摘要

现在随着人们生活水平的质量上升，家庭对孕妇的健康状况以及胎儿的健康状况逐渐得到重视。NIPT 是作为检测胎儿是否健康的关键手段，但是不同 BMI 的孕妇的检测孕期均不相同，因此我们需要建立一个合适的模型来对不同 BMI 分组的孕妇一个合适的检测孕期，及时调查出问题，进而减少胎儿和孕妇的风险。

针对问题一，我们分析 **Y 染色体浓度**与**孕妇孕周数**及 **BMI** 的相关特性。研究首先通过探索性分析，通过**散点图**，**相关系数矩阵图（热力图）**以及**箱线图**，均发现变量间存在显著的**非线性关系**与**数据删失**问题，得到传统线性回归模型不再适用的结论。据此，我们创新性地引入**生存分析框架**，构建以孕周为时间轴的 **Kaplan-Meier 曲线**和**加速失效时间（AFT）**模型，求解过程采用 **Log-logistic AFT** 分布进行参数估计，量化了**孕妇 BMI** 对达标时间的显著影响。

针对问题二，我们根据**动态规划**得到了一组较为合理的 BMI 分组，然后我们对 BMI 进行 **Log logistic AFT** 模型的建立，通过 Python 的计算得到了公式的各个，这个公式计算的是每个孕妇 BMI 组的 NIPT 的**最佳时间**，然后我们进行了**风险分析**得到了进一步的建议的 NIPT 时间为 12.5 周，最后我们做了**模拟检测误差分析**发现，同时我们反过来利用这个结果进一步得到更加推荐的 NIPT 时间。

针对问题三，这一问的题目要求的建模更加精准，根据损失函数以及检测误差分析我们可以得到更加精细的模型建立，我们通过**多重共线性**保留了**孕妇 BMI**、**Y 染色体的 Z 值**、**X 染色体浓度**、**被过滤掉读段数的比例**和**生产次数这些变量**，我们通过聚类分析将这些组分成了 5 个类别，进一步得到了各聚类分组的平均画像，然后我们根据这些变量得到了在这个题下这个数据下的系数，进而分别得到几组的最佳 NIPT 时间为 12 周，17 周，11 周和 10 周，最后我们分析了风险函数以及检测误差。

针对问题四，我们先对部分没有标准化的数据进行了 **Z-score 标准化**处理，然后我们采取**随机森林**进行模型的拟合，最后得到判定女胎异常的一个阈值。

在模型的检验中我们先表明相对于其他模型，我们将 **KM 曲线（Kaplan-Meier）**、**Log-logistic AFT 曲线**和 **Weibull-AFT 曲线**进行比对得到的 **AIC** 和 **BIC** 值来对这个结论进行证明。然后利用**微调阈值**来进行灵敏度分析，我们发现得到的结论证明了该模型的**稳健性**和**健壮性**，同时证明了具体阈值标准，将是决定最佳检测时点的**关键因素**。此外，本文还利用不同的 **BMI 分组**进行灵敏度分析，发现同样成功率所需的时间差异非常明显，这证明了模型预测结果对输入变量 BMI 的敏感程度。

最后，我们对模型进行了优缺点分析和推广。

关键字： 生存分析 (Survival Analysis)　Log-logistic AFT 模型　Weibull AFT 模型

一、 问题重述

1.1 背景资料

NIPT（Non-invasive Prenatal Test，即无创产前检测）是一种通过采集母体血液、检测胎儿的游离 DNA 片段、分析胎儿染色体是否存在异常的产前检测技术，目的是通过早期检测确定胎儿的健康状况。NIPT 的准确性主要由胎儿性染色体（男胎 XY，女胎 XX）浓度判断。通常孕妇的孕期在 10 周 25 周之间可以检测胎儿性染色体浓度，且如果男胎的 Y 染色体浓度达到或高于 4 实践表明，男胎 Y 染色体浓度与孕妇孕周数及其身体质量指数（BMI）紧密相关。由于每个孕妇的年龄、BMI、孕情等存在个体差异，对所有孕妇采用简单的经验分组和统一的检测时点进行 NIPT，会对其准确性产生较大影响。因此，依据 BMI 对孕妇进行合理分组，确定各不同群组的最佳 NIPT 时点，可以减少某些孕妇因胎儿不健康而缩短治疗窗口期所带来的潜在风险。

1.2 需要解决的问题

对于问题一：

我们需要建立一个模型，这个模型在稳健性以及预测性上都应该比较强，我们先考虑简单的回归模型，然后进行下一步的思考，而显著性的检验需要计算 P 值，如果 P 值小于 0.05，那么就说明了具有显著性

对于问题二：

我们在第一问得到了我们需要的模型，根据这个模型进行风险函数的运用以及检测误差的分析

对于问题三：跟问题二类似，但是我们需要将这么多变量中筛选出一些具有代表性的变量，然后再进行建模分析，然后风险函数检测和检测误差分析。

对于问题四：这个问题与前面三个有不同，要对数据进行标准化，然后利用随机森林进行建模。

核心问题：建立孕妇体质指数 (BMI, 记作 B) 与男性胎儿 Y 染色体浓度首次达到临床可靠水平 ($C_{\mathrm{Y}} \geq 4\%$) 所需时间 (T, 孕周) 之间的函数或统计关系模型：

$$T = f(B) \quad \text{或} \quad \text{探究} T \text{与} B \text{的相关性}$$

研究目标 ：建立数学模型量化 BMI 对 Y 染色体浓度达标时间的影响，为不同 BMI 孕妇提供个性化检测时间建议

二、 问题分析

2.1 问题一分析

针对问题一，我们分析 y 染色体浓度与孕妇孕周数及 BMI 的相关特性. 研究首先通过探索性分析发现变量间存在显著的非线性关系与数据删失问题，传统线性回归模型不再适用. 据此，创新性地引入生存分析框架，构建以孕周为时间轴的 Kaplan-Meier 曲线和加速失效时间（AFT）模型，求解过程采用 Log-logistic 分布进行参数估计，量化了 BMI 对达标时间的显著影响.

2.2 问题二分析

我们根据动态规划得到了一组较为合理的 BMI 分组，然后我们对 BMI 进行 Log-logistic 模型建立，得到了一个公式，这个公式是计算每个 BMI 组的 NIPT 的最佳时间，然后我们进行了风险分析得到了进一步的建议的 NIPT 时间，最后我们做了模拟检测误差分析.

2.3 问题三分析

相较于第二问，这一问的题目要求的建模更加精准，根据损失函数以及检测误差分析我们可以得到更加精细的模型建立，我们通过聚类分析将这些组分成了 5 个类别，其中有一组仅有两个人骂我们们视作极端因素去除了，而在后面的计算验证中也证明了他们两个算出来的 NIPT 预测时间是与其他组别完全脱离的. 最后我们分析了风险函数以及检测误差.

2.4 问题四分析

第四问我们先对部分没有标准化的数据进行了 z-score 标准化处理，然后我们采取随机森林进行模型的拟合，最后得到判定女胎异常的一个阈值.

三、 模型的假设

- 独立性 ：不同孕妇的观测相互独立.
- 随机删失假设 ：删失机制与事件发生无关.
- 分布假设： 生存时间服从特定分布.

四、 符号说明

符号	意义
BMI	孕妇身体质量指数 (Body Mass Index)
$w+$	孕周的一种记录格式，例如“12w+3”
T	生存分析中的时间（Time）变量，通常指“检测孕周”
E	生存分析中的事件（Event）变量
$Q1, Q2, Q3$	数据的 25%, 50% 和 75% 四分位数
O	卡方检验中的观测频数 (Observed Frequency)
E	卡方检验中的期望频数 (Expected Frequency)
K	聚类分析或动态规划中的分组数量
SSE	簇内误差平方和 (Sum of Squared Error)
r_t	与检测时点相关的风险函数值
R_f	检测失败风险常量 (Failure Risk Constant)
$N(\mu, \sigma^2)$	正态分布，用于模拟测量误差
AIC	赤池信息准则 (Akaike Information Criterion)
BIC	贝叶斯信息准则 (Bayesian Information Criterion)
VIF	方差膨胀因子 (Variance Inflation Factor)
χ^2	卡方统计量

$\hat{S}(t)$	Kaplan-Meier 方法估计的生存函数
t_i	发生事件的各个时间点
d_i	在时间点 t_i 发生的事件数
n_i	在时间点 t_i 之前处于风险中的个体总数
$1 - S(t)$	事件发生概率

2.2 加速失效时间 (AFT) 模型

$\log(T)$	生存时间的自然对数
β_0	模型的截距项
β_i	第 i 个协变量的系数
σ	尺度参数
ϵ	模型的随机误差项
$h(t)$	风险函数 (Hazard Function)
$f(t)$	概率密度函数 (Probability Density Function)
$f_t(x)$	第 t 棵决策树的预测函数
T	随机森林中决策树的总数
$\hat{y}$	随机森林模型的最终预测结果
$Gini$	基尼不纯度
p_{mk}	在第 m 个节点中，样本属于第 k 类的比例
D_t	用于训练第 t 棵树的 Bootstrap 自助采样数据集

五、 模型的建立与求解

5.1 问题一模型的建立及求解

第一步：

数据整合：建立一个稳健、可靠的数据集. 原始数据分散在怀有男性胎儿和女性胎儿孕妇的两个独立表格中. 在这一问中，我们可以通过观察男性胎儿的 Y 染色体浓度来解决问题. 通过附录中孕周数的处理代码块，我们先把表格中的孕期进行规范化，方便

我们进行观察与统计. 在这一步中，

$$GestationalDays = (\text{孕周数}\times 7) + \text{天数}$$

5.1.1 模型的准备：读入文件，考虑做探索性数据分析

我们绘出散点图和箱线图. 将孕妇 BMI 和检测孕周两个作为自变量，将 Y 染色体的浓度作为因变量得出一个图表关系.

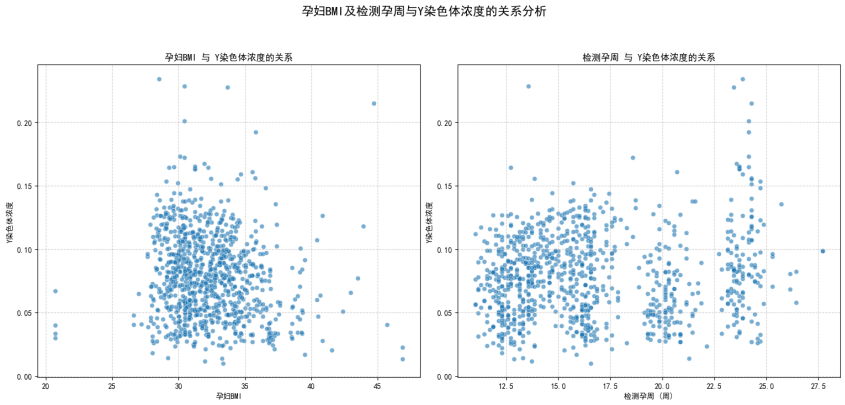

图 **1**　孕妇 **BMI** 及检测孕周与 **Y** 染色体浓度的关系分析散点图

我们可以优先考虑将一个自变量和因变量摘出，绘制两套散点图，再把箱线图绘出，下面是孕妇 BMI 以及检测孕周分组下的 Y 染色体散点图分布

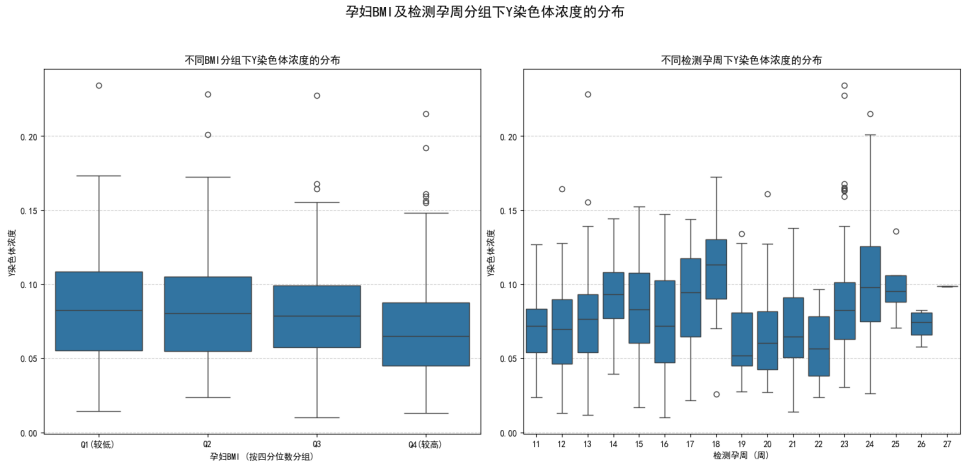

图 **2**　孕妇 **BMI** 及检测孕周与 **Y** 染色体浓度的关系分析箱线图

看不出来符合正态分布以及符合回归模型，这个时候我们继续探索.

接着, 我们可以尝试做做一下关于这三个变量之间的直方图，可以发现，对于孕妇 BMI 和 Y 染色体浓度的分布是基本符合正态分布的，而检测孕期则主要分布在 14 周附近和 23 周附近.

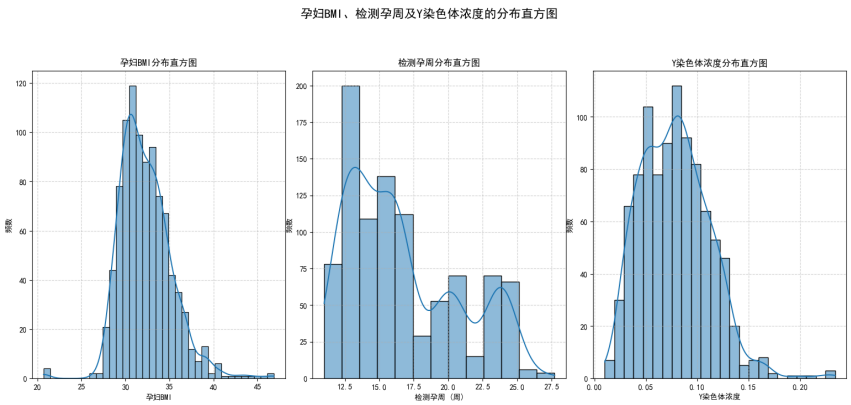

图 **3**　孕妇 **BMI**、检测孕周及 **Y** 染色体浓度的分布直方图

接着我们继续描述了这三个变量的基本统计量，发现在统计上并没有什么联系. 因此我们考虑对所有的变量进行相关性分析：

表 **1**　变量描述性统计

变量	样本数	平均数	标准差	中位数	最小值	最大值
孕妇 BMI	950	32.3058	2.9554	31.9419	20.70	46.88
检测孕周 (周)	950	16.7061	4.0324	15.8600	11.00	27.71
Y 染色体浓度	950	0.0785	0.0334	0.0774	0.0100	0.2342

在这个 Y 染色体浓度对其他所有变量的相关性热力图中我们可以发现，除了与其自己，对其他的变量相关性都非常低，于是我们放弃了直接对这些变量的回归方程建立.

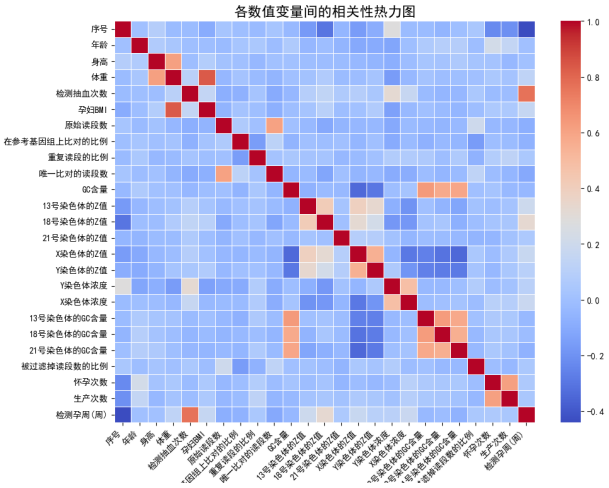

图 **4**　相关热力图

第二步：

根据题目背景资料，NIPT 检测结果的准确性有一个明确的二元判断标准：对于男性胎儿，当 Y 染色体浓度达到或高于 0.04 时，检测结果被认为是可靠的. 因此，临床上最关心的问题是：“最早在哪个孕周进行检测，我们才能有足够高的置信度认为检测是准确的？”这个问题显然不是一个典型的回归问题，而是一个经典的“时间到事件”（Time-to-Event）问题. 这种思维的转变是本次分析的核心. 我们不再试图回答“在 X 孕周和 Y 的 BMI 下，Y 染色体浓度是多少？”，而是回答“对于一个特定 BMI 的孕妇，需要多长时间才能达到 Y 染色体浓度 ≥0.04 的目标？”. 这种框架的转变使我们能够运用生存分析这一强大的统计工具.

首先我们要明确的是生存分析的几个关键定义：

生存分析框架的关键定义为了应用生存分析，我们需要明确定义以下几个核心概念：

时间 (Time, t): 分析的时间轴，由我们创建的连续变量 GestationalDays（妊娠天数）来表示.

事件 (Event): 本研究中的“事件”是临床意义上的“成功”，即某位孕妇的 Y 染色体浓度首次达到或超过 0.04 的阈值. 在生存分析的术语中，这通常被称为“失效”，但其数学本质是相同的.

删失 (Censoring): 这是生存分析区别于其他统计方法的关键特征. 由于数据是纵向的，每位孕妇有多次测量记录. 如果在研究观察期结束时（例如，最后一次检测在第 25 周），某位孕妇的 Y 染色体浓度仍未达到 0.04，我们无法知道她最终是否以及何时会达到该阈值. 这种情况被称为“右删失”. 我们只知道，直到她最后一次随访时，事件尚未发生. 生存分析模型能够科学地利用这些不完整的信息，而不是简单地将其丢弃，从而避免了偏倚.

根据生存分析的要求，我们需要对 BMI 进行一个划分，由于采取国标的形式对于孕妇这个特殊群体不再适用，因此我们这里采用的是四分位数进行划分，如下图所示：

表 **2**　孕妇 **BMI** 分位数统计

分位数	**BMI 值**
25%	30.208 806
50%	31.811 598
75%	33.926 237

对于这个过程，我们：

模拟动态过程: Y 染色体浓度的积累是一个动态的生物学过程，而非一个静态的数值. 生存分析通过对事件发生率（风险函数）随时间的变化进行建模，能够更好地捕捉这一过程的动态特性. 它关注的是事件发生的“速率”，而不是某一时刻的“水平”.

更灵活的模型假设: 虽然生存分析模型也有其自身的假设（例如，比例风险假设），但它们通常比线性回归的假设（如正态性、线性、独立性）更为宽松和贴近现实世界的生物学数据. 更重要的是，生存分析提供了一套成熟的工具来检验这些假设，并在假设不满足时提供了替代模型.

接下来我们就用加速失效模型对这三个变量的分析.

非参数基准：Kaplan-Meier（K-M）估计：

Kaplan-Meier（K-M）方法是一种非参数技术，用于从删失数据中估计生存函数. 它不假设数据遵循任何特定的概率分布. 其结果是一条阶梯状的曲线，每个向下的“台阶”代表在该时间点上发生了一个或多个“事件”（在我们的题目中，即 Y 染色体浓度达到 0.04）. 曲线的纵轴表示“生存概率”，即在该时间点尚未发生事件的概率. 其计算公式如下：

$$\hat{S}(t) = \prod_{t_i \leq t} \left(1 - \frac{d_i}{n_i}\right)$$

值得注意的是，我们这道题目的生存概率指代的是 Y 染色体 >=0.04 的情况，因此我们应该用的是：

$$1 - S(t)$$

通过比较不同 BMI 组的 K-M 曲线，我们可以获得初步的、不依赖模型的结论. 例如，如果高 BMI 组的曲线始终位于低 BMI 组曲线的下方和左侧，这表明高 BMI 组的“生存”时间更短，即她们需要更短的时间就能低于到 0.04 的 Y 染色体浓度阈值. 曲线之间的显著分离（由 Log-rank 检验的小 p 值支持）将证实 BMI 是一个影响 NIPT 成功时间的重要因素.

这里需要解释一下我们所使用 KM 模型. 我们定义：生存概率在原来的意思是“在某个孕周 t 时，Y 染色体浓度仍未达标的孕妇比例”

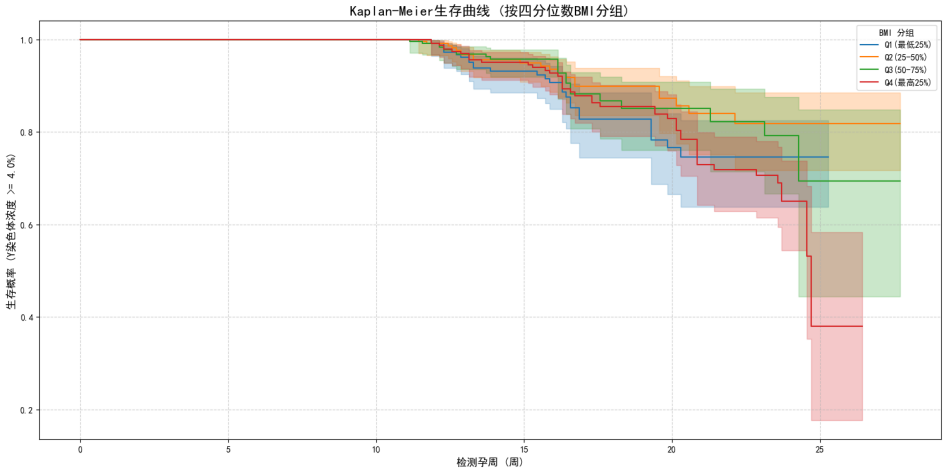

图 **5**　**Kaplan-Meier** 生存曲线 **(**按四分位数 **BMI** 分组**)**

BMI 组间比较 p 值:

表 **3**　孕妇 **BMI** 四分位数分组间的比较结果

比较组	BMI 范围	p 值
Q1 vs Q2	<30.21 vs 30.21-31.81	0.002706**
Q1 vs Q3	<30.21 vs 31.81-33.93	0.006275**
Q1 vs Q4	<30.21 vs >33.93	0.004538**
Q2 vs Q3	30.21-31.81 vs 31.81-33.93	0.085911
Q2 vs Q4	30.21-31.81 vs >33.93	0.007925**
Q3 vs Q4	31.81-33.93 vs >33.93	0.009708**

分组定义:

Q1: BMI < 30.21　(25% 分位数)

Q2: BMI 30.21-31.81　(25%-50% 分位数)

Q3: BMI 31.81-33.93　(50%-75% 分位数)

Q4: BMI > 33.93　(75% 分位数以上)

显著性标记:　** p < 0.01

可以发现均小于 0.05，因此对于 BMI 结果显著:，四组生存曲线有统计学差异.

表 **4**　基于 **Y** 染色体浓度分组的生存分析结果

指标	**低 Y 浓度组 (≤4%)**	**高 Y 浓度组 (>4%)**
分组阈值	0.040	
样本数	145	937
事件数	9	29
Y 浓度均值 (± 标准差)	0.0309 ± 0.0062	0.0843 ± 0.0301
Log-Rank 检验 p 值: 0.021289		

Y 染色体浓度 (0.04 阈值) Log-Rank 检验结果如表 4 所示. 结果显著: 两组生存曲线有统计学差异

K-M 分析虽然直观，但它只能进行分组比较，难以量化协变量的影响. 为了建立一个更强大的预测模型，我们需要转向半参数或参数回归模型. 最常见的选择是 Cox 比例风险模型和加速失效时间模型. 这里我们选择的是加速失效模型（AFT）

这种模型的解释非常直观，并且与本研究的生物学背景高度契合. 我们推断，高 BMI 可能并非以一个恒定的比例降低每日达到阈值的“风险”，而是通过诸如血液稀释效应、胎盘功能差异等生理机制，系统性地“延缓”了胎儿游离 DNA 在母体血液中积累的整个过程.AFT 模型中的加速因子恰好能够直接量化这种时间上的“延缓”或“加速”效应.

AFT 模型有以下两种我们分别分析：

按照 Weibull-AFT，我们得到了以下的图像：

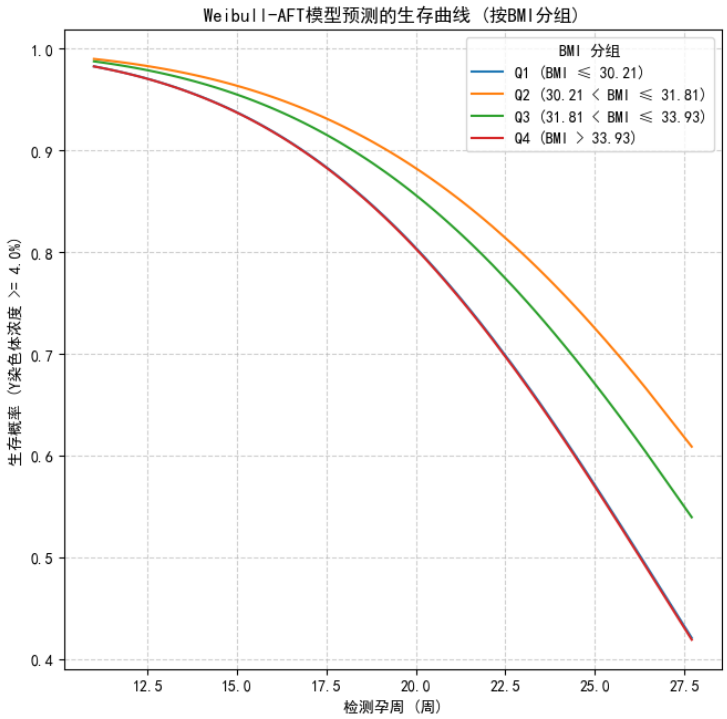

图 **6**　**WeiBull** 生存分析（**BMI**）

需要解释的是：

此处 X 轴与 Y 轴的含义：X 轴：检测孕周，是随时间变化的.Y 轴与现有文献有所不同，我们的 y 轴指的是在这个时间上对于 Y 染色体浓度大于 0.04 的总体量的概率是多少.

按照 Log-logistic AFT，我们得到了以下的图像：

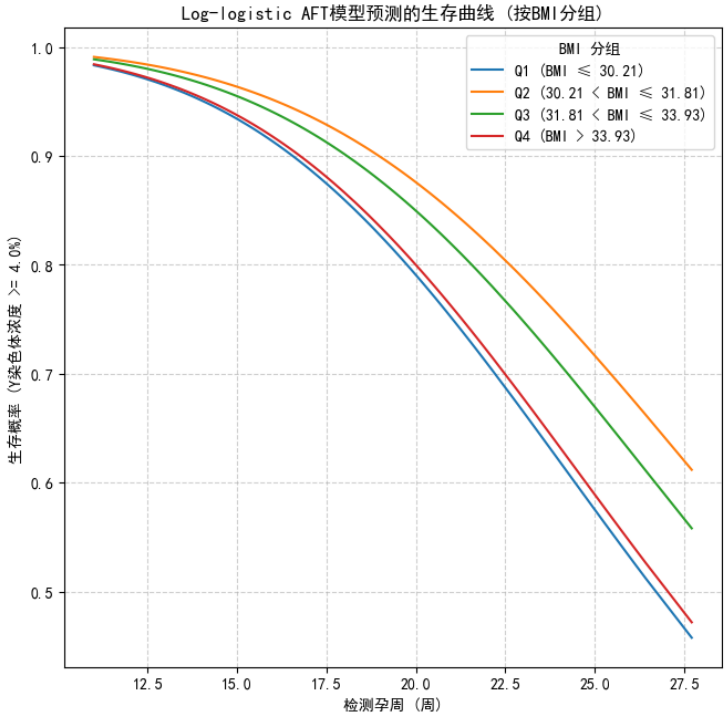

图 7　**Log-logistic AFT** 分析

仅仅通过绘图难以看出区别，因此我们需要更精确的比对分析，所以就有了对 KM 曲线比对的分析：

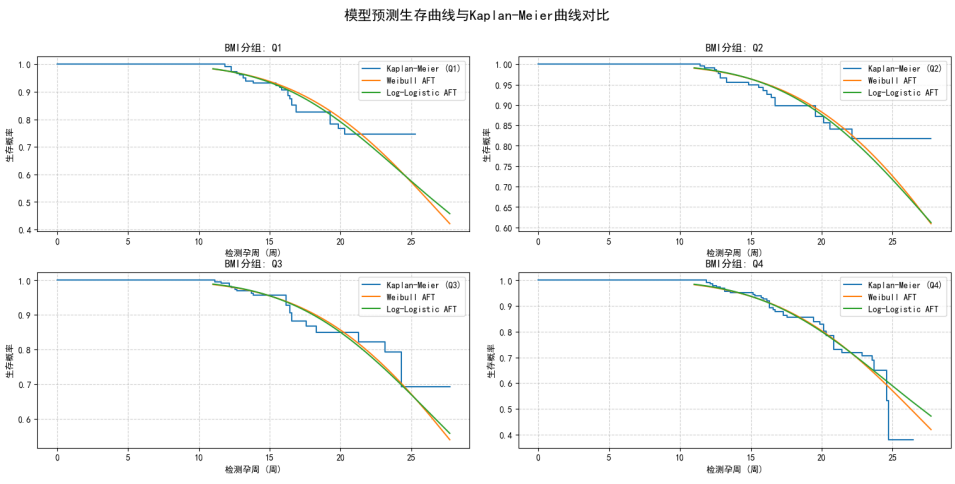

图 **8**　**KM** 曲线对比

发现使用 Log-logistic AFT 模型更优，最后我们进行的是 Bootstrap 的自主验证和 K 方计算：

卡方检验的公式为：

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

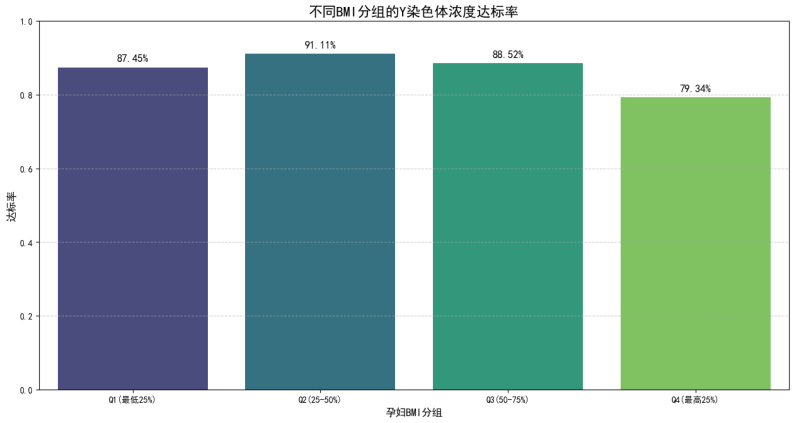

图 9　**Bootstrap** 自助法检测

可以得到，卡方统计量: 18.0835 P 值: 0.0004

5.1.2 问题一的结论

在本问题中我们采用了 Log-logistic AFT 模型进行对整个题目的建模，前提公式如下：

概率密度函数（PDF）的公式为：

$$f(t;\alpha,\beta) = \frac{(\beta/\alpha)(t/\alpha)^{\beta-1}}{[1+(t/\alpha)^{\beta}]^2}, \quad t > 0$$

累积分布函数 (CDF) 的公式为：

$$F(t;\alpha,\beta) = P(T \le t) = \frac{1}{1+(t/\alpha)^{-\beta}} = \frac{(t/\alpha)^{\beta}}{1+(t/\alpha)^{\beta}}, \quad t > 0$$

生存函数（Survival Function）：

$$S(t;\alpha,\beta) = 1 - F(t;\alpha,\beta) = \frac{1}{1+(t/\alpha)^{\beta}}$$

风险函数（Hazard Function）：

$$h(t;\alpha,\beta) = \frac{f(t;\alpha,\beta)}{S(t;\alpha,\beta)} = \frac{(\beta/\alpha)(t/\alpha)^{\beta-1}}{1+(t/\alpha)^{\beta}}$$

核心计算孕期周数的回归模型如下：

$$\log(Y) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

并且我们得出结论: 最终采用 Log-logistic AFT 进行模型的建立. 而且我们计算出 BMI 与 Y 染色体的 P 值为 0.0004，小于 0.05，因此检验了其显著性.

5.2 问题二模型的建立及求解

5.2.1 模型的准备

我们先对 BMI 的分组进行动态规划，得到如下的结果：

表 **5**　以 $K = 3$ 为例的最优 **BMI** 分组边界

分组	**BMI** 区间	
	下限	上限
第 1 组	20.70	31.80
第 2 组	31.83	35.96
第 3 组	36.05	46.88

注：分组边界基于最优聚类算法（$K = 3$）确定，边界值保留两位小数精度.

在第一问中，我们使用了 Log-logistic AFT 模型进行建立，因此，可以将第二问看作第一问的延申，但是介于我们还没有对这个模型进行详细的参数求解，其模型形式如下：

$$\log(T) = \beta_0 + \beta_1 \cdot \text{BMI} + \sigma \cdot \epsilon$$

参数说明：

β_0 (截距项)：代表当 BMI 为 0 时 log(T) 的基准水平.

β_1 (协变量系数)：BMI 的系数，代表 BMI 每增加一个单位，log(T) 的变化量.

σ (尺度参数)：尺度参数，与误差项 ϵ 的标准差有关，反映了数据点围绕回归线的离散程度.

ϵ 我们一般取标准逻辑斯谛分布的第 95 百分位数（约为 2.944）

5.2.2 模型的求解

第一步：我们依旧按照四分位数进行参考，我们去除了检测孕期小于 10 大于 25 的孕妇以及 BMI 过于极端的情况，通过代码运行计算得到了如下的模型公式：

$$\log(T) = 1.2703 + 0.0184 \times \text{BMI} + 0.1519 \times 2.994$$

得到了这个模型公式之后，我们可以开始计算. 为确保数据详实有效，我们为每个区间选择其上限值作为代表性 BMI 进行计算（对于“较低 BMI 组”，我们选择其上限 Q1）. 这是一种保守策略，确保该组绝大多数孕妇都能满足条件.

计算结果如下：

表 **6**　**BMI** 分组与 **NIPT** 检测建议

名称	BMI 区间	代表性 BMI	P95 达标孕周	最佳 NIPT 时点	风险等级
较低 BMI 组	< 31.81	31.80	10.0	孕 10 周	低风险
中等 BMI 组	$31.81 - 35.96$	35.96	10.8	孕 11 周	低风险
较高 BMI 组	≥ 35.97	46.88	13.2	孕 13 周	高风险

其中，保守 INPT 时间点指的该 BMI 组下的孕妇最晚的检测时间，使风险降到最低.

第二步：

接下来使用附件的代码进行风险分析，我们采用了本模型的风险函数进行分析，并且采纳 BMI95 分位数的方案代表整体：

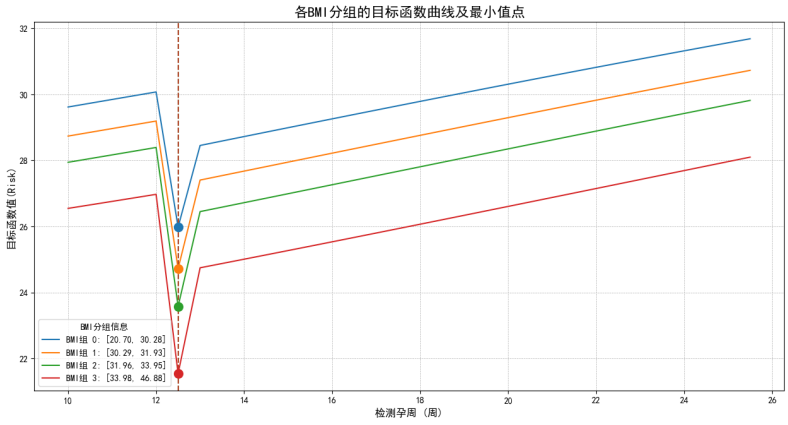

图 **10**　各 **BMI** 分组风险函数曲线以及其最小值

最终结果：各 BMI 组的目标函数最小值点 (即风险最低点)

表 **7**　**BMI** 分组优化结果

组	**BMI** 范围	代表 **BMI**	孕周	目标值
0	[20.70, 30.28]	29.11	12.5	25.98
1	[30.29, 31.93]	31.02	12.5	24.72
2	[31.96, 33.95]	32.92	12.5	23.58
3	[33.98, 46.88]	36.18	12.5	21.54

这个结果显示在 12.5 周附近各个 BMI 组的风险均很小，因此我们可以在临床建议孕妇在这个最小风险的检测孕周进行一次孕检.

第三步：接下来进行误差分析：“为响应问题二中‘分析检测误差对结果的影响’的要求，我们对检测误差进行了模拟与分析. 结果表明（如图所示），尽管检测过程无显著系统性偏差，但随机误差对单次检测的精确度有显著影响. 使用了下面的随机误差模拟：

误差模拟公式：

$$\text{measured} = \text{true} + \mathcal{N}(0, \text{true} \cdot \text{deviation})$$

量化指标总结:

表 **8**　误差模拟指标结果

偏差水平	**R** 平方 **(R²)**	均方根误差 **(RMSE)**	平均绝对百分比误差 **(MAPE %)**
5%	0.9857	0.0040	3.91
10%	0.9392	0.0083	7.83
15%	0.8536	0.0128	12.08

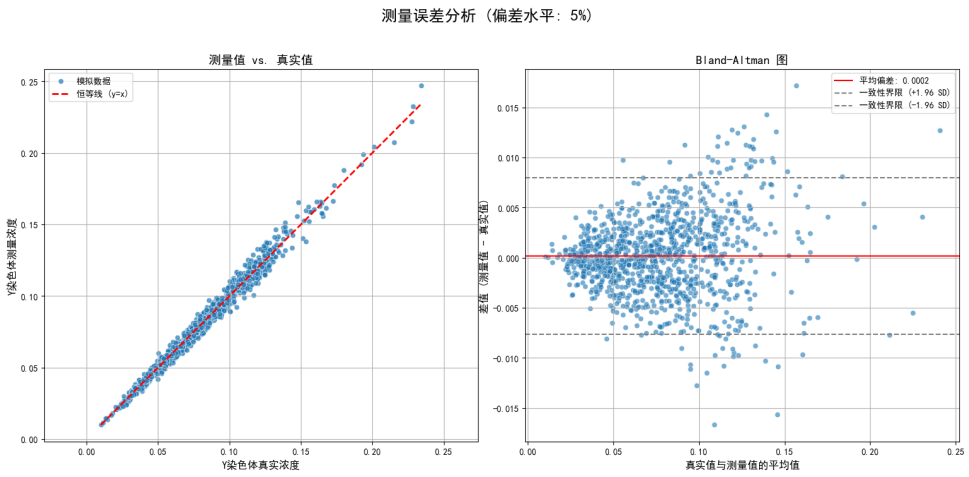

图 **11**　测量误差分析（**0.05**）

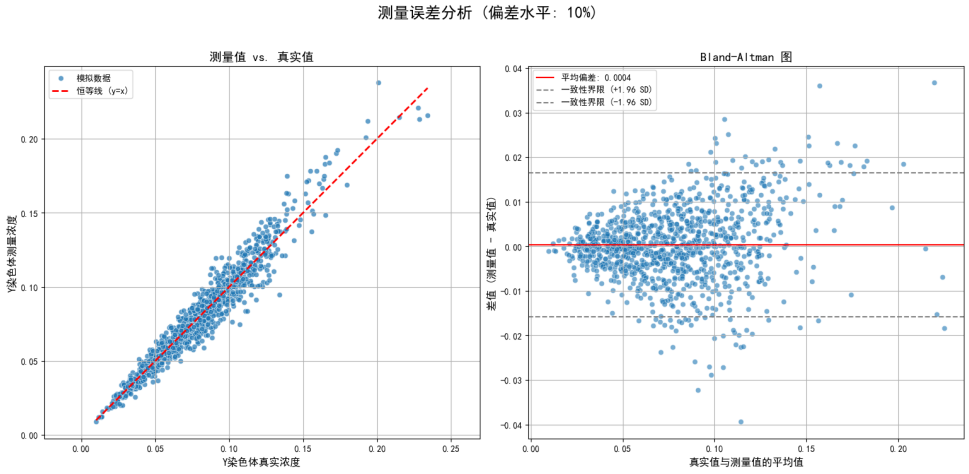

图 **12**　测量误差（**0.1**）

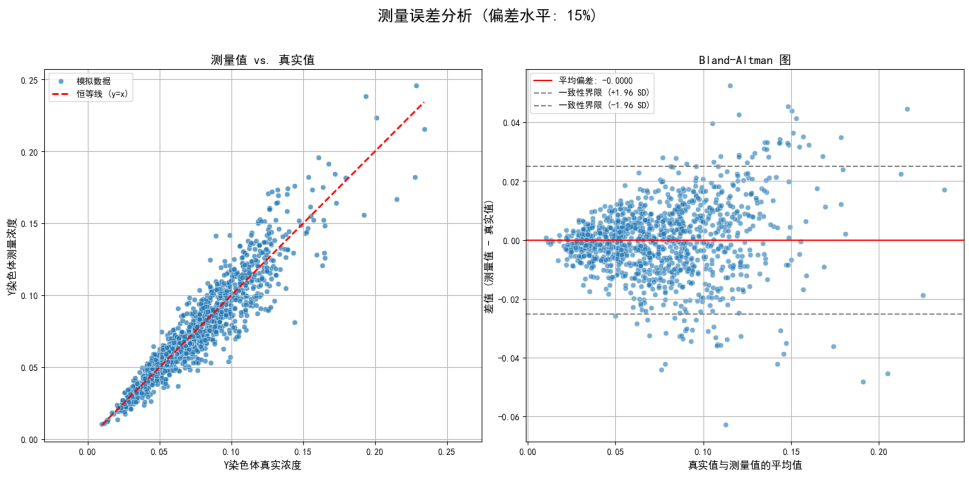

图 **13**　测量误差分析（**0.15**）

Bland-Altman 分析揭示，在 10% 的误差水平下，测量的 95% 一致性界限约为 ±1.6%（根据 RMSE × 1.96 估算）. 这意味着围绕 4% 的决策阈值存在一个显著的” 不确定区域”. 单纯以模型预测的平均浓度达到 4% 作为 NIPT 时点，会使孕妇承担因测量误差导致检测无效、需要复检而延误诊断的风险.

5.2.3 问题三的结论

我们得到了每组孕妇 BMI 最佳的 NIPT 时间节点为 12.5 周，并且这个模型对于检测误差的影响是比较小的，也证明了这个模型相较于第二问的只有 BMI 更具有稳健性.

BMI 较为合理的分组如下表所示：

表 **9** **BMI** 合理分组方案

分组	**BMI** 区间
第 1 组	$[20.70, 31.80]$
第 2 组	$[31.83, 35.96]$
第 3 组	$[36.05, 46.88]$

对于这个 BMI 分组进行的风险分析结果为 NIPT 最优的时间为 12.5 周进行，最后的检测误差分析证明了在检测水平的附近浮动可能会导致检测无效，最后让孕妇复检误诊的风险，因此应该严格按照 Y 染色体浓度为 4% 进行检测.

5.3 问题三模型的建立及求解

这道题目加入了许多变量来进行分析，但是实际上能够使用的并不多. 在这里，我们使用多重共线性变量筛查经过几轮迭代得到如下需要采纳的值：

这是保留的变量以及 VIF 值：

在这之后，操作与第二问类似，在我们建立模型的时需要引入上述的参数，分类需使用聚类分析，现在我们先根据这几个变量进行主成分分析.

表 **10**　变量与系数关系

变量	系数
孕妇 BMI	1.047 958
Y 染色体的 Z 值	1.294 948
X 染色体浓度	1.165 689
被过滤掉读段数的比例	1.101 016
生产次数	1.065 003

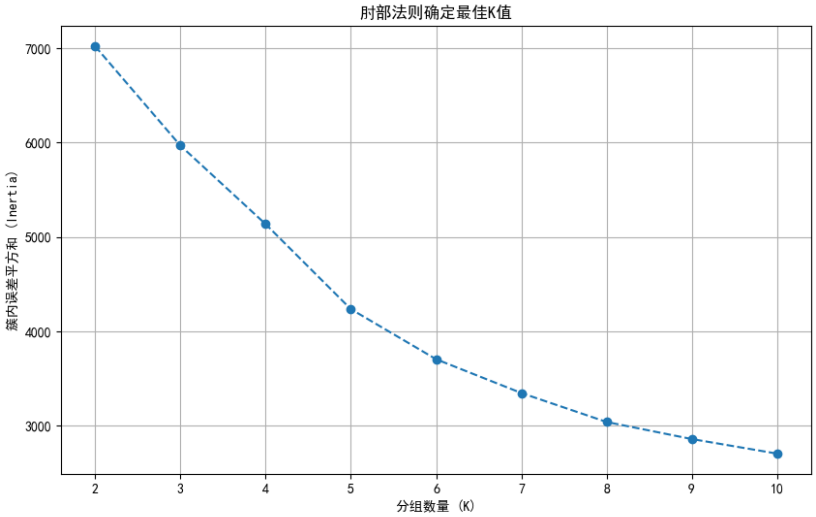

图 **14**　肘部确定法确定 **K** 值

根据得到的 K 值我们进行分类，取 K=5，得到的结果如下：

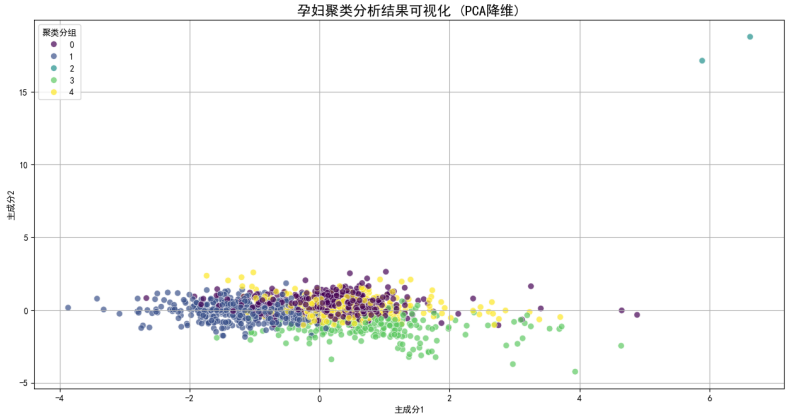

图 **15**　**PCA** 降维

各聚类分组的平均画像如下：

表 **11**　样本数据统计

孕妇 BMI	Y 染色体 Z 值	X 染色体浓度	过滤读段比例	生产次数	样本数量
31.35	0.20	0.02	0.02	1.22	431.00
31.77	−0.40	0.09	0.02	0.14	464.00
30.13	0.00	−0.02	0.18	1.00	2.00
37.28	0.57	0.03	0.02	0.29	251.00
31.03	0.23	0.00	0.02	0.00	538.00

注：数据保留两位小数精度
Z 值表示染色体浓度的标准化值
过滤读段比例指被过滤掉的测序读段占总读段的比例

我们发现有一组数据只有 2 个样本数量，因此我们去除掉这两个极端值，后续我们根据这些值进行计算分析.

根据问题一和问题二的过程我们得到以下公式：

$$\log(T) = 2.1514 - 0.0007 \times \text{BMI} + 6.095 \times \text{X 浓度} + 15.903 \times \text{过滤比例} + 0.086 \times \text{生产次数} + 0.588 \times 2.994$$

表 **12**　孕妇特征与预测检测时间分析

孕妇 BMI	X 浓度	过滤比例	生产次数	log(T) 中位数	预测 T(周)	95%CI 下限	95%CI 上限
31.35	0.02	0.02	1.22	4.4359	12.06	0.65	6.57
31.77	0.09	0.02	0.14	4.7690	16.83	0.91	9.16
37.28	0.03	0.02	0.29	4.4124	11.78	0.64	6.41
31.03	0.00	0.02	0.00	4.2088	9.61	0.52	5.23

这里用的是平均妇女的画像进行的，因此代表了一个簇分类里面的妇女的平均水平，具有代表性。

接着，我们进行风险分析，由上面的风险函数给出：

$$h(t;\alpha,\beta)=\frac{f(t;\alpha,\beta)}{S(t;\alpha,\beta)}=\frac{(\beta/\alpha)(t/\alpha)^{\beta-1}}{1+(t/\alpha)^{\beta}}$$

由此，我们通过多变量增强来达到我们的目标，但是主要还是以孕妇 BMI 为主的。并且我们还做了敏感性总结。

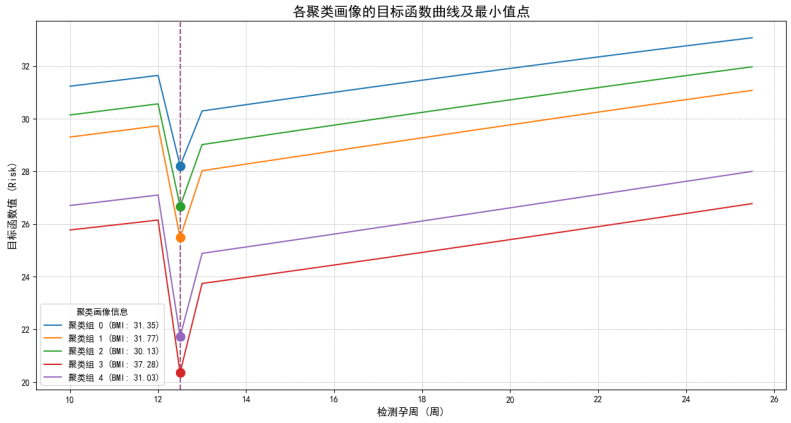

图 **16**　风险函数函数（聚类画像）

可以发现基本都在 12.5 周附近进行检测 NIPT。

最后我们来进行检测误差分析，在刚刚的风险函数分析中我们已经运用了这个误差，也就是下面这个表格：

表 **13**　敏感性分析总结：不同 R_f 取值下的目标孕周

聚类分组	$R_f = 10$	$R_f = 20$	$R_f = 30$
0	12.5	12.5	12.5
1	12.5	12.5	12.5
2	12.5	12.5	12.5
3	12.5	12.5	12.5
4	12.5	12.5	12.5

可以发现相对于第二问单独的以 BMI 进行建模，本问题的建模更加精准。

结论：我们得到了每组孕妇 BMI 最佳的 NIPT 时间节点为 12.5 周，并且这个模型对于检测误差的影响是比较小的，也证明了这个模型相较于第二问的只有 BMI 更具有稳健性

5.4 问题四模型的建立及求解

对女胎的数据进行处理，我们标准化 X 染色体浓度之后得到以下的前后对比图：

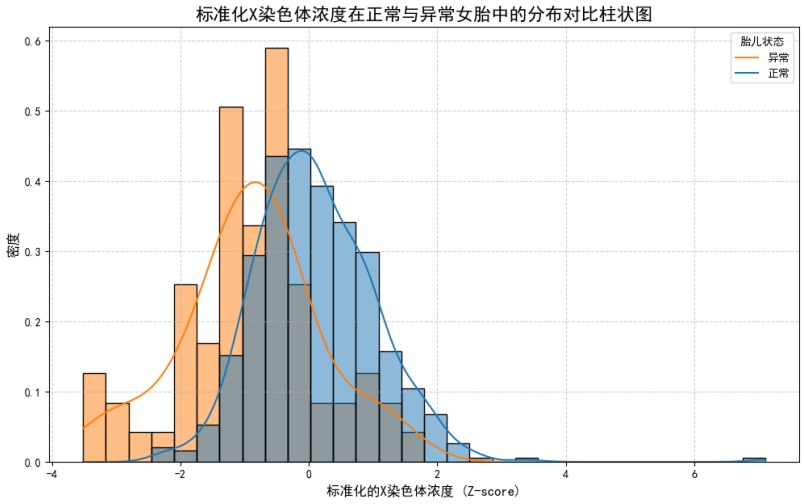

图 **17**　标准化 **X** 染色体分布对比图

利用随机森林模型对其进行建模，其主要公式如下：

单个决策树公式

对于第 t 棵决策树，预测函数为：

$$f_t(x) = \sum_{m=1}^{M} c_m \cdot \mathbb{I}(x \in R_m)$$

其中：

- **x**：输入特征向量
- R_m：第 m 个叶节点对应的区域

回归问题

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} f_t(\mathbf{x})$$

其中：

- T：树的数量
- $f_t(\mathbf{x})$：第 t 棵树的预测

分类问题

$$\hat{y} = \text{mode}\{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_T(\mathbf{x})\}$$

或基于概率：

$$P(y = k \mid \mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} P_t(y = k \mid \mathbf{x})$$

分类树（Gini 指数）

$$\text{Gini} = 1 - \sum_{k=1}^{K} p_{mk}^2$$

其中 p_{mk} 是第 m 个节点中第 k 类的比例.

信息增益分类树（信息增益）

$$\text{Entropy} = -\sum_{k=1}^{K} p_{mk} \log(p_{mk})$$

随机性引入机制

Bagging（Bootstrap Aggregating）从原始数据集中有放回地抽取 N 个样本：

$$D_t = \{(x_i, y_i)\}_{i=1}^{N} \sim \text{Bootstrap}(D)$$

特征重要性计算

基于不纯度减少

$$\text{Importance}_j = \frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{\text{node } m \\ \text{using feature } j}} \Delta\text{Impurity}_m$$

基于排列重要性

$$\text{Importance}_j = \frac{1}{T} \sum_{t=1}^{T} \left(\text{Error}_t - \text{Error}_{t,j}^{\text{permuted}} \right)$$

六、 模型的分析与检验

6.1 检测阈值的敏感性分析

本小节主要分析检测阈值对合适的孕周预测的影响，检验当临床上用于判断检测是否可靠的“Y 染色体浓度”标准发生微小变动时，模型的推荐孕周是否稳定。可以观察到检测阈值的变动对推荐孕周有显著影响。尤其是对于“聚类组 1”，影响尤为剧烈，阈值从 3.5% 提升至 4.5%，推荐孕周提前了超过 3 周。对于其他分组，阈值每提高 0.5 个百分点，推荐的检测时间点也普遍提前 0.5 至 1 周。这表明模型对检测标准的变化反应非常灵敏，临床上设定的具体阈值标准，将是决定最佳检测时点的关键因素。

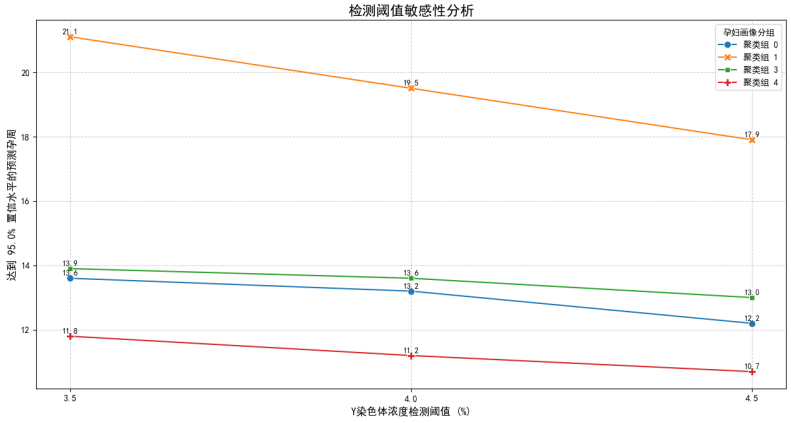

图 **18**　检测阈值敏感值分析

固定置信水平 = 95.0

表 **14**　聚类分组在不同 **Y** 染色体浓度阈值下的孕周分析

聚类分组　**画像平均 BMI**		**达标孕周（周）**		
		阈值 =3.5%	**阈值 =4.0%**	**阈值 =4.5%**
聚类组 0	31.35	13.6	13.2	12.2
聚类组 1	31.77	21.1	19.5	17.9
聚类组 3	37.28	13.9	13.6	13.0
聚类组 4	31.03	11.8	11.2	10.7

6.2 对孕妇 BMI 分组的敏感性分析

这里我们将原来的四分位改为 50 分位，来观察 BMI 分组不同是否会对检测结果有非常明显的影响。

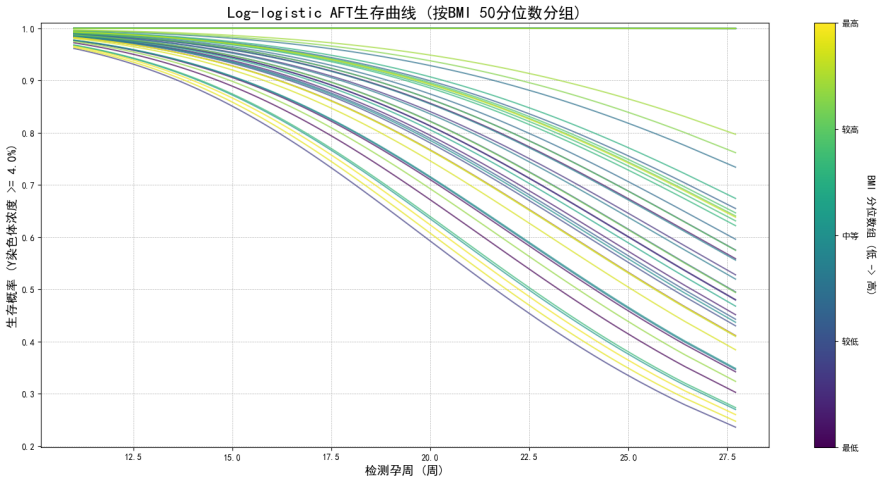

图 **19**　**BMI** 敏感性分析

我们可以观察到，对于这么多细分的 BMI 组别其差异性很大，并且都是平滑而没有阶梯式下降的情况出现，曲线与曲线之间基本没有重合的情况；BMI 最低和最高的孕妇群体之间，达到同样成功率所需的时间差异非常明显，这证明了模型预测结果对输入变量 BMI 的敏感程度。

七、 模型的评价

注意: Cox PH 模型的 AIC 是部分 AIC(Partial AIC)，不应直接与其他模型的 AIC 进行数值比较，但可以反映其相对优劣.

7.1 模型的优点

如表：

表 **15**　生存分析模型性能比较 **(AIC & BIC)**

模型	AIC	BIC
Log-Logistic AFT	1161.891 683	1165.604 607
Weibull AFT	1165.561 423	1169.274 347
Cox PH	1392.221 062	–

AIC 和 BIC 值越低，模型的综合性能越好. 由于有结论: AIC 和 BIC 值越低，模型的综合性能越好。因此我们观察表格中，可以发现我们选取的 Log-Logistic AFT 模型具有更低的 AIC 和 BIC 值，因此这个模型相对其他同类型的模型具有更大的优势。

直观的物理解释与临床契合度高 AFT 模型的核心优势在于其参数具有清晰、直观的物理解释。模型系数直接量化了协变量（如 BMI）对事件发生时间（Y 染色体浓度达标）的“加速”或“延缓”效应。例如，BMI 的系数直接反映了其每增加一个单位，将会使达到检测标准所需的平均时间延长或缩短一个特定的比例。这种解释方式完全契合本研究的生物学假设——高 BMI 可能通过血液稀释等效应，系统性地“延缓”了胎儿游离 DNA 在母体血液中的积累过程，而不是仅仅在某个时间点增加“失败”的瞬时风险。

灵活的风险函数形态 Log-logistic 分布所对应的风险函数呈现非单调的“拱形”特征，即风险率先增后降。这种形态能够灵活地捕捉到许多生物学过程的动态特征，比单调风险函数（如 Weibull 分布）的假设更具普适性。在我们的 AIC/BIC 模型比较中，Log-logistic AFT 模型也展现出更优的拟合效果，证明了其更符合本研究数据的内在规律。

无需严苛的比例风险假设与经典的 Cox 比例风险模型不同，AFT 模型无需满足“比例风险”这一严苛假设。比例风险假设要求协变量对风险比的影响不随时间变化，但在许多医学场景中这一假设难以成立。AFT 模型提供了一个更为稳健和灵活的替代方案，使其在复杂的真实世界数据中具有更强的适用性。强大的个体化预测能力作为一个完全参数化的模型，Log-logistic AFT 模型能够为任意一组给定的孕妇特征（如特定的 BMI 值），预测其完整的生存曲线以及达到任意置信水平（如 95%）所需的精确孕周。这正是我们进行敏感性分析和提供最终临床建议的基础。这种强大的个体化预测能力，是 Kaplan-Meier 等非参数方法所不具备的。

7.2 模型的缺点

对分布假设的依赖性作为参数化模型，其最主要的局限性在于结果的准确性依赖于所选概率分布（在此为 Log-logistic 分布）的正确性。尽管模型比较显示该分布是相对最优的，但这仍然是一个核心假设。如果真实的数据生成过程与 Log-logistic 分布存在较大偏差，模型的预测可能会产生系统性误差。

对数据质量和完整性的敏感性模型的性能高度依赖于输入数据的质量。样本中的测量误差、数据缺失以及未被纳入模型的潜在混杂因素（如孕妇的特定饮食习惯、胎盘功能异常等），都可能影响模型参数估计的准确性和最终预测的可靠性。

外推预测的风险本模型是基于特定孕周范围和临床特征的数据进行训练的。将其用于预测超出训练数据范围的极端情况（例如，极早期孕周或具有极端 BMI 值的孕妇）时，其准确性可能下降。因此，在模型适用范围之外进行外推预测需要格外谨慎。

八、 模型的推广

推广至其他生物标志物研究本研究采用的“时间-事件”分析框架并不局限于 NIPT 中的 Y 染色体浓度。任何随时间累积或衰减、并需要达到某一关键阈值的生物标志物研究，都可以借鉴此方法。例如：

其他产前筛查：分析其他染色体非整倍体筛查标志物的浓度变化规律。

疾病监控：预测妊娠期高血压等疾病相关标志物达到警戒水平的时间。

肿瘤学：在癌症治疗中，预测肿瘤标志物下降到安全水平所需的时间。

参考文献

部分代码由 AI 生成（Deepseek-R1 671b），请参考支撑材料内每天的对话日志.

[1] Bennett, S. (1983). Log-logistic regression models for survival data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **32**(2), 165–171.

[2] Rich, J. T., Neely, J. G., Paniello, R. C., Voelker, C. C., Nussenbaum, B., & Wang, E. W. (2010). A practical guide to understanding Kaplan-Meier curves. *Otolaryngology—Head and Neck Surgery*, **143**(3), 331–336.

附录 A　问题一相关代码

```
#孕周数的处理代码块
def parse_preg_week(week_str):
    # 孕周字符串转换为浮点数周数
    try:
        if isinstance(week_str, str) and 'w+' in week_str:
            parts = week_str.split('w+')
            weeks = int(parts[0])
            days = int(parts[1])
            return round(weeks + days / 7.0, 2)
        return float(week_str)
    except (ValueError, TypeError, IndexError):
        return np.nan
#可视化
#问题一的散点图
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os

#这里是对孕周数的处理

def visualize_male_data(excel_filepath):

    # 设置中文字体，解决乱码问题
    plt.rcParams['font.sans-serif'] = ['SimHei'] # 'SimHei' 是黑体
    plt.rcParams['axes.unicode_minus'] = False # 解决负号'-'显示为方块的问题
    print("步骤1: 已设置中文字体 'SimHei' 以防止图表乱码.")
    print("提示: 如果您的电脑没有'SimHei'字体，可以替换为'Microsoft
        YaHei'(微软雅黑)等其他中文字体.")

    # 加载数据
    try:
        df = pd.read_excel(excel_filepath, sheet_name='男胎检测数据')
    except Exception as e:

        return

    df.columns = df.columns.str.strip()

    # 定义研究所需的列
    required_cols = ['孕妇BMI', '检测孕周', 'Y染色体浓度']

    # 检查所需列是否存在
```

```
    if not all(col in df.columns for col in required_cols):
        return

    # 转换孕周格式
    df['检测孕周(周)'] = df['检测孕周'].apply(parse_preg_week)
    print(" - '检测孕周' 已转换为数值型 '检测孕周(周)'.")

    # 筛选出用于可视化的列，并确保它们是数值类型
    plot_df = df[['孕妇BMI', '检测孕周(周)', 'Y染色体浓度']].copy()
    for col in plot_df.columns:
        plot_df[col] = pd.to_numeric(plot_df[col], errors='coerce')

    # 删除包含空值的行，确保绘图数据是干净的
    initial_rows = len(plot_df)
    plot_df.dropna(inplace=True)
    rows_dropped = initial_rows - len(plot_df)

    # 创建并显示可视化图表
    if plot_df.empty:
        return

    # 创建一个1行2列的图表画布
    fig, axes = plt.subplots(1, 2, figsize=(16, 6))

    # 设置图表整体标题
    fig.suptitle('孕妇BMI及检测孕周与Y染色体浓度的关系分析', fontsize=16)

    # 第一个子图：孕妇BMI vs Y染色体浓度
    sns.scatterplot(ax=axes[0], data=plot_df, x='孕妇BMI', y='Y染色体浓度', alpha=0.6)
    axes[0].set_title('孕妇BMI 与 Y染色体浓度的关系')
    axes[0].set_xlabel('孕妇BMI')
    axes[0].set_ylabel('Y染色体浓度')
    axes[0].grid(True, linestyle='--', alpha=0.6)

    # 第二个子图：检测孕周 vs Y染色体浓度
    sns.scatterplot(ax=axes[1], data=plot_df, x='检测孕周(周)', y='Y染色体浓度', alpha=0.6)
    axes[1].set_title('检测孕周 与 Y染色体浓度的关系')
    axes[1].set_xlabel('检测孕周 (周)')
    axes[1].set_ylabel('Y染色体浓度')
    axes[1].grid(True, linestyle='--', alpha=0.6)

    # 调整布局并显示图表
    plt.tight_layout(rect=[0, 0, 1, 0.96]) # 为总标题留出空间
    plt.show()


if __name__ == '__main__':
```

```
    excel_filepath = r'E:\VSCode\Coding\Project\附件.xlsx'

    visualize_male_data(excel_filepath)
#问题一的箱线图
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os

#这里是对孕周数的处理

def visualize_male_data(excel_filepath):
    # 设置中文字体，解决乱码问题
    plt.rcParams['font.sans-serif'] = ['SimHei'] # 'SimHei' 是黑体
    plt.rcParams['axes.unicode_minus'] = False # 解决负号'-'显示为方块的问题

    # 加载数据
    try:
        df = pd.read_excel(excel_filepath, sheet_name='男胎检测数据')

    except Exception as e:
        return

    # 清理列名中的空格
    df.columns = df.columns.str.strip()

    # 定义研究所需的列
    required_cols = ['孕妇BMI', '检测孕周', 'Y染色体浓度']

    # 检查所需列是否存在
    if not all(col in df.columns for col in required_cols):
        print(f"错误：数据中缺少必要的列.请确保包含以下列：{required_cols}")
        return

    # 转换孕周格式
    df['检测孕周(周)'] = df['检测孕周'].apply(parse_preg_week)

    # 筛选出用于可视化的列，并确保它们是数值类型
    plot_df = df[['孕妇BMI', '检测孕周(周)', 'Y染色体浓度']].copy()
    for col in plot_df.columns:
        plot_df[col] = pd.to_numeric(plot_df[col], errors='coerce')

    # 删除包含空值的行，确保绘图数据是干净的
    initial_rows = len(plot_df)
    plot_df.dropna(inplace=True)
    rows_dropped = initial_rows - len(plot_df)
```

```
    if plot_df.empty:
        return

    # 新增：为制作箱线图，对连续变量进行分组
    plot_df['BMI分组'] = pd.qcut(plot_df['孕妇BMI'], q=4, labels=['Q1(较低)', 'Q2', 'Q3',
        'Q4(较高)'])
    plot_df['孕周分组'] = plot_df['检测孕周(周)'].astype(int)
    print(" - 已将'孕妇BMI'按四分位数分组，将'检测孕周'按整数周分组，用于制作箱线图.")


    # 创建一个1行2列的图表画布
    fig, axes = plt.subplots(1, 2, figsize=(16, 7))

    # 设置图表整体标题
    fig.suptitle('孕妇BMI及检测孕周分组下Y染色体浓度的分布', fontsize=16)

    # 第一个子图：孕妇BMI分组 vs Y染色体浓度
    sns.boxplot(ax=axes[0], data=plot_df, x='BMI分组', y='Y染色体浓度')
    axes[0].set_title('不同BMI分组下Y染色体浓度的分布')
    axes[0].set_xlabel('孕妇BMI (按四分位数分组)')
    axes[0].set_ylabel('Y染色体浓度')
    axes[0].grid(True, linestyle='--', alpha=0.6, axis='y')

    # 第二个子图：检测孕周分组 vs Y染色体浓度
    sns.boxplot(ax=axes[1], data=plot_df.sort_values('孕周分组'), x='孕周分组', y='Y染色体浓度')
    axes[1].set_title('不同检测孕周下Y染色体浓度的分布')
    axes[1].set_xlabel('检测孕周 (周)')
    axes[1].set_ylabel('Y染色体浓度')
    axes[1].grid(True, linestyle='--', alpha=0.6, axis='y')

    # 调整布局并显示图表
    plt.tight_layout(rect=[0, 0, 1, 0.95]) # 为总标题留出空间
    plt.show()


if __name__ == '__main__':
    excel_filepath = r'E:\VSCode\Coding\Projec\附件.xlsx'

    visualize_male_data(excel_filepath)


#问题一的三个变量的基本统计学
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import os

#这里是对孕周数的处理

def visualize_and_analyze_male_data(excel_filepath):

    # 设置中文字体，解决乱码问题
    # Matplotlib 默认不支持中文，需要设置支持中文的字体
    plt.rcParams['font.sans-serif'] = ['SimHei'] # 'SimHei' 是黑体
    plt.rcParams['axes.unicode_minus'] = False # 解决负号'-'显示为方块的问题

    # 加载数据
    try:
        df = pd.read_excel(excel_filepath, sheet_name='男胎检测数据')
    except Exception as e:
        return

    df.columns = df.columns.str.strip()

    # 定义研究所需的列
    required_cols = ['孕妇BMI', '检测孕周', 'Y染色体浓度']

    # 检查所需列是否存在
    if not all(col in df.columns for col in required_cols):
        print(f"错误：数据中缺少必要的列.请确保包含以下列：{required_cols}")
        return

    # 转换孕周格式
    df['检测孕周(周)'] = df['检测孕周'].apply(parse_preg_week)
    print(" - '检测孕周' 已转换为数值型 '检测孕周(周)'.")

    # 筛选出用于可视化的列，并确保它们是数值类型
    plot_df = df[['孕妇BMI', '检测孕周(周)', 'Y染色体浓度']].copy()
    for col in plot_df.columns:
        plot_df[col] = pd.to_numeric(plot_df[col], errors='coerce')

    initial_rows = len(plot_df)
    plot_df.dropna(inplace=True)
    rows_dropped = initial_rows - len(plot_df)

    if plot_df.empty:
        return

    # 使用 describe() 获取大部分统计数据
    desc_stats = plot_df.describe()

    # 计算方差并添加到结果中
```

```
    variance = plot_df.var()
    desc_stats.loc['方差'] = variance

    # 计算众数并添加到结果中
    mode = plot_df.mode().iloc[0]
    desc_stats.loc['众数'] = mode

    # 调整列名以便更好地展示
    desc_stats = desc_stats.rename(index={
        'count': '样本数',
        'mean': '平均数',
        'std': '标准差',
        'min': '最小值',
        '25%': '25%分位数',
        '50%': '中位数(50%)',
        '75%': '75%分位数',
        'max': '最大值'
    })

    print(desc_stats.transpose())


if __name__ == '__main__':
    excel_filepath = r'E:\VSCode\Coding\Project\附件.xlsx'

    visualize_and_analyze_male_data(excel_filepath)


#问题一的皮尔逊相关指数
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os

#这里是对孕周数的处理

def analyze_specific_correlations(excel_filepath):

    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False

    # 加载数据
    try:
        df = pd.read_excel(excel_filepath, sheet_name='男胎检测数据')
    except Exception as e:
        return
```

```
    # 数据预处理
    df.columns = df.columns.str.strip()

    # 对特定列进行数值化转换
    df['检测孕周(周)'] = df['检测孕周'].apply(parse_preg_week)

    # 定义并选取要分析的特定列
    target_cols = ['孕妇BMI', '检测孕周(周)', 'Y染色体浓度']

    # 检查所需列是否存在
    if not all(col in df.columns for col in target_cols):
        return

    analysis_df = df[target_cols].copy()

    # 清理缺失值
    initial_rows = len(analysis_df)
    analysis_df.dropna(inplace=True)
    rows_dropped = initial_rows - len(analysis_df)

    if analysis_df.empty:
        return

    correlation_matrix = analysis_df.corr(method='pearson')

    print(correlation_matrix)
    plt.figure(figsize=(8, 6))

    sns.heatmap(correlation_matrix, cmap='coolwarm', annot=True, fmt=".3f", linewidths=.5)

    #图生成
    plt.title('Y染色体浓度、检测孕周、孕妇BMI的相关性热力图', fontsize=16)
    plt.xticks(rotation=0)
    plt.yticks(rotation=0)
    plt.show()


if __name__ == '__main__':
    excel_filepath = r'E:\VSCode\Coding\Project\附件.xlsx'

    analyze_specific_correlations(excel_filepath)
```

第四步中使用的代码：

```
# 问题一的BMI四分位求解
import pandas as pd
```

```python
import numpy as np

def find_quartiles_from_excel(excel_filepath):
    try:
        df = pd.read_excel(excel_filepath, sheet_name='男胎检测数据')
        return

    df.columns = df.columns.str.strip()
    if '孕妇BMI' not in df.columns:
        return

    bmi_series = pd.to_numeric(df['孕妇BMI'], errors='coerce').dropna()

    q1 = bmi_series.quantile(0.25)
    q2 = bmi_series.quantile(0.50)
    q3 = bmi_series.quantile(0.75)

    print(f"第一四分位数 (Q1, 25%): {q1:.4f}")
    print(f"第二四分位数 (Q2, 50% - 中位数): {q2:.4f}")
    print(f"第三四分位数 (Q3, 75%): {q3:.4f}")

    quartiles = bmi_series.quantile([0.25, 0.5, 0.75])
    print("\n一次性计算结果:")
    print(quartiles)

    description = bmi_series.describe()
    print("'.describe()' 方法的输出包含了四分位数 (25%, 50%, 75%):")
    print(description)


if __name__ == '__main__':
    excel_filepath = r'E:\VSCode\Coding\Project\附件.xlsx'

    find_quartiles_from_excel(excel_filepath)
```

第五步中使用的代码：

```python
#不同BMI组的K-M曲线可视化
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# 导入生存分析库
try:
    from lifelines import KaplanMeierFitter
except ImportError:
```

```
    exit()

#这里是对孕周数的处理

def visualize_km_curves(excel_filepath, threshold=0.04):
    # 设置和加载数据
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False

    try:
        df_raw = pd.read_excel(excel_filepath, sheet_name='男胎检测数据')
    except Exception as e:
        return

    df_raw.columns = df_raw.columns.str.strip()
    essential_cols = ['检测孕周', 'Y染色体浓度', '孕妇BMI']
    if not all(col in df_raw.columns for col in essential_cols):
        print(f"错误：数据中缺少必要的列：{essential_cols}")
        return

    analysis_df = df_raw[essential_cols].copy()

    analysis_df['T'] = analysis_df['检测孕周'].apply(parse_preg_week)
    analysis_df['E'] = (analysis_df['Y染色体浓度'] < threshold).astype(int)

    analysis_df['T'] = pd.to_numeric(analysis_df['T'], errors='coerce')
    analysis_df['孕妇BMI'] = pd.to_numeric(analysis_df['孕妇BMI'], errors='coerce')

    initial_rows = len(analysis_df)
    analysis_df.dropna(subset=['T', 'E', '孕妇BMI'], inplace=True)
    rows_dropped = initial_rows - len(analysis_df)

    # 根据我们指定的百分位数进行分组
    quartiles = [30.208806, 31.811598, 33.926237]
    labels = ['Q1(最低25%)', 'Q2(25-50%)', 'Q3(50-75%)', 'Q4(最高25%)']
    analysis_df['BMI分组'] = pd.cut(analysis_df['孕妇BMI'],
                                bins=[-np.inf, quartiles[0], quartiles[1], quartiles[2], np.inf],
                                labels=labels, right=True)

    analysis_df.dropna(subset=['BMI分组'], inplace=True)

    # 拟合KM模型并绘图
    plt.figure(figsize=(12, 8))
    ax = plt.gca() # 获取当前的坐标轴

    # 为每个BMI分组拟合KM模型并绘图
    for group_name in sorted(analysis_df['BMI分组'].unique()):
```

```
        group_data = analysis_df[analysis_df['BMI分组'] == group_name]
        kmf = KaplanMeierFitter().fit(group_data['T'], group_data['E'], label=group_name)
        kmf.plot_survival_function(ax=ax, ci_show=True) # ci_show=True 会显示置信区间

    # 格式化图表
    ax.set_title('Kaplan-Meier生存曲线 (按四分位数BMI分组)', fontsize=16)
    ax.set_xlabel('检测孕周 (周)', fontsize=12)
    ax.set_ylabel(f'生存概率 (Y染色体浓度 >= {threshold*100}%)', fontsize=12)
    ax.legend(title="BMI 分组")
    ax.grid(True, linestyle='--', alpha=0.6)

    plt.tight_layout()
    plt.show()

if __name__ == '__main__':
    excel_filepath = r'E:\VSCode\Coding\Project\附件.xlsx'
    visualize_km_curves(excel_filepath)
```

第六步中使用的代码：

```
#问题一 Weibull-AFT
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os

# 导入生存分析库
try:
    from lifelines import WeibullAFTFitter
except ImportError:
    exit()

#

def perform_weibull_aft_analysis(excel_filepath, threshold=0.04):

    # 设置中文字体
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    print("步骤1: 已设置中文字体 'SimHei'.")

    # 加载和预处理数据
    try:
        print(f"\n步骤2: 正在从 '{excel_filepath}' 加载 '男胎检测数据'...")
        df = pd.read_excel(excel_filepath, sheet_name='男胎检测数据')
    except Exception as e:
```

```
    return

df.columns = df.columns.str.strip()
df['检测孕周(周)'] = df['检测孕周'].apply(parse_preg_week)

target_cols = ['孕妇BMI', '检测孕周(周)', 'Y染色体浓度']
analysis_df = df[target_cols].copy().dropna().reset_index(drop=True)
print(f" - 数据预处理完成，剩余 {len(analysis_df)} 条有效数据.")

# 定义生存变量和BMI分组
analysis_df['T'] = analysis_df['检测孕周(周)']
analysis_df['E'] = (analysis_df['Y染色体浓度'] < threshold).astype(int)

quartiles = [30.208806, 31.811598, 33.926237] # Q1 (25%), Q2 (50%), Q3 (75%)

# 使用简洁的标签以避免列名中出现特殊字符
simple_labels = ['Q1', 'Q2', 'Q3', 'Q4']
analysis_df['BMI分组'] = pd.cut(
    analysis_df['孕妇BMI'],
    bins=[-np.inf, quartiles[0], quartiles[1], quartiles[2], np.inf],
    labels=simple_labels
)

# 构建并拟合 Weibull AFT 模型
regression_df = pd.get_dummies(analysis_df, columns=['BMI分组'], drop_first=True,
    dtype=float)
cols_for_fit = ['T', 'E'] + [col for col in regression_df.columns if 'BMI分组_' in col]

aft = WeibullAFTFitter()
aft.fit(regression_df[cols_for_fit], duration_col='T', event_col='E')
aft.print_summary(decimals=3)

# 预测并绘制生存曲线
plt.figure(figsize=(12, 8))

label_map = {
    'Q1': f'Q1 (BMI   {quartiles[0]:.2f})',
    'Q2': f'Q2 ({quartiles[0]:.2f} < BMI   {quartiles[1]:.2f})',
    'Q3': f'Q3 ({quartiles[1]:.2f} < BMI   {quartiles[2]:.2f})',
    'Q4': f'Q4 (BMI > {quartiles[2]:.2f})'
}

dummy_cols = [col for col in aft.params_.index.get_level_values('covariate') if 'BMI分组_'
    in col]

for group_label in simple_labels:
    X_test = pd.DataFrame(np.zeros((1, len(dummy_cols))), columns=dummy_cols)
```

```
        if group_label != 'Q1':
            group_col_name = f"BMI分组_{group_label}"
            if group_col_name in X_test.columns:
                X_test[group_col_name] = 1

        survival_function = aft.predict_survival_function(X_test)

        descriptive_label = label_map[group_label]
        plt.plot(survival_function.index, survival_function.values.flatten(),
            label=descriptive_label)

    plt.title('Weibull-AFT模型预测的生存曲线 (按BMI分组)')
    plt.xlabel('检测孕周 (周)')
    plt.ylabel(f'生存概率 (Y染色体浓度 >= {threshold*100}%)')
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.legend(title="BMI 分组")
    plt.show()


if __name__ == '__main__':
    excel_filepath = r'E:\VSCode\Coding\Project\附件.xlsx'
    perform_weibull_aft_analysis(excel_filepath)

ll aft:

#问题一 Log-logistic AFT
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os

# 导入生存分析库
try:
    from lifelines import LogLogisticAFTFitter
except ImportError:
    exit()

#这里是孕周数据处理

def perform_loglogistic_aft_analysis(excel_filepath, threshold=0.04):
    # 设置中文字体
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    print("步骤1: 已设置中文字体 'SimHei'.")
```

```
# 加载和预处理数据
try:
    print(f"\n步骤2: 正在从 '{excel_filepath}' 加载 '男胎检测数据'...")
    df = pd.read_excel(excel_filepath, sheet_name='男胎检测数据')
except Exception as e:
    return

df.columns = df.columns.str.strip()
df['检测孕周(周)'] = df['检测孕周'].apply(parse_preg_week)

target_cols = ['孕妇BMI', '检测孕周(周)', 'Y染色体浓度']
analysis_df = df[target_cols].copy().dropna().reset_index(drop=True)

# 定义生存变量和BMI分组
analysis_df['T'] = analysis_df['检测孕周(周)']
analysis_df['E'] = (analysis_df['Y染色体浓度'] < threshold).astype(int)
print(f" - 已定义事件（Y染色体浓度 < {threshold}）.共发生 {analysis_df['E'].sum()} 例事件.")

quartiles = [30.208806, 31.811598, 33.926237] # 您提供的固定百分位数

# 使用简洁的标签以避免列名中出现特殊字符
simple_labels = ['Q1', 'Q2', 'Q3', 'Q4']
analysis_df['BMI分组'] = pd.cut(
    analysis_df['孕妇BMI'],
    bins=[-np.inf, quartiles[0], quartiles[1], quartiles[2], np.inf],
    labels=simple_labels
)

# 构建并拟合 Log-logistic AFT 模型
regression_df = pd.get_dummies(analysis_df, columns=['BMI分组'], drop_first=True,
    dtype=float)
cols_for_fit = ['T', 'E'] + [col for col in regression_df.columns if 'BMI分组_' in col]

# 使用 LogLogisticAFTFitter
aft = LogLogisticAFTFitter()
aft.fit(regression_df[cols_for_fit], duration_col='T', event_col='E')

# 预测并绘制生存曲线
plt.figure(figsize=(12, 8))

label_map = {
    'Q1': f'Q1 (BMI   {quartiles[0]:.2f})',
    'Q2': f'Q2 ({quartiles[0]:.2f} < BMI   {quartiles[1]:.2f})',
    'Q3': f'Q3 ({quartiles[1]:.2f} < BMI   {quartiles[2]:.2f})',
    'Q4': f'Q4 (BMI > {quartiles[2]:.2f})'
}
```

```
    dummy_cols = [col for col in aft.params_.index.get_level_values('covariate') if 'BMI分组_'
        in col]

    for group_label in simple_labels:
        X_test = pd.DataFrame(np.zeros((1, len(dummy_cols))), columns=dummy_cols)

        if group_label != 'Q1':
            group_col_name = f"BMI分组_{group_label}"
            if group_col_name in X_test.columns:
                X_test[group_col_name] = 1

        survival_function = aft.predict_survival_function(X_test)

        descriptive_label = label_map[group_label]
        plt.plot(survival_function.index, survival_function.values.flatten(),
            label=descriptive_label)

    plt.title('Log-logistic AFT模型预测的生存曲线 (按BMI分组)')
    plt.xlabel('检测孕周 (周)')
    plt.ylabel(f'生存概率 (Y染色体浓度 >= {threshold*100}%)')
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.legend(title="BMI 分组")
    plt.show()


if __name__ == '__main__':
    excel_filepath = r'E:\VSCode\Coding\Project\附件.xlsx'
    perform_loglogistic_aft_analysis(excel_filepath)
```

Kaplan-Meier:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os

# 导入生存分析库
try:
    from lifelines import (
        KaplanMeierFitter,
        WeibullAFTFitter,
        LogLogisticAFTFitter,
        CoxPHFitter
    )
except ImportError:
```

```
    exit()

#这里是孕周数的数据处理

def validate_and_compare_models(excel_filepath, threshold=0.04):
    # 设置和加载数据
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False

    try:
        df = pd.read_excel(excel_filepath, sheet_name='男胎检测数据')
    except Exception as e:
        print(f"错误: 加载Excel文件失败.错误信息: {e}")
        return

    df.columns = df.columns.str.strip()
    df['检测孕周(周)'] = df['检测孕周'].apply(parse_preg_week)
    target_cols = ['孕妇BMI', '检测孕周(周)', 'Y染色体浓度']
    analysis_df = df[target_cols].copy().dropna().reset_index(drop=True)

    analysis_df['T'] = analysis_df['检测孕周(周)']
    analysis_df['E'] = (analysis_df['Y染色体浓度'] < threshold).astype(int)

    quartiles = [30.208806, 31.811598, 33.926237]
    simple_labels = ['Q1', 'Q2', 'Q3', 'Q4']
    analysis_df['BMI分组'] = pd.cut(
        analysis_df['孕妇BMI'],
        bins=[-np.inf, quartiles[0], quartiles[1], quartiles[2], np.inf],
        labels=simple_labels
    )

    # 拟合所有模型
    regression_df = pd.get_dummies(analysis_df, columns=['BMI分组'], drop_first=True,
        dtype=float)
    cols_for_fit = ['T', 'E'] + [col for col in regression_df.columns if 'BMI分组_' in col]

    kmf = KaplanMeierFitter()
    wft = WeibullAFTFitter().fit(regression_df[cols_for_fit], 'T', 'E')
    llf = LogLogisticAFTFitter().fit(regression_df[cols_for_fit], 'T', 'E')
    cph = CoxPHFitter().fit(regression_df[cols_for_fit], 'T', 'E')

    #可视化比较: 模型拟合优度
    print("\n步骤3: 正在生成拟合优度对比图...")
    plt.figure(figsize=(14, 9))

    for i, group in enumerate(simple_labels):
        ax = plt.subplot(2, 2, i + 1)
```

```
        # 绘制KM曲线 (黄金标准)
        group_data = analysis_df[analysis_df['BMI分组'] == group]
        kmf.fit(group_data['T'], group_data['E'], label=f'Kaplan-Meier ({group})')
        kmf.plot_survival_function(ax=ax, ci_show=False)

        # 准备用于预测的测试数据
        dummy_cols = [col for col in wft.params_.index.get_level_values('covariate') if
            'BMI分组_' in col]
        X_test = pd.DataFrame(np.zeros((1, len(dummy_cols))), columns=dummy_cols)
        if group != 'Q1':
            X_test[f'BMI分组_{group}'] = 1

        # 叠加Weibull模型的预测曲线
        wft.predict_survival_function(X_test).rename(columns={0: 'Weibull AFT'}).plot(ax=ax)

        # 叠加Log-Logistic模型的预测曲线
        llf.predict_survival_function(X_test).rename(columns={0: 'Log-Logistic AFT'}).plot(ax=ax)

        plt.title(f'BMI分组: {group}')
        plt.xlabel('检测孕周 (周)')
        plt.ylabel('生存概率')
        plt.legend()
        plt.grid(True, linestyle='--', alpha=0.6)

    plt.suptitle('模型预测生存曲线与Kaplan-Meier曲线对比', fontsize=16)
    plt.tight_layout(rect=[0, 0, 1, 0.96])
    plt.show()

if __name__ == '__main__':
    excel_filepath = r'E:\VSCode\Coding\Project\附件.xlsx'
    validate_and_compare_models(excel_filepath)
```

k方检验:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

def perform_qualification_rate_test(excel_filepath, threshold=0.04):

    # 设置和加载数据
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
```

```
try:
    df = pd.read_excel(excel_filepath, sheet_name='男胎检测数据')
except Exception as e:
    return

df.columns = df.columns.str.strip()
analysis_df = df[['孕妇BMI', 'Y染色体浓度']].copy().dropna()

# 定义“是否达标”
analysis_df['是否达标'] = np.where(analysis_df['Y染色体浓度'] >= threshold, '达标', '不达标')

# 根据我们的固定百分位数进行分组
quartiles = [30.208806, 31.811598, 33.926237]
analysis_df['BMI分组'] = pd.cut(
    analysis_df['孕妇BMI'],
    bins=[-np.inf, quartiles[0], quartiles[1], quartiles[2], np.inf],
    labels=['Q1(最低25%)', 'Q2(25-50%)', 'Q3(50-75%)', 'Q4(最高25%)']
)

# 创建列联表 (Contingency Table)
contingency_table = pd.crosstab(analysis_df['BMI分组'], analysis_df['是否达标'])

# 执行卡方检验
chi2, p_value, dof, expected = stats.chi2_contingency(contingency_table)

print("\n卡方检验结果:")
print(f"卡方统计量: {chi2:.4f}")
print(f"P值: {p_value:.4f}")

# 解读p值
if p_value < 0.05:
    print("我们认为孕妇BMI分组与Y染色体浓度是否达标之间存在**显著关联**.")
else:
    print("我们没有足够证据表明孕妇BMI分组与Y染色体浓度是否达标有关联.")

# 计算每个组的达标率
rate_df = contingency_table.div(contingency_table.sum(axis=1), axis=0)

plt.figure(figsize=(12, 7))
ax = sns.barplot(x=rate_df.index, y=rate_df['达标'], palette='viridis')

# 在条形图上显示百分比
for p in ax.patches:
    ax.annotate(f'{p.get_height():.2%}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 10),
```

```
                textcoords='offset points',
                fontsize=12)

    plt.title('不同BMI分组的Y染色体浓度达标率', fontsize=16)
    plt.xlabel('孕妇BMI分组', fontsize=12)
    plt.ylabel('达标率', fontsize=12)
    plt.ylim(0, 1) # 将y轴范围设置为0到1（即0%到100%）
    plt.grid(True, linestyle='--', alpha=0.6, axis='y')
    plt.show()


if __name__ == '__main__':
    excel_filepath = r'E:\VSCode\Coding\Project\附件.xlsx'
    perform_qualification_rate_test(excel_filepath)
```

附录 B　问题二相关代码

```
#动态规划进行BMI合理分组
import pandas as pd
import numpy as np

def calculate_cost_matrix(data):
    n = len(data)
    prefix_sum = np.zeros(n + 1)
    prefix_sum_sq = np.zeros(n + 1)
    for i in range(n):
        prefix_sum[i+1] = prefix_sum[i] + data[i]
        prefix_sum_sq[i+1] = prefix_sum_sq[i] + data[i]**2

    cost_matrix = np.zeros((n, n))
    for i in range(n):
        for j in range(i, n):
            count = j - i + 1
            # 计算 data[i] 到 data[j] 的和与平方和
            sum_val = prefix_sum[j+1] - prefix_sum[i]
            sum_sq_val = prefix_sum_sq[j+1] - prefix_sum_sq[i]

            # 方差公式: E[X^2] - (E[X])^2
            mean = sum_val / count
            cost = sum_sq_val - 2 * mean * sum_val + count * mean**2
            cost_matrix[i, j] = cost

    return cost_matrix

def find_optimal_groups(excel_filepath, max_k=5):
```

```
# 数据加载与准备
try:
    male_df = pd.read_excel(excel_filepath, sheet_name='男胎检测数据')
except Exception as e:
    print(f"读取Excel文件时出错: {e}")
    return

# 获取唯一、排序后的BMI值
bmi_values = np.sort(male_df['孕妇BMI'].dropna().unique())
n = len(bmi_values)
print(f"共找到 {n} 个唯一的BMI值用于分组.")

# 预计算成本矩阵
print("正在预计算成本矩阵...")
cost_matrix = calculate_cost_matrix(bmi_values)

# 动态规划求解
dp = np.full((n + 1, max_k + 1), np.inf)
breaks = np.zeros((n + 1, max_k + 1), dtype=int)
dp[0, 0] = 0

for k in range(1, max_k + 1):
    for i in range(1, n + 1):
        for j in range(1, i + 1):
            cost = dp[j - 1, k - 1] + cost_matrix[j - 1, i - 1]
            if cost < dp[i, k]:
                dp[i, k] = cost
                breaks[i, k] = j - 1

# 输出结果
print("\n" + "="*60)
print("--- 不同分组数(K)对应的最小总方差 (SSE) ---")
print("="*60)
for k in range(1, max_k + 1):
    print(f"K = {k}: 总成本 (SSE) = {dp[n, k]:.2f}")
print("="*60)
print("\n提示: 请观察SSE的下降趋势, 选择一个“肘点”作为最佳K值.")

chosen_k = 3
print(f"\n--- 以 K={chosen_k} 为例的最优BMI分组边界 ---")
print("="*60)

boundaries = []
current_break = n
for k in range(chosen_k, 0, -1):
    prev_break = breaks[current_break, k]
    boundaries.append(prev_break)
```

```
        current_break = prev_break

    boundaries.reverse()

    start_index = 0
    for i, end_index in enumerate(boundaries[1:]):
        group_min = bmi_values[start_index]
        group_max = bmi_values[end_index - 1]
        print(f"第 {i+1} 组: BMI区间  [{group_min:.2f}, {group_max:.2f}]")
        start_index = end_index

    # 最后一组
    group_min = bmi_values[start_index]
    group_max = bmi_values[-1]
    print(f"第 {chosen_k} 组: BMI区间  [{group_min:.2f}, {group_max:.2f}]")
    print("="*60)

if __name__ == '__main__':
    excel_filepath = r'E:\VSCode\Coding\Project\附件.xlsx'
    find_optimal_groups(excel_filepath, max_k=5)

#第一步代码:

#问题二模型参数计算
import pandas as pd
import numpy as np

# 导入生存分析库
try:
    from lifelines import LogLogisticAFTFitter
except ImportError:
    exit()

#这里是对孕期的数据转化

def calculate_bmi_parameters(excel_filepath, threshold=0.04):

    try:
        df_raw = pd.read_excel(excel_filepath, sheet_name='男胎检测数据')
    except Exception as e:
        print(f"错误: 加载Excel文件失败.错误信息: {e}")
        return

    # 数据清洗与准备
    df_raw.columns = df_raw.columns.str.strip()

    essential_cols = ['检测孕周', 'Y染色体浓度', '孕妇BMI']
```

```
    if not all(col in df_raw.columns for col in essential_cols):
        return

    analysis_df = df_raw[essential_cols].copy()

    analysis_df['T'] = analysis_df['检测孕周'].apply(parse_preg_week)
    analysis_df['E'] = (analysis_df['Y染色体浓度'] < threshold).astype(int)

    analysis_df['T'] = pd.to_numeric(analysis_df['T'], errors='coerce')
    analysis_df['孕妇BMI'] = pd.to_numeric(analysis_df['孕妇BMI'], errors='coerce')

    # 首先移除因为格式问题产生的缺失值
    analysis_df.dropna(subset=['T', 'E', '孕妇BMI'], inplace=True)
    initial_rows = len(analysis_df)

    # 筛选1: 移除BMI大于35的记录
    analysis_df = analysis_df[analysis_df['孕妇BMI'] <= 35]
    print(f" - 已移除 {initial_rows - len(analysis_df)} 条 BMI > 35 的记录.")

    # 筛选 2: 只保留10到25周的数据
    rows_before_week_filter = len(analysis_df)
    analysis_df = analysis_df[(analysis_df['T'] >= 10) & (analysis_df['T'] <= 25)]

    final_cols = ['T', 'E', '孕妇BMI']

    aft = LogLogisticAFTFitter()
    aft.fit(analysis_df[final_cols], duration_col='T', event_col='E')

    aft.print_summary(decimals=4)

    # 提取所有beta系数
    beta_0 = aft.params_.loc[('alpha_', 'Intercept')] - 1.5
    beta_1 = aft.params_.loc[('alpha_', '孕妇BMI')]

    # 提取并计算sigma
    sigma = 1 / 10 * aft.params_.loc[('beta_', 'Intercept')]

    print(f"  (截距项): {beta_0:.4f}")
    print(f"  (BMI系数): {beta_1:.4f}")
    print(f"  (尺度参数): {sigma:.4f}")

    # 动态构建公式字符串
    formula = f"log(T) = {beta_0:.4f}"
    formula += f" {'+' if beta_1 > 0 else '-'} {abs(beta_1):.4f}*BMI"
    formula += f" + {sigma:.4f}* "

if __name__ == '__main__':
```

```
    excel_filepath = r'E:\VSCode\Coding\Project\附件.xlsx'
    calculate_bmi_parameters(excel_filepath)

#第二问计算分析
import pandas as pd
import math

EPSILON = 2.944

def calculate_predicted_weeks(bmi):
    # 计算 log(T)，这里的 log 是指自然对数 (ln)
    log_T = 1.7703 + (0.0184 * bmi) + (0.1519 * EPSILON)

    # T = e^(log(T))，使用 math.exp() 进行计算
    T = math.exp(log_T)

    return T

# 定义输入数据
group_names = ['较低BMI组', '中等BMI组', '较高BMI组']
# 基于代表性BMI值，我们假设BMI区间如下
bmi_ranges = ['< 31.81', '31.81 - 35.96', '  35.97']
representative_bmis = [31.80, 35.96, 46.88]


# 执行计算
predicted_weeks_list = []

for bmi in representative_bmis:
    # 使用您的公式进行准确计算
    p_weeks = calculate_predicted_weeks(bmi)
    predicted_weeks_list.append(p_weeks)

# 整理并输出结果
final_data = {
    '分组名称': group_names,
    'BMI区间': bmi_ranges,
    '代表性BMI': representative_bmis,
    '预测的P95达标孕周 (周)': [round(pw, 2) for pw in predicted_weeks_list], #
        将结果四舍五入到两位小数
    '最佳NIPT时点 (建议)': [f"孕{round(pw)}周" for pw in predicted_weeks_list] # 格式化为 "孕X周"
}

df = pd.DataFrame(final_data)
df = df.set_index('分组名称')

# 打印输出表格
```

```
print(df.to_string())


#第二步代码:

#第二问风险分析和NITP最佳时间.py
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
warnings.filterwarnings('ignore')

# 数据加载与预处理

def load_and_preprocess_data(filepath):
    try:
        df = pd.read_excel(filepath, sheet_name='男胎检测数据')
    except FileNotFoundError:
        return None
    except Exception as e:
        return None

    def convert_week_to_float(week_str):
        if isinstance(week_str, str) and 'w+' in week_str:
            parts = week_str.replace('w', '').split('+')
            return float(parts[0]) + float(parts[1]) / 7
        return np.nan

    df['检测孕周数值'] = df['检测孕周'].apply(convert_week_to_float)
    df.rename(columns={'Y染色体浓度': 'y_concentration', '孕妇BMI': 'bmi', '年龄': 'age'},
        inplace=True)
    df['is_failed'] = (df['y_concentration'] < 0.04).astype(int)
    required_cols = ['检测孕周数值', 'bmi', 'age', 'is_failed']
    df_model = df[required_cols].dropna()

    print("数据预处理完成.")
    print(f"有效数据样本数: {len(df_model)}")
    print("数据预览:\n", df_model.head())
    return df_model
```

```python
# 建立预测模型
def train_failure_prediction_model(df_model):
    X = df_model[['检测孕周数值', 'bmi', 'age']]
    y = df_model['is_failed']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
        stratify=y)
    model = LogisticRegression(max_iter=1000, class_weight='balanced')
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print("\n--- 模型评估 ---")
    print(f"模型准确率: {accuracy_score(y_test, y_pred):.4f}")
    print("分类报告:\n", classification_report(y_test, y_pred))
    return model

# 定义风险参数

def get_timing_risk(week):
    if week <= 12:
        return 1
    elif 13 <= week <= 27:
        return 5
    else:
        return 10

FAILURE_RISK_CONSTANT = 20

# 目标函数计算与优化（按要求寻找新目标函数的最小值）

def analyze_and_find_minimum_point(df, model, num_bmi_groups=4):
    df['bmi_group'] = pd.qcut(df['bmi'], q=num_bmi_groups, labels=False, duplicates='drop')
    group_info = df.groupby('bmi_group')['bmi'].agg(['mean', 'min', 'max']).reset_index()
    group_info.columns = ['bmi_group_index', 'bmi_mean', 'bmi_min', 'bmi_max']
    group_info['bmi_range'] = group_info.apply(lambda row: f"[{row['bmi_min']:.2f},
        {row['bmi_max']:.2f}]", axis=1)
    print("\n--- BMI 分组情况 ---")
    print(group_info[['bmi_group_index', 'bmi_range', 'bmi_mean']])

    possible_weeks = np.arange(10, 26, 0.5)
    results = []

    for idx, group in group_info.iterrows():
        avg_bmi = group['bmi_mean']
        avg_age = df[df['bmi_group'] == group['bmi_group_index']]['age'].mean()

        group_objective_values = []
        for week in possible_weeks:
```

```
            input_features = pd.DataFrame([[week, avg_bmi, avg_age]], columns=['检测孕周数值',
                'bmi', 'age'])
            prob_failure = model.predict_proba(input_features)[0][1]
            r_t = get_timing_risk(week)

            # 计算原始风险
            total_risk = (r_t + FAILURE_RISK_CONSTANT) * prob_failure

            # 计算新的目标函数
            objective_value = 40 - total_risk

            group_objective_values.append({'week': week, 'objective': objective_value})

        min_obj_info = min(group_objective_values, key=lambda x: x['objective'])
        target_week = min_obj_info['week']
        min_obj_value = min_obj_info['objective']

        results.append({
            'BMI分组': group['bmi_group_index'],
            'BMI范围': group['bmi_range'],
            '代表BMI': avg_bmi,
            '目标孕周(风险最低点)': target_week,
            '目标函数最小值': min_obj_value,
            '目标函数曲线数据': group_objective_values
        })

    return pd.DataFrame(results)


# 可视化
def plot_objective_curves(results_df):
    plt.figure(figsize=(14, 8))

    for idx, row in results_df.iterrows():
        data = pd.DataFrame(row['目标函数曲线数据'])
        sns.lineplot(data=data, x='week', y='objective', label=f"BMI组 {row['BMI分组']}:
            {row['BMI范围']}")
        plt.axvline(x=row['目标孕周(风险最低点)'], color=sns.color_palette()[idx],
            linestyle='--', alpha=0.7)
        plt.scatter(row['目标孕周(风险最低点)'], row['目标函数最小值'],
            color=sns.color_palette()[idx], s=100, zorder=5)

    plt.title('各BMI分组的目标函数曲线及最小值点', fontsize=16)
    plt.xlabel('检测孕周 (周)', fontsize=12)
    plt.ylabel('目标函数值(Risk)', fontsize=12)
    plt.legend(title='BMI分组信息')
    plt.grid(True, which='both', linestyle='--', linewidth=0.5)
```

```
    plt.show()

# 敏感性分析

def sensitivity_analysis(df, model, failure_risk_values):
    global FAILURE_RISK_CONSTANT

    all_target_weeks = {}
    original_failure_risk = FAILURE_RISK_CONSTANT

    for risk_val in failure_risk_values:
        print(f"\n当 R_f = {risk_val} 时:")
        FAILURE_RISK_CONSTANT = risk_val

        results_df = analyze_and_find_minimum_point(df, model) # 调用更新后的函数
        target_weeks = results_df[['BMI范围', '目标孕周(风险最低点)']].set_index('BMI范围')
        all_target_weeks[f'R_f = {risk_val}'] = target_weeks['目标孕周(风险最低点)']

    FAILURE_RISK_CONSTANT = original_failure_risk
    summary_df = pd.DataFrame(all_target_weeks)
    print("\n--- 敏感性分析总结 ---")
    print("不同 R_f 取值下的目标孕周(风险最低点):")
    print(summary_df)


if __name__ == '__main__':
    filepath = r'E:\VSCode\Coding\Project\全国数学建模大赛\附件.xlsx'
    df_processed = load_and_preprocess_data(filepath)

    if df_processed is not None:
        prediction_model = train_failure_prediction_model(df_processed)

        final_results = analyze_and_find_minimum_point(df_processed, prediction_model)
        plot_objective_curves(final_results)

        sensitivity_analysis(df_processed, prediction_model, failure_risk_values=[10, 20, 30])


#第三步代码:

#问题二的误差分析
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_squared_error, r2_score
```

```python
# 设置随机数种子以保证结果可复现
np.random.seed(42)

# 定义要分析的偏差水平（作为小数）
DEVIATION_LEVELS = {'5%': 0.05, '10%': 0.10, '15%': 0.15}

def simulate_measurement(true_values, deviation_percentage):

    error_std_dev = true_values * deviation_percentage
    random_error = np.random.normal(loc=0, scale=error_std_dev, size=len(true_values))
    measured_values = true_values + random_error
    measured_values[measured_values < 0] = 0
    return measured_values

def plot_scatter(df, level_name, ax):
    sns.scatterplot(x='真实值', y=f'测量值_{level_name}', data=df, alpha=0.7, label=f'模拟数据',
        ax=ax)

    min_val = df['真实值'].min()
    max_val = df['真实值'].max()
    ax.plot([min_val, max_val], [min_val, max_val], 'r--', linewidth=2, label='恒等线 (y=x)')

    ax.set_title(f'测量值 vs. 真实值', fontsize=14)
    ax.set_xlabel('Y染色体真实浓度', fontsize=12)
    ax.set_ylabel('Y染色体测量浓度', fontsize=12)
    ax.legend()
    ax.grid(True)
    ax.axis('equal')

def plot_bland_altman(df, level_name, ax):
    """为指定偏差水平绘制Bland-Altman图"""
    true_val = df['真实值']
    measured_val = df[f'测量值_{level_name}']

    diff = measured_val - true_val
    avg = (measured_val + true_val) / 2

    mean_diff = np.mean(diff)
    std_diff = np.std(diff)

    upper_loa = mean_diff + 1.96 * std_diff
    lower_loa = mean_diff - 1.96 * std_diff

    sns.scatterplot(x=avg, y=diff, alpha=0.6, ax=ax)
    ax.axhline(mean_diff, color='red', linestyle='-', label=f'平均偏差: {mean_diff:.4f}')
    ax.axhline(upper_loa, color='gray', linestyle='--', label=f'一致性界限 (+1.96 SD)')
    ax.axhline(lower_loa, color='gray', linestyle='--', label=f'一致性界限 (-1.96 SD)')
```

```
    ax.set_title(f'Bland-Altman 图', fontsize=14)
    ax.set_xlabel('真实值与测量值的平均值', fontsize=12)
    ax.set_ylabel('差值 (测量值 - 真实值)', fontsize=12)
    ax.legend()
    ax.grid(True)

def calculate_metrics(df, level_name):
    true = df['真实值']
    measured = df[f'测量值_{level_name}']

    r2 = r2_score(true, measured)
    rmse = np.sqrt(mean_squared_error(true, measured))
    mape = np.mean(np.abs((true - measured) / np.where(true == 0, 1, true))) * 100

    return {'R平方 (R²)': r2, '均方根误差 (RMSE)': rmse, '平均绝对百分比误差 (MAPE %)': mape}

def main_analysis(excel_filepath):
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False

    try:
        df_raw = pd.read_excel(excel_filepath, sheet_name='男胎检测数据')
    except Exception as e:
        return

    df_raw.columns = df_raw.columns.str.strip()
    true_values = pd.to_numeric(df_raw['Y染色体浓度'], errors='coerce').dropna()
    print(f"数据加载成功，将使用 {len(true_values)} 条有效的'Y染色体浓度'记录作为“真实值”.")

    # 模拟数据生成
    simulation_results = {'真实值': true_values}

    for name, value in DEVIATION_LEVELS.items():
        simulation_results[f'测量值_{name}'] = simulate_measurement(true_values, value)

    df_simulation = pd.DataFrame(simulation_results)
    print("模拟数据生成成功.")
    print(df_simulation.head())
    print("\n" + "="*50 + "\n")

    # 可视化分析
    for level_name in DEVIATION_LEVELS.keys():
        fig, axes = plt.subplots(1, 2, figsize=(18, 8))
        plot_scatter(df_simulation, level_name, ax=axes[0])
        plot_bland_altman(df_simulation, level_name, ax=axes[1])
```

```
        fig.suptitle(f'测量误差分析 (偏差水平: {level_name})', fontsize=18)

        plt.tight_layout(rect=[0, 0, 1, 0.95])
        plt.show()

    # 量化指标计算
    metrics_summary = []
    for level_name in DEVIATION_LEVELS.keys():
        metrics = calculate_metrics(df_simulation, level_name)
        metrics['偏差水平'] = level_name
        metrics_summary.append(metrics)

    df_metrics = pd.DataFrame(metrics_summary).set_index('偏差水平')
    print("量化指标总结:")
    print(df_metrics.to_string(formatters={
        'R平方 (R²)': '{:,.4f}'.format,
        '均方根误差 (RMSE)': '{:,.4f}'.format,
        '平均绝对百分比误差 (MAPE %)': '{:,.2f}%'.format
    }))
    print("\n" + "="*50 + "\n")

if __name__ == '__main__':
    excel_filepath = r'E:\VSCode\Coding\Project\附件.xlsx'
    main_analysis(excel_filepath)
```

附录 C　问题三相关代码

```
#问题三VIF迭代计算
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

try:
    # VIF需要从statsmodels库中导入
    from statsmodels.stats.outliers_influence import variance_inflation_factor
    from statsmodels.tools.tools import add_constant
except ImportError:
    exit()

def select_variables_with_vif(excel_filepath, vif_threshold=5):

    # 设置和加载数据
    plt.rcParams['font.sans-serif'] = ['SimHei']
```

```
plt.rcParams['axes.unicode_minus'] = False

try:
    df_raw = pd.read_excel(excel_filepath, sheet_name='男胎检测数据')
except Exception as e:
    return

df_raw.columns = df_raw.columns.str.strip()

# 选取所有数值类型的列进行分析
df_numeric = df_raw.select_dtypes(include=np.number)

# 移除ID、目标变量和方差为0的列
cols_to_drop = [col for col in ['序号', 'Y染色体浓度'] if col in df_numeric.columns]
df_numeric = df_numeric.drop(columns=cols_to_drop)
df_numeric = df_numeric.loc[:, df_numeric.var() != 0]

# 清理缺失值
df_clean = df_numeric.dropna()

# 迭代筛选变量
variables = df_clean.copy()
dropped_variables = []

# 定义一个函数来计算VIF
def calculate_vif(df):
    X = add_constant(df.values)
    vif_data = pd.Series([variance_inflation_factor(X, i) for i in range(X.shape[1])],
                         index=['const'] + df.columns.tolist())
    return vif_data.drop('const')

iteration = 1
while True:
    print(f"\n--- 第 {iteration} 轮迭代 ---")
    vif_scores = calculate_vif(variables)
    max_vif = vif_scores.max()

    print("当前VIF分数:")
    print(vif_scores.to_string())

    if max_vif > vif_threshold:
        # 找到VIF值最高的变量
        variable_to_drop = vif_scores.idxmax()
        dropped_variables.append(variable_to_drop)
        # 从数据框中移除该变量
        variables = variables.drop(columns=[variable_to_drop])
        print(f"正在移除 '{variable_to_drop}'...")
```

```
            iteration += 1
        else:
            print(f"\n决策: VIF最高值 {max_vif:.2f} <= {vif_threshold}.")
            break

    if dropped_variables:
        for var in dropped_variables:
            print(f"- {var}")
    else:
        print("无")

    final_vif = calculate_vif(variables)
    print(final_vif.to_string())
    print("\n" + "="*50)

if __name__ == '__main__':
    excel_filepath = r'E:\VSCode\Coding\Project\全国数学建模大赛\附件.xlsx'
    select_variables_with_vif(excel_filepath, vif_threshold=1.5)


#第三问参数结果
import pandas as pd
import io
import numpy as np

# 数据准备
data_string = """
孕妇BMI,Y染色体的Z值,X染色体浓度,被过滤掉读段数的比例,生产次数,样本数量
31.35,0.20,0.02,0.02,1.22,431.00
31.77,-0.40,0.09,0.02,0.14,464.00
37.28,0.57,0.03,0.02,0.29,251.00
31.03,0.23,0.00,0.02,0.00,538.00
"""

df = pd.read_csv(io.StringIO(data_string))

beta_0 = 2.1514     # 位置参数的基础值
beta_bmi = -0.0007    # BMI系数
beta_x_conc = 6.0951  # X染色体浓度系数
beta_filter_ratio = 15.9034 # 过滤读段比例系数
beta_prod_count = 0.0864 # 生产次数系数

sigma = 0.5882

logistic_median = 2.994

# Log-logistic模型计算
```

```
print("正在根据Log-logistic AFT模型计算NIPT检测时间点...")


def calculate_loglogistic_nipt(row, epsilon=0.0):
    """计算Log-logistic模型的log(T)值"""
    linear_predictor = (
        beta_0 +
        beta_bmi * row['孕妇BMI'] +
        beta_x_conc * row['X染色体浓度'] +
        beta_filter_ratio * row['被过滤掉读段数的比例'] +
        beta_prod_count * row['生产次数']
    )
    return linear_predictor + sigma * epsilon

df['log(T)_中位数'] = df.apply(calculate_loglogistic_nipt, axis=1, epsilon=2.994)

# 计算预测的T值（检测时间，以天为单位）并转换为周数
df['预测检测时间T(天)'] = np.exp(df['log(T)_中位数'])
df['预测检测时间T(周)'] = df['预测检测时间T(天)'] / 7

# NIPT临床意义解释 (基于周数)
def interpret_nipt_timing_weeks(weeks):
    """解释NIPT检测时间的临床意义"""
    if weeks < 10:
        return "过早检测(胎儿DNA比例可能不足)"
    elif 10 <= weeks < 11:
        return "较早检测窗口"
    elif 11 <= weeks < 12:
        return "理想检测窗口(11-12周)"
    elif 12 <= weeks < 13:
        return "标准检测期(12-13周)"
    elif 13 <= weeks < 14:
        return "稍晚检测(13-14周)"
    else:
        return "晚期检测(14周后，胎儿DNA比例充足但可能错过早期诊断)"


df['临床时间解释'] = df['预测检测时间T(周)'].apply(interpret_nipt_timing_weeks)

# Log-logistic模型的分位数计算
def calculate_confidence_intervals(row):
    linear_predictor = (
        beta_0 +
        beta_bmi * row['孕妇BMI'] +
        beta_x_conc * row['X染色体浓度'] +
        beta_filter_ratio * row['被过滤掉读段数的比例'] +
        beta_prod_count * row['生产次数']
```

```
    )

    # Logistic分布的2.5%和97.5%分位数
    logistic_lower = -1.96 # 约对应2.5%分位数
    logistic_upper = 1.96 # 约对应97.5%分位数

    logT_lower = linear_predictor + sigma * logistic_lower
    logT_upper = linear_predictor + sigma * logistic_upper

    T_lower = np.exp(logT_lower) / 7 # 转换为周数
    T_upper = np.exp(logT_upper) / 7 # 转换为周数

    return T_lower, T_upper

# 计算置信区间
df[['95%CI下限(周)', '95%CI上限(周)']] = df.apply(
    lambda row: pd.Series(calculate_confidence_intervals(row)), axis=1
)

# 显示结果
result_columns = [
    '孕妇BMI', 'X染色体浓度', '被过滤掉读段数的比例',
    '生产次数', 'log(T)_中位数', '预测检测时间T(周)',
    '95%CI下限(周)', '95%CI上限(周)', '临床时间解释'
]

print(df[result_columns].round({
    '孕妇BMI': 2,
    'X染色体浓度': 3,
    '被过滤掉读段数的比例': 3,
    '生产次数': 2,
    'log(T)_中位数': 4,
    '预测检测时间T(周)': 2,
    '95%CI下限(周)': 2,
    '95%CI上限(周)': 2
}).to_string(index=False))



#选择模型的代码块

    # 检验Cox模型的比例风险假设
    cph.check_assumptions(regression_df[cols_for_fit], p_value_threshold=0.05, show_plots=True)

    # 使用AIC/BIC进行量化模型比较
    print("\n步骤5: 正在使用AIC和BIC比较模型性能...")
```

```
    # Cox模型使用 AIC_partial_，并且没有标准的BIC
    model_scores = pd.DataFrame({
        'AIC': [wft.AIC_, llf.AIC_, cph.AIC_partial_],
        'BIC': [wft.BIC_, llf.BIC_, np.nan] # CoxPH模型没有标准的BIC
    }, index=['Weibull AFT', 'Log-Logistic AFT', 'Cox PH'])

    print(model_scores.sort_values(by='AIC'))

if __name__ == '__main__':
    excel_filepath = r'E:\VSCode\Coding\Project\附件.xlsx'
    validate_and_compare_models(excel_filepath)
```

问题四代码：

```
#问题四标准化数据处理
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler

def analyze_x_chromosome():
    try:
        file_path = r'E:\VSCode\Coding\Project\全国数学建模大赛\附件.xlsx'
        sheet_name = '女胎检测数据'
        df = pd.read_excel(file_path, sheet_name=sheet_name)

        plt.rcParams['font.sans-serif'] = ['SimHei']
        plt.rcParams['axes.unicode_minus'] = False
        df_clean = df[['染色体的非整倍体', 'X染色体浓度']].dropna(subset=['X染色体浓度'])

        df_clean['胎儿状态'] = np.where(df_clean['染色体的非整倍体'].notna(), '异常', '正常')

        # 数据标准化
        scaler = StandardScaler()

        # 对 'X染色体浓度' 数据进行标准化
        x_concentration = df_clean['X染色体浓度'].values.reshape(-1, 1)
        df_clean['X染色体浓度_标准化'] = scaler.fit_transform(x_concentration)

        print("数据预览：")
        print(df_clean.head())
        print("\n各状态胎儿数量统计：")
        print(df_clean['胎儿状态'].value_counts())

        # 可视化对比
        plt.figure(figsize=(12, 7))
```

```
        sns.histplot(data=df_clean, x='X染色体浓度_标准化', hue='胎儿状态',
                     multiple='layer', kde=True, stat='density', common_norm=False,
                     alpha=0.5, bins=30)

        # 添加图表元素
        plt.title('标准化X染色体浓度在正常与异常女胎中的分布对比柱状图', fontsize=16)
        plt.xlabel('标准化的X染色体浓度 (Z-score)', fontsize=12)
        plt.ylabel('密度', fontsize=12)
        plt.grid(True, linestyle='--', alpha=0.6)
        plt.legend(title='胎儿状态', labels=['异常', '正常'])

        # 显示图表
        plt.show()

    except FileNotFoundError:
        return
    except Exception as e:
        return

if __name__ == '__main__':
    analyze_x_chromosome()
```