0933	15.6887	15.8924	16.3889	16.2883	15.5722	15.8856	15.8614

疫苗类型 The vaccine type	生产工位 Production location	25	26	27	28	29	30	31	32
YM1	CJ1	13.7172	12.6966	11.9311	12.2451	13.3192	13.6277	13.8622	16.5784
YM1	CJ2	14.8076	15.4882	15.1978	14.7563	15.7223	13.067	13.8865	14.7744
YM1	CJ3	21.379	19.7221	18.4229	21.1275	18.2498	19.4664	19.4555	21.7119
YM1	CJ4	17.8264	20.2755	18.7269	21.891	19.5453	20.7004	18.8028	19.6628
YM2	CJ1	11.2347	10.2761	9.052	6.9708	10.3503	11.6039	10.3071	9.8377
YM2	CJ2	21.0061	20.9248	20.3502	19.3484	21.0205	20.5812	20.4264	22.2294
YM2	CJ3	18.137	17.8232	17.7461	19.1385	18.1644	18.3275	18.677	18.8261
YM2	CJ4	18.1777	19.3501	18.6842	19.7873	18.43	19.6076	17.1946	19.1136
YM3	CJ1	20.0458	20.312	20.6001	20.4018	18.9484	21.636	18.9797	21.2616
YM3	CJ2	16.7989	16.7596	15.8682	15.7617	16.6011	14.2501	16.6125	15.007
YM3	CJ3	15.9019	14.9308	14.3914	16.3312	16.0294	14.1343	15.2908	13.8333
YM3	CJ4	15.0475	14.6666	15.144	14.9858	15.8641	15.5509	14.7023	15.1766
YM4	CJ1	8.6417	7.9564	6.6385	6.1037	6.2746	7.9978	7.9521	8.1498
YM4	CJ2	11.0486	10.053	10.5551	10.4093	9.7868	9.7505	11.7382	9.773
YM4	CJ3	6.0699	5.7593	5.9406	5.9573	5.6861	6.0809	5.9546	5.6935
YM4	CJ4	17.6464	17.9379	17.0637	17.2269	17.2221	16.3942	17.0101	16.6148
YM5	CJ1	9.0614	9.6527	9.2778	10.1104	9.4489	8.7133	8.5	8.4007
YM5	CJ2	15.2333	12.0352	12.8536	13.9157	13.6091	14.3926	14.5979	14.9424
YM5	CJ3	12.6798	11.5197	14.2815	13.0256	13.5301	13.0397	14.1963	12.8083
YM5	CJ4	10.3288	11.3656	11.6098	10.2222	11.6478	10.7172	10.6767	12.5261
YM6	CJ1	18.6477	17.6767	17.3375	19.1472	18.635	21.1066	18.3282	18.5766
YM6	CJ2	21.9278	19.047	21.3418	18.6951	19.0394	18.4678	21.026	19.1001
YM6	CJ3	13.6517	13.4139	15.1393	14.1562	13.7249	15.3623	12.5449	12.7344
YM6	CJ4	14.512	13.9531	15.0306	15.4725	13.7037	13.1335	13.1648	14.3953

YM7	CJ1	10.4174	10.8873	11.0063	10.7472	10.0323	10.4192	10.6256	11.1602
YM7	CJ2	16.2719	15.6787	15.5	16.163	18.1842	16.5201	16.2141	16.0362
YM7	CJ3	11.6792	11.7843	12.6201	9.6528	12.0407	14.4366	13.089	13.2033
YM7	CJ4	19.8967	19.0611	18.8898	18.7248	20.7553	19.2426	17.2442	19.9914
YM8	CJ1	16.0378	16.5458	14.2611	16.7948	15.7649	16.087	18.0783	16.4043
YM8	CJ2	9.1702	8.2577	8.4075	8.7539	8.0403	8.8833	8.8275	8.8363
YM8	CJ3	18.1412	17.7213	17.3961	18.1144	17.3806	17.0742	18.0831	17.2036
YM8	CJ4	15.585	17.6744	15.8316	16.1111	16.0222	16.738	17.5842	15.6714
YM9	CJ1	13.8043	14.458	13.7797	15.0778	14.5312	14.573	14.1512	15.4121
YM9	CJ2	10.9543	11.9596	12.8671	12.1354	14.7485	11.168	13.5338	12.975
YM9	CJ3	7.0373	6.9797	7.3555	6.9663	7.0274	6.9685	7.7906	7.4429
YM9	CJ4	9.1619	9.4194	8.4154	9.2957	8.816	8.7777	8.3937	8.9064
YM10	CJ1	12.8912	13.2607	12.7798	12.9658	12.5589	13.3814	13.5386	13.4797
YM10	CJ2	6.7361	6.2157	6.919	6.3879	7.2507	6.5716	7.5501	7.3215
YM10	CJ3	7.9742	9.0691	8.7758	9.45	9.394	8.6883	8.7864	9.0528
YM10	CJ4	15.8129	16.3005	16.4755	16.9602	16.2367	15.7255	14.9571	16.0038

疫苗类型 The vaccine type	生产工位 Production location	33	34	35	36	37	38	39	40
YM1	CJ1	12.9369	14.409	14.6302	13.2939	12.1905	14.3703	13.3129	14.0933
YM1	CJ2	16.1006	13.9384	15.8886	14.8226	16.5877	15.2157	17.5855	14.561
YM1	CJ3	18.9278	19.8023	18.9418	20.7015	20.508	20.3502	19.7143	17.9974
YM1	CJ4	23.789	19.7352	20.5003	19.2858	19.4977	20.9902	20.0346	19.6854
YM2	CJ1	9.5162	11.5301	9.7704	9.7388	9.2589	9.543	9.971	10.0983
YM2	CJ2	19.7219	20.0662	19.3491	20	21.2503	21.1921	20.8617	17.8076
YM2	CJ3	19.0078	18.6487	17.7081	15.8679	16.5714	15.5031	18.7477	18.6647
YM2	CJ4	18.1392	18.7159	18.9058	17.1641	19.4286	18.1241	17.9743	18.8822
YM3	CJ1	18.5186	19.6914	19.9362	21.8045	20.5939	21.4702	20.3975	19.5749
YM3	CJ2	16.1992	16.9421	16.1202	15.3428	16.5954	16.2296	16.0923	16.9105
YM3	CJ3	14.8315	16.7783	13.1644	14.4927	13.7774	14.5811	14.6549	13.9569
YM3	CJ4	15.951	16.4158	16.7463	15.7135	13.3613	13.8445	15.1134	15.683
YM4	CJ1	8.5667	8.8797	8.4255	8.5824	8.3476	5.872	8.2882	8.0931
YM4	CJ2	8.6784	10.777	10.6607	8.7116	9.4432	9.0474	9.8655	10.5037
YM4	CJ3	5.9264	6.2042	5.8542	6.2076	5.3536	5.9349	5.9045	5.8599
YM4	CJ4	18.4967	18.1732	18.0464	19.199	16.7309	18.8366	18.316	18.6615
YM5	CJ1	9.2704	9.1614	7.1539	8.2657	9.6395	8.0104	8.6367	8.8027
YM5	CJ2	13.039	13.7143	14.6103	16.6052	14.5476	12.008	14.4092	15.3018
YM5	CJ3	12.7954	13.5256	13.8175	13.5404	12.1903	13.3083	12.0479	12.5494
YM5	CJ4	10.3054	11.5047	12.1867	14.5267	10.3521	9.9351	10.6824	12.1522
YM6	CJ1	19.4659	19.181	18.8252	19.1283	20.9437	21.2957	18.1519	18.2842

YM6	CJ2	20.9877	19.2255	19.8238	20.3539	17.5005	18.9941	18.3662	18.6631
YM6	CJ3	14.2177	14.9149	15.4126	14.7449	13.5741	15.5973	14.4431	14.4519
YM6	CJ4	14.3319	15.3396	14.0114	13.8839	14.3275	13.8839	12.5031	13.5782
YM7	CJ1	9.1699	11.1667	11.2226	10.9612	11.6865	12.1948	11.2021	11.8751
YM7	CJ2	15.7038	15.2097	17.4896	16.6611	16.7165	15.3673	16.8099	14.9078
YM7	CJ3	12.6134	11.9079	11.2156	11.8474	11.7133	10.2864	11.341	12.3024
YM7	CJ4	19.9015	19.1538	19.4123	19.2167	20.5724	19.2411	19.9315	18.8994
YM8	CJ1	13.5753	16.6878	15.6364	18.0099	16.017	16.0714	17.8773	16.1736
YM8	CJ2	8.3387	8.6464	8.5184	8.8277	7.9676	8.977	8.9818	9.2019
YM8	CJ3	18.4081	18.6298	19.2897	17.7995	17.6313	19.1469	18.5353	18.0053
YM8	CJ4	16.102	18.7351	16.9666	17.6784	18.202	16.0407	15.4271	15.9046
YM9	CJ1	15.0151	15.5265	15.3653	15.1223	14.6238	15.0807	14.9534	15.0777
YM9	CJ2	11.346	12.994	12.5057	12.6972	11.8546	12.4178	10.487	12.4979
YM9	CJ3	6.5214	7.3012	7.3741	6.6749	7.0277	7.0357	7.3006	6.6233
YM9	CJ4	9.1448	9.5184	9.1407	9.071	8.5023	8.1729	9.4364	8.4594
YM10	CJ1	12.2915	12.8179	13.3981	12.5045	12.5758	13.3912	12.2477	12.4089
YM10	CJ2	6.5912	8.1603	7.6208	6.0779	6.7555	5.9503	7.2702	7.0242
YM10	CJ3	9.0899	9.5496	8.7758	8.8108	9.0842	8.2344	9.1491	8.3249
YM10	CJ4	15.7376	16.5333	16.3477	15.663	15.6047	16.4806	16.6828	15.937

疫苗类型 The vaccine type	生产工位 Production location	41	42	43	44	45	46	47	48
YM1	CJ1	13.3188	15.7694	13.7147	14.4172	13.4889	12.2127	10.0557	11.2885
YM1	CJ2	14.9932	16.1174	16.5442	17.3505	14.2352	14.8039	14.1955	13.8342
YM1	CJ3	20.3035	19.8059	20.961	18.7922	19.5314	17.9482	20.282	19.7009
YM1	CJ4	21.528	19.1165	22.4832	21.1871	20.587	21.293	20.8292	19.8982
YM2	CJ1	10.1352	9.8539	9.288	9.751	8.4938	10.4434	9.4922	11.2424
YM2	CJ2	19.6272	20.3376	20.4227	20.6524	20.2571	19.9451	20.9298	18.3882
YM2	CJ3	18.8577	18.5362	15.8763	18.8257	18.3018	19.1454	17.9791	18.4413
YM2	CJ4	20.8586	18.0953	19.7847	18.9133	19.3362	20.036	17.964	19.3199
YM3	CJ1	19.553	20.4754	20.1555	18.9034	20.6113	19.2769	17.814	19.6732
YM3	CJ2	15.9451	16.975	14.479	16.3005	16.5712	15.3961	17.0468	16.44
YM3	CJ3	15.1129	13.1457	14.7815	16.2231	15.0668	15.2358	15.3165	14.8597
YM3	CJ4	16.1437	15.9707	14.568	13.3955	15.1554	15.3174	14.2399	14.9905
YM4	CJ1	9.7013	9.4049	6.6174	10.0389	6.6853	6.9935	7.8182	6.8231
YM4	CJ2	9.5698	8.8511	9.3639	10.6224	12.5088	9.6288	9.1049	10.3173
YM4	CJ3	5.9675	5.7804	5.7278	5.8252	6.0654	5.8308	5.7826	6.3889
YM4	CJ4	19.1978	16.0432	19.0822	17.4945	17.2071	18.8017	18.498	16.8717
YM5	CJ1	9.383	8.4104	8.3472	8.7318	8.6017	9.5406	8.919	7.1712
YM5	CJ2	12.7187	14.0937	13.3463	13.5376	14.0591	14.9724	15.5651	14.8412

YM5	CJ3	13.1203	12.1342	10.7985	14.5233	13.4902	12.9085	11.7632	12.0618
YM5	CJ4	11.7665	11.1685	10.5381	10.5991	11.7907	10.8876	13.6173	9.2316
YM6	CJ1	19.5756	19.3616	20.8536	19.2442	18.5193	17.5576	17.9153	21.7526
YM6	CJ2	20.8707	20.6347	20.3929	20.7868	19.7562	21.597	19.8324	20.7907
YM6	CJ3	12.2577	13.8507	12.0908	13.9429	15.5024	13.1718	14.6361	14.1124
YM6	CJ4	12.724	13.1294	13.8839	13.0309	13.956	12.8533	14.6521	12.3667
YM7	CJ1	9.5483	11.2874	10.5509	8.8435	11.7795	11.0881	10.1451	11.6061
YM7	CJ2	16.8768	15.6354	16.5643	16.8034	17.4371	17.9153	17.3373	17.612
YM7	CJ3	13.7849	12.522	13.6829	11.7559	11.635	12.0337	12.598	11.7629
YM7	CJ4	18.7426	20.0771	19.3947	18.2414	19.5475	17.6019	19.5605	19.7547
YM8	CJ1	13.778	15.2061	15.7162	14.6066	16.1496	17.4167	16.2192	15.2263
YM8	CJ2	8.8466	9.2378	8.5702	8.8492	9.557	8.338	9.2825	8.7671
YM8	CJ3	18.1578	18.7253	19.0724	17.9208	17.5051	18.1367	17.1618	18.7873
YM8	CJ4	15.3835	16.6801	14.8427	16.6283	17.2611	16.6874	16.3393	16.5073
YM9	CJ1	14.9429	14.0629	13.8355	15.2648	17.4124	15.9623	15.0524	15.7947
YM9	CJ2	14.7335	12.2897	13.275	12.4284	12.332	12.1688	11.6138	12.8199
YM9	CJ3	7.2179	6.9258	6.9031	7.0854	7.2654	6.8246	7.9947	7.5825
YM9	CJ4	8.1086	8.5813	8.8948	9.7125	9.323	9.0613	8.938	8.5382
YM10	CJ1	12.9826	12.8295	13.1	12.5869	12.241	12.8734	12.8798	12.2896
YM10	CJ2	7.0875	6.9936	6.9368	7.2072	6.9212	7.1442	6.8886	6.8049
YM10	CJ3	9.2554	9.5642	8.5083	9.3266	8.2244	9.0715	8.4398	9.2523
YM10	CJ4	16.118	15.5312	16.5642	14.9504	16.4388	15.4524	15.7553	15.7211

疫苗类型 The vaccine type	生产工位 Production location	49	50	平均	方差	最大值	最小值	极距	statistic 正态分布	pvalue
YM1	CJ1	12.1351	14.1093	13.1222	0.974366	14.1093	12.1351	1.9742	0.983827	0.72025
YM1	CJ2	14.3331	13.2053	13.7692	0.317983	14.3331	13.2053	1.1278	0.978293	0.482443
YM1	CJ3	19.1686	20.9642	20.0664	0.806045	20.9642	19.1686	1.7956	0.983192	0.691952
YM1	CJ4	20.0769	20.9378	20.50735	0.185287	20.9378	20.0769	0.8609	0.991302	0.971692
YM2	CJ1	10.1825	10.0414	10.11195	0.004977	10.1825	10.0414	0.1411	0.963879	0.129222
YM2	CJ2	20.0012	17.6807	18.84095	1.34618	20.0012	17.6807	2.3205	0.959956	0.08852
YM2	CJ3	17.727	18.0852	17.9061	0.032077	18.0852	17.727	0.3582	0.980891	0.590097
YM2	CJ4	18.0913	19.6992	18.89525	0.646336	19.6992	18.0913	1.6079	0.980313	0.565225
YM3	CJ1	19.2481	20.5894	19.91875	0.449771	20.5894	19.2481	1.3413	0.9857	0.801241
YM3	CJ2	17.7298	16.8671	17.29845	0.186063	17.7298	16.8671	0.8627	0.984007	0.72821
YM3	CJ3	16.0128	14.7299	15.37135	0.411458	16.0128	14.7299	1.2829	0.984914	0.767918
YM3	CJ4	15.3984	16.1706	15.7845	0.149073	16.1706	15.3984	0.7722	0.985831	0.806681
YM4	CJ1	6.4058	7.6218	7.0138	0.369664	7.6218	6.4058	1.216	0.974828	0.359465
YM4	CJ2	8.8286	9.1073	8.96795	0.019418	9.1073	8.8286	0.2787	0.992045	0.981988
YM4	CJ3	5.7324	5.6739	5.70315	0.000856	5.7324	5.6739	0.0585	0.977069	0.436017

YM4	CJ4	19.4065	20.1385	19.7725	0.133956	20.1385	19.4065	0.732	0.978848	0.504452
YM5	CJ1	7.9794	9.4056	8.6925	0.508512	9.4056	7.9794	1.4262	0.955186	0.05592
YM5	CJ2	12.8576	13.4064	13.132	0.075295	13.4064	12.8576	0.5488	0.978081	0.474161
YM5	CJ3	13.854	13.1092	13.4816	0.138682	13.854	13.1092	0.7448	0.989957	0.945159
YM5	CJ4	12.769	9.8535	11.31125	2.125035	12.769	9.8535	2.9155	0.968645	0.203749
YM6	CJ1	18.2352	18.7195	**18.4774**	0.058637	18.7195	18.2352	0.4843	0.979102	0.514734
YM6	CJ2	20.7612	18.5262	**19.6437**	1.248806	20.7612	18.5262	2.235	0.977519	0.452746
YM6	CJ3	13.8652	15.6484	**14.7568**	0.794951	15.6484	13.8652	1.7832	0.979172	0.517596
YM6	CJ4	13.0952	13.0573	**13.0763**	0.000359	13.0952	13.0573	0.0379	0.991253	0.970911
YM7	CJ1	10.6521	12.3954	11.52375	0.759774	12.3954	10.6521	1.7433	0.987197	0.860339
YM7	CJ2	16.7163	15.7742	16.24525	0.221888	16.7163	15.7742	0.9421	0.971391	0.263431
YM7	CJ3	11.3695	12.0583	11.7139	0.118611	12.0583	11.3695	0.6888	0.976442	0.413496
YM7	CJ4	19.8253	17.375	18.60015	1.500993	19.8253	17.375	2.4503	0.96696	0.173631
YM8	CJ1	16.6092	16.4536	16.5314	0.006053	16.6092	16.4536	0.1556	0.97569	0.38761
YM8	CJ2	8.3874	9.5962	8.9918	0.365299	9.5962	8.3874	1.2088	0.980522	0.574157
YM8	CJ3	18.483	19.1391	18.81105	0.107617	19.1391	18.483	0.6561	0.986671	0.840392
YM8	CJ4	16.6297	17.5492	17.08945	0.21137	17.5492	16.6297	0.9195	0.982607	0.66587
YM9	CJ1	15.3535	16.7909	16.0722	0.51653	16.7909	15.3535	1.4374	0.979469	0.52979
YM9	CJ2	12.434	14.3156	13.3748	0.885105	14.3156	12.434	1.8816	0.969283	0.216386
YM9	CJ3	6.7242	6.7389	6.73155	5.4E-05	6.7389	6.7242	0.0147	0.990589	0.958932
YM9	CJ4	8.9344	9.1457	9.04005	0.011162	9.1457	8.9344	0.2113	0.981015	0.595498
YM10	CJ1	13.2166	13.2919	13.25425	0.001418	13.2919	13.2166	0.0753	0.96696	0.173631
YM10	CJ2	7.4954	6.6676	7.0815	0.171313	7.4954	6.6676	0.8278	0.980709	0.582222
YM10	CJ3	8.9181	8.4189	8.6685	0.0623	8.9181	8.4189	0.4992	0.977198	0.440792
YM10	CJ4	15.1811	16.1498	15.66545	0.234595	16.1498	15.1811	0.9687	0.980462	0.571615

附录 19：第二题结果

```
184.6549
4    5    10   7    8    1    2    3    6    9
```

附录 20：第二题结果显示

```
总时间:
184.6549
顺序：
4    5    10   7    8    1    2    3    6    9
编号     开始     结束
4    0.0000   41.9896
5    7.9887   55.9020
10   16.7587 71.9544
7    29.7409 91.0420
8    40.9011 107.8734
1    56.9212 127.8863
```

```
2    70.2052 146.8287
3    80.0761 161.9451
6    100.1345    175.7052
9    119.2086    184.6549
```

附录 21：第三题结果

```
214.2449
204.8139
215.2513
213.4338
210.8892
219.1996
209.4289
210.8102
210.8892
210.8102
215.2513
210.8102
213.9206
216.2191
215.0129
213.5152
210.7944
210.8102
210.6963
215.0129
214.3142
213.4338
210.7944
210.8093
214.3142
210.8102
214.3142
214.3142
213.6408
206.0121

min:204.8139
4    10  7    8    2    6    1    3    5    9
```

附录 22：第三题结果分箱分析

```
0
1
4
```

3
6
9
23
11
12
10
8
7
2
3
1

附录 23：第四题结果

18113.7076
9084.5382
11178.5646
16419.4460
14408.6627
29622.5738
32017.9142
13423.6051
8069.5593
13721.7620
共 166060.3336 分钟

附录 24：第五题 lingo 运行结果

LINGO/WIN64 19.0.24 (26 Oct 2020), LINDO API 13.0.4099.232

Licensee info: Eval Use Only
License expires: 17 FEB 2022

Global optimal solution found.
Objective value: 312228.3
Infeasibilities: 0.000000
Total solver iterations: 2
Elapsed runtime seconds: 0.13

Model Class: LP

Total variables: 10
Nonlinear variables: 0
Integer variables: 0

Total constraints:	25
Nonlinear constraints:	0
Total nonzeros:	70
Nonlinear nonzeros:	0

Variable	Value	Reduced Cost
X1	0.000000	9.819107
X2	500.0000	0.000000
X3	600.0000	0.000000
X4	0.000000	0.1268263
X5	1200.000	0.000000
X6	1546.861	0.000000
X7	492.0732	0.000000
X8	800.0000	0.000000
X9	600.0000	0.000000
X10	900.0000	0.000000

Row	Slack or Surplus	Dual Price
1	312228.3	1.000000
2	1000.000	0.000000
3	0.000000	7.500464
4	0.000000	1.408939
5	1000.000	0.000000
6	0.000000	

11.64347		
	7	53.13886
0.000000		
	8	1307.927
0.000000		
	9	0.000000
2.848483		
	10	0.000000
14.27236		
	11	0.000000
4.241975		
	12	0.000000
0.000000		
	13	500.0000
0.000000		
	14	600.0000
0.000000		
	15	0.000000
0.000000		
	16	1200.000
0.000000		
	17	1546.861
0.000000		
	18	492.0732
0.000000		
	19	800.0000
0.000000		
	20	600.0000
0.000000		
	21	900.0000
0.000000		
	22	0.000000
0.9427237		
	23	281.6751
0.000000		
	24	7897.096
0.000000		
	25	0.000000
1.963532		

附录 25：第五题 lingo 代码

```
max42x1+54x2+50x3+43x4+42x5+45x6+48x7+51x8+46x9+48x10
st
    x1<=1000
```

```
    x2<=500
    x3<=600
    x4<=1000
    x5<=1200
    x6<=1600
    x7<=1800
    x8<=800
    x9<=600
    x10<=900

    x1>=0
    x2>=0
    x3>=0
    x4>=0
    x5>=0
    x6>=0
    x7>=0
    x8>=0
    x9>=0
    x10>=0

    13.2840x1+9.8709x2+20.0584x3+7.9887x4+8.7701x5+19.0741x6+11.1601x7+16.020
1x8+15.0146x9+12.9822x10<=96000
    14.9621x1+19.9075x2+15.9726x3+9.9366x4+13.7220x5+20.0944x6+16.4961x7+8.72
89x8+12.0351x9+7.0110x10<=96000
    19.8460x1+17.9282x2+14.9704x3+5.9359x4+13.0052x5+14.1485x6+12.0137x7+17.9
794x8+7.0419x9+9.0492x10<=96000
    20.0129x1+18.9424x2+15.1164x3+18.1284x4+11.2495x5+13.7601x6+19.0876x7+16.
8314x8+8.9497x9+16.0524x10<=96000
end
```

附录 26：第五题线性规划结果

线性规划结果：产量（剂）
0
500
600
0
1200
1546
492
800
600
900

加粗的存在小数

每工位的多余时间
第 1 个工位剩余时间： 17.2175
第 2 个工位剩余时间： 300.2219
第 3 个工位剩余时间： 7910.1760
第 4 个工位剩余时间： 13.2403

附录 27：第五题生产顺序安排。

	时间序列：	结束当前工作的时间						
	初始	7 个 10 号	1 个 2	1 个 3	1 个 7	490 个 (5,7)	840 个 10	499 个 (2,3)
工位 1	0	90.8754	100.7463	120.8047	131.9648	9897.7628	20802.8108	35737.5315
工位 2	12.9822	97.8864	120.6538	136.7773	153.2734	14960.1424	20849.3824	38753.5523
工位 3	19.9932	106.9356	138.582	153.5524	165.5661	14972.1561	22573.4841	38989.8855
工位 4	29.0424	141.4092	160.3516	175.468	194.5556	15059.7346	28543.7506	45539.0918
		浪费的时间						
工位 1	0	0	0	0	0	0	0	0
工位 2	12.9822	48.8094	51.6693	51.8202	51.8202	51.8202	51.8202	51.8202
工位 3	19.9932	43.5912	57.3094	57.3094	57.3094	2604.6384	2604.6384	2604.6384
工位 4	29.0424	29.0424	29.0424	29.0424	29.0424	29.0424	29.0424	29.0424
	100 个 3	600 个 (5,9)	53 个 10	110 个 (5,8)	550 个 (6,8)	465 个 6	300 个 6	140 个 8
工位 1	37743.3715	52014.1915	52702.25	55429.17	74730.9801	83600.4366	89322.6666	91565.4806
工位 2	40350.8123	55805.0723	55875.18	58344.78	74197.5963	83541.4923	89569.8123	91574.2095
工位 3	40486.9255	55818.0775	55908.57	59316.88	76987.2205	83566.273	89583.9608	92101.0768
工位 4	47050.7318	59170.2518	59330.78	62419.67	79244.9998	85643.4463	89771.4763	92127.8723
	浪费的时间				**做完后还剩 996 个 6，140 个 8**		**231,140**	
工位 1	0	0	0	0	0	0	0	0

工位 2	51.8202	51.8202	51.8202	51.8202	51.8202	51.8202	51.8202	834.1714
工位 3	2604.6384	5907.5304	5907.53	5907.53	5907.5304	5907.5304	7680.6682	7680.6682
工位 4	29.0424	29.0424	29.0424	29.0424	29.0424	29.0424	29.0424	29.0424
					550	465	300	140
	最后的6							
工位 1	95723.6344							
工位 2	95954.7887							
工位 3	95968.9372							
工位 4	95982.6973							
	浪费的时间							
工位 1	0							
工位 2	834.1714							
工位 3	8464.1556							
工位 4	884.1656218							

附录 28：第五题实际生产结果

实际：

0
500
600
0
1200
1533
491
800
600
900