《生存分析》课程报告

左截断数据基于面积法的假设检验方法

成员：陈奕昕（3227042017）（组长）

杨礼佳（3227042038）

雷林琦（3227042024）

吴育雄（3227042018）

黄荣鹏（3227042015）

应用(生物)统计学专业 2022 级

完成时间：2024 年 11 月 1 日

授课教师：陈 征

课程报告组员分工：

雷林琦：面积法假设检验方法、bootsrap 面积法程序实现，以及对应报告和 PPT 内容的撰写。

杨礼佳：非参数估计方法程序实现及对应报告内容的撰写，报告内容的整合。

陈奕昕：非参数估计方法程序实现，各部分程序修改和连接整合。PPT 内容的修改整合。

吴育雄：置换面积法程序实现，MC 模拟结果分析及对应报告内容的撰写。

黄荣鹏：左截断生存数据的模拟生成程序实现，及对应报告内容的撰写。

目录

【摘要】

目的：本文基于现有针对左截断数据的几种非参数估计方法，自编通用 R 程序实现左截断生存数据的生存率及其方差的计算，在此基础上构造左截断数据下面积法的统计量Δ，并编写对应的面积法、置换面积法和 Bootstrap 面积法的假设检验的通用程序，并对上述方法进行比较研究。**方法**：使用非参极大似然估计法（NPMLE）、Breslow-Fleming-Harrington 估计法（BFH）、加权 NPMLE、条件 NPMLE 和加权 BFH 五种方法进行非参数估计，基于非参数估计的结果，利用面积法、置换面积法和 Bootstrap 面积法进行假设检验，通过模拟研究比较上述方法的Ⅰ类错误和检验效能。**结果**：研究结果显示，在小风险集时加权 BFH 法和加权 NPMLE 法的估计精度较高。模拟结果显示，在不同样本量和删失率下面积法和置换面积法在控制 I 类错误方面表现优异，同时置换面积法在多种情形，特别是在生存曲线交叉和部分发散时具有更高的检验效能。**结论**：结合模拟结果和实际例子，存在左截断数据时，本文所实现的 R 程序能够较好地避免生存率估计的偏倚和小风险集的影响，同时在比较两组生存率的差异时，左截断数据下的置换面积法在各个情况中的检验效能都较高。

【关键词】：生存分析、左截断、小风险集、非参数估计面积法、置换检验、蒙特卡洛模拟

【Abstract】

Objective: This study develops a general R program to estimate survival rates and their variances for left-truncated data using several non-parametric methods. The program constructs the statistical measure Δ for the area-based test and provides implementations for the area-based test, permutation area-based test, and Bootstrap area-based test. These methods are compared through simulations. **Methods:**Five non-parametric estimation methods are used: non-parametric maximum likelihood estimation (NPMLE), Breslow-Fleming-Harrington (BFH) estimation, weighted NPMLE, conditional NPMLE, and weighted BFH. Hypothesis testing is conducted using the area-based test, permutation area-based test, and Bootstrap area-based test. Simulations compare the type I error and power of these methods.**Results:**Weighted BFH and weighted NPMLE methods provide higher estimation accuracy in small risk sets. Simulations show that the area-based test and permutation area-based test excel in controlling type I error across different sample sizes and censoring rates. The permutation area-based test demonstrates higher power in various scenarios, especially when survival curves cross or diverge partially**Conclusion:**The R program developed in this study effectively avoids bias in survival rate estimation and handles small risk sets. The permutation area-based test shows consistently high power in detecting differences in survival rates under left-truncated data.

【Keywords】Survival analysis, Left truncation, Small risk sets, Non-parametric estimation, Area-based test, Permutation test, Monte Carlo simulation

1 前言

1.1 背景

在医学临床研究中，准确评估和比较不同群体的生存率差异对于理解疾病进展和治疗效果至关重要。而左截断数据在实际研究中非常普遍，例如在传染病研究中，患者往往在症状明显时才被纳入研究，而实际的感染时间却早于纳入研究的时间，如果将左截断数据忽略，按照一般处理右删失数据的方法进行估计，通过传统的生存分析方法，如 Kaplan-Meier 估计，往往无法妥善处理左截断数据，这可能导致生存率的高估和分析结果的偏倚[1]。为了解决这一问题，我们参考了多种非参数估计方法，如非参数极大似然估计法（NPMLE）、Breslow-Fleming-Harrington（BFH）估计法等，以更准确地估计生存函数和处理截断数据。查阅资料发现，目前已有的几种左截断数据的几种非参数估计方法并没有对应的通用程序。

对于两组生存率差异的假设检验方法，当风险率成比例假定失效，特别是生存曲线交叉时，基于面积法构造统计量的方法被证明是较好的假设检验方法。然而现有的程序大多都是针对有删失数据，即将左截断数据忽略，通过传统 KM 法估计的生存率来进行检验的，这在一定程度上会导致统计推断的结果有偏倚。因此，将处理左截断数据的非参数方法与基于面积法的假设检验方法结合，并建立相关方法的 R 程序是非常有必要的。

1.2 目的

基于现有的针对左截断数据的几种非参数估计方法[2]，自编通用程序实现左截断生存数据的生存率及其方差的计算，在此基础上，构造左截断数据下面积法的统计量Δ，并编写对应的面积法、置换面积法和 Bootstrap 面积法的假设检验的通用程序。同时，通过模拟生成不同情形下含有左截断信息的生存数据，通过蒙特卡洛模拟评价上述检验方法的稳健性和适用性，最后通过实例分析对自编程序进行演示。

1.3 研究对象

左截断数据指个体的生存时间开始点是在实验随访之前，也就是在随访开始时其已经度过了一段生存期，相当是延迟进入随访研究，一直到发生终点事件或右删失为止，如在艾滋病研究中，患者可能在感染病毒后的很长一段时间内都处于潜伏期，没有表现出明显的症状。因此，在潜伏期内的数据（如 CD4 细胞计数、病毒载量等）是无法被准确记录的。只有当患者出现症状或进行 HIV 检测时被确诊为艾滋病患者，其后的数据才开始被记录和观察。这种情况下，潜伏期内的数据也是左截断的，也就是说从感染发病到随访之间有一段延迟的时期。

2 理论推导

2.1 左截断数据生存率的估计

本次研究有两个组别，但由于相互之间不影响各自的生存率计算，故此处只表达一个组别的生存率的理论推导。

该组样本量设为 n，令 X_1，X_2，…，X_n 表示一组独立同分布的随机生存时间变量{X_i}，其中 i=1，2，…，n，令{C_i}是对应且独立于{X_i}的一组随机删失时间变量。如存在左截断信息时，则假定一组独立同分布的随机左截断时间变量为{T_i},使得 X_i，只有在 $X_i \geqslant T_i$ 时才能被观察到。如 X_i 是不完整的观察，则 n 个个体最终观察数据各自自定义为（T_i，$\tilde{X}_i$，δ_i），其中 $\tilde{X}_i$ =min（X_i，C_i），$\delta_i = I_{|X_i \leq T_i|}$且$X_i \geq T_i$。另外用#A 表示集合 A 中元素的个数。

2.1.1NPMLE

如存在左截短信息时，需特别注意的是此时 i=1，2，…，m(n)，其中 m(n)=inf $\{m: \sum_{i=1}^{m} I_{|X_i \geq T_i|} = n\}$。

则 t 时刻生存率 NPMLE 定义如下[3]：

$$1 - F_n^*(t) = \prod_{s \leq t} \{1 - \frac{\Delta L_{m(n)}(s)}{R_{m(n)}(s)}\} \quad (1)$$

方差定义如下：

$$\hat{V}\left(1 - F_n^*(t)\right) = (1 - F_n^*(t))^2 \sum_{s \leq t} \frac{\Delta L_{m(n)}(s)}{R_{m(n)}(s)(R_{m(n)}(s) - \Delta L_{m(n)}(s))} \quad (2)$$

其中$\Delta L_{m(n)}(s) = L_{m(n)}(s) - L_{m(n)}(s-), L_m(s) = \#\{i \leq n: \widetilde{X}_i \leq s\} = \sum_{i=1}^{n} I_{|T_i \leq \widetilde{X}_i \leq s|}$,$R_m(s) = \#\{i \leq n: \widetilde{X}_i \geq s \geq T_i\} = \sum_{i=1}^{n} I_{|T_i \leq s \leq \widetilde{X}_i|}$,可以发现$\Delta L_m(s)$表示 s 时刻下死亡数，$R_m(s)$表示 s 时刻下风险集数，是指从早于 s 时刻的$T_i$时刻开始至少生存到 s 时刻的人数，即在$T_i \leq s \leq \tilde{X}_I$时刻不同于传统 KM 法风险集数，传统 KM 法风险集数是指从 0 时刻开始至少生存到 s 时刻的人数。

这种方法在存在出现风险集数$R_m(s)$和死亡数$\Delta L_m(s)$相等的小风险集情形时，很容易导致生存率 1-$F_n^*(t)$突然出现急速下降，甚至下降为 0，并且这种不稳定会因为公式（1）连续乘积作用而影响到后边生存率的计算，导致估计出现较大偏差，甚至得出错误结果，所以需要进行适当的校正。

2.1.2Breslow-Fleming-Harrington 估计法（BFH 法）

BFH 法通过累积危险率来估计生存率，累积危险率估计式为$H_n^*(t) = \sum_{s \leq t} \frac{\Delta L_{m(n)}(s)}{R_{m(n)}(s)}$，其中$R_m(s)$和$\Delta L_m(s)$与 NPMLE 定义一样，则 BFH 法定义如下：

$$\begin{aligned} S_n^*(t) &= \exp\left(-H_n^*(t)\right) \\ &= \exp\left(-\sum_{s \leq t} \frac{\Delta L_{m(n)}(s)}{R_{m(n)}(s)}\right) \end{aligned} \quad (3)$$

方差定义如下：

$$\hat{V}(\hat{S}_n(t)) = (\hat{S}_n(t))^2 \sum_{s\le t} \frac{\Delta L_{m(n)}(s)}{R_{m(n)}(s)^2}\left[1 - \frac{\Delta L_{m(n)}(s)}{R_{m(n)}(s)}\right] \quad (4)$$

BFH 法虽然在$R_m(s)$和$\Delta L_m(s)$相等的情形下，可以固定保持为正数，表现优于 NPMLE 等于 0，然而依旧无法避免出现低估生存率的情况，在出现小风险集时生存率也会骤降，也需要进行适当的校正。

2.1.3加权非参数极大似然估计法（加权 NPMLE）

Lai-Ying[2]提出第一种校正方法，称为加权 NPMLE 法，通过加权可以确保估计量的一致性。首先是先选择常数 c> 0和$0 < \alpha < 1$， n 代表样本量，

则加权 NPMLE 法定义如下：

$$1 - F_n^*(t) = \prod_{s\le t}\{1 - \frac{\Delta L_{m(n)}(s)}{R_{m(n)}(s)} I_{|R_{m(n)}(s)\ge cn^{\alpha}|}\} \quad (5)$$

方差定义如下：

$$\hat{V}\left(1 - F_n^*(t)\right) = (1 - F_n^*(t))^2 \sum_{s\le t} \frac{\Delta L_{m(n)}(s) I_{|R_{m(n)}(s)\ge cn^{\alpha}|}}{R_{m(n)}(s)(R_{m(n)}(s) - \Delta L_{m(n)}(s) I_{|R_{m(n)}(s)\ge cn^{\alpha}|})} \quad (6)$$

相比于公式（1），公式（5）设置了权重函数$I_{|R_{m(n)}(s)\ge cn^{\alpha}|}$，即只有在风险集数$R_m(s)$大于等于某一设定值$cn^{\alpha}$时，权重函数设为 1，否则设为 0，此时加权 NPMLE 弱收敛于正态分布，有较强的一致性，则可以避免小风险集的影响。

根据陈金宝 37 等的结论，我们将选用加权 NPMLE 方法进行下一步的假设检验。

2.1.4加权 Breslow-Fleming-Harrington 估计法（加权 BFH 法）

基于 BFH 法（公式（3））和 Lai-Ying [错误!未找到引用源。] 的加权校正思想，陈金宝[1]提出加权 BFH 估计法。首先选择常数 c> 0和$0 < \alpha < 1$，令权重函数为$I_{|R_{m(n)}(s)\ge cn^{\alpha}|}$，则累计危险率估计式为$\widehat{H}_n(t) = \sum_{s\le t} \frac{\Delta L_{m(n)}(s)}{R_{m(n)}(s)} I_{|R_{m(n)}(s)\ge cn^{\alpha}|}$类似加权 NPMLE 法（公式（2）），只有在$R_m(s)$等于某一设定值 cn^{α} 时，权重函数才设为 1，否则为 0。

最终加权 BFH 定义如下：

$$\begin{aligned}\hat{S}_n(t) &= \exp\left(-\widehat{H_n}(t)\right) \\ &= \exp\left(-\sum_{s\le t} \frac{\Delta L_{m(n)}(s)}{R_{m(n)}(s)} I_{|R_{m(n)}(s)\ge cn^{\alpha}|}\right)\end{aligned} \quad (7)$$

方差定义如下：

$$\hat{V}(\hat{S}_n(t)) = (\hat{S}_n(t))^2 \sum_{s\le t} \frac{\Delta L_{m(n)}(s) I_{|R_{m(n)}(s)\ge cn^{\alpha}|}}{R_{m(n)}(s)^2}\left[1 - \frac{\Delta L_{m(n)}(s)}{R_{m(n)}(s)} I_{|R_{m(n)}(s)\ge cn^{\alpha}|}\right] \quad (8)$$

通过实例分析，加权 BFH 法能避免出现风险集数$R_m(s)$和死亡数$\Delta L_m(s)$相等的小风险集情形，固定保持为正数，有效避免了低估生存率的情况。

2.1.5条件非参数极大似然估计法（条件 NPMLE）

加权 NPMLE 法（公式（5））使用了加权思路，另一种有效且自然的解决方法[2]是通过估计（t|t>u）条件分布来计算生存率，其中 u 是事先定义的恰当时间点。

则条件 NPMLE 定义如下：

$$1-F_{n,u}^{*}(t)=\prod_{u\le s\le t}\{1-\frac{\Delta L_{m(n)}(s)}{R_{m(n)}(s)}\} \tag{9}$$

方差定义如下：

$$\hat{V}\left(1-F_{n}^{*}(t)\right)=(1-F_{n}^{*}(t))^{2}\sum_{a<s\le t}\frac{\Delta L_{m(n)}(s)}{R_{m(n)}(s)(R_{m(n)}(s)-\Delta L_{m(n)}(s))} \tag{10}$$

此时的条件 NPMLE 有较强的一致性。通过选择和改变不同的时间点 u，可以获得足够的样本信息和避免出现小风险集情形。

在左截断类型数据的实际应用中，应注意是否有小风险集情形的出现， 对应地采用合理的方法，避免做出错误的结论

2.2 基于面积法的假设检验

2.2.1面积法

组间生存率差异的检验是常见的研究问题之一。Log-rank 检验是目前最常用的方法，其在风险率成比例假定成立时具有较高的检验效能，但当不满足比例风险假定，特别是两条生存曲线交叉时，交叉点前后生存率高低逆转可能导致正负差异相互抵消，从而使 Log-rank 检验的检验效能大幅降低。对此我们根据文献[5]实现一种基于两条生存曲线间面积值大小，构造左截断数据下对应统计量的面积检验法，不管风险率是否成比例、生存曲线是否交叉，其统计量构造均不受影响，避免了交叉点前后正负差异相互抵消的问题；并在原假设下其检验统计量假设服从标准正态分布。

具体的步骤如下:

先设定第 k 组样本量 n_k， 其中 k=1、2 ，t_{ki}表示第 k 组中第 i 个个体的生存时间，δ_{ki}表示第 k 组中第 i 个个体的生存状态，δ_{ki}=表示发生终点事件，δ_{ki}=0 表示删失；且$t_{k1}<t_{k2}<\cdots<t_{kn_k}$，$d_{ki}$表示第 k 组在时间点$t_{ki}$发生的事件数量，也就是发生结局事件的数量，$Y_{ki}$表示第 k 组在时间点$t_{ki}$的风险集人数；根据 Lin.X 的方法，推到得上述的非参数估计对应的方差[7]公式如下：

$$\hat{\sigma}_{k}(t)^{2}=\hat{S}_{k}{}^{2}(t)\sum_{t_{ki}\le t}\frac{d_{ki}}{Y_{ki}(Y_{ki}-d_{ki})} \tag{11}$$

将两条生存曲线之间的绝对面积值大小作为衡量差异的指标，根据上述面积指标定义可以将其表示为如下所示，其中对于连续性的变量，采取的是进行两条生存曲线下面积的积分，而对于离散型的变量，则进行的是各个时间不同生存率与时间间隔进行相乘然后进行累加；

$$\begin{aligned}\Delta &= \int_0^v \left|\hat{S}_1(t) - \hat{S}_2(t)\right| dt \\ &= \sum_{j|t_j<v} \left|\hat{S}_1(t_j) - \hat{S}_2(t_j)\right|(t_{j+1} - t_j)\end{aligned} \tag{12}$$

$$\Delta^* = \frac{(\Delta - \hat{E}(\Delta))}{\sqrt{\hat{V}(\Delta)}} \tag{13}$$

而计算完成曲线先面积Δ后，需要进行构造检验统计量，参照上面Δ*统计量的构造，理论上来说，在原假设下$\hat{S}_1(t) - \hat{S}_2(t)$的分布服从正态分布，其均值为 0，方差为$\hat{\sigma}^2_{S_1}(t) + \hat{\sigma}^2_{S_2}(t)$，进一步$|\hat{S}_1(t) - \hat{S}_2(t)|$均值和方差的分别估计为：

$$\hat{E}(\Delta) = \sum_{j|t_j \le v} \left\{\frac{2}{\pi}\left[\hat{\sigma}^2_{S_1}(t_j) + \hat{\sigma}^2_{S_2}(t_j)\right]\right\}^{\frac{1}{2}} (t_{j+1} - t_j) \tag{14}$$

$$\hat{\mathrm{V}}\left(\left|\hat{\mathrm{S}}_1(t) - \hat{\mathrm{S}}_2(t)\right|\right) = \left(1 - \frac{2}{\pi}\right)\left(\hat{\sigma}^2_{\mathrm{S}_1}(t) + \hat{\sigma}^2_{\mathrm{S}_2}(t)\right) \tag{15}$$

利用正态近似的原理，面积指标 Δ 均值估计值和方差估计值可以分别估计为:

$$\hat{E}(\Delta) = \sum_{j|t_j \leqslant v} \left\{\frac{2}{\pi}\left[\hat{\sigma}^2_{S_1}(t_j) + \hat{\sigma}^2_{S_2}(t_j)\right]\right\}^{\frac{1}{2}} (t_{j+1} - t_j) \tag{16}$$

$$\begin{aligned}\hat{V}(\Delta) = &\sum_{j|t_j<v} (t_{j+1} - t_j)^2 \left(1 - \frac{2}{\pi}\right)\left[\sigma^2_{S_1}(t_j) + \sigma^2_{S_2}(t_j)\right] + \\ &\sum_{j<j<1} 2\rho_{j,j}(t_{j+1} - t_j)(t_{j'+1} - t_{j'})(1 - \frac{2}{\pi}) \times \{[\sigma^2_{S_1}(t_j) + \\ &\sigma^2_{S_2}(t_j)][\sigma^2_{S_1}(t_{j'}) + \sigma^2_{S_2}(t_{j'})]\}^{\frac{1}{2}}\end{aligned} \tag{17}$$

其中，对于方差$Var(\Delta)$，包含了自相关项和互相关项两个部分。首先对于自相关项$\sum_{j|t_j<v} (t_{j+1} - t_j)^2(1 - \frac{2}{\pi})(\hat{\sigma}^2_{S_1}(t_j) + \hat{\sigma}^2_{S_2}(t_j))$，这部分考虑了每个时间点$t_j$上对方差的贡献，包括时间间隔的评平方包括时间间隔的平方，正态分布绝对值的方差以及两个生存函数估计值的方差之和。对于互相关项$\sum_{j<j\uparrow t_{j,}t_{j'}<v} (t_{j+1} - t_j)(t_{j'+1} - t_{j'})(1 - \frac{2}{\pi}) \times \{[\hat{\sigma}^2_{S_1}(t_j) + \hat{\sigma}^2_{S_2}(t_j)][\hat{\sigma}^2_{S_1}(t_j) + \hat{\sigma}^2_{S_2}(t_j)]\}^{\frac{1}{2}}$，考虑了不同时间点之间的互相关性对于方差的贡献，包括任意两个时间间隔的乘积，正态分布绝对值的方差，以及两个时间点上生存函数估计值方差之和的几何平均值，用于衡量两个时间点之间的互相关关系。$\rho_{j,j'}$是任意两个时间点上两组生存率之差绝对值的相关系数

研究[6]表明，当$\rho_{j,j'}$在取值为 0.5 时，检验统计量Δ不会膨胀一类错误率，同时保持较高的检验功效,化简后公式如下：

$$\hat{V}(\Delta)=\sum_{j|t_j<v}(t_{j+1}-t_j)^2(1-\frac{2}{\pi})(\hat{\sigma}^2_{S_1}(t_j)+\hat{\sigma}^2_{S_2}(t_j))+$$
$$\sum_{j<j'|t_j,t_{j'}<v}(t_{j+1}-t_j)(t_{j'+1}-t_{j'})(1-\frac{2}{\pi})\times\{[\hat{\sigma}^2_{S_1}(t_j)+ \tag{18}$$
$$\hat{\sigma}^2_{S_2}(t_j)][\hat{\sigma}^2_{S_1}(t_j)+\hat{\sigma}^2_{S_2}(t_j)]\}^{\frac{1}{2}}$$

$$\hat{V}(\hat{S}(x))=\hat{S}(x)^2\sum_{x_i\le x}\frac{dN_i(x)I\left[\hat{R}_i(x)\ge cn^{\alpha}\right]}{\hat{R}_i(x)\left(\hat{R}_i(x)-dN_i(x)I\left[\hat{R}_i(x)\ge cn^{\alpha}\right]\right)} \tag{19}$$

在进行两组生存率比较时，以$\Delta^*=\frac{\left(\Delta-\hat{\boldsymbol{E}}(\Delta)\right)}{\sqrt{\hat{\boldsymbol{V}}(\Delta)}}$作为检验统计量，在原假设下该检验统计量服从标准正态分布。

2.2.2置换面积法

2.2.2.1 方法介绍

置换面积检验法是一种强大的非参数统计方法，旨在比较两组生存数据的差异。这种方法特别适用于生存曲线交叉或风险率不成比例的情况，因为它不依赖于传统生存分析中常用的比例风险假设。通过计算两组生存曲线之间的面积差异，置换面积检验法能够有效评估生存率的差异，并通过置换检验构建统计量的分布，从而获得 P 值。

2.2.2.2 方法步骤

步骤 1：数据置换 在进行左截断处理之前，首先将两组数据合并，并进行随机置换。这一过程的目的是模拟原假设（即两组生存曲线没有显著差异）下的数据分布。通过随机置换，我们能够打破原始数据组之间的关联，为后续的统计量构造提供基础。（置换方法）

步骤 2：左截断处理 左截断数据指的是在研究开始之前，一些个体的事件已发生，因此这些数据点是不完整的。在置换后的数据中进行左截断处理，即只考虑在研究开始后发生的事件。（本文 2.1 左截断数据处理方法）

步骤 3：统计量构造 在左截断数据的基础上，我们构造统计量来比较两组生存曲线之间的差异。统计量通常基于两组生存曲线下的面积差异。通过计算这些面积差异，我们可以量化两组之间的生存率差异。构造的统计量在原假设下的分布能够通过前面置换得到的结果进行评估。 （本文 2.2.1 面积检验法）

2.2.3Bootstrap 面积法

2.2.3.1 方法介绍

这个方法结合了 Bootstrap 重抽样、左截断非参数估计和面积法，分析存在左截断现象的数据，相比于不使用 Bootstrap 的方法，引入 Bootstrap 重抽样技术的主要优势在于它能够提供

一种不依赖于分布假设的灵活统计框架，从而在样本量较小、数据分布未知或非正态分布的情况下，更准确地估计统计量和 p 值，增强结果的稳健性和普适性。

2.2.3.2 方法步骤

自助抽样（Bootstrap）：自助抽样是一种重抽样技术，用于估计统计量的分布。通过从原始数据中随机抽取样本（允许重复），可以生成多个“自助样本”。每个样本反映了原始数据的特性，从而为后续的统计分析提供基础。（Bootstrap 方法）

左截断估计：在完成自助抽样后，是对数据进行左截断估计。这一过程涉及排除在观察开始之前已发生的事件，确保分析的准确性。（本文 2.2.1 面积检验法）

面积法检验：在左截断数据的基础上，最后一步是使用面积法进行检验。面积法检验通过计算两组生存曲线下的面积差异来评估生存率的差异。（本文 2.2.1 面积检验法）

3 程序实现

主要分为非参数估计和假设检验两部分的程序，其中非参数估计分为风险集函数和五种非参数估计函数的构建，假设检验分为面积法函数和置换面积法函数的构建，模拟的程序将放在模拟比较部分。

3.1 非参数估计

非参数估计的程序主要分为两个部分，分别为计算 s 时刻下死亡数和 s 时刻下风险集数的 riskdadta()函数以及五种非参数估计的函数。

3.1.1风险集函数

我们构建 riskdata()函数来产生生存数据在 s 时刻下的死亡数和风险集数，用生存数据中的生存时间变量创建新的时间序列，以确保每个时间的死亡（或删失）人数不为零。

其中 s 时刻的死亡（或删失）人数直接计数

```
# 计算每个时间点的死亡人数
 death_table <- table(death_times)
 death_counts <- death_table[match(death_times.sort, names(death_table))]
  # 计算每个时间点的删失人数
 cen_table <- table(cen_times)
 cen_counts <- cen_table[match(cen_times.sort, names(cen_table))]
```

然后我们通过 s 时刻进入风险集的人数加上 s-1 时刻的风险集人数再减去 s-1 时刻的死亡（或删失）人数，最后存储该风险集人数为 s 时刻的风险集人数。

```
for (i in 2:length(t1)) {
risk_set_counts[1] <- sum(origin_data$ageentry <= t1[1])
 # 添加进入风险集的个体
```

```
  current_risk_set_size1 <- current_risk_set_size1+sum(origin_data$ageentry <=
t1[i]&origin_data$ageentry > t1[i-1])
  # 减去离开风险集的个体（死亡或删失）
  current_risk_set_size1 <- current_risk_set_size1- sum(origin_data$age == t1[i-1])
  # 存储当前时间点的风险集人数
  risk_set_counts[i] <- current_risk_set_size1
 }
```

完整函数程序见附录。

3.1.2非参数估计函数实现

我们构建 NPMLE()、BFH()、WNPMLE()、WBFH()、以及 CNPMLE()函数，通过上述几个方法来估计左截断数据的生存率，和其对应的方差。输入参数为经过计算各个事件发生时间点的风险集，及对应死亡人数和删失人数处理后的风险集数据。

对于非参数估计（NPMLE），由于不需要考虑其风险集人数和死亡人数之间的关系，可直接利用公式(1)(2)即可计算各时间点上的生存率和对应的方差。

```
## NPMLE
NPMLE <- function(data){#输入风险集数据
  risk_set_data <- data
  s_t.NPMLE <- numeric()
  c.NPMLE <- numeric()
  var.NPMLE <-numeric()
  s_t.NPMLE[1] <- 1-risk_set_data$DeathCount[1]/risk_set_data$RiskSetCount[1]
  c.NPMLE[1] <-
risk_set_data$DeathCount[1]/(risk_set_data$RiskSetCount[1]*(risk_set_data$RiskSetCount[1]-
risk_set_data$DeathCount[1]))
  var.NPMLE[1]<-s_t.NPMLE[1]^2*(c.NPMLE[1])
  for(i in 2:length(risk_set_data$Time)){#循环计算各个时间点上的生存率和方差
   s_t.NPMLE[i] <- s_t.NPMLE[i-1]*(1-
risk_set_data$DeathCount[i]/risk_set_data$RiskSetCount[i])
   c.NPMLE[i]<-c.NPMLE[i-
1]+risk_set_data$DeathCount[i]/(risk_set_data$RiskSetCount[i]*(risk_set_data$RiskSetCount[i]-
risk_set_data$DeathCount[i]))
   var.NPMLE[i]<-s_t.NPMLE[i]^2*c.NPMLE[i]
  }
  npmle <- cbind(s_t.NPMLE,var.NPMLE,c.NPMLE)#储存结果
```

```
  return(npmle)

}
```

对于 Breslow-Fleming-Harrington 估计法（BFH），计算其累计危险率，进一步利用公式(3)(4)求得其对应的生存率和方差值。

```
## BFH
BFH <- function(data){#输入风险集数据
 risk_set_data <- data
 s_t.BFH <- numeric()
 H_t.BFH <- numeric()
 c.BFH <- numeric()
 var.BFH <-numeric()
 H_t.BFH[1] <- risk_set_data$DeathCount[1]/risk_set_data$RiskSetCount[1]#计算累计危险率
 s_t.BFH[1] <- exp(-H_t.BFH[1])
 c.BFH[1] <- (risk_set_data$DeathCount[1]/risk_set_data$RiskSetCount[1]^2)*(1-
risk_set_data$DeathCount[1]/risk_set_data$RiskSetCount[1])
 var.BFH[1] <-s_t.BFH[1]^2*c.BFH[1]
 for(i in 2:length(risk_set_data$Time)){#循环计算各个时间点上的生存率和方差
  H_t.BFH[i] <- (H_t.BFH[i-1]+risk_set_data$DeathCount[i]/risk_set_data$RiskSetCount[i])
  s_t.BFH[i] <- exp(-H_t.BFH[i])
  c.BFH[i] <-c.BFH[i-1]+(risk_set_data$DeathCount[i]/risk_set_data$RiskSetCount[i]^2)*(1-
risk_set_data$DeathCount[i]/risk_set_data$RiskSetCount[i])
  var.BFH[i] <-s_t.BFH[i]^2*c.BFH[i]
 }
 bfh<-cbind(s_t.BFH,var.BFH,c.BFH)
 return(bfh)
}
```

对于加权 NPMLE 法（WNPMLE），需要输入权重参数,构建权重系数和权重函数 I(R)，相似的，根据公式（5）（6）计算其在权重系数下的生存率和方差，完整程序见附录。

权重函数的构架如下：

```
###权重系数
 Coef <- ceiling(c*(n^alpha))
###权重函数
 I <- function(R){
```

```
  test <- ifelse(R>=Coef,1,0)
  return(test)
 }
```

对于加权 BFH 法，相似的，利用权重函数控制其风险集的纳入条件，利用公式（7）（8）计算，完成程序见附录。

对于条件极大似然估计法（CNPMLE）,输入条件参数 u 控制其进入公式的风险集，利用公式（9）（10）计算生存率和方差。

```
##条件 NPMLE
CNPMLE <- function(data,u){ ###条件参数 u
 risk_set_data <- data
 s_t.CNPMLE <- numeric()
 c.CNPMLE<-numeric()
 var.CNPMLE<-numeric()
 if (risk_set_data$Time[1]>u) #判断是否满足条件
 {s_t.CNPMLE[1] <- 1-risk_set_data$DeathCount[1]/risk_set_data$RiskSetCount[1]
 c.CNPMLE[1]<-
risk_set_data$DeathCount[1]/(risk_set_data$RiskSetCount[1]*(risk_set_data$RiskSetCount[1]-
risk_set_data$DeathCount[1]))
 var.CNPMLE[1]<-s_t.CNPMLE[1]^2*c.CNPMLE[1]
 }
 else {
  s_t.CNPMLE[1] <- 1
  c.CNPMLE[1]<-0
  var.CNPMLE[1]<-0
 }

 for(i in 2:length(risk_set_data$Time)){
  if (risk_set_data$Time[i]>u)
  {s_t.CNPMLE[i] <- s_t.CNPMLE[i-1]*(1-
risk_set_data$DeathCount[i]/risk_set_data$RiskSetCount[i])
  c.CNPMLE[i]<-c.CNPMLE[i-
1]+risk_set_data$DeathCount[i]/(risk_set_data$RiskSetCount[i]*(risk_set_data$RiskSetCount[i]-
risk_set_data$DeathCount[i]))
  var.CNPMLE[i]<-s_t.CNPMLE[i]^2*c.CNPMLE[i]
  }
```

```
  else
  {s_t.CNPMLE[i] <- 1
  c.CNPMLE[i]<-0
  var.CNPMLE[i]<-0
  }
 }
 cnpmle<-cbind(s_t.CNPMLE,var.CNPMLE,c.CNPMLE)
 return(cnpmle)
}
```

最后通过 non_parametric_estmation（）函数将五种方法的结果进行整合，完整函数程序见附录。

3.1.3实例分析

非参数估计的实例是 R 语言的 survival 包中的 channing 数据集，这个数据集是指与“钱宁之家”（Channing House）相关的数据集。我们将 channing 中的数据分性别进行了非参数估计并且做出了相应的累积生存曲线。

```
Cumulative_Survival_Rate_Graph<-function(data_1,data_2)
{
#绘制女性生存曲线图
 df1_1<-as.data.frame(data_1)
 colnames(df1_1) <- c("Time" ,"RiskSetCount","DeathCount","cenCount","NPMLE" ,"var.NPMLE
","c.NPMLE","BFH" ,"var.BFH","c.BFH","WNPMLE","var.WNPMLE" ,"c.WNPMLE" ,"WBFH" ,"
var.WBFH" ,"c.WBFH" ,"CNPMLE" ,"var.CNPMLE","c.CNPMLE")
 df2_1 <- melt(df1_1, id.vars =
"Time",measure.vars=c("NPMLE","BFH","WNPMLE","WBFH","CNPMLE"), variable.name =
"group", value.name = "y")

 gg1<-ggplot(df2_1, aes(x = Time, y = y, color = group)) +
  geom_line() +
  labs(title = "女性累计生存率图", x = "年龄（月）", y = "累积生存率", color = "非参数估计方
法") +
  theme_minimal()

  #绘制男性生存曲线图
 df1_2<-as.data.frame(data_2)
```

```
 colnames(df1_2) <- c("Time" ,"RiskSetCount","DeathCount","cenCount","NPMLE" ,"var.NPMLE
","c.NPMLE","BFH" ,"var.BFH","c.BFH","WNPMLE","var.WNPMLE" ,"c.WNPMLE" ,"WBFH" ,"
var.WBFH" ,"c.WBFH" ,"CNPMLE" ,"var.CNPMLE","c.CNPMLE")
 df2_2 <- melt(df1_2, id.vars =
"Time",measure.vars=c("NPMLE","BFH","WNPMLE","WBFH","CNPMLE"), variable.name =
"group", value.name = "y")

 gg2<-ggplot(df2_2, aes(x = Time, y = y, color = group)) +
  geom_line() +
  labs(title = "男性累计生存率图", x = "年龄（月）", y = "累积生存率", color = "非参数估计方
法") +
  theme_minimal()
 print(gg1)
 print(gg2)
}
```

3.2 假设检验

3.2.1面积法

通过应用左截断数据的非参数最大似然估计（NPMLE）、Breslow-Fleming-Harrington（BFH）估计、加权 NPMLE（WNPMLE）、加权 BFH（WBFH）和条件 NPMLE（CNPMLE）方法，我们能够估计出生存时间以及在每个时间点的生存概率。利用这些生存时间和相应的生存概率，我们可以计算两条生存曲线下的面积（AUC）。

为了进行正态近似，我们需要结合生存时间及其对应时间点的生存概率的方差信息，以估算期望值和 AUC 差异的总方差。通过这种正态近似，我们能够推导出一个统计量，进而用于计算比较两条生存曲线的统计显著性，即 p 值。

首先根据公式（12），计算两条生存曲线之间的绝对面积之大小，作为衡量差异的指标：

```
#定义计算生存曲线面积差的函数
diffarea <- function(data1,data2){
 merged_data <- Mergeddata(data1,data2)#合并两组数据时间点
 t <- merged_data$t  # 提取生存曲线的估计值和时间点
 # 初始化面积变量
 diffarea <- 0  # 计算每个时间点的面积并累加
 for (i in 1:(length(t) - 1)) {
  diffarea <- diffarea + abs(merged_data$S1[i] - merged_data$S2[i]) * (t[i+1]-t[i])
```

```
  }
  return(diffarea)
}
```

根据公式（16）（17），我们可以计算统计量的期望值和方差。在面积检验法中，面积指标 Δ 的均值和方差的计算涉及到多个时间点 t_j 和 $t_{j'}$。这些时间点是从两组数据的时间点 t_{1i} 和 t_{2i} 中提取出来的，并合并成一个统一的时间序列。

```
# 计算 Delta 期望函数
calculate_EhatD <- function(data){
  t <- data$t
  dt <- diff(t)# 计算时间间隔
  a <- data$sigma2_S1
  b <- data$sigma2_S2
  # 初始化期望
  Ehat_D <- 0
  for (j in 1:(length(t) - 1)) {
    # 计算平方根内的表达式
    variance_sum <- a[j] + b[j]
    # 计算加权和
    weighted_sum <- (2 / pi) * sqrt(variance_sum) * dt[j]
    # 累加结果
    Ehat_D <- Ehat_D + weighted_sum
  }
  return(Ehat_D)
}
# 计算 delta 方差函数
calculate_VarD <- function(data){
  t <- data$t
  S1 <- data$S1
  S2 <- data$S2
  sigma2_S1 <- data$sigma2_S1
sigma2_S2 <- data$sigma2_S2
  # 初始化方差
  var_D <- 0
# 计算自相关项
```

```
 for (j in 1:(length(t) - 1)) {
  t_j <- t[j]
  t_j1 <- t[j + 1]
  var_D <- var_D + (t_j1 - t_j)^2 * (1 - 2/pi) * (sigma2_S1[j] + sigma2_S2[j])
 }
 # 计算互相关项
 for (j in 1:(length(t) - 1)) {
  t_j <- t[j]
  t_j1 <- t[j + 1]
  for (j_prime in 1:(length(t) - 1)) {
   if (j != j_prime) {
    t_jp <- t[j_prime]
    t_jp1 <- t[j_prime + 1]
    var_D <- var_D + (t_j1 - t_j) * (t_jp1 - t_jp) * (1 - 2/pi) *
     sqrt((sigma2_S1[j] + sigma2_S2[j]) * (sigma2_S1[j_prime] + sigma2_S2[j_prime]))
   }
  }
 }

 return(var_D) # 返回结果
}
```

对于 Var(Δ)方差的计算，需要构建内外两个循环并遍历每一个时间点上的互相关性。

3.2.2置换面积法

在采用面积法的基础上，我们首先应用置换方法于左截断数据的非参数估计之前。通过先进行数据的随机置换，然后执行非参数估计和面积法检验，我们能够获得以下优势：这种方法不依赖于分布的假设，适用于样本量较小或数据分布不明的情况，并且能够灵活适应复杂的实验设计。

过程：置换数据、非参数估计、面积法检验

```
permutation_method <- function(origindata,original_statistic){
channingdata_f2 <- origindata[origindata$group==0,]#提取性别（0 男 1 女）
 channingdata_m2 <- origindata[origindata$group==1,]
 library(dplyr)
 comb_df <-  bind_rows(channingdata_f2, channingdata_m2)
 set.seed(123)
```

```
 n <- 500
 perm_stats <- numeric(n)
 for(k in 1:n){
  #permuted <- sample(comb_df)
  qw <- length(channingdata_f2$time)

  #sampled_rows <- comb_df[sample(nrow(comb_df), size = 300), ]

  total_rows <- nrow(comb_df)
  sampled_indices <- sample(total_rows, size = qw)
  perm_group1 <- comb_df[sampled_indices, ]
  perm_group2 <- comb_df[-sampled_indices, ]

  #非参数估计主程序
  ##置换数据生成
  risk_data1 <- riskdata(perm_group1)#女性风险集数据
  risk_data2 <- riskdata(perm_group2)#男性风险集数据
  risk_data <- riskdata(origindata)#总体风险集数据


  #最终结果置换结果
  estmation_data1 <- non_parametric_estmation(risk_data1,u=1)
  estmation_data2 <- non_parametric_estmation(risk_data2,u=1)

  #面积法假设检验（以 WNPMLE 为例）
  #合并数据时间点函数置换
merged_data <- Mergeddata(estmation_data1,estmation_data2)
  # 计算置换生存曲线面积差
  all_diff <- diffarea(estmation_data1,estmation_data2)

  # 计算统计量和对应的 p 值
  # 计算 Delta 期望函数和方差函数
  Ehat_D <- calculate_EhatD(merged_data)
```

```
  Varhat_D <- calculate_VarD(merged_data)
  #Dstar <- (all_diff - Ehat_D) / (Varhat_D^(1/2))
  perm_stats[k]<- (all_diff - Ehat_D) / (Varhat_D^(1/2))

 }
 PMT_P_value <- mean(abs(perm_stats) >= abs(original_statistic))
 #result <- data.frame(perm_stats,PMT_P_value)
 return(PMT_P_value)
}
```

3.2.3Bootstrap 面积法

在采用面积法的基础上，我们首先应用 Bootstrap 方法于左截断数据的非参数估计之前。通过先进行数据的重采样，然后执行非参数估计和面积法检验，我们能够获得以下优势：这种方法不依赖于分布的假设，适用于样本量较小或数据分布不明的情况，并且能够灵活适应复杂的实验设计。

过程：Bootstrap、非参数估计、面积法

```
#bootstrap 循环
 for (k in 1:n_bootstrap) {
  # 获取样本数量
  num_female_samples <- nrow(channingdata_f2)
  num_male_samples <- nrow(channingdata_m2)
  # 自助抽样：回到原假设，从总体样本中抽取两组的样本，允许重复
  bootstrap_group_female <- origindata[sample(nrow(origindata), size = num_female_samples,
replace = TRUE), ]
  bootstrap_group_male <- origindata[sample(nrow(origindata), size = num_male_samples, replace =
TRUE), ]

    # 非参数估计主程序
  risk.data_f <- riskdata(bootstrap_group_female)  # 使用 bootstrap_group_female
  risk.data_m <- riskdata(bootstrap_group_male) # 使用 bootstrap_group_male
  result_f <- non_parametric_estmation(risk.data_f,u=781)
  result_m <- non_parametric_estmation(risk.data_m,u=781)
    # 合并数据时间点
  merged_data <- Mergeddata(result_f, result_m)
```

```
  # 计算生存曲线面积

  all_diff[k] <- diffarea(result_f,result_m)

  # 计算统计量和对应的 p 值
  Ehat_D[k] <- calculate_EhatD(merged_data)
  Varhat_D[k] <- calculate_VarD(merged_data)
  boot_stats_results[k] <- (all_diff[k] - Ehat_D[k]) / sqrt(Varhat_D[k])
 }
```

4 模拟比较

4.1 模拟数据生成

4.1.1模拟设计

为了评价 ABS 和 ABSP test 在基于左截断数据的条件下的稳健性和适用性，我们通过模拟数据实验，计算检验统计量Δ，并计算两者的一类错误和检验效能，将他们与 Aalen model Supremum-test 检验进行比较。一共有六类生存数据数据，第一类为两组生存时间均有参数一样的指数分布产生用于生成Ⅰ类错误数据，后五类数据用于模拟比较检验效能的五种情形：第二类两组生存时间组间风险率满足成比例假定，第三类两条生存曲线前中期发散后期收敛，第四类为前中期收敛后期发散，第五类两条生存曲线交叉与中期，即交叉点位于 S(t)=0.4~0.6 附近，第六类两条生存曲线交叉于后期，即交叉点位于 S(t)=0.2~0.4 附近。六类生存时间数据曲线理论图见图 4-1 所示，L 为左截断数据函数，X 为生存时间函数，其具体模拟数据参数如下（l 代表指数分布参数，w 表示 weibull 分布）：

Ⅰ类（Ⅰ类错误）：

$L_1(t) = 0.25l; L_2(t) = 0.25l$

$X_1(t) = 0.15l; X_2(t) = 0.15l$

Ⅱ类（风险率成比例）：

$L_1(t) = 0.25l; L_2(t) = 0.5l$

$X_1(t) = 0.15l; X_2(t) = 0.3l$

Ⅲ类（前中期差异）：

$L_1(t) = 1.2l(t \le 0.8) + 0.1l(0.8 < t \le 1.8) + 0.5l(1.8 < t \le 2.6) + l(t \ge 2.6)$

$L_2(t) = 0.5l(t \le 0.8) + 0.1l(0.8 < t \le 1.8) + 1.2l(1.8 < t \le 2.6) + l(t \ge 2.6)$

$X_1(t) = 0.9l(t \le 0.8) + 0.06l(0.8 < t \le 1.8) + 0.35l(1.8 < t \le 2.6) + 0.8l(t \ge 2.6)$

$X_2(t) = 0.35l(t \le 0.8) + 0.06l(0.8 < t \le 1.8) + 0.9l(1.8 < t \le 2.6) + 0.8l(t \ge 2.6)$

Ⅳ类（后期差异）

$L_1(t) = l(t \le 0.8) + 2.5l(t \ge 0.8)$

$L_2(t) = l(t \le 0.8) + 0.5l(t \ge 0.8)$

$X_1(t) = 0.75l(t \le 0.8) + 1.875l(t \ge 0.8)$

$X_2(t) = 0.75l(t \le 0.8) + 0.375l(t \ge 0.8)$

Ⅴ类（中期交叉）

$L_1(t) = \frac{1}{12}l$

$L_2(t) = 0.25l(t \le 2) + \frac{1}{35}l(t \ge 2)$

$X_1(t) = (\frac{1}{12} - 0.02)l$

$X_2(t) = 0..1l(t \le 2) + (\frac{1}{35} - 0.015)l(t \ge 2)$

Ⅵ类（后期交叉）

$L_1(t) = W(1.5,5)$

$L_2(t) = 0.5l(t \le 2.2) + 0.1l(t \ge 2.2)$

$X_1(t) = W(1.5,5)$

$X_2(t) = 0.5l(t \le 2.2) + 0.1l(t \ge 2.2)$

4.1.2模拟数据程序以Ⅱ类数据为例：

```
class2_data<- function(n1,n2,n_sim,w1,w2){
 rate2_1 <- 0.25;rate2_2 <- 0.5  #先输入参数

 data_list_2 <- list() #建立空列表储存数据

 for(i in 1:n_sim){  #n_sim 为模拟次数，本次模拟均为 500 次

  id1 <- c(1:n1)
  id2 <- c(1:n2)

  count1 <- 0
  count2 <- 0
  turn_data1 <- data.frame()
  turn_data2 <- data.frame()

   #生成固定样本量的左截断数据
  while(count1<n1){
   l <- rexp(n1,rate = rate2_1)
```

```
  x <- rexp(n1,rate = rate2_1-0.1)
  sup_data <- data.frame(l[l<x],x[l<x])
  names(sup_data) <- c("L1","X1")
  turn_data1 <- bind_rows(turn_data1,sup_data)
  count1 <- nrow(turn_data1)
}
data1 <- cbind(data.frame(id1),turn_data1[sample(nrow(turn_data1),n1),])

  #生成固定样本量的生存时间数据

while(count2<n2){
  l <- rexp(n2,rate = rate2_2)
  x <- rexp(n2,rate = rate2_2-0.2)
  sup_data <- data.frame(l[l<x],x[l<x])
  names(sup_data) <- c("L2","X2")
  turn_data2 <- bind_rows(turn_data2,sup_data)
  count2 <- nrow(turn_data2)
}
data2 <- cbind(data.frame(id2),turn_data2[sample(nrow(turn_data2),n2),])


  #设置研究时间窗口长度 w，这里我们假设 w 是一个固定的值或者从某个分布中抽取
  #这里我们从随机分布
wrep1 <- runif(n1,0,w1)
wrep2 <- runif(n2,0,w2)

L1 <- data1$L1
X1 <- data1$X1
L2 <- data2$L2
X2 <- data2$X2
  #计算删失时间 C
C1 <- L1+ wrep1
C2 <- L2+ wrep2

censored1 <- X1 < C1
```

```
  censored2 <- X2 < C2
  count_TRUE1 <- sum(censored1)
  count_TRUE2 <- sum(censored2)
  censored_rate1 <- count_TRUE1/n1
  censored_rate2 <- count_TRUE2/n2

  ageentry <- trunc(c(L1,L2)*100)
  age <- trunc(c(X1,X2)*100)
  death <- 1-c(censored1,censored2)
  time <- age-ageentry
  obs <- c(1:(n1+n2))
  group <- c(rep(0,n1),rep(1,n2))
  censored_rate <- c(rep(censored_rate1,n1),rep(censored_rate2,n2))
  wtime <- c(rep(w1,n1),rep(w2,n2))

  data_list_2[[i]] <- data.frame(obs,death,ageentry,age,time,group,censored_rate,wtime)
```

#将每次模拟的结果存储在列表中

```
 }
 return(data_list_2)
}
```

在每一种参数下都模拟设置了删失比例为 0%，20%的情形，同时考虑了均衡设计（$N_1 = N_2 = 50,100,150$）以及非均衡设计（$N_1 = 50, N_2 = 100$）;对各种检验方法统计性能的影响。每一个参数组合下模拟 500 次，显著性水平为 α =0.05.首先产生服从某一特定分布（如指数分布、weibull 分布）左截断时间 L，生存时间 X（生存时间 X 服从分布的参数比左截断时间 L 所服从的分布大）；设计窗口时间 w 来控制模拟数据的删失率，在模拟中窗口时间为可以为固定的数值，也可以是从某一分布中随机抽取，本实验模拟的数据的两组窗口时间 w 分别由服从 U（0，w1）和 U（0，w2）的均匀分布产生，记作生存时间 X=min（X，L+w），通过设定参数 w1 和 w2 的值控制每组平均删失比例。模拟数据生存曲线如图 4-2 所示。

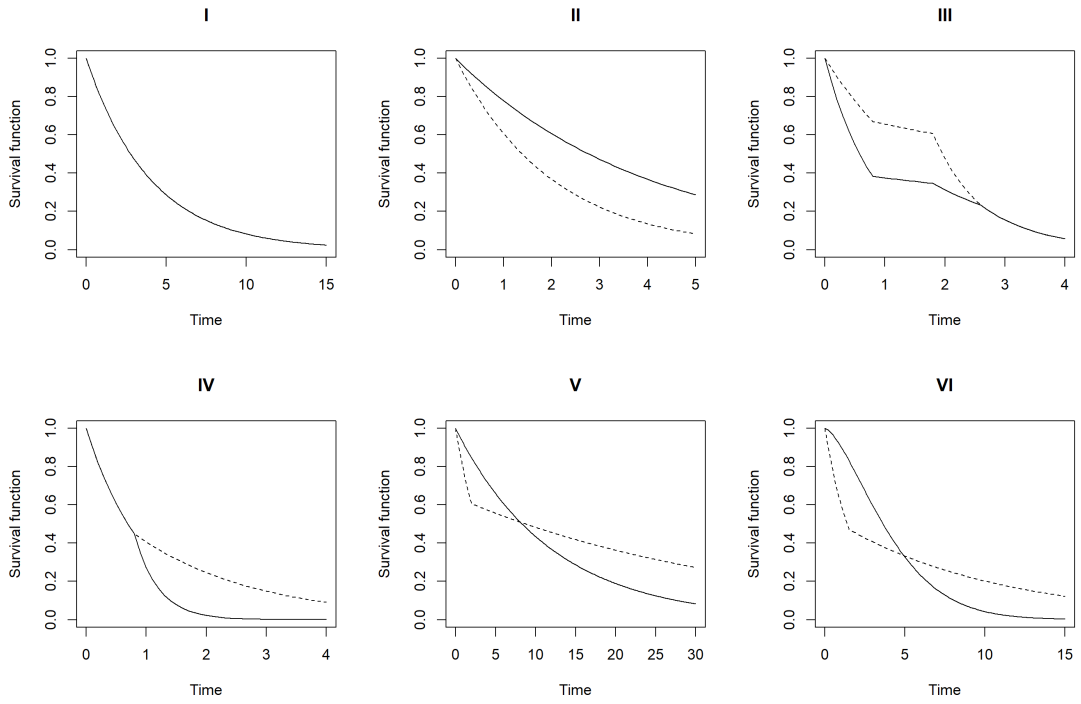


图 1 六种模拟理论图

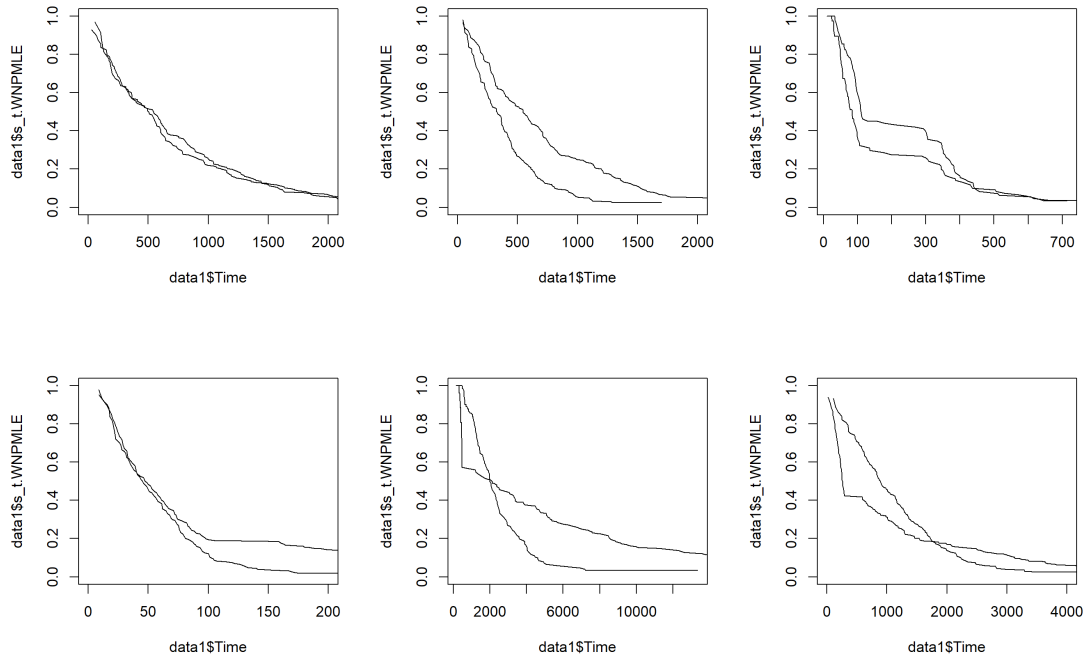


图 2 六种模拟生存函数图

4.1.2 **模拟结果**

（1）面积检验和置换面积检验统计量Δ的正态性研究

在原假设得到不同的样本量、删失率组合下的检验统计量Δ，根据检验统计量的取值绘制出相应对应的密度直方图，计算其数据偏离正态分布程度的 p 值，结合正态性检验判断检验统计量Δ是否服从于正态分布。我们考虑删失比例（Censoring Rate，CR）为 0%、20%的情形，同时考虑了样本量的均衡设计（$N_1 = N_2 = 50,100,150$）和非均衡设计（$N_1 = 50, N_2 = 100$）。

表 1 不同样本量、删失率组合下检正态性检验 p 值

	Ⅰ型	Ⅱ型	Ⅲ型	Ⅳ型	Ⅴ型	Ⅵ型
（0%，面积）	2.2e-16	0.02072	0.06957	0.0002594	1.133e-6	4.935e-9
（20%，面积）	2.2e-16	0.1941	0.8012	0.04746	0.00167	0.000113
（0%，置换）	2.2e-6	2.2e-16	2.62e-15	2.6e-16	2.2e-16	2.2e-16
（20%，置换）	4.4e-6	2.911e-13	2.2e-16	2.2e-16	1.869e-14	2.2e-16

图 3 不同样本量、删失率组合下检验统计量Δ的概率密度分布图

图 3 是检验统计量Δ的密度直方图和拟合曲线（横坐标为Δ取值，纵坐标为密度分布）。通过模拟研究发现，在六种样本量与两种删失率的不同组合下（所得 p 值图见表 4-1）：p 值大部分都小于 0.05，在根据图 3 所示，正态性检验结果显示在不同样本量和删失率组合下检验统计量不太服从正太分布。

4.1.3模拟结果

（1）面积检验和置换面积检验统计量△的正态性研究

在原假设得到不同的样本量、删失率组合下的检验统计量△，根据检验统计量的取值绘制出相应对应的密度直方图，计算其数据偏离正态分布程度的 p 值，结合正态性检验判断检验统计量△是否服从于正态分布。我们考虑删失比例（Censoring Rate，CR）为 0%、20%的情形，同时考虑了样本量的均衡设计（$N_1 = N_2 = 50,100,150$）和非均衡设计（$N_1 = 50, N_2 = 100$）。图 4-3 是检验统计量△的密度直方图和拟合曲线（横坐标为△取值，纵坐标为密度分布）。通过模拟研究发现，在六种样本量与两种删失率的不同组合下（所得 p 值图见表 4-1）：p 值大部分都小于 0.05，在根据图 4-3 所示，正态性检验结果显示在不同样本量和删失率组合下检验统计量不太服从正太分布。

表 2 不同样本量、删失率组合下检正态性检验 p 值

	Ⅰ型	Ⅱ型	Ⅲ型	Ⅳ型	Ⅴ型	Ⅵ型
（0%，面积）	2.2e-16	0.02072	0.06957	0.0002594	1.133e-6	4.935e-9
（20%，面积）	2.2e-16	0.1941	0.8012	0.04746	0.00167	0.000113
（0%，置换）	2.2e-6	2.2e-16	2.62e-15	2.6e-16	2.2e-16	2.2e-16
（20%，置换）	4.4e-6	2.911e-13	2.2e-16	2.2e-16	1.869e-14	2.2e-16

4.2 模拟分析

根据上面模拟数据的生成函数，我们根据各个不同类型的数据生成函数，我们生成对于情形一（一类错误）、情形二（风险率成比例）、情形三（前中期差异）、情形四（后期差异）、情形五（中期交叉）、情形六（后期交叉）这六种情形，每种情形设置了（50，100）、（100，100）、（150，150）三种不同样本量，以及零删失和 20%附近删失两组删失情况，一共 36 中不同的情形，每种不一样的情形用 500 组数据进行模拟面积法和置换面积法的检验结果（对于 bootstrapBootstrap 面积法，可能由于时间原因）；探究其的一类错误，以及在不同样本量不同删失率下的检验效能；

表 3 不同检验方法的模拟Ⅰ类错误

样本	ErrorⅠ面积法	ErrorⅠ置换面积	Cox	pval.test
（50，100）	0.016	0.056	0.070	0.130
（100，100）	0.010	0.048	0.040	0.088
（150，150）	0.024	0.064	0.046	0.070
（50，100） 0.2 删失	0.014	0.058	0.088	0.126
（100，100） 0.2 删失	0.002	0.050	0.052	0.072
（150，150） 0.2 删失	0.008	0.048	0.020	0.068

4.2.1面积法的表现：

在所有样本量和删失条件下，面积法的 I 类错误率始终保持在较低水平，表现出极佳的 I 类错误控制能力。这说明面积法在假设检验中很少出现错误拒绝零假设的情况，具有较高的准确性。面积法在样本量较小时（如 50）也能保持低 I 类错误率，且在样本量增大时进一步下降，表明其对样本量的鲁棒性较强。即便在删失率达到 0.2 的情况下，面积法的 I 类错误控制依然稳定，基本维持在低水平（如样本量 100 和删失 0.2 时，I 类错误降至 0.002）。因此，面积法在处理数据缺失和删失时表现优异。

4.2.2置换面积法的表现：

置换面积法的 I 类错误控制也较为良好，但相比面积法略高。它在各个条件下都能保持相对稳定的 I 类错误率，适合用于对 I 类错误控制要求不特别严格的情境。置换面积法在样本量增大（如从 50 增至 150）时 I 类错误率有所改善。这表明它在样本量增加的情况下对 I 类错误控制更好。在有 0.2 删失的情况下，置换面积法的 I 类错误率会略微上升，但并不显著（如样本量 100 和删失 0.2 时，I 类错误为 0.050）。因此，它在有删失数据的情况下仍能提供一定的 I 类错误控制效果。

用于参考比较的 cox 模型的 I 类错误控制相对较稍差，尤其在样本量较小的情况下，I 类错误率稍微偏高。Cox 模型在小样本情境中较容易出现错误拒绝零假设的情况，可能导致较多假阳性结果。但 I 类错误随着样本量增加也呈现下降，其 I 类错误率接近面积法和置换面积法。总的来说，面积法是三种方法中控制 I 类错误比较稳定的方法，尤其适用于数据完整性较差（存在删失）或样本量较小的情况。置换面积法表现良好，在大样本或中等删失的情况下仍能保持较低的 I 类错误，因此是一个较为稳健的选择，尤其适合中等至大样本的情境。加性风险模型的 pval.test 在小样本下 I 类错误率较高，容易出现假阳性，稳定性稍差，更为激进，但在样本量较大时表现会有所改善，在小样本或高删失条件下建议谨慎使用。

表 4 情形二（风险成比例）的模拟检验效能

	面积法-power	置换面积-power	cox -power
（50，100）	0.542	0.806	0.976
（100，100）	0.808	0.924	0.988
（150，150）	0.964	0.988	1
（50，100） 0.2 删失	0.564	0.882	0.964
（100，100） 0.2 删失	0.768	0.980	0.998
（150，150） 0.2 删失	0.970	1.000	1.000

在情形二（风险率成比例）时，在没有删失的情况下，三种方法的检验效能都相对较高，尤其是 cox 模型，这表明在完整数据的情况下，模型的检测能力较强。并且随着样本量的增加从(50,100)到(150,150)，三种方法的检验效能普遍提高，这表明更大的样本量有助于提高检验效能。当引入 0.2 的删失后，三种方法的检验效能都有所提高，这可能表明在此种情形下，删失数据的情况下，这些方法能够更好地适应数据的不完整性，从而提高检验效能。

在删失情况下，(150,150)样本量的 cox 模型 power 达到了 1.000，这意味着在这种样本量下，即使有删失，cox 模型也能够完全检测到效应。并且在所有样本量和删失情况下，置换面积法的检验效能都高于面积法。这意味着在检测实际存在的效应时，置换面积法更有可能正确地拒绝零假设。置换面积法在不同的样本量和删失情况下都保持了较高的检验效能。

表 5 情形三（前中期差异）的模拟检验效能

	面积法-power	置换面积-power	cox -power
（50，100）	0.632	0.726	0.688
（100，100）	0.732	0.820	0.802
（150，150）	0.862	0.906	0.916
（50，100） 0.2 删失	0.466	0.572	0.492
（100，100） 0.2 删失	0.602	0.702	0.648
（150，150） 0.2 删失	0.816	0.894	0.830

在情形三（前中期差异）情况下，首先是随着样本量的增加，个方法检验效能普遍提高，更大的样本量有助于提高检验效能。引入 0.2 的删失后，所有方法的检验效能都有所下降，但下降的幅度因方法而异。这表明在前中期差异这种情况下删失会降低检验效能，但不同方法对删失的敏感性不同，即不同的方法对于删失数据检验效能的降低程度有所不同，其中面积法和置换面积法对于删失数据的检验效能降低程度较小，可能更适合在有删失的情况下使用。

置换面积方法在所有情况下都显示出比面积法和 cox 更高的检验效能。这表明置换面积法在检测前中期差异方面更为有效。cox 在没有删失的情况下表现良好，但在删失情况下，其检验效能下降幅度较大，尤其是在较小样本量时。在所有测试的样本量和删失情况下，置换面积法的检验效能都高于面积法和 cox。这意味着在实际应用中，使用置换面积法更有可能检测到实际存在的组间效应。

表 6 情形四（后期差异）的模拟检验效能

	面积法-power	置换面积-power	cox -power
（50，100）	0.394	0.678	0.934
（100，100）	0.670	0.856	1.000
（150，150）	0.556	0.976	1.000
（50，100） 0.2 删失	0.418	0.966	0.996
（100，100） 0.2 删失	0.862	1.000	1.000
（150，150） 0.2 删失	0.998	1.000	1.000

在情形四（后期差异）时，对于所有方法，首先是随着样本量的增加，检验效能提高。在引入 0.2 的删失后，所有方法的检验效能都有所提高，尤其是面积法与置换面积法。这可能是因为在后期差异的情况下，删失可能去除了早期事件的影响，使得后期的差异更加明显。其中置换面积法和 cox 在这种情形下表现出了极高的检验效能，这可能与模拟数据时候，由于生成数据的函数设置的组间参数过于极端，使得两组差异较大，更容易检验出两组生存数据的差异，这在下面的某些情形中也有出现。但是从中也可以体现出置换面积法对于面积法在改进之后对于检验效能的提升是显著的。

表 7 情形五（中期交叉）的模拟检验效能

	X5 面积法	X5 置换面积	pval.test
（50，100）	0.144	0.258	0.990
（100，100）	0.208	0.624	1.000
（150，150）	0.826	0.950	1.000
（50，100） 0.2 删失	0.836	1.000	1.000
（100，100） 0.2 删失	0.962	0.996	1.000
（150，150） 0.2 删失	1.000	1.000	1.000

注（下表同）：pval.test：Aalen model Supremum-test of significance p-value

在情况五（生存曲线中期交叉）时，由于作为对比的 aalen model 里面的 pval.test 特别适合与两组生存率不成比例的情况，而生存曲线交叉则刚好符合这种情形，则用来对比的的则用来对比的 pval.test 的检验效能格外的高，而作为计算两条生存曲线间面积作为面积指标构造检验统

计量的面积法在小样本情况下的检验效能不高，但在稍大样本的检验之中，检验效能非常不错，且在 pval.test 存在着Ⅰ类错误较大的缺点时，面积法和置换面积法会有着更为稳健的检验结果。

表 8 情形六（后期交叉）的模拟检验效能

	面积法-power	置换面积-power	pval.test-power
（50，100）	0.490	0.592	0.828
（100，100）	0.787	0.875	0.827
（150，150）	0.968	0.976	0.958
（50，100） 0.2 删失	0.122	0.778	0.806
（100，100） 0.2 删失	0.544	0.994	0.858
（150，150） 0.2 删失	0.500	1.000	0.906

在情形六（后期交叉）时，对于所有方法，随着样本量的增加，检验效能普遍提高。引入删失后，所有方法的检验效能都有所下降，删失会降低检验效能，但不同方法对删失的敏感性不同。置换面积方法在部分情况下显示出比面积法和 pval.test 方法更高的检验效能。这表明置换面积法在检测后期交叉效应方面更为有效。pval.test 在没有删失的情况下表现良好，但在删失情况下，其检验效能下降幅度较大，尤其是在较小样本量时。总的来说，置换面积法的检验效能高于面积法和 pval.test 方法。这意味着在实际应用中，使用置换面积法更有可能检测到实际存在的效应。置换面积法在处理后期交叉的统计检验中表现出了明显的优势，尤其是在样本量较小或数据存在删失时。这种优势可能来源于置换面积法不依赖于数据的分布假设，因此在处理删失数据或数据分布未知时可能更为有效。此外，置换测试通过随机化的方法来评估统计量的分布，这可能为检验提供了更为精确的 p 值估计，从而提高了检验效能。

表 9 各情况的方法检验效能均值

2 面积法	2 置换面积	3 面积法	3 置换面积	4 面积法	4 置换面积	5 面积法	5 置换面积	6 面积法	6 置换面积
0.769	0.930	0.680	0.770	0.705	0.913	0.663	0.805	0.568	0.865

总的来说，由于各个情况下模拟检验的次数（500 次）和置换次数（500 次）较少，会有较大的随机误差，模拟检验的结果有瑕疵；且个别情况的引入删失的数据生成可能组间差异过于明显，使得检验效能偏高；但各情形下的平均检验效能仍然具有一定的参考价值和意义，从中可以看出来的面积法在控制了Ⅰ类错误的情况下有着较为保守稳健的检验效能，而置换面积，既较好地控制了 I 类错误，又在生存曲线交叉、生存曲线部分发散(收敛) 时，较于普通面积法提高了检验效能，特别是组间风险率成比例假设失效或生存曲线交叉时具有较高效能。

5 实例分析

5.1 实例介绍

非参数估计的实例是 R 语言的 survival 包中的 channing 数据集，这个数据集是指与“钱宁之家”（Channing House）相关的数据集。钱宁之家是位于加利福尼亚州帕洛阿尔托的一个退休中心，该数据集是在 1964 年该中心开业至 1975 年 7 月 1 日期间收集的。数据集包含 462 行和 5 列，主要涉及经过该中心的 97 名男性和 365 名女性的相关信息。

这些数据可能包含有关这些人在退休中心的居住时间、健康状况、服务使用情况等方面的信息，通常用于生存分析。

5.2 实例分析

在 t 图 5-1 中，NPMLE 法所做的生存曲线在 800 个月之前急速下降为 0，BFH 法虽然避免突然降低为 0，但仍然在 800 个月之前骤降；参考陈金宝中的参数令 c=1，$\alpha=\frac{1}{3}$，使得权重函数$I_{|R_{m(n)}(s)\geq cn^{\alpha}|}$中 cn^{α} 男性和女性分别约为 5 和 8，合理地避免风险集数 $R_m(s)$过小情形，加权 BFH 和加权 NPMLE 两条实线基本重合；条件 NPMLE（ t ＞781 个月）和两条实线都基本重合，三种方法都避免了生存率突然降低为 0 的情况。

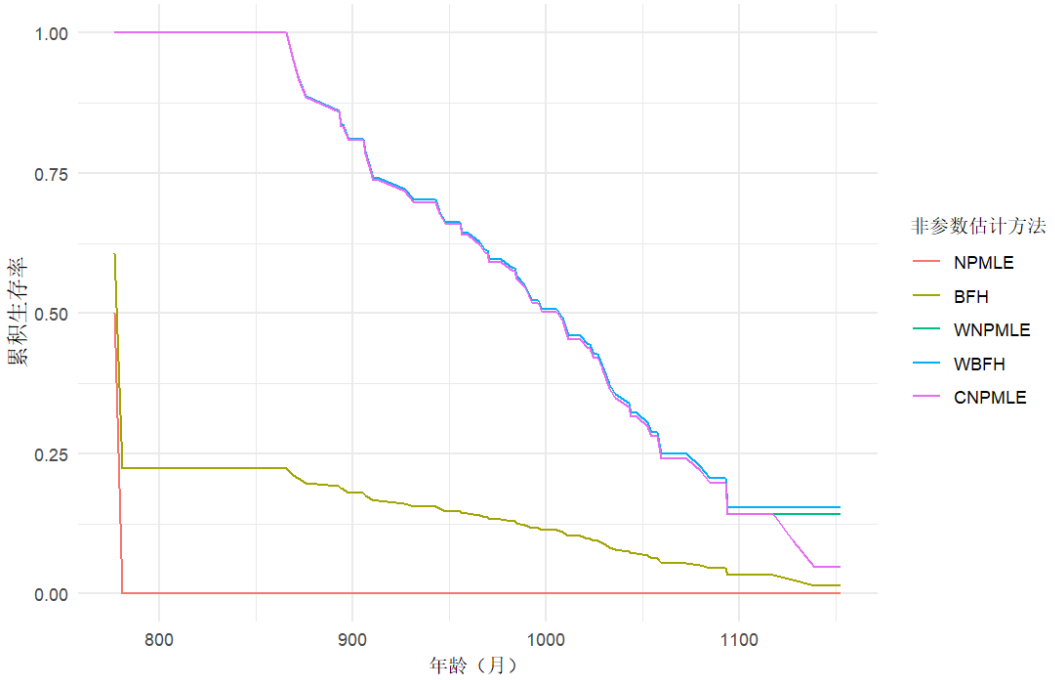


图 4 基于 Channing 数据的男性累积生存曲线

在图 5-2 中，发现不存在风险集数$R_m(s)$过小或 $R_m(s)$和死亡数 $\Delta L_{m(n)}(s)$相等的小风险集情形时，校正与未校正的非参数估计法结果基本一致， 5 条图线基本重合在一块。

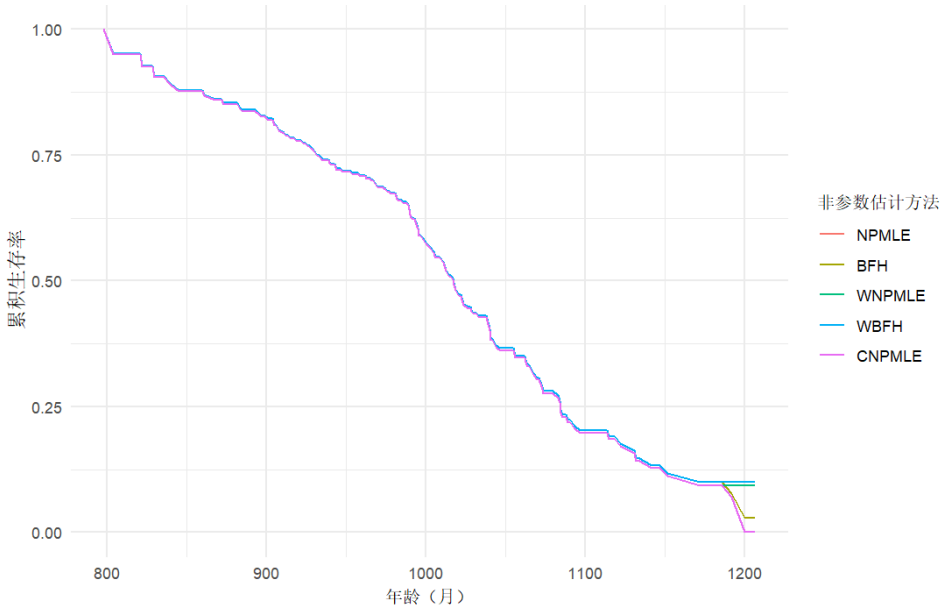


图 5 基于 Channing 数据的女性累积生存曲线

6 结果讨论

在本次模拟研究中，由于小组成员的设备性能有限，模拟检验和置换次数设定并不高，每种情形下的模拟检验次数只为 500 次，这个数字相对较小。500 次的模拟可能不足以完全捕捉到所有可能的变异性，导致结果中存在较大的随机误差。由于模拟次数较少，得到的估计可能不够稳定，这可能导致对 I 类错误和检验效能的估计不够准确。在实际应用中，这可能意味着我们对方法性能的评估存在一定的不确定性。在模拟中引入删失可能会影响数据的分布和组间差异。如果删失引入的方式导致组间差异过于明显，那么检验效能可能会被人为地提高，因为差异更明显的情况下，更容易检测出效应。如果模拟数据生成过程中组间差异设置得过于极端，那么在实际应用中可能并不具有代表性。这可能会高估某些方法在实际数据中的表现，因为实际数据中的差异可能不会如此明显。

尽管存在上述问题，各情形下的平均检验效能仍然具有一定的参考价值。它提供了一个总体的趋势，表明在不同情况下，不同方法可能的表现。面积法在控制 I 类错误的同时，显示出较为保守但稳健的检验效能。这意味着面积法不太可能产生假阳性结果，即使在样本量较小或数据完整性较差的情况下也能保持较低的 I 类错误率。置换面积法在控制 I 类错误方面表现良好，并且在生存曲线交叉或部分发散（收敛）的情况下，相比于普通面积法，显示出更高的检验效能。这表明置换面积法在处理复杂生存曲线时更为有效，尤其是在组间风险率成比例假设不成立或生存曲线出现交叉时。尽管模拟检验的次数和置换次数较少，可能导致结果存在一定的随机误差，且个别情况下删失数据生成可能导致组间差异过于明显，影响检验效能的估计，但平均检验效能的结果仍然为我们提供了有价值的信息。置换面积法在不同情况下的表现，尤

其是在控制 I 类错误并没有明显提高的情况下，拥有更优于面积法的检验效能，为我们选择适合的统计方法提供了参考和新的选择思路。在实际应用中，我们应考虑到这些方法的优缺点，并结合具体的研究设计和数据特点，选择最合适的方法来确保研究的有效性和可靠性。

7 评价与推广

7.1 优点

（1） **方法多样性**

采用了五种非参数估计方法（NPMLE、BFH、加权 NPMLE、条件 NPMLE、加权 BFH）和三种假设检验方法（面积法、置换面积法、Bootstrap 面积法），展示了多种方法在处理左截断数据时的性能。

（2） **创新性**

将处理左截断数据的非参数估计方法与基于面积法的假设检验方法相结合，解决了传统的乘积极限估计和 Log-rank 检验在左截断数据存在和生存曲线交叉时效能降低的问题，为处理左截断数据提供了新的工具。

（3） **程序实现完善**

对每种方法的理论推导和公式解释较为详尽，并编写了详细的 R 语言程序，实现了非参数估计和假设检验的功能，为实际应用提供了技术支持。

（4） **实例分析**

使用了 R 语言 survival 包中的 channing 数据集进行实例分析，展示了方法在实际数据中的应用效果。并通过绘制生存曲线图，直观展示了不同方法的估计结果，便于理解和比较。

7.2 不足之处

（1） **模拟情况覆盖不广**

由于时间限制，未能进行更广泛的模拟和更深入的分析，影响了研究的全面性和深度。仅考虑了 0%和 20%的删失率，未能涵盖更多实际研究中可能出现的不同删失率情况。同时我们仅测试了（50,100）、（100,100）和（150,150）三种样本量组合，未能全面评估不同样本量对方法性能的影响。模拟次数仅为 500 次，可能不足以充分捕捉方法的稳定性和变异性。

（2） **模拟数据生成的参数设置极端**

生成模拟数据时，参数设置较为极端，导致两组数据的差异较为显著，使得模拟出来的检验效能偏高，与实际情况存在较大差距。

（3） **模拟分析比较的方法不够多**

只比较了面积法、置换面积法和 Bootstrap 面积法，未能涵盖更多的假设检验方法，如 Log-rank 检验等，这限制了方法对比的全面性。

（4） **实例分析的数据差异较小**

实例分析中使用的数据差异较小，可能未能充分展示所提方法在处理复杂数据时的优势和局限性。

7.3 改进方向

增加不同删失率（如 10%, 30%, 50%）和更多样本量组合的模拟，以全面评估方法的性能；增加模拟次数（如 1000 次或更多），以提高结果的稳定性和可靠性；调整模拟数据生成的参数，使其更加接近实际研究中的数据特征，避免参数设置过于极端；同时增加置换程序的次数，以确保置换结果的稳定性。

8 参考文献、书籍

实例数据来源于:

Hyde, J. (1980) Testing survival with incomplete observations. Biostatistics Casebook. R.G. Miller, B. Efron, B.W. Brown and L.E. Moses (editors), 31–46. John Wiley.

方法出处：

[1] 陈金宝, 侯雅文, 陈征. 左截断右删失数据下非参数估计方法的研究[J]. 中国卫生统计, 2017, 34(03):386-389+396.

[2] Leung T L ,Zhiliang Y .Estimating a Distribution Function with Truncated and Censored Data[J].The Annals of Statistics,2007,19(1):417-442.

[3] Ingo,Langner.Survival Analysis: Techniques for Censored and Truncated Data[J].Biometrics, 2006.DOI:10.1111/j.1541-0420.2006.00589_9.x.

[4] 黄兴辉.删失数据中组间疗效比较的统计推断方法研究[D].南方医科大学,2019.

[5] Huang X ,Lyu J ,Hou Y , et al.A nonparametric statistical method for two crossing survival curves[J].Communications in Statistics - Simulation and Computation,2020,51(9):1-10.

[6] 黄兴辉, 陈金宝, 杨紫荆, 等. 基于两条生存曲线间面积的非参数统计推断方法研究[J]. 中国卫生统计, 2019, 36(01):8-12.

[7] X. Lin, Q. Xu, A new method for the comparison of survival distributions, Pharmaceutical Statistics, 9 (2010) 67-76.

[8] Stefania G , Grazia V M .Multivariate permutation test to compare survival curves for matched data[J].Bmc Medical Research Methodology, 2013, 13(1):16.DOI:10.1186/1471-2288-13-16.

[9] Gasparini M , Gandini M .A comparison of nonparametric estimators of survival under left-truncation and right-censoring motivated by a case study[J].Statistica, 2011, 71(3):391-406.DOI:10.6092/issn.1973-2201/3622.

[10] 李慧敏,韩栋,陈征,等.生存曲线交叉时统计推断方法的比较和选择[J].中国卫生统计,2013,30(05):668-672+675.

[11] 卢天仪. 左截断区间删失数据的回归分析[D]. 广州大学, 2024. DOI:10.27040/d.cnki.ggzdu.2024.000947.

9 附录

附录 1：
支撑材料文件列表
R 程序.R：文章所用程序源代码； R 数据文件：含有模拟分析所用数据及其结果的文件夹； 参考文献：含有文中所用参考文献的文件夹；

附录 2：
完整 R 程序代码

```
# 第一部分：非参数估计准备函数 --------------------------------------------------------
#调用程序包
library(survival)
library(KMsurv)
library(ggplot2)
library(reshape2)
library(dplyr)
library(stats)
#设置随机种子数
set.seed(5344)
# 风险集生成函数
riskdata <- function(origin_data){
 #data 的格式为 ageentry-进入时间；age-离开时间； death-死亡 1 删失 0；
 death_times <- origin_data$age[origin_data$death == 1]# 提取死亡时间
 cen_times <- origin_data$age[origin_data$death == 0]#提取删失时间
 cen_times.sort <- sort(unique(cen_times))
 death_times.sort <- sort(unique(death_times))
 times <- c(cen_times.sort,death_times.sort)
 t1 <- sort(unique(times))# 创建时间点序列 {t}
 # 计算每个时间点的死亡人数
 death_table <- table(death_times)
 death_counts <- death_table[match(death_times.sort, names(death_table))]
 # 计算每个时间点的删失人数
 cen_table <- table(cen_times)
```

```
  cen_counts <- cen_table[match(cen_times.sort, names(cen_table))]
  # 初始化风险集人数
  risk_set <- numeric(length(t1))
  current_risk_set_size1 <- sum(origin_data$ageentry <= t1[1])
  risk_set_counts <- numeric()
  # 计算每个时间点的风险集人数
  for (i in 2:length(t1)) {
   risk_set_counts[1] <- sum(origin_data$ageentry <= t1[1])

    # 添加进入风险集的个体
   current_risk_set_size1 <- current_risk_set_size1+
    sum(origin_data$ageentry <= t1[i]&origin_data$ageentry > t1[i-1])

   # 减去离开风险集的个体（死亡或删失）
   current_risk_set_size1 <- current_risk_set_size1- sum(origin_data$age == t1[i-1])

   # 存储当前时间点的风险集人数
   risk_set_counts[i] <- current_risk_set_size1

  }
  # 创建一个数据框来存储时间点和对应人数
  death_data <- data.frame(Time = death_times.sort,DeathCount = as.vector(death_counts))
  cen_data <- data.frame(Time = cen_times.sort,cenCount = as.vector(cen_counts))
  risk_set_data <- data.frame(Time = t1, RiskSetCount = risk_set_counts)
  risk_set_data <- merge(risk_set_data, death_data, by = "Time", all.x = TRUE)
  risk_set_data <- merge(risk_set_data, cen_data, by = "Time", all.x = TRUE)
  risk_set_data$DeathCount[is.na(risk_set_data$DeathCount)] <- 0# 填充缺失值
  risk_set_data$cenCount[is.na(risk_set_data$cenCount)] <- 0
  return(risk_set_data)
}
# 非参数估计函数
## NPMLE
NPMLE <- function(data){#输入风险集数据
  risk_set_data <- data
```

```
  s_t.NPMLE <- numeric()
  c.NPMLE <- numeric()
  var.NPMLE <-numeric()

  s_t.NPMLE[1] <- 1-risk_set_data$DeathCount[1]/risk_set_data$RiskSetCount[1]
  c.NPMLE[1] <-risk_set_data$DeathCount[1]/
    (risk_set_data$RiskSetCount[1]*(risk_set_data$RiskSetCount[1]-
risk_set_data$DeathCount[1]))
  var.NPMLE[1]<-s_t.NPMLE[1]^2*(c.NPMLE[1])

  for(i in 2:length(risk_set_data$Time)){
    s_t.NPMLE[i] <- s_t.NPMLE[i-1]*(1-
risk_set_data$DeathCount[i]/risk_set_data$RiskSetCount[i])
    c.NPMLE[i]<-c.NPMLE[i-1]+risk_set_data$DeathCount[i]/
      (risk_set_data$RiskSetCount[i]*(risk_set_data$RiskSetCount[i]-
risk_set_data$DeathCount[i]))
    var.NPMLE[i]<-s_t.NPMLE[i]^2*c.NPMLE[i]
  }
  npmle <- cbind(s_t.NPMLE,var.NPMLE,c.NPMLE)
  return(npmle)
}
##  BFH
BFH <- function(data){
  risk_set_data <- data
  s_t.BFH <- numeric()
  H_t.BFH <- numeric()
  c.BFH <- numeric()
  var.BFH <-numeric()

  H_t.BFH[1] <- risk_set_data$DeathCount[1]/risk_set_data$RiskSetCount[1]
  s_t.BFH[1] <- exp(-H_t.BFH[1])
  c.BFH[1] <- (risk_set_data$DeathCount[1]/risk_set_data$RiskSetCount[1]^2)*
    (1-risk_set_data$DeathCount[1]/risk_set_data$RiskSetCount[1])
  var.BFH[1] <-s_t.BFH[1]^2*c.BFH[1]
```

```
  for(i in 2:length(risk_set_data$Time)){
   H_t.BFH[i] <- (H_t.BFH[i-1]+risk_set_data$DeathCount[i]/risk_set_data$RiskSetCount[i])
   s_t.BFH[i] <- exp(-H_t.BFH[i])
   c.BFH[i] <-c.BFH[i-1]+(risk_set_data$DeathCount[i]/risk_set_data$RiskSetCount[i]^2)*
    (1-risk_set_data$DeathCount[i]/risk_set_data$RiskSetCount[i])
   var.BFH[i] <-s_t.BFH[i]^2*c.BFH[i]
  }
  bfh<-cbind(s_t.BFH,var.BFH,c.BFH)
  return(bfh)
}
##加权 NPMLE
WNPMLE <- function(data,c=1,alpha=1/3,n){### 输入权重参数
  risk_set_data <- data
   ###权重系数
  Coef <- ceiling(c*(n^alpha))
   ###权重函数
  I <- function(R){
   test <- ifelse(R>=Coef,1,0)
   return(test)
  }
  s_t.WNPMLE <- numeric()
  c.WNPMLE <- numeric()
  var.WNPMLE <- numeric()

  s_t.WNPMLE[1] <- 1-(risk_set_data$DeathCount[1]/risk_set_data$RiskSetCount[1]*
                  I(risk_set_data$RiskSetCount[1]))
  c.WNPMLE[1] <- (risk_set_data$DeathCount[1]*I(risk_set_data$RiskSetCount[1]))/
   (risk_set_data$RiskSetCount[1]*(risk_set_data$RiskSetCount[1]-
risk_set_data$DeathCount[1]*
                                I(risk_set_data$RiskSetCount[1])))
  var.WNPMLE[1] <- s_t.WNPMLE[1]^2*c.WNPMLE[1]

  for(i in 2:length(risk_set_data$Time)){
   s_t.WNPMLE[i] <- s_t.WNPMLE[i-1]*(1-
(risk_set_data$DeathCount[i]/risk_set_data$RiskSetCount[i])*
```

```
                                I(risk_set_data$RiskSetCount[i]))
   c.WNPMLE[i]<-c.WNPMLE[i-
1]+(risk_set_data$DeathCount[i]*I(risk_set_data$RiskSetCount[i]))/
    (risk_set_data$RiskSetCount[i]*(risk_set_data$RiskSetCount[i]-
risk_set_data$DeathCount[i]*
                                I(risk_set_data$RiskSetCount[i])))
   var.WNPMLE[i]<-s_t.WNPMLE[i]^2*c.WNPMLE[i]
  }
  wnpmle<-cbind(s_t.WNPMLE,var.WNPMLE,c.WNPMLE)
  return(wnpmle)

}
##加权 BFH
WBFH<-function(data,c=1,alpha=1/3,n){### 输入权重参数
  risk_set_data <- data
   ###权重系数
  Coef <- ceiling(c*(n^alpha))
   ###权重函数
  I <- function(R){
   test <- ifelse(R>=Coef,1,0)
   return(test)
  }

  H_t.WBFH <- numeric()
  S_t.WBFH<-numeric()
  c.WBFH <- numeric()
  var.WBFH <- numeric()

  H_t.WBFH[1] <-
risk_set_data$DeathCount[1]/risk_set_data$RiskSetCount[1]*I(risk_set_data$RiskSetCount[1])
  S_t.WBFH[1]<-exp(-H_t.WBFH[1])
  c.WBFH[1] <-
((risk_set_data$DeathCount[1]*I(risk_set_data$RiskSetCount[1]))/risk_set_data$RiskSetCount[1
]^2)*(1-
```

```
(risk_set_data$DeathCount[1]/risk_set_data$RiskSetCount[1]*I(risk_set_data$RiskSetCount[1]))
)
  var.WBFH[1] <- S_t.WBFH[1]^2*c.WBFH[1]

  for(i in 2:length(risk_set_data$Time)){
    H_t.WBFH[i] <- H_t.WBFH[i-
1]+(risk_set_data$DeathCount[i]/risk_set_data$RiskSetCount[i])*I(risk_set_data$RiskSetCount[i
])
    S_t.WBFH[i]<-exp(-H_t.WBFH[i])
    c.WBFH[i] <- c.WBFH[i-
1]+((risk_set_data$DeathCount[i]*I(risk_set_data$RiskSetCount[i]))/risk_set_data$RiskSetCount
[i]^2)*(1-
(risk_set_data$DeathCount[i]/risk_set_data$RiskSetCount[i]*I(risk_set_data$RiskSetCount[i])))
    var.WBFH[i] <- S_t.WBFH[i]^2*c.WBFH[i]
  }

  wbfh<-cbind(S_t.WBFH,var.WBFH,c.WBFH)
  return(wbfh)
}

##条件 NPMLE
###条件参数
CNPMLE <- function(data,u){
  risk_set_data <- data
  s_t.CNPMLE <- numeric()#创建空向量储存结果
  c.CNPMLE<-numeric()
  var.CNPMLE<-numeric()
  if (risk_set_data$Time[1]>u) #根据条件参数判定是否满足条件
  {s_t.CNPMLE[1] <- 1-risk_set_data$DeathCount[1]/risk_set_data$RiskSetCount[1]
  c.CNPMLE[1]<-risk_set_data$DeathCount[1]/
    (risk_set_data$RiskSetCount[1]*(risk_set_data$RiskSetCount[1]-
risk_set_data$DeathCount[1]))
  var.CNPMLE[1]<-s_t.CNPMLE[1]^2*c.CNPMLE[1]
  }
  else {
```

```
   s_t.CNPMLE[1] <- 1
   c.CNPMLE[1]<-0
   var.CNPMLE[1]<-0
  }

  for(i in 2:length(risk_set_data$Time)){
   if (risk_set_data$Time[i]>u)
   {s_t.CNPMLE[i] <- s_t.CNPMLE[i-1]*(1-
risk_set_data$DeathCount[i]/risk_set_data$RiskSetCount[i])
   c.CNPMLE[i]<-c.CNPMLE[i-
1]+risk_set_data$DeathCount[i]/(risk_set_data$RiskSetCount[i]*(risk_set_data$RiskSetCount[i]-
risk_set_data$DeathCount[i]))
   var.CNPMLE[i]<-s_t.CNPMLE[i]^2*c.CNPMLE[i]
   }
   else
   {s_t.CNPMLE[i] <- 1
   c.CNPMLE[i]<-0
   var.CNPMLE[i]<-0
   }

  }
  cnpmle<-cbind(s_t.CNPMLE,var.CNPMLE,c.CNPMLE)
  return(cnpmle)
}
non_parametric_estmation <- function(data,u){
  NPMLE.result_f <- NPMLE(data)
  BFH.result_f <- BFH(data)
  WNPMLE.result_f <- WNPMLE(data,n = length(data$Time))
  WBFH.result_f <- WBFH(data,n = length(data$Time))
  CNPMLE.result_f <- CNPMLE(data,u=u)
  result <-
cbind(data,NPMLE.result_f,BFH.result_f,WNPMLE.result_f,WBFH.result_f,CNPMLE.result_f)
  return(result)
}
# 第二部分：面积法假设检验函数准备（以 WNPMLE 为例） ----------------------------
```

```
#合并两组数据时间点函数
Mergeddata <- function(data1,data2){
 t <- sort(unique(c(data1$Time,data2$Time)))
  # 构造合并后的数据框
 merged_data <- data.frame(
  t = t,
  S1 = NA,
  sigma2_S1 = NA,
  S2 = NA,
  sigma2_S2 = NA
 )
 i <- 1
  # 填充缺失值
 for (i in 1:length(t)) {
    # 填充 S1 和 sigma2_S1
  if (any(data1$Time == t[i])) {
   merged_data$S1[i] <- data1$s_t.WNPMLE [data1$Time == t[i]]
   merged_data$sigma2_S1[i] <- data1$var.WNPMLE[data1$Time == t[i]]
  } else if(all(data1$Time >= t[i])){
   merged_data$S1[i] <- 1
   merged_data$sigma2_S1[i] <- 0
  }else {
   last_index <- max(which(data1$Time < t[i]))
   if (last_index > 0) {
    merged_data$S1[i] <- data1$s_t.WNPMLE[last_index]
    merged_data$sigma2_S1[i] <- data1$var.WNPMLE[last_index]
   }
  }

  # 填充 S2 和 sigma2_S2
  if (any(data2$Time == t[i])) {
   merged_data$S2[i] <- data2$s_t.WNPMLE[data2$Time == t[i]]
   merged_data$sigma2_S2[i] <- data2$var.WNPMLE[data2$Time == t[i]]
  } else if(all(data2$Time >= t[i])){
   merged_data$S2[i] <- 1
```

```
    merged_data$sigma2_S2[i] <- 0
   }else {
    last_index <- max(which(data2$Time < t[i]))
    if (last_index > 0) {
     merged_data$S2[i] <- data2$s_t.WNPMLE[last_index]
     merged_data$sigma2_S2[i] <- data2$var.WNPMLE[last_index]
    }
   }
  }
  return(merged_data)
}
#定义计算生存曲线面积差的函数
diffarea <- function(data1,data2){
  merged_data <- Mergeddata(data1,data2)#合并两组数据时间点
  # 提取生存曲线的估计值和时间点
  t <- merged_data$t
   # 初始化面积变量
  diffarea <- 0
   # 计算每个时间点的面积并累加
  for (i in 1:(length(t) - 1)) {
   diffarea <- diffarea + abs(merged_data$S1[i] - merged_data$S2[i]) * (t[i+1]-t[i])
  }
  return(diffarea)
}
# 计算 Delta 期望函数
calculate_EhatD <- function(data){
  t <- data$t
  dt <- diff(t)# 计算时间间隔
  a <- data$sigma2_S1
  b <- data$sigma2_S2
   # 初始化期望
  Ehat_D <- 0
  for (j in 1:(length(t) - 1)) {
     # 计算平方根内的表达式
```

```
    variance_sum <- a[j] + b[j]
      # 计算加权和
    weighted_sum <- (2 / pi) * sqrt(variance_sum) * dt[j]
    # 累加结果
    Ehat_D <- Ehat_D + weighted_sum
  }
  return(Ehat_D)
}
# 计算 delta 方差函数
calculate_VarD <- function(data){
  t <- data$t
  S1 <- data$S1
  S2 <- data$S2
  sigma2_S1 <- data$sigma2_S1
  sigma2_S2 <- data$sigma2_S2
   # 初始化方差
  var_D <- 0

   # 计算自相关项
  for (j in 1:(length(t) - 1)) {
    t_j <- t[j]
    t_j1 <- t[j + 1]
    var_D <- var_D + (t_j1 - t_j)^2 * (1 - 2/pi) * (sigma2_S1[j] + sigma2_S2[j])
  }

  # 计算互相关项
  for (j in 1:(length(t) - 1)) {
    t_j <- t[j]
    t_j1 <- t[j + 1]
    for (j_prime in 1:(length(t) - 1)) {
      if (j != j_prime) {
        t_jp <- t[j_prime]
        t_jp1 <- t[j_prime + 1]
        var_D <- var_D + (t_j1 - t_j) * (t_jp1 - t_jp) * (1 - 2/pi) *
```

```
        sqrt((sigma2_S1[j] + sigma2_S2[j]) * (sigma2_S1[j_prime] + sigma2_S2[j_prime]))
    }
   }
  }

  # 返回结果
 return(var_D)
}
# 两组统计量计算函数
compare_twogroups <- function(origindata){
 origindata1 <- origindata[origindata$group==0,]
 origindata2 <- origindata[origindata$group==1,]

 risk_data1 <- riskdata(origindata1)
 risk_data2 <- riskdata(origindata2)

 data1 <- non_parametric_estmation(risk_data1,u=1)
 data2 <- non_parametric_estmation(risk_data2,u=1)



 merged_data <- Mergeddata(data1,data2)#合并两组数据时间点
 all_diff <- diffarea(data1,data2)
 Ehat_D <- calculate_EhatD(merged_data)
 Varhat_D <- calculate_VarD(merged_data)
 Dstar <- (all_diff - Ehat_D) / (Varhat_D^(1/2))
 return(Dstar)
}
# 第三部分：优化的面积法程序 ------------------------------------------------------
----------------

# 置换面积法
# 输入原始数据和面积法得到的原始统计量
permutation_method <- function(origindata,original_statistic){
 channingdata_f2 <- origindata[origindata$group==0,]#提取性别（0 男 1 女）
```

```
channingdata_m2 <- origindata[origindata$group==1,]
library(dplyr)
comb_df <-  bind_rows(channingdata_f2, channingdata_m2)
set.seed(123)
n <- 500
perm_stats <- numeric(n)
for(k in 1:n){
 #permuted <- sample(comb_df)
 qw <- length(channingdata_f2$time)

 #sampled_rows <- comb_df[sample(nrow(comb_df), size = 300), ]

 total_rows <- nrow(comb_df)
 sampled_indices <- sample(total_rows, size = qw)
 perm_group1 <- comb_df[sampled_indices, ]
 perm_group2 <- comb_df[-sampled_indices, ]



   #非参数估计主程序
   ##置换数据生成
risk_data1 <- riskdata(perm_group1)#女性风险集数据
risk_data2 <- riskdata(perm_group2)#男性风险集数据
risk_data <- riskdata(origindata)#总体风险集数据


   #最终结果,,,,置换
estmation_data1 <- non_parametric_estmation(risk_data1,u=1)
estmation_data2 <- non_parametric_estmation(risk_data2,u=1)

   #面积法假设检验（以 WNPMLE 为例）
 #合并数据时间点函数置换

merged_data <- Mergeddata(estmation_data1,estmation_data2)
```

```
    # 计算置换生存曲线面积差
    all_diff <- diffarea(estmation_data1,estmation_data2)

      # 计算统计量和对应的 p 值
      # 计算 Delta 期望函数和方差函数
    Ehat_D <- calculate_EhatD(merged_data)
    Varhat_D <- calculate_VarD(merged_data)
    #Dstar <- (all_diff - Ehat_D) / (Varhat_D^(1/2))
    perm_stats[k]<- (all_diff - Ehat_D) / (Varhat_D^(1/2))

  }
  PMT_P_value <- mean(abs(perm_stats) >= abs(original_statistic))
  #result <- data.frame(perm_stats,PMT_P_value)
  return(PMT_P_value)
}
## bootstrap 程序
Boost_method <- function(origindata,original_statistic){
  n_bootstrap <- 500  # 自助抽样次数
  boot_stats <- numeric(n_bootstrap)
  boot_stats_results <- numeric(n_bootstrap) # 初始化自助抽样统计量
  channingdata_f2 <- origindata[origindata$group==1,]
  channingdata_m2 <- origindata[origindata$group==0,]
  Ehat_D  <- numeric()
  Varhat_D <- numeric()
  all_diff <- numeric()
  for (k in 1:n_bootstrap) {
      # 获取样本数量
    num_female_samples <- nrow(channingdata_f2)
    num_male_samples <- nrow(channingdata_m2)
      # 自助抽样：回到原假设，从总体样本中抽取两组的样本，允许重复
    bootstrap_group_female <- origindata[sample(nrow(origindata), size = num_female_samples,
replace = TRUE), ]
    bootstrap_group_male <- origindata[sample(nrow(origindata), size = num_male_samples,
replace = TRUE), ]
```

```
  # 非参数估计主程序
  risk.data_f <- riskdata(bootstrap_group_female) # 使用 bootstrap_group_female
  risk.data_m <- riskdata(bootstrap_group_male)  # 使用 bootstrap_group_male
  result_f <- non_parametric_estmation(risk.data_f,u=781)
  result_m <- non_parametric_estmation(risk.data_m,u=781)
   # 合并数据时间点
  merged_data <- Mergeddata(result_f, result_m)

    # 计算生存曲线面积

  all_diff[k] <- diffarea(result_f,result_m)

    # 计算统计量和对应的 p 值
  Ehat_D[k] <- calculate_EhatD(merged_data)
  Varhat_D[k] <- calculate_VarD(merged_data)



  boot_stats_results[k] <- (all_diff[k] - Ehat_D[k]) / sqrt(Varhat_D[k])

 }
  # 移除 NA 值
 #valid_boot_stats <- na.omit(boot_stats_results)
 bootstrap_P_value <- mean(abs(boot_stats_results) >= abs(original_statistic))
 return(bootstrap_P_value)
}

# 第四部分：模拟数据生成 ----------------------------------------------------
n_sim <- 500 #设置重复次数
## Ⅰ型数据
#两条生存曲线左截断数据和生存数据均分别符合参数为 0.25 和 0.15 的指数分布
class1_data<- function(n1,n2,n_sim,w){
 rate1 <- 0.25
```

```
data_list_1 <- list()

for(i in 1:n_sim){

 id1 <- c(1:n1)
 id2 <- c(1:n2)

 count1 <- 0
 count2 <- 0
 turn_data1 <- data.frame()
 turn_data2 <- data.frame()

   #生成固定样本量的左截断数据
 while(count1<n1){
  l <- rexp(n1,rate = rate1)
  x <- rexp(n1,rate = rate1-0.1)
  sup_data <- data.frame(l[l<x],x[l<x])
  names(sup_data) <- c("L1","X1")
  turn_data1 <- bind_rows(turn_data1,sup_data)
  count1 <- nrow(turn_data1)
 }
 data1 <- cbind(data.frame(id1),turn_data1[sample(nrow(turn_data1),n1),])

 while(count2<n2){
  l <- rexp(n2,rate = rate1)
  x <- rexp(n2,rate = rate1-0.1)
  sup_data <- data.frame(l[l<x],x[l<x])
  names(sup_data) <- c("L2","X2")
  turn_data2 <- bind_rows(turn_data2,sup_data)
  count2 <- nrow(turn_data2)
 }
 data2 <- cbind(data.frame(id2),turn_data2[sample(nrow(turn_data2),n2),])


   #设置研究时间窗口长度 w，这里我们假设 w 是一个固定的值或者从某个分布中抽取
```

```
    #这里我们从均匀分布里抽取
  wrep1 <- runif(n1,0,w)
  wrep2 <- runif(n2,0,w)

  L1 <- data1$L1
  X1 <- data1$X1
  L2 <- data2$L2
  X2 <- data2$X2
   #计算删失时间 C
  C1 <- L1+ wrep1
  C2 <- L2+ wrep2

  censored1 <- X1 < C1
  censored2 <- X2 < C2
  count_TRUE1 <- sum(censored1)
  count_TRUE2 <- sum(censored2)
  censored_rate1 <- count_TRUE1/n1
  censored_rate2 <- count_TRUE2/n2

  ageentry <- trunc(c(L1,L2)*100)
  age <- trunc(c(X1,X2)*100)
  death <- 1-c(censored1,censored2)
  time <- age-ageentry
  obs <- c(1:(n1+n2))
  group <- c(rep(0,n1),rep(1,n2))
  censored_rate <- c(rep(censored_rate1,n1),rep(censored_rate2,n2))
  wtime <- c(rep(w,n1),rep(w,n2))

  data_list_1[[i]] <- data.frame(obs,death,ageentry,age,time,group,censored_rate,wtime)#将每次
模拟的结果存储在列表中

 }
 return(data_list_1)
}
##Ⅱ型数据
```

```
#风险率成比例，两个左截断生存数据来自不同参数的指数分布
#一个为 0.25 和 0.15，另一个为 0.5 和 0.3
#为了两个生存数据保持相近的删失率，w2 可能要小于 w1
class2_data<- function(n1,n2,n_sim,w1,w2){
 rate2_1 <- 0.25;rate2_2 <- 0.5

 data_list_2 <- list()

 for(i in 1:n_sim){

  id1 <- c(1:n1)
  id2 <- c(1:n2)

  count1 <- 0
  count2 <- 0
  turn_data1 <- data.frame()
  turn_data2 <- data.frame()

    #生成固定样本量的左截断数据
  while(count1<n1){
   l <- rexp(n1,rate = rate2_1)
   x <- rexp(n1,rate = rate2_1-0.1)
   sup_data <- data.frame(l[l<x],x[l<x])
   names(sup_data) <- c("L1","X1")
   turn_data1 <- bind_rows(turn_data1,sup_data)
   count1 <- nrow(turn_data1)
  }
  data1 <- cbind(data.frame(id1),turn_data1[sample(nrow(turn_data1),n1),])

  while(count2<n2){
   l <- rexp(n2,rate = rate2_2)
   x <- rexp(n2,rate = rate2_2-0.2)
   sup_data <- data.frame(l[l<x],x[l<x])
   names(sup_data) <- c("L2","X2")
   turn_data2 <- bind_rows(turn_data2,sup_data)
```

```
  count2 <- nrow(turn_data2)
}
data2 <- cbind(data.frame(id2),turn_data2[sample(nrow(turn_data2),n2),])



 #设置研究时间窗口长度 w，这里我们假设 w 是一个固定的值或者从某个分布中抽取
   #这里我们从随机分布
wrep1 <- runif(n1,0,w1)
wrep2 <- runif(n2,0,w2)

L1 <- data1$L1
X1 <- data1$X1
L2 <- data2$L2
X2 <- data2$X2
 #计算删失时间 C
C1 <- L1+ wrep1
C2 <- L2+ wrep2

censored1 <- X1 < C1
censored2 <- X2 < C2
count_TRUE1 <- sum(censored1)
count_TRUE2 <- sum(censored2)
censored_rate1 <- count_TRUE1/n1
censored_rate2 <- count_TRUE2/n2

ageentry <- trunc(c(L1,L2)*100)
age <- trunc(c(X1,X2)*100)
death <- 1-c(censored1,censored2)
time <- age-ageentry
obs <- c(1:(n1+n2))
group <- c(rep(0,n1),rep(1,n2))
censored_rate <- c(rep(censored_rate1,n1),rep(censored_rate2,n2))
wtime <- c(rep(w1,n1),rep(w2,n2))
```

```
  data_list_2[[i]] <- data.frame(obs,death,ageentry,age,time,group,censored_rate,wtime)#将每次模拟的结果存储在列表中


 }
 return(data_list_2)
}
##III型数据
#本类型数据为前中期存在差异，所以整条生存曲线包含四段不同参数分布的曲线
#分别为 s1=1.2l(<0.8)+0.1l(0.8~1.8)+0.5l(1.8~2.6)+l(>2.6)
#s2=0.5(<0.8)+0.1l(0.8~1.8)+1.2l(1.8~2.6)+l(>2.6)
#因为参数太多，每段曲线都单独设计 w 决定删失不太现实，所以就同一用统一的数值较小的窗口值
class3_data<- function(n1,n2,n_sim,w1,w2){
 rate3_1_1 <- 1.2;rate3_1_2 <- 0.1;rate3_1_3 <- 0.5;rate3_1_4 <- 1
 rate3_2_1 <- 0.5;rate3_2_2 <- 0.1;rate3_2_3 <- 1.2;rate3_2_4 <- 1
 t3_1 <- 0.8;t3_2 <- 1.8;t3_3 <- 2.6

 data_list_3 <- list()

 for(i in 1:n_sim){

  id1 <- c(1:n1)
  id2 <- c(1:n2)

  count1 <- 0
  count2 <- 0
  turn_data1 <- data.frame()
  turn_data2 <- data.frame()

  #生成固定样本量的左截断数据
  while(count1<n1){
   l1 <- rexp(n1,rate = rate3_1_1)
   x1 <- rexp(n1,rate = rate3_1_1-0.3)
   t3_1x <- qexp(pexp(t3_1, rate = rate3_1_1),rate = rate3_1_1-0.3)
```

```
  time1 <- data.frame(l1,x1)
  time1 <- subset(time1,x1<=t3_1x&l1<x1)
  names(time1) <- c("l","x")

  l2 <- rexp(n1,rate = rate3_1_2)
  x2 <- rexp(n1,rate = rate3_1_2-0.04)
  t3_2x <- qexp(pexp(t3_2, rate = rate3_1_2),rate = rate3_1_2-0.04)
  time2 <- data.frame(l2,x2)
  time2 <- subset(time2,x2>t3_1x&x2<=t3_2x&l2<x2)
  names(time2) <- c("l","x")

  l3 <- rexp(n1,rate = rate3_1_3)
  x3 <- rexp(n1,rate = rate3_1_3-0.15)
  t3_3x <- qexp(pexp(t3_3, rate = rate3_1_3),rate = rate3_1_3-0.15)
  time3 <- data.frame(l3,x3)
  time3 <- subset(time3,x3>t3_2x&x3<=t3_3x&l3<x3)
  names(time3) <- c("l","x")

  l4 <- rexp(n1,rate = rate3_1_4)
  x4 <- rexp(n1,rate = rate3_1_4-0.2)
  time4 <- data.frame(l4,x4)
  time4 <- subset(time4,x4>t3_3x&l4<x4)
  names(time4) <- c("l","x")

  timeall1 <- rbind(time1,time2,time3,time4)
  sup_data <- subset(timeall1,l<x)
  names(sup_data) <- c("L1","X1")
  turn_data1 <- bind_rows(turn_data1,sup_data)
  count1 <- nrow(turn_data1)
}
data1 <- cbind(data.frame(id1),turn_data1[sample(nrow(turn_data1),n1),])

while(count2<n2){
  l1 <- rexp(n2,rate = rate3_2_1)
  x1 <- rexp(n2,rate = rate3_2_1-0.15)
```

```
  t3_1x <- qexp(pexp(t3_1, rate = rate3_2_1),rate = rate3_2_1-0.15)
  time1 <- data.frame(l1,x1)
  time1 <- subset(time1,x1<=t3_1x&l1<x1)
  names(time1) <- c("l","x")

  l2 <- rexp(n2,rate = rate3_2_2)
  x2 <- rexp(n2,rate = rate3_2_2-0.04)
  t3_2x <- qexp(pexp(t3_2, rate = rate3_2_2),rate = rate3_2_2-0.04)
  time2 <- data.frame(l2,x2)
  time2 <- subset(time2,x2>t3_1x&x2<=t3_2x&l2<x2)
  names(time2) <- c("l","x")

  l3 <- rexp(n2,rate = rate3_2_3)
  x3 <- rexp(n2,rate = rate3_2_3-0.3)
  t3_3x <- qexp(pexp(t3_3, rate = rate3_2_3),rate = rate3_2_3-0.3)
  time3 <- data.frame(l3,x3)
  time3 <- subset(time3,x3>t3_2x&x3<=t3_3x&l3<x3)
  names(time3) <- c("l","x")

  l4 <- rexp(n2,rate = rate3_2_4)
  x4 <- rexp(n2,rate = rate3_2_4-0.2)
  time4 <- data.frame(l4,x4)
  time4 <- subset(time4,x4>t3_3x&l4<x4)
  names(time4) <- c("l","x")

  timeall2 <- rbind(time1,time2,time3,time4)
  sup_data <- subset(timeall2,l<x)
  names(sup_data) <- c("L2","X2")
  turn_data2 <- bind_rows(turn_data2,sup_data)
  count2 <- nrow(turn_data2)
 }
data2 <- cbind(data.frame(id2),turn_data2[sample(nrow(turn_data2),n2),])


#设置研究时间窗口长度 w，这里我们假设 w 是一个固定的值或者从某个分布中抽取
```

```
  wrep1 <- rep(w1,n1)
  wrep2 <- rep(w2,n2)

  L1 <- data1$L1
  X1 <- data1$X1
  L2 <- data2$L2
  X2 <- data2$X2
  #计算删失时间 C
  C1 <- L1+ wrep1
  C2 <- L2+ wrep2

  censored1 <- X1 < C1
  censored2 <- X2 < C2
  count_TRUE1 <- sum(censored1)
  count_TRUE2 <- sum(censored2)
  censored_rate1 <- count_TRUE1/n1
  censored_rate2 <- count_TRUE2/n2



  ageentry <- trunc(c(L1,L2)*100)
  age <- trunc(c(X1,X2)*100)
  death <- 1-c(censored1,censored2)
  time <- age-ageentry
  obs <- c(1:(n1+n2))
  group <- c(rep(0,n1),rep(1,n2))
  censored_rate <- c(rep(censored_rate1,n1),rep(censored_rate2,n2))
  wtime <- c(rep(w1,n1),rep(w2,n2))

  data_list_3[[i]] <- data.frame(obs,death,ageentry,age,time,group,censored_rate,wtime)#将每次
模拟的结果存储在列表中

 }
 return(data_list_3)
}
##Ⅳ型数据
```

```
#后期差异,生存曲线分两段分布
#分别为 s1=l(<0.8)+2.5(>0.8)
#s2=l(<0.8)+0.5l(>0.8)
class4_data<- function(n1,n2,n_sim,w1,w2){
 rate4_1_1 <- 1;rate4_1_2 <- 2.5
 rate4_2_1 <- 1;rate4_2_2 <- 0.5
 t4 <- 0.8

 data_list_4 <- list()

 for(i in 1:n_sim){

  id1 <- c(1:n1)
  id2 <- c(1:n2)

  count1 <- 0
  count2 <- 0
  turn_data1 <- data.frame()
  turn_data2 <- data.frame()

  #生成固定样本量的左截断数据
  while(count1<n1){
   l1 <- rexp(n1,rate = rate4_1_1)
   x1 <- rexp(n1,rate = rate4_1_1-0.25)
   t4x <- qexp(pexp(t4, rate = rate4_1_1),rate = rate4_1_1-0.25)
   time1 <- data.frame(l1,x1)
   time1 <- subset(time1,x1<=t4x&l1<x1)
   names(time1) <- c("l","x")

   l2 <- rexp(n1,rate = rate4_1_2)
   x2 <- rexp(n1,rate = rate4_1_2-0.625)
   time2 <- data.frame(l2,x2)
   time2 <- subset(time2,x1>t4x&l2<x2)
   names(time2) <- c("l","x")
```

```
  timeall1 <- rbind(time1,time2)
  sup_data <- subset(timeall1,l<x)
  names(sup_data) <- c("L1","X1")
  turn_data1 <- bind_rows(turn_data1,sup_data)
  count1 <- nrow(turn_data1)
 }
data1 <- cbind(data.frame(id1),turn_data1[sample(nrow(turn_data1),n1),])

while(count2<n2){
  l1 <- rexp(n2,rate = rate4_2_1)
  x1 <- rexp(n2,rate = rate4_2_1-0.25)
  t4x <- qexp(pexp(t4, rate = rate4_2_1),rate = rate4_2_1-0.25)
  time1 <- data.frame(l1,x1)
  time1 <- subset(time1,x1<=t4x&l1<x1)
  names(time1) <- c("l","x")

  l2 <- rexp(n2,rate = rate4_2_2)
  x2 <- rexp(n2,rate = rate4_2_2-0.125)
  time2 <- data.frame(l2,x2)
  time2 <- subset(time2,x1>t4x&l2<x2)
  names(time2) <- c("l","x")

  timeall2 <- rbind(time1,time2)
  sup_data <- subset(timeall2,l<x)
  names(sup_data) <- c("L2","X2")
  turn_data2 <- bind_rows(turn_data2,sup_data)
  count2 <- nrow(turn_data2)
 }
data2 <- cbind(data.frame(id2),turn_data2[sample(nrow(turn_data2),n2),])


#设置研究时间窗口长度 w，这里我们假设 w 是一个固定的值或者从某个分布中抽取
wrep1 <- runif(n1,0,w1)
wrep2 <- runif(n2,0,w2)
```

```
  L1 <- data1$L1
  X1 <- data1$X1
  L2 <- data2$L2
  X2 <- data2$X2
  #计算删失时间 C
  C1 <- L1+ wrep1
  C2 <- L2+ wrep2

  censored1 <- X1 < C1
  censored2 <- X2 < C2
  count_TRUE1 <- sum(censored1)
  count_TRUE2 <- sum(censored2)
  censored_rate1 <- count_TRUE1/n1
  censored_rate2 <- count_TRUE2/n2



  ageentry <- trunc(c(L1,L2)*100)
  age <- trunc(c(X1,X2)*100)
  death <- 1-c(censored1,censored2)
  time <- age-ageentry
  obs <- c(1:(n1+n2))
  group <- c(rep(0,n1),rep(1,n2))
  censored_rate <- c(rep(censored_rate1,n1),rep(censored_rate2,n2))
  wtime <- c(rep(w1,n1),rep(w2,n2))

  data_list_4[[i]] <- data.frame(obs,death,ageentry,age,time,group,censored_rate,wtime)#将每次
模拟的结果存储在列表中


 }
 return(data_list_4)
}
##V 型数据
#中期交叉数据
#分别为 s1=1/12l
```

```
#s2=0.25l(<=2)+1/35l(>2)
class5_data<- function(n1,n2,n_sim,w1,w2){
 rate5_1 <- 1/12
 rate5_2_1 <- 0.25;rate5_2_2 <- 1/35
 t5 <- 2

 data_list_5 <- list()

 for(i in 1:n_sim){

  id1 <- c(1:n1)
  id2 <- c(1:n2)

  count1 <- 0
  count2 <- 0
  turn_data1 <- data.frame()
  turn_data2 <- data.frame()

  #生成固定样本量的左截断数据
  while(count1<n1){
   l <- rexp(n1,rate = rate5_1)
   x <- rexp(n1,rate = rate5_1-0.02)
   sup_data <- data.frame(l[l<x],x[l<x])
   names(sup_data) <- c("L1","X1")
   turn_data1 <- bind_rows(turn_data1,sup_data)
   count1 <- nrow(turn_data1)
  }
  data1 <- cbind(data.frame(id1),turn_data1[sample(nrow(turn_data1),n1),])

  while(count2<n2){
   l1 <- rexp(n2,rate = rate5_2_1)
   x1 <- rexp(n2,rate = rate5_2_1-0.15)
   t5x <- qexp(pexp(t5, rate = rate5_2_1),rate = rate5_2_1-0.15)
   time1 <- data.frame(l1,x1)
   time1 <- subset(time1,x1<=t5x&l1<x1)
```

```
  names(time1) <- c("l","x")

  l2 <- rexp(n2,rate = rate5_2_2)
  x2 <- rexp(n2,rate = rate5_2_2-0.015)
  time2 <- data.frame(l2,x2)
  time2 <- subset(time2,x2>t5x&l2<x2)
  names(time2) <- c("l","x")

  timeall2 <- rbind(time1,time2)
  sup_data <- subset(timeall2,l<x)
  names(sup_data) <- c("L2","X2")
  turn_data2 <- bind_rows(turn_data2,sup_data)
  count2 <- nrow(turn_data2)
}
data2 <- cbind(data.frame(id2),turn_data2[sample(nrow(turn_data2),n2),])


  #设置研究时间窗口长度 w，这里我们假设 w 是一个固定的值或者从某个分布中抽取
wrep1 <- rep(w1,n1)
wrep2 <- rep(w2,n2)

L1 <- data1$L1
X1 <- data1$X1
L2 <- data2$L2
X2 <- data2$X2
#计算删失时间 C
C1 <- L1+ wrep1
C2 <- L2+ wrep2

censored1 <- X1 < C1
censored2 <- X2 < C2
count_TRUE1 <- sum(censored1)
count_TRUE2 <- sum(censored2)
censored_rate1 <- count_TRUE1/n1
censored_rate2 <- count_TRUE2/n2
```

```
  ageentry <- trunc(c(L1,L2)*100)
  age <- trunc(c(X1,X2)*100)
  death <- 1-c(censored1,censored2)
  time <- age-ageentry
  obs <- c(1:(n1+n2))
  group <- c(rep(0,n1),rep(1,n2))
  censored_rate <- c(rep(censored_rate1,n1),rep(censored_rate2,n2))
  wtime <- c(rep(w1,n1),rep(w2,n2))

  data_list_5[[i]] <- data.frame(obs,death,ageentry,age,time,group,censored_rate,wtime)#将每次
模拟的结果存储在列表中

 }
 return(data_list_5)
}
##VI型数据
#后期交叉数据
#分别为 weibull 分布 s1=w(1.5,5)
#s2=0.15l(<=2.2)+0.1l(>2.2)
class6_data<- function(n1,n2,n_sim,w1,w2){
 shape <- 1.5;scale <- 5
 rate6_2_1 <- 0.5;rate6_2_2 <- 0.1
 t6 <- 2.2

 data_list_6 <- list()

 for(i in 1:n_sim){

  id1 <- c(1:n1)
  id2 <- c(1:n2)

  count1 <- 0
  count2 <- 0
```

```
turn_data1 <- data.frame()
turn_data2 <- data.frame()

#生成固定样本量的左截断数据
while(count1<n1){
 l <- rweibull(n1,shape = shape,scale = scale)
 x <- rweibull(n1,shape = shape,scale = scale)
 sup_data <- data.frame(l[l<x],x[l<x])
 names(sup_data) <- c("L1","X1")
 turn_data1 <- bind_rows(turn_data1,sup_data)
 count1 <- nrow(turn_data1)
}
data1 <- cbind(data.frame(id1),turn_data1[sample(nrow(turn_data1),n1),])

while(count2<n2){
 l1 <- rexp(n2,rate = rate6_2_1)
 x1 <- rexp(n2,rate = rate6_2_1)
 t6x <- qexp(pexp(t6, rate = rate6_2_1),rate = rate6_2_1)
 time1 <- data.frame(l1,x1)
 time1 <- subset(time1,x1<=t6x&l1<x1)
 names(time1) <- c("l","x")

 l2 <- rexp(n2,rate = rate6_2_2)
 x2 <- rexp(n2,rate = rate6_2_2)
 time2 <- data.frame(l2,x2)
 time2 <- subset(time2,x2>t6x&l2<x2)
 names(time2) <- c("l","x")

 timeall2 <- rbind(time1,time2)
 sup_data <- subset(timeall2,l<x)
 names(sup_data) <- c("L2","X2")
 turn_data2 <- bind_rows(turn_data2,sup_data)
 count2 <- nrow(turn_data2)
}
data2 <- cbind(data.frame(id2),turn_data2[sample(nrow(turn_data2),n2),])
```

```
  #设置研究时间窗口长度 w，这里我们假设 w 是一个固定的值或者从某个分布中抽取
  wrep1 <- rep(w1,n1)
  wrep2 <- rep(w2,n2)

  L1 <- data1$L1
  X1 <- data1$X1
  L2 <- data2$L2
  X2 <- data2$X2
  #计算删失时间 C
  C1 <- L1+ wrep1
  C2 <- L2+ wrep2

  censored1 <- X1 < C1
  censored2 <- X2 < C2
  count_TRUE1 <- sum(censored1)
  count_TRUE2 <- sum(censored2)
  censored_rate1 <- count_TRUE1/n1
  censored_rate2 <- count_TRUE2/n2


  ageentry <- trunc(c(L1,L2)*100)
  age <- trunc(c(X1,X2)*100)
  death <- 1-c(censored1,censored2)
  time <- age-ageentry
  obs <- c(1:(n1+n2))
  group <- c(rep(0,n1),rep(1,n2))
  censored_rate <- c(rep(censored_rate1,n1),rep(censored_rate2,n2))
  wtime <- c(rep(w1,n1),rep(w2,n2))

  data_list_6[[i]] <- data.frame(obs,death,ageentry,age,time,group,censored_rate,wtime)#将每次
模拟的结果存储在列表中

 }
```

```
 return(data_list_6)
}

##我们分别设计了均衡设计和非均衡设计
#分别为样本量为（50，100），（100，100），（150，150）
#在每一种参数下同时考虑了删失率不同的情况，分别为 0%和 20%
data_list_1_1 <- class1_data(50,100,500,0)
data_list_1_2 <- class1_data(100,100,500,0)
data_list_1_3 <- class1_data(150,150,500,0)
data_list_1_4 <- class1_data(50,100,500,3.5)
data_list_1_5 <- class1_data(100,100,500,3.2)
data_list_1_6 <- class1_data(150,150,500,3.5)

data_list_2_1 <- class2_data(50,100,500,0,0)
data_list_2_2 <- class2_data(100,100,500,0,0)
data_list_2_3 <- class2_data(150,150,500,0,0)
data_list_2_4 <- class2_data(50,100,500,2,2)
data_list_2_5 <- class2_data(100,100,500,2.1,2.1)
data_list_2_6 <- class2_data(150,150,500,2.1,2.1)

data_list_3_1 <- class3_data(50,100,500,0,0)
data_list_3_2 <- class3_data(100,100,500,0,0)
data_list_3_3 <- class3_data(150,150,500,0,0)
data_list_3_4 <- class3_data(50,100,500,0.23,0.24)
data_list_3_5 <- class3_data(100,100,500,0.22,0.22)
data_list_3_6 <- class3_data(150,150,500,0.22,0.22)

data_list_4_1 <- class4_data(50,100,500,0,0)
data_list_4_2 <- class4_data(100,100,500,0,0)
data_list_4_3 <- class4_data(150,150,500,0,0)
data_list_4_4 <- class4_data(50,100,500,0.25,0.45)
data_list_4_5 <- class4_data(100,100,500,0.24,0.45)
data_list_4_6 <- class4_data(150,150,500,0.24,0.45)

data_list_5_1 <- class5_data(50,100,500,0,0)
```

```
data_list_5_2 <- class5_data(100,100,500,0,0)
data_list_5_3 <- class5_data(150,150,500,0,0)
data_list_5_4 <- class5_data(50,100,500,4,3.5)
data_list_5_5 <- class5_data(100,100,500,4,2.87)
data_list_5_6 <- class5_data(150,150,500,4,3)

data_list_6_1 <- class6_data(50,100,500,0,0)
data_list_6_2 <- class6_data(100,100,500,0,0)
data_list_6_3 <- class6_data(150,150,500,0,0)
data_list_6_4 <- class6_data(50,100,500,3,1.2)
data_list_6_5 <- class6_data(100,100,500,2,1.4)
data_list_6_6 <- class6_data(150,150,500,2,1.2)
data.all <-
list(data_list_1_1,data_list_1_2,data_list_1_3,data_list_1_4,data_list_1_5,data_list_1_6,
            data_list_2_1,data_list_2_2,data_list_2_3,data_list_2_4,data_list_2_5,data_list_2_6,
            data_list_3_1,data_list_3_2,data_list_3_3,data_list_3_4,data_list_3_5,data_list_3_6,
            data_list_4_1,data_list_4_2,data_list_4_3,data_list_4_4,data_list_4_5,data_list_4_6,
            data_list_5_1,data_list_5_2,data_list_5_3,data_list_5_4,data_list_5_5,data_list_5_6,
            data_list_6_1,data_list_6_2,data_list_6_3,data_list_6_4,data_list_6_5,data_list_6_6)
##各类型理论图
# class1
S1 <- function(x){
 S1 <- exp(-0.25 * (x))
 S2 <- exp(-0.25 * (x))
 s <- data.frame(
   S1,
   S2
 )
 return(s)
}
t1= seq(0,15,0.1)
S11 <- S1(t1)$S1
S12 <- S1(t1)$S1
plot(t1,S11,type = "l",main = "I",xlab = "Time",ylab ="Survival function",ylim = c(0, 1))
lines(t1,S12,lty=2)
```

```
# class2
S2 <- function(x){
  S1 <- exp(-0.25 * (x))
  S2 <- exp(-0.5 * (x))
  s <- data.frame(
    S1,
    S2
  )
  return(s)
}
t2 = seq(0,5,0.1)
S21<- S2(t2)$S1
S22<- S2(t2)$S2
plot(t2,S21,type = "l",main = "II",xlab = "Time",ylab ="Survival function",ylim = c(0, 1))
lines(t2,S22,lty=2)
# class3
S3 <- function(x){
  if(x<=0.8){
    S1=exp(-1.2*x)
    S2=exp(-0.5*x)
  }else if(x<=1.8){
    S1=exp(-1.2*0.8)*exp(-0.1*(x-0.8))
    S2=exp(-0.5*0.8)*exp(-0.1*(x-0.8))
  }else if(x<=2.6){
    S1=exp(-1.2*0.8)*exp(-0.1*1)*exp(-0.5*(x-1.8))
    S2=exp(-0.5*0.8)*exp(-0.1*1)*exp(-1.2*(x-1.8))
  }else{
    S1=exp(-1.2*0.8)*exp(-0.1*1)*exp(-0.5*(0.8))*exp(-1*(x-2.6))
    S2=exp(-0.5*0.8)*exp(-0.1*1)*exp(-1.2*(0.8))*exp(-1*(x-2.6))
  }
  s <- data.frame(
    S1,
    S2
  )
  return(s)
```

```
}
t3 <- seq(0,4,0.1)
s31 <- numeric()
s32 <- numeric()
for(i in 1:length(t3)){
  s31[i] <- S3(t3[i])$S1
  s32[i] <- S3(t3[i])$S2
}
plot(t3,s31,type = "l",main = "III",xlab = "Time",ylab ="Survival function",ylim = c(0, 1) )
lines(t3,s32,lty = 2)
# class4
S4 <- function(x){
  if(x<=0.8){
    S1=exp(-1*x)
    S2=exp(-1*x)
  }else{
    S1=exp(-1*0.8)*exp(-2.5*(x-0.8))
    #S2=exp(-0.5*0.8)-(1-exp(-0.3*(x-0.8)))
    S2=exp(-1*0.8)*exp(-0.5*(x-0.8))
  }
  s <- data.frame(
    S1,
    S2
  )
  return(s)
}
t4 <- seq(0,4,0.1)
s41 <- numeric()
s42 <- numeric()
for(i in 1:length(t4)){
  s41[i] <- S4(t4[i])$S1
  s42[i] <- S4(t4[i])$S2
}
plot(t4,s41,type = "l",main = "IV",xlab = "Time",ylab ="Survival function",ylim = c(0, 1) )
lines(t4,s42,lty = 2)
```

```
# class5
S5 <- function(x){
 if(x<=2){
  S1=exp(-(1/12)*x)
  S2=exp(-0.25*x)
 }else{
  S1=exp(-(1/12)*x)
  S2=exp(-0.25*2)*exp(-(1/35)*(x-2))
 }
 s <- data.frame(
  S1,
  S2
 )
 return(s)
}
t5 <- seq(0,30,0.1)
s51 <- numeric()
s52 <- numeric()
for(i in 1:length(t5)){
 s51[i] <- S5(t5[i])$S1
 s52[i] <- S5(t5[i])$S2
}
plot(t5,s51,type = "l",main = "V",xlab = "Time",ylab ="Survival function",ylim = c(0, 1))
lines(t5,s52,lty = 2)
# class6
S6 <- function(x){
 if(x<=1.5){
  S1=exp(-0.1*x^1.5)
  S2=exp(-0.5*x)
 }else{
  S1=exp(-0.1*x^1.5)
  S2=exp(-0.5*1.5)*exp(-0.1*(x-1.5))
 }
 s <- data.frame(
  S1,
```

```
    S2
  )
  return(s)
}
t6 <- seq(0,15,0.1)
s61 <- numeric()
s62 <- numeric()
for(i in 1:length(t6)){
  s61[i] <- S6(t6[i])$S1
  s62[i] <- S6(t6[i])$S2
}
plot(t6,s61,type = "l",main = "VI",xlab = "Time",ylab ="Survival function",ylim = c(0, 1))
lines(t6,s62,lty = 2)
##绘画模拟数据图
for(i in 1:6){
  for(j in 1:6){
    q <- sample(c(1:500),1)
    data <- data.all[[i]][[j]][[q]]
    origindata.h3 <- data
    origindata1 <- origindata.h3[origindata.h3$group==0,]
    origindata2 <- origindata.h3[origindata.h3$group==1,]

    risk_data1 <- riskdata(origindata1)
    risk_data2 <- riskdata(origindata2)

    data1 <- non_parametric_estmation(risk_data1,u=1)
    data2 <- non_parametric_estmation(risk_data2,u=1)

    plot(data1$Time,data1$s_t.WNPMLE,type = "l",ylim = c(0,1))
    lines(data2$Time,data2$s_t.WNPMLE)
  }
}
##正态性检验
for(i in 1:6){
  for(j in 1:6){
```

```
  data <- data.all[[i]][[j]]
  origindata <- data.frame()
  Dstar <- numeric()
  original_statistic <- numeric()
  origin_p_value <- numeric()
  PMT_P_value <- numeric()
  for(i in 1:500){#模拟一百次
   origindata <- data[[i]]
   Dstar[i] <- compare_twogroups(origindata)
   original_statistic[i] <- Dstar[i]
   origin_p_value[i] <- 1 - pnorm(Dstar[i])
   #PMT_P_value[i] <- permutation_method(origindata,original_statistic[i])
   cat("模拟完成了",i,"次  ",origin_p_value[i],"\n")
  }
  er <- c(original_statistic)
  er
  data_test_N <- er
  breaks <- seq(-2,7,by=0.05)
  hist(data_test_N,breaks=breaks,xlab="",ylab="",main="1 面积 0 删失",probability
=1,col="lightblue",axes=FALSE)
  data <- density(data_test_N)
  lines(data$x,data$y)

  N_test_result <- shapiro.test(data_test_N)
  N_test_result




  origindata <- data.all[[i]][[j]][[1]]
  Dstar <- compare_twogroups(origindata)
  original_statistic <- Dstar

  er <- permutation_method_N_t(origindata,original_statistic)
  er
```

```
  data_test_N <- c(er)
  breaks <- seq(-2,4,by=0.05)
  hist(data_test_N,breaks=breaks,xlab="",ylab="",main="4 置换 0 删失",probability
=1,col="lightblue",axes=FALSE)
  data <- density(data_test_N)
  lines(data$x,data$y)

  N_test_result <- shapiro.test(data_test_N)
  N_test_result
 }
}
# 第五部分：模拟分析 ----------------------------------------------------------------
## 模拟分析程序
simulation <- function(data,n){#输入数据和模拟次数
 origindata <- data.frame()
 Dstar <- numeric()
 original_statistic <- numeric()
 origin_p_value <- numeric()
 PMT_P_value <- numeric()
 for(i in 1:n){#模拟一百次
  origindata <- data[[i]]
  Dstar[i] <- compare_twogroups(origindata)
  original_statistic[i] <- Dstar[i]
  origin_p_value[i] <- 1 - pnorm(Dstar[i])
  PMT_P_value[i] <- permutation_method(origindata,original_statistic[i])
 }
 result<- cbind(original_statistic,
                origin_p_value,
                PMT_P_value)
 return(result)
}
result.all <- list(list())
for(i in 1:6){
 for(j in 1:6){
  data <- data.all[[i]][[j]]
```

```
    result.all[[i]][[j]] <- simulation(data,500)
  }
}
#模拟
# 模拟数据整合
load("data_all.RData")#加载原始数据
load("result_all.RData")#加载模拟结果
#求解一类错误
typeI_errorr <- matrix(nrow=6,ncol=2)
for(i in 1:2){
  for(j in 1:6){

    typeI_errorr[j,i] <- mean(result.all[["class1"]][[j]][,i+1]<=0.05)
  }
}
#求解 Power
class2Power <- matrix(nrow=6,ncol=2)
class3Power <- matrix(nrow=6,ncol=2)
class4Power <- matrix(nrow=6,ncol=2)
class5Power <- matrix(nrow=6,ncol=2)
class6Power <- matrix(nrow=6,ncol=2)
#result.all[["class1"]][[1]][,2]
for(i in 1:2){
  for(j in 1:6){

    class2Power[j,i] <- mean(result.all[["class2"]][[j]][,i+1]<=0.05)
    class3Power[j,i] <- mean(result.all[["class3"]][[j]][,i+1]<=0.05)
    class4Power[j,i] <- mean(result.all[["class4"]][[j]][,i+1]<=0.05)
    class5Power[j,i] <- round(mean(result.all[["class5"]][[j]][,i+1]<=0.05),3)
    class6Power[j,i] <- mean(result.all[["class6"]][[j]][,i+1]<=0.05)
  }
}
typeI_error <- data.frame(
  "面积法"=typeI_errorr[,1],
```

```
  "置换面积法"=typeI_errorr[,2]
)
Power <- data.frame(
  "class2"=class2Power,
  "class3"=class3Power,
  "class4"=class4Power,
  "class5"=class5Power,
  "class6"=class6Power
)
print(typeI_error)
print(Power)

##aalen 加性风险函数
install.packages("timereg")
library(timereg)
aalen_model_test <- function(listall){
  aalenP_calss <- numeric()
  for(k in 1:36){
    aalenP <- numeric()
    for(i in 1:500){
      data11 <-listall[[k]][[i]]
      #data11$ageentry
      # 创建左截断的生存对象，格式为 Surv(entry_time, time, status)
      surv_obj <- Surv(time =data11$ageentry, time2 =data11$age, event = data11$death)
      # 使用 Aalen 加性风险模型
      aalen_model <- aalen(surv_obj ~ group, data = data11)
      #summary_aareg_model <- summary(aalen_model)

      aalenP[i] <- aalen_model[["pval.testBeq0"]][["group"]]
      cat("模拟完成了",i,"次  ",aalenP[i],"\n")
    }

    aalenP_calss[k]<- mean(aalenP<=0.05)
    cat("模拟完成了",k,"个数据框  ",aalenP_calss[k],"\n")
```

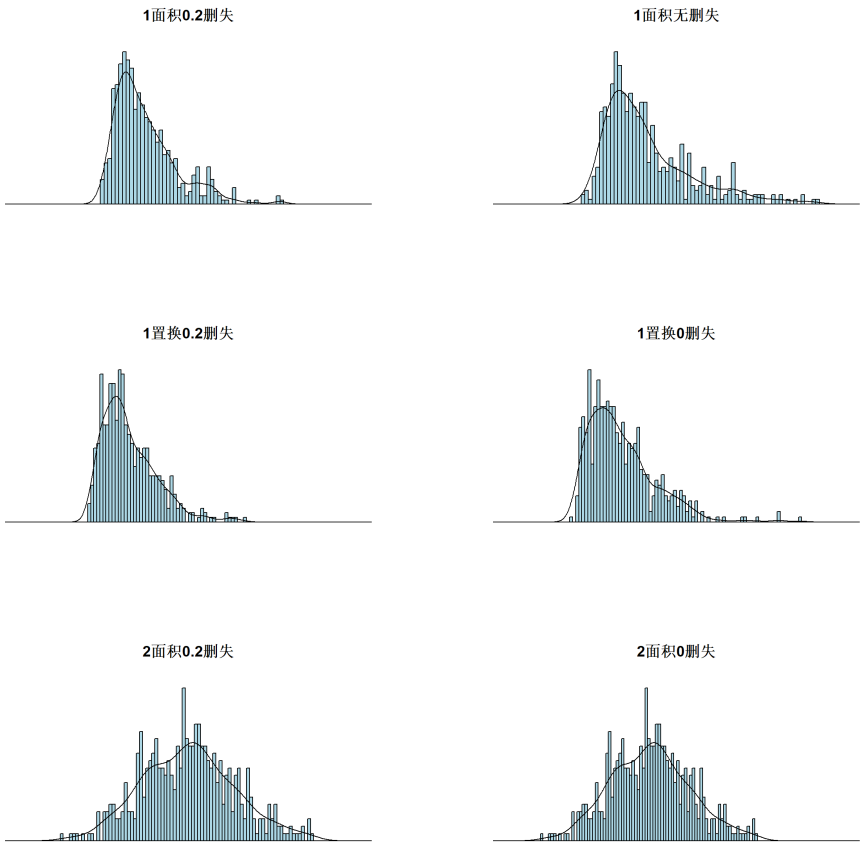
```
 }
 return(aalenP_calss)
}
aalen_p_use <- aalen_model_test(listall)
aalen_model_P <- c(aalenP_calss)

# 第六部分：实例分析 ------------------------------------------------------------
#加载数据 数据预处理
data(channing)
channing$gender <- ifelse(channing$gender == 1, 0, 1)#将性别列改为 0,1
channingdata <- channing[channing$ageentry < channing$age, ]#提取满足生存时间大于截断时
间的数据
channingdata_f <- channingdata[channingdata$gender==1, ]#提取性别（0 男 1 女）
channingdata_m <- channingdata[channingdata$gender==0, ]
#非参数估计主程序
##数据生成
risk.data_f <- riskdata(channingdata_f)#女性风险集数据
risk.data_m <- riskdata(channingdata_m)#男性风险集数据
risk.data <- riskdata(channingdata)#总体风险集数据
#最终结果
result_f <- non_parametric_estmation(risk.data_f,u=781)
result_m <- non_parametric_estmation(risk.data_m,u=781)
#绘制累积生存曲线图
Cumulative_Survival_Rate_Graph<-function(data_1,data_2)
{
 #绘制女性生存曲线图
 df1_1<-as.data.frame(data_1)
 colnames(df1_1) <-
c("Time" ,"RiskSetCount","DeathCount","cenCount","NPMLE" ,"var.NPMLE
","c.NPMLE","BFH" ,"var.BFH","c.BFH",
```

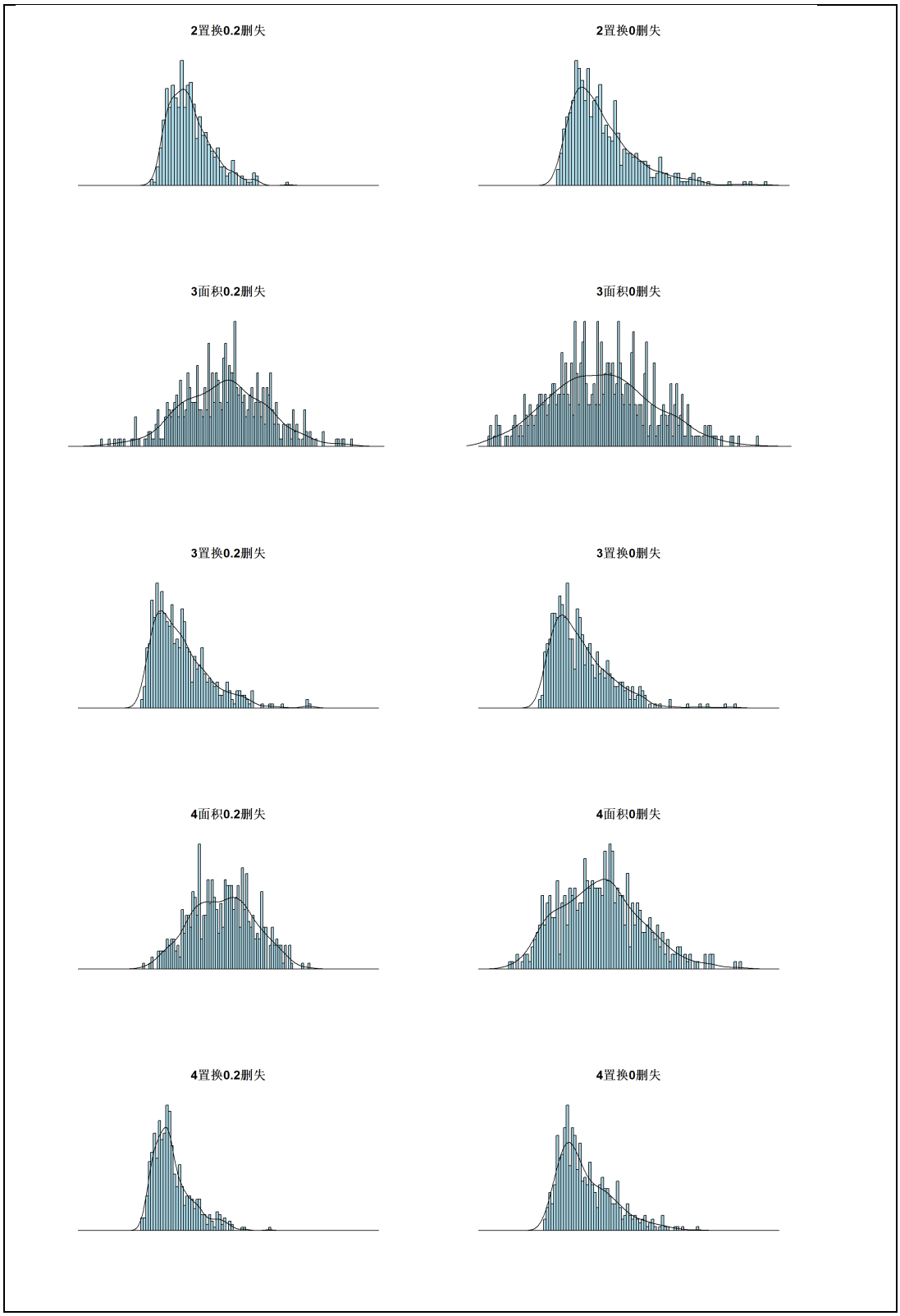
```
"WNPMLE","var.WNPMLE" ,"c.WNPMLE" ,"WBFH" ,"var.WBFH" ,"c.WBFH" ,"CNPMLE" ,
"var.CNPMLE","c.CNPMLE")
 df2_1 <- melt(df1_1, id.vars =
"Time",measure.vars=c("NPMLE","BFH","WNPMLE","WBFH","CNPMLE"), variable.name =
"group", value.name = "y")

 gg1<-ggplot(df2_1, aes(x = Time, y = y, color = group)) +
   geom_line() +
   labs(title = "女性累计生存率图", x = "年龄（月）", y = "累积生存率", color = "非参数估计
方法") +
   #geom_text_repel(aes(label=labers),family = 'STHeiti')+
   theme_minimal()


 #绘制男性生存曲线图
 df1_2<-as.data.frame(data_2)
 colnames(df1_2) <-
c("Time" ,"RiskSetCount","DeathCount","cenCount","NPMLE" ,"var.NPMLE
","c.NPMLE","BFH" ,"var.BFH","c.BFH",

"WNPMLE","var.WNPMLE" ,"c.WNPMLE" ,"WBFH" ,"var.WBFH" ,"c.WBFH" ,"CNPMLE" ,
"var.CNPMLE","c.CNPMLE")
 df2_2 <- melt(df1_2, id.vars =
"Time",measure.vars=c("NPMLE","BFH","WNPMLE","WBFH","CNPMLE"), variable.name =
"group", value.name = "y")

 gg2<-ggplot(df2_2, aes(x = Time, y = y, color = group)) +
   geom_line() +
   labs(title = "男性累计生存率图", x = "年龄（月）", y = "累积生存率", color = "非参数估计
方法") +
   #geom_text_repel(aes(label=labers),family = 'STHeiti')+
   theme_minimal()
 print(gg1)
 print(gg2)
```

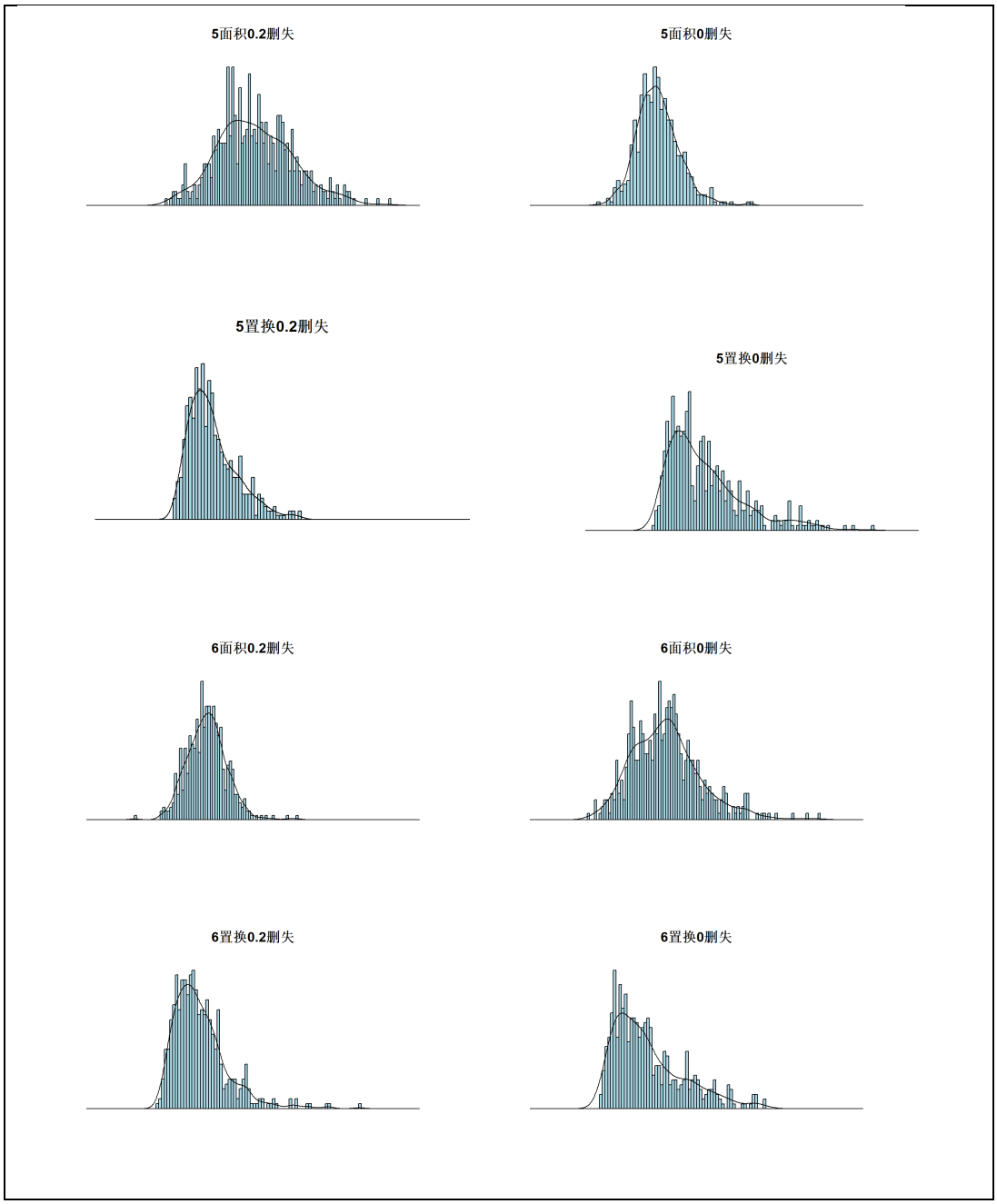
```
}
#输出实例生存曲线图
Cumulative_Survival_Rate_Graph(result_f,result_m)
#channing 数据假设检验
channingdata <- rename(channingdata, group = group)
D_star.channing <- compare_twogroups(channingdata)
origin_p_value.channing  <- 1 - pnorm(D_star.channing)
PMT_P_value.channing <- permutation_method(channingdata,D_star.channing)
Bootstrap_P_value <- Boost_method(channingdata,D_star.channing)
cat("面积法统计量为",D_star.channing,"P 值为",origin_p_value.channing ,"置换面积法 p 值为
",PMT_P_value.channing,"Bootstrap 法 p 值为",Bootstrap_P_value)
```

附录 3：

各种情况下的统计量分布图




2置换0.2删失
2置换0删失
3面积0.2删失
3面积0删失
3置换0.2删失
3置换0删失
4面积0.2删失
4面积0删失
4置换0.2删失
4置换0删失


5面积0.2删失
5面积0删失
5置换0.2删失
5置换0删失
6面积0.2删失
6面积0删失
6置换0.2删失
6置换0删失