第十三章 多重检验

本章介绍经典的 p 值假设检验。假设检验通常设置零假设，并试图证伪零假设以获得我们想要证明的目的 (就像数学中的反证法)。在现代统计学中，我们可能希望同时检验大量的零假设，并且小心地避免错误地拒绝过多的零假设

13.1 快速回顾假设检验

假设检验提供了一个严格的框架以回答统计学中“是或否”的问题

13.1.1 检验一个假设

实施假设检验共有 4 步：

1. 定义零假设和备择假设
2. 构建检验统计量
3. 计算 p 值
4. 基于 p 值，决定是否拒绝零假设

第一步, 定义零假设和备择假设
假设检验将可能的情况分为两类，即**原假设 (null hypothesis)** 和 **备择假设 (alternative hypothesis)**。原假设记为 H_0，它一般是我们希望证伪的事实陈述，虽然它很可能是真的；备择假设记为 H_a，代表我们希望证明的事实，通常与原假设相反。一个经典的零假设可以是：

$$\text{There is no difference between the expected blood pressure of mice in the control and treatment grou}$$

H_0 与 H_a 的处理并不对称。若我们的数据支持否定 H_0，则支持 H_a，且认为 H_0 不成立；若不支持拒绝 H_0，我们就不清楚是否是样本量太小还是 H_0 确实成立

第二步，构造检验统计量
检验统计量 (test statistic) 提供了反对原假设的证据，符号为 T，表示我们的数据与 H_0 的一致程度。以上面的小鼠实验为例，记 $x_1^t, \dots, x_{n_t}^t$ 表示 n_t 个实验组的样本，$x_1^c, \dots, x_{n_c}^c$ 表示 n_c 个对照组的样本，且 $\mu_t = E(X^t), \mu_c = E(X^c)$ 分别表示两组均值，为了检验原假设 $H_0 : \mu_t = \mu_c$，我们需要构造**两个样本下的 t 统计量 (two-sample t-statistic)**：

$$T = \frac{\hat{\mu}_t - \hat{\mu}_c}{s\sqrt{\frac{1}{n_t} + \frac{1}{n_c}}}$$

其中：

$$\hat{\mu}_t = \frac{1}{n_t}\sum_{i=1}^{n_t} x_i^t, \quad \hat{\mu}_c = \frac{1}{n_c}\sum_{i=1}^{n_c} x_i^c$$

且：

$$s = \sqrt{\frac{(n_t - 1)s_t^2 + (n_c - 1)s_c^2}{n_t + n_c - 2}}$$

它表示两个样本合并标准差的估计。这里 s_t^2 和 s_c^2 是两样本方差的无偏估计。一个较大的 T 值提供了更强的拒绝 H_0 的证据

第三步，计算 p 值
在上一步中，我们发现双样本 t 统计量的绝对数值反映了拒绝 H_0 的强度。那么，多大才算大？**p 值 (p-value)** 定义为 H_0 为真的情况下，观察到等于或更极端的检验统计量的概率，因此较小的 p 值提供了反对 H_0 的证据

具体来说，我们设上一步的统计量值为 $T = 2.33$，在 H_0 成立条件下，T 服从标准正态分布，则观察到不小于 T 的统计量的概率为 $1 - 0.98 = 2\%$。因此，我们得到的 p 值就是 0.02

在 H_0 假设下的统计量分布就是**零分布 (null distribution)**，常用的检验统计量在满足足够多样本和其他一些假设下，都是遵循已知分布的 (如正态分布、χ^2 分布等)

p 值可能是统计学中最常用和滥用的概念。请注意，p 值不是 H_0 成立的概率，其唯一解释是：重复多次实验，预期看到如此极端检验统计量的比例。它衡量的是数据在 H_0 条件下成立的可能性，而不是 H_0 本身成立的可能性

第四步，决定是否拒绝原假设
一旦计算出 p 值，我们就可以决定是否拒绝原假设。事实上，拒绝原假设的阈值取决于观测者，因为 p 值实际上反映的是若 H_0 成立，我们看到如此小的 p 值的概率。若设定阈值不足以支持我们拒绝原假设，我们只好认为原假设成立

13.1.2 两类错误

若原假设真实成立，则它是一个**真原假设 (true null hypothesis)**；否则称为**假原假设 (false null hypothesis)**。我们无法先验地知道原假设是否成立，因此需要假设检验

执行假设检验后，我们的行为如下表所示：

Decision	**Truth** (H_0)	**Truth** (H_a)
Reject (H_0)	**Type I Error**	Correct
Do Not Reject (H_0)	Correct	**Type II Error**

显然，当我们在 H_0 为假时拒绝 H_0，或在 H_0 为真时不拒接 H_0，则我们得到了正确结果；然而，若我们在 H_0 正确时错误地拒绝了 H_0，则我们犯了**第一类错误 (Type I error)**。**第一类错误率 (Type I error rate)** 定义为当 H_0 成立时，错误拒绝 H_0 的概率。类似地，若我们在 H_0 错误时接受了 H_0，则称为**第二类错误 (Type II error)**。假设检验的**功效 (power)** 定义为在 H_a 成立的条件下，正确拒绝 H_0 的概率

我们通常不能同时降低两类错误率。实践中，常认为第一类错误比第二类错误严重，因为前者通常涉及不正确的科学发现。所以我们希望较低的第一类错误率，如 $\alpha = 0.05$，同时假设检验

的功效较大。可以证明，p 值和第一类错误率之间存在对应关系。通过在 p 值小于 α 时拒绝 H_0 就可以确保第一类错误率 $\leq \alpha$

13.2 多重检验面临的问题

当我们希望进行**多重检验 (multiple testing)**，我们很容易在大量的实验下错误地拒绝部分原假设，即犯第一类错误。例如，对于 m 个原假设 $H_{01}, \ldots, H_{0m}$，我们设定 $\alpha = 0.01$，则预期会拒绝 $\alpha \cdot m$ 个假设检验，当 m 很大时将会是大量的第一类错误。后续的章节将会讨论如何降低这种风险

13.3 族系错误率

族系错误率 (falmily-wise error rate, FWER) 定义为对 m 个原假设 $H_{01}, \ldots H_{0m}$ 进行假设检验时，至少发生一次第一类错误的概率

	H_0 **is True**	H_0 **is False**	**Total**
Reject H_0	V	S	R
Do Not Reject H_0	U	W	$m - R$
Total	m_0	$m - m_0$	m

根据上表，族系错误率即为：

$$\text{FWER} = \Pr(V \geq 1)$$

拒绝 p 值低于 α 的任何原假设的策略（即在 α 级别控制每个原假设的 I 类误差）会导致 FWER 为：

$$\begin{aligned}
\text{FWER}(\alpha) &= 1 - \Pr(V = 0) \\
&= 1 - \Pr(\text{do not falsely reject any null hypotheses}) \\
&= 1 - \Pr\left(\bigcap_{j=1}^{m} \{\text{do not falsely reject } H_{0j}\}\right)
\end{aligned}$$

当我们作一个极强的假设，即 m 个检验都是独立的，且 m 个原假设都正确时，有：

$$\text{FWER}(\alpha) = 1 - \prod_{j=1}^{m}(1 - \alpha) = 1 - (1 - \alpha)^m$$

根据上式，当我们只检验一个原假设时，族系错误率等同于第一类错误率；若执行 $m = 100$ 个独立测试，则族系错误率为 $1 - (1 - \alpha)^{100}$，当 $\alpha = 0.05$ 时，这个错误率将达到 0.994，几乎相当于必然出现一个第一类错误。因此，将族系错误率控制在 α 水平，比将第一类错误率控制在 α 水平的要求要高得多

13.3.2 控制族系错误率的方法

Bonferroni 方法

设 A_j 表示事件：我们对第 j 个原假设犯第一类错误，其中 $j = 1, \ldots, m$，则：

$$\begin{aligned}\text{FWER} &= \Pr(\text{falsely reject at least one null hypothesis}) \\ &= \Pr(\bigcup_{j=1}^{m} A_j) \\ &\leq \sum_{j=1}^{m} \Pr(A_j).\end{aligned}$$

在上式中，我们基于不等式 $P(A \cup B) \leq P(A) + P(B)$ 得出不论 A 与 B 是否独立，**Bonferroni 方法 (Bonferronni method)** 或**Bonferroni 校验 (Bonferronni correciton)** 将拒绝每个假设检验的阈值设置为 α/m，因此 $P(A_j) \leq \alpha/m$，也即：

$$\text{FWER}(\alpha/m) \leq m \times \frac{\alpha}{m} = \alpha$$

这也就相当于把每个零假设的阈值除以总数 m 来避免族系错误

Bonferroni 方法让我们不会错误地拒绝过多原假设，但是它相当保守，使我们拒绝了很少的原假设，提升了犯第二类错误的概率。Bonferroni 校正是迄今为止所有统计中最著名和最常用的多重性校正。它的普遍性在很大程度上是由于它非常易于理解和实施，还因为它成功地控制了 I 类错误，而不管 m 假设检验是否独立

> 根据不等式，我们得到的 FWER 比目标 FWER 往往低很多，这表明 Bonferroni 方法过于保守

Holm 逐步递减法

Holm 逐步递减法 (Holm's step-down procedure) 比 Bonferroni 方法稍微复杂，在这个方法中，我们否定每个原假设的阈值取决于所有 m 个 p 值的值，与 Bonferroni 方法要求每个原假设的阈值都低于 α/m 不同。Holm 方法也不对 m 个原假设作独立假设，因此它的功效始终更大

Holm 方法的步骤如下：

1. 给定 FWER 的 α
2. 对 m 个原假设 $H_{01}, \ldots, H_{0m}$，计算对应 p 值 $p_1, \ldots, p_m$
3. 按 p 值对原假设排序
4. 定义

$$L = \min \left\{ j : p_{(j)} > \frac{\alpha}{m + 1 - j} \right\}$$

5. 拒绝首个 $p_j < p_{(L)}$ 及其之后的原假设，并接受之前的原假设

Tukey 方法

前两类方法不对零假设和检验统计量分布作任何假设。在某些特定条件下，我们可使用更适合任务的方法来控制 FWER，**Tukey 方法 (Tukey's method)** 是其中之一

以 5 个基金经理的表现为例，在观察五个基金经理的表现后，我们发现 1 号和二号的均值差异最大，因此可能希望进行假设检验 $H_0 : \mu_1 \neq \mu_2$，且得到了一个较小的 $p = 0.0349$。然而，这并非是我们一开始就执行的假设检验，而是在比较了 5 个样本之后执行的，实际上相当于进行了 $5 \times (5-1)/2 = 10$ 次的假设检验。因此 FWER 需要控制到 0.05，则基于 Bonferroni 方法每对原假设的 $\alpha = 0.005$，我们反而不应该拒绝原假设

不过，Bonferroni 校验在这里有点严格，因为没有考虑 $m = 10$ 对原假设的关系：实际上 m 个成对比较的 m 个 p 值不是独立的，因此我们的校验可以不那么严格。Tukey 方法允许我们对 $m = G(G-1)/2$ 对均值比较时，在 α 水平控制 FWER，同时拒绝 $p < \alpha_T$，这里 α_T 稍微 $> \alpha/m$

> 原书未提及如何计算 α_T，但代码已封装好

Scheffé 方法
假设我们希望对五个基金经理进行检验：

$$H_0 : \frac{1}{2}(\mu_1 + \mu_3) = \frac{1}{3}(\mu_2 + \mu_4 + \mu_5)$$

这个检验可用双样本 t 检验进行，也能得到一个很小的 $p = 0.004$。然而，这实际上也是我们查看了 5 个观测值得到的，据 Bonferroni 校正需要更小的 p 值才能拒绝原假设

Scheffé 方法 (Scheffé's method) 让我们通过计算几个 α_S 值来拒绝上面的原假设，同时确保族错误率低于 α。一旦计算了 α_S 值，我们就可以进行类似于：

$$H_0 : \frac{1}{3}(\mu_1 + \mu_2 + \mu_3) = \frac{1}{2}(\mu_4 + \mu_5)$$

这样的假设检验而不需要调整 α_S 值

> 这里没有提到怎么计算 α_S

Bonferroni 方法和 Holm 方法适用于任何情况的多重假设检验。但是，在一些特定条件下，使用 Scheffé' 方法或者 Tukey 方法可以在保证降低第一类错误率的同时提高假设检验的功效

13.3 FWER 和功效的权衡

回顾 FWER 和功效的定义：

- FWER 表示我们在多重检验中犯至少一次第一类错误的概率
- 功效表示在原假设为假时，成功拒绝原假设的概率，即拒绝零假设的数量除以假零假设的总数
  这里 FWER 保证了我们不发现谬误，而功效则体现我们假设检验获得新发现的作用

随着原假设数量 m 的增大，为了控制 FWER 在一个合适的范围 α 内，我们不得不降低每个假设检验的阈值，这将会导致功效的迅速降低。实践中，当 m 很大时，我们可能愿意容忍假阳性以获得更多的发现，这就是错误发现率背后的动机

13.4 错误发现率

13.4.1 错误发现率背后的直觉

	H_0 is True	H_0 is False	Total
Reject H_0	V	S	R
Do Not Reject H_0	U	W	$m-R$
Total	m_0	$m-m_0$	m

根据上表，我们定义假阳性 V 和总阳性 $V+S=R$ 的比率 V/R 为**错误发现比例 (false discovery proportion, FDP)**。当 m 很大时，我们试图确保错误发现率足够低，以便大多数被拒绝的原假设不是假阳性

实践中，控制 FDP 很吸引人，但是几乎不可能完成。因为我们不能知道数据集上哪些假设是错误的，哪些是真实的，即使能控制 FWER 保证 $P(V\geq 1)\leq \alpha$，但是也不能保证 $V=0$，除非不拒绝任何零假设

因此，我们转向控制 **错误发现率 (false discovery rate, FDR)**。它定义为：

$$\mathrm{FDR}=\mathrm{E}(\mathrm{FDP})=\mathrm{E}(V/R).$$

也就是错误发现比例 FDP 的期望。例如，当设定 FDR 为 20% 时，我们会尽可能多的拒绝原假设，同时保证假阳性的概率不超过 20%

> 期望表示多次重复试验下得到的假阳性率。一次特定实验下，这个值可能超过 FDR

与 p 值不同，p 值通常将低于阈值 (如 0.05) 的结果视为得出阳性结果的最低证据标准，而 FDR 不存在一个广泛接受的阈值，它的阈值选择取决于上下文甚至数据集，例如愿意付出调查的经费数量

13.4.2 Benjamini–Hochberg 程序

Benjamini–Hochberg 程序 (Benjamini–Hochberg Procedure) 是一种用于控制错误发现率 FDR 的方法，可将 m 个 p 值联系到一个设置的好的 FDR 值 q 上。它的步骤如下：

1. 确定 FDR 值 q
2. 计算 m 个原假设的 p 值 $p_1,\dots,p_m$
3. 按从小到大的顺序排序，即令 $p_{(1)}\leq\cdots\leq p_{(m)}$
4. 定义：

$$L=\max\{j:p_{(j)}<qj/m\}$$

5. 拒绝所有 满足 $P_j\leq p_{(L)}$ 的 H_{0j} 原假设
   简单来说，就是排序后找到一个最大的序号 j，使得 $p_{(j)}\leq \frac{j}{m}q$，拒绝所有前面的原假设，同时后续的所有原假设都不拒绝以控制第一错误率

当 m 个 p 值独立或轻度依赖时， Benjamini–Hochberg 程序就能保证：

$$FDR \leq q$$

换言之，这个过程可以保证平均来说被拒绝的原假设不超过 q 的部分为假阳性，且无论多少个原假设为真，也无论原假设的 p 值是如何分布的

> Benjamini-Hochberg 程序类似于 Holm 方法，拒绝的阈值不是确定的，而是依赖全部数据的 p 值得出的函数，无法预先知道如 $p = 0.01$ 的原假设是否应被拒绝，这取决于其他的原假设 p 值情况

13.5 p 值和错误发现率的重采样方法

之前的讨论基于用某些统计量来检验特定零假设 H_0，这些统计量在 H_0 下具有特定分布，如正态分布、t 分布、χ^2 分布等，这被称为**理论零分布 (theoretical null distribution)**，我们用这些分布计算 p 值。对于绝大多数零假设，只要对数据作出严格假设，我们就可以使用理论零分布计算 p 值。有时候，我们的原假设 H_0 和检验统计量 T 不再寻常，或者很难建立合适假设 (如小样本)，我们则不能使用理论零分布

本章讨论利用计算机的快速计算近似 T 的零分布，我们需要通过对特定问题实例化来实现这个方法。本章考虑的示例是用双样本 t 检验检验两个随机变量的均值是否相等

13.5.1 关于 p 值的重采样方法

考虑零假设 $H_0 : E(X) = E(Y)$，备择假设 $H_a : E(X) \neq E(Y)$，记来自 X 和 Y 的样本数 n_X 和 n_Y，则双样本 t 检验统计量：

$$T = \frac{\hat{\mu}_X - \hat{\mu}_Y}{s\sqrt{\frac{1}{n_X} + \frac{1}{n_Y}}}$$

其中 $\hat{\mu}_X = \frac{1}{n_X}\sum_{i=1}^{n_X} x_i$，$\hat{\mu}_Y = \frac{1}{n_Y}\sum_{i=1}^{n_Y} y_i$，$\quad s = \sqrt{\frac{(n_X-1)s_X^2 + (n_Y-1)s_Y^2}{n_X+n_Y-2}}$，且 s_X^2 和 s_Y^2 分别是两组样本的方差的无偏估计。显然，一个较大的 T 值提供了拒绝 H_0 的证据

当 n_X 和 n_Y 都很大时，上述 T 近似服从 $N(0,1)$ 分布；但是当二者很小时，在未知 X 和 Y 的分布情况下，我们就无法得到 T 的理论零分布。此时，我们需要用**重采样 (re-sampling)** 或者具体称为**排列 (permutation)** 的方法进行估计

我们先进行思想实验。设 H_0 成立，且 X 与 Y 同分布，则我们随机交换 X 和 Y 中的观测值，则交换统计值后的检验统计量 T 与原始统计量 T 应当具有相同分布，即仅当 H_0 成立且 X 和 Y 同分布时成立

上述想法提供了一个近似 T 的零分布的方案。取出所有的 $n_X + n_Y$ 的观测样本，并进行 B 次重新排列，这里 B 是一个较大的数，每次排列后重新分配观测样本 X' 和 Y'，并计算 $T^{*1}, \ldots, T^{*B}$ 表示新统计量 T。回顾 p 值是在 H_0 下至少在此极端值观测到检验统计量的概率，因此：

$$p\text{-value} = \frac{\sum_{b=1}^{B} l_{(|T^{*b}| \geq |T|)}}{B}$$

这就是检验统计量的值至少和原始数据的检验统计量一样极端的值的比例。其中 l 是一个指示函数，当排列 T 不小于原始 T 时值为 1

通过实验可以发现。当样本数据分布较为偏态或数据量较小的情况下 (此时理论零分布准确性较低)，重抽样得到的 p 与理论 p 之间差异明显；否则，重抽样 p 与理论 p 之间差异很小

> 建议在数据偏态或小样本下使用重抽样方法。其他时候，优势并不明显

13.5.2 错误发现率的重采样方法

假设我们希望控制 FDR，对于 m 个零假设 $H_{01},\dots,H_{0m}$，为了避免使用理论零分布，考虑使用 Benjamini–Hochberg 程序计算 m 个检验统计量 $T_1,\dots,T_m$，然后获得临界的 $p_{(L)}$ 值并拒绝对应原假设。事实上，可用重采样方法避免计算 p 值

由 FDR 定义 $\mathrm{FDR}=E(V/R)$，则作近似估计：

$$\mathrm{FDR}=E\left(\frac{V}{R}\right)\approx\frac{E(V)}{R}$$

假设我们希望拒绝任何检验统计量超过 c 的任何原假设，然后计算：

$$R=\sum_{j=1}^{m}l_{(|T_j|\geq c)}$$

这里 R 是所有拒绝的原假设数量。然而，我们很难计算 V，因为我们不知道哪些被拒绝的 H_0 是假阳性。实际上我们可使用重采样方法模拟 $H_{01},\dots H_{0m}$ 下的数据，然后计算得到的检验统计量 T，以超过 c 的重采样检验统计量 T^* 的数量提供 V 的估计值

记 $x_1^{(j)},\dots,x_{n_X}^{(j)}$ 和 $y_1^{(j)},\dots,y_{n_Y}^{(j)}$ 表示和第 j 个零假设有关的统计样本，我们按照排列的方法再次随机排布 n_X+n_Y 个样本，并重新分配到两组样本中以计算双样本统计量 T_i，并以超过设定统计量大小 c 的 T_i 数量为 $E(V)$，即错误拒绝 H_0 的数量。重复排列 一个大数 B 次，我们就能得到相应的 $E(V)$ 估计

计算步骤：

1. 选择一个阈值 c，其中 $c>0$
2. 对于 $j=1,\cdots,m$：
   - 计算 $T^{(j)}$，即第 j 个原假设 H_{0j} 的原始检验统计量
   - 从 $b=1,\dots,B$，其中 B 是一个大数
     - 计算 n_X+n_Y 对观测值的随机排列，重新分配观测值到两个样本中
     - 对新的两对样本，计算 $T^{(j),*b}$
3. 计算 $R=\sum_{j=1}^{m}l_{(|T^{(j)}\geq c)}$
4. 计算 $\hat{V}=\frac{\sum_{b=1}^{B}\sum_{j=1}^{m}1_{\left(|T^{(j),*b}|\geq c\right)}}{B}$
5. 估计 FDR，使用 $\hat{V}/R$

事实上，如果定义 p 值：

$$p_j = \frac{\sum_{j'=1}^{m}\sum_{b=1}^{B} 1_{(|T_{j'}^{*b}| \geq |T_j|)}}{Bm}$$

其中 $j = 1, \ldots, m$，以代替之前定义的 p 值：

$$p-\text{value} = \frac{\sum_{b=1}^{B} 1_{(|T^{*b}| \geq |T|)}}{B}$$

并使用 Benjamini-Hochberg 程序计算这些重采样的 p 值，实际上与本节介绍的重采样方法等价

> 第一个计算方法一次性考虑了 m 对假设检验的重采样结果，适合多重检验；第二个计算方法适合单对的假设检验

13.5.3 什么时候应用重采样

下面两种情况，使用重采样比较有用：

1. 也许无理论零分布可用。例如，检验异常的原假设 H_0 或使用不常规的检验统计量 T
2. 存在理论零分布时，其有效性假设不成立。例如，仅当观测值正态分布时，双样本检验统计量 t 才服从 $t_{n_X+n_Y-2}$ 分布，且仅 n_X 和 n_Y 很大时，才服从 $N(0,1)$ 分布。当假设不成立时，理论 p 无效

此外，如果还有其他有效的方法重新采样或者重新排列数据，都可以用以重采样变体以计算估计 p 值或者估计 FDR

13.6 实验：多重检验

基础引入：

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from ISLP import load_data
```

本章新的引入：

```
from scipy.stats import \
    (ttest_1samp,
     ttest_rel,
     ttest_ind,
     t as t_dbn)
from statsmodels.stats.multicomp import \
     pairwise_tukeyhsd
from statsmodels.stats.multitest import \
     multipletests as mult_test
```

13.6.1 假设检验回顾

首先进行单样本假设检验。生成 100 个样本，其中前五十个样本的均值为 0.5，方差为 1；后五十个样本的均值为 0，方差为 1：

```
# 生成数据
rng = np.random.default_rng(12)
X = rng.standard_normal((10, 100)) # 标准正态分布
true_mean = np.array([0.5]*50 + [0]*50) # 列表，前50个元素值为0.5，后50个元素值为0
X += true_mean[None,:] # 为true_mean添加维度称为(1, 100)，然后进行广播
```

我们用 `scipy.stats` 中的 `ttest_1samp()` 检验 $H_0 : \mu_1 = 0$：

```
# 单样本假设检验
result = ttest_1samp(X[:,0], 0) # 选择第一列的元素，第二个参数表示原假设设定的检验均值
print(result.pvalue) # 输出0.931
```

结果显示完全不能拒绝整体的均值，注意设定的真实均值为 $\mu_1 = 0.5$，因此犯了第二类错误

现在，测试所有 100 个列的均值 $H_{0,j} : \mu_j = 0$。我们计算 100 个 p 值然后构造一个向量，记录第 j 个 p 值是否 ≤ 0.05，此时拒绝 H_{0j}：

```
# 进行100对假设检验
p_values = np.empty(100) # 空数组
for i in range(100):
   p_values[i] = ttest_1samp(X[:,i], 0).pvalue
decision = pd.cut(p_values,
                  [0, 0.05, 1],
                  labels=['Reject H0',
                          'Do not reject H0'])
truth = pd.Categorical(true_mean == 0, # 分类用的bool数组
                       categories=[True, False], # 标签
                       ordered=True)
```

这里使用 `pd.cut` 将连续的数值离散化，即划分 p 值到各种不同区间，这里划分的区间是 $[0, 0.05]$ 和 $[0.05, 1]$，同时指定区间的标签。`pd.Categorical` 转换了一个 `Categorical` 对象，它的第一个参数是布尔数组，第二个参数标签，第三个参数指定有序，然后我们生成混淆矩阵：

```
# 生成混淆矩阵
pd.crosstab(decision,
            truth,
     rownames=['Decision'],
     colnames=['H0'])
```

```
"""
H0                True  False
Decision
Reject H0            5     15
Do not reject H0    45     35
"""
```

结果表明，我们共拒绝了 20 个零假设，其中 5 个错误拒绝，属于第一类错误

由于对于假零假设，均值与标准差比率仅有 $0.5/1 = 0.5$，表明一个很弱的信号，容易导致大量第二类错误。下面使用更强信号的模拟数据，可以看到我们只犯了 10 个二类错误：

```
true_mean = np.array([1]*50 + [0]*50)
X = rng.standard_normal((10, 100))
X += true_mean[None,:]
for i in range(100):
   p_values[i] = ttest_1samp(X[:,i], 0).pvalue
decision = pd.cut(p_values,
                  [0, 0.05, 1],
                  labels=['Reject H0',
                          'Do not reject H0'])
truth = pd.Categorical(true_mean == 0,
                       categories=[True, False],
                       ordered=True)
pd.crosstab(decision,
            truth,
            rownames=['Decision'],
            colnames=['H0'])
```

这里可以看出假设检验的功效增大了

13.6.2 族错误率

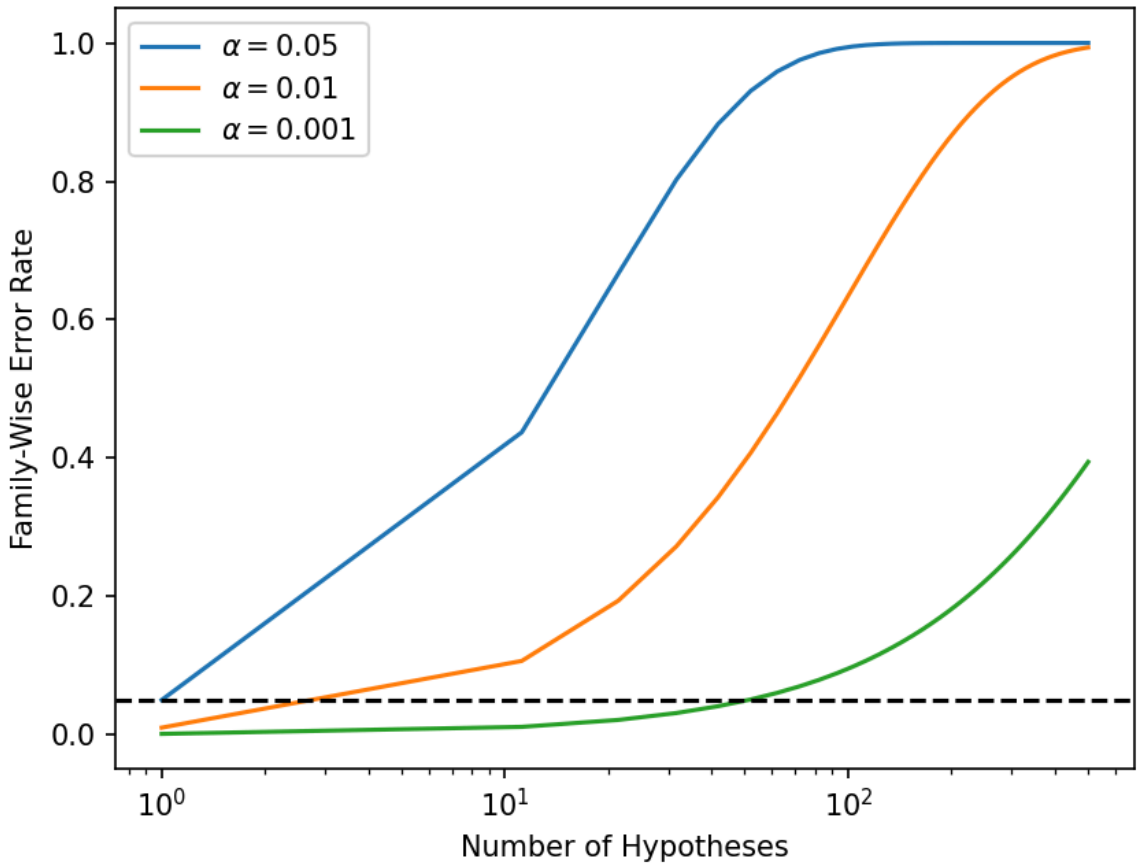
回顾族错误率的定义：

$$\mathrm{FWER} = 1 - (1 - \alpha)^m$$

其中 m 表示所有独立假设检验的数量，FWER 即至少发生一次第一类错误的概率。对 $m = 1, \ldots, 500$，$\alpha = 0.05, 0.01, 0.001$，可绘制这些 α 下随着 m 增大的族错误率：

```
# 绘制族错误率随m增大的变化曲线
m = np.linspace(1, 501)
fig, ax = plt.subplots()
[ax.plot(m,
         1 - (1 - alpha)**m,
         label=r'$\alpha=%s$' % str(alpha))
         for alpha in [0.05, 0.01, 0.001]] # 用列表推导式绘制不同α下随m变化的族错
误率曲线
```

```
ax.set_xscale('log')
ax.set_xlabel('Number of Hypotheses')
ax.set_ylabel('Family-Wise Error Rate')
ax.legend()
ax.axhline(0.05, c='k', ls='--') # 在0.05水平处绘制横线
plt.show()
```



观察图像发现，即使对一个较小的 $m = 50$，α 下的错误率也很容易超过 0.05，除非设置 $\alpha = 0.001$，但这样假设检验的功效很低

考虑对 `Fund` 数据集中前 5 位基金经理进行单样本 t 检验，假设检验为 $H_{0j} : \mu_j = 0$：

```
# 对5条数据进行假设检验
Fund = load_data('Fund')
fund_mini = Fund.iloc[:,:5]
fund_mini_pvals = np.empty(5)
for i in range(5):
    fund_mini_pvals[i] = ttest_1samp(fund_mini.iloc[:,i], 0).pvalue
print(fund_mini_pvals)
"""
输出
array([0.00620236, 0.91827115, 0.01160098, 0.6005396 , 0.75578151])
"""
```

注意到 1 和 3 的 p 值很低，但是仅凭单样本假设检验不足以拒绝它们的原假设，需要使用 Bonferroni 方法或 Holm 方法

用来自 `statsmodels` 的 `multipletests()` 函数 (此处重命名为 `mult_test()`) 进行上面两种检验。给定 p 值，对于 Holm 或 Bonferroni 方法，这个函数输出**校正 p 值 (adjusted p-values)**，可视为为多重检验更新的 p 值。若校正 p 值 $\leq \alpha$，则可在保持 FWER $\leq \alpha$ 的情况下拒绝原假设。也就是说，我们可以简单地将校正 p 值与需要的 FWER 比较，以确定是否拒绝每个原假设，这个方法随后应用于控制 FDR

`mult_test()` 函数采用 p 值和方法参数，以及可选的 α 参数，返回决策以及调整后的 p 值

```
# Bonferroni校正
reject, bonf = mult_test(fund_mini_pvals, method = "bonferroni")[:2]
print(reject)
"""
输出
array([ True, False, False, False, False])
"""
```

这里 `mult_test()` 可选 `alpha=` 参数，默认为 0.05，我们查看校正 p 值：

```
print(bonf,
 np.minimum(fund_mini_pvals * 5, 1)) # 将大于1的值截断为1
```

这里手动模拟了 Bonferrnoi 校正过程，可以看到两个列表值相同。结果表明我们只能拒绝 1 的原假设，改用 Holm 方法：

```
# Holm方法
print(mult_test(fund_mini_pvals, method = "holm", alpha=0.05)[:2])
"""
输出
(array([ True, False,  True, False, False]),
 array([0.03101178, 1.        , 0.04640393, 1.        , 1.        ]))
"""
```

这里结果表明我们能拒绝 1 和 3 的原假设

我们查看原始数据的均值：

```
print(fund_mini.mean())
"""
输出
Manager1    3.0
Manager2   -0.1
Manager3    2.8
Manager4    0.5
```

```
Manager5    0.3
dtype: float64
"""
```

现在希望检验两位经理之间的绩效之间是否存在有意义的差异，可使用 `scipy.stats` 的 `ttest_rel()` 函数检验均值：

```
# 执行均值检验
print(ttest_rel(fund_mini['Manager1'],
          fund_mini['Manager2']).pvalue)
```

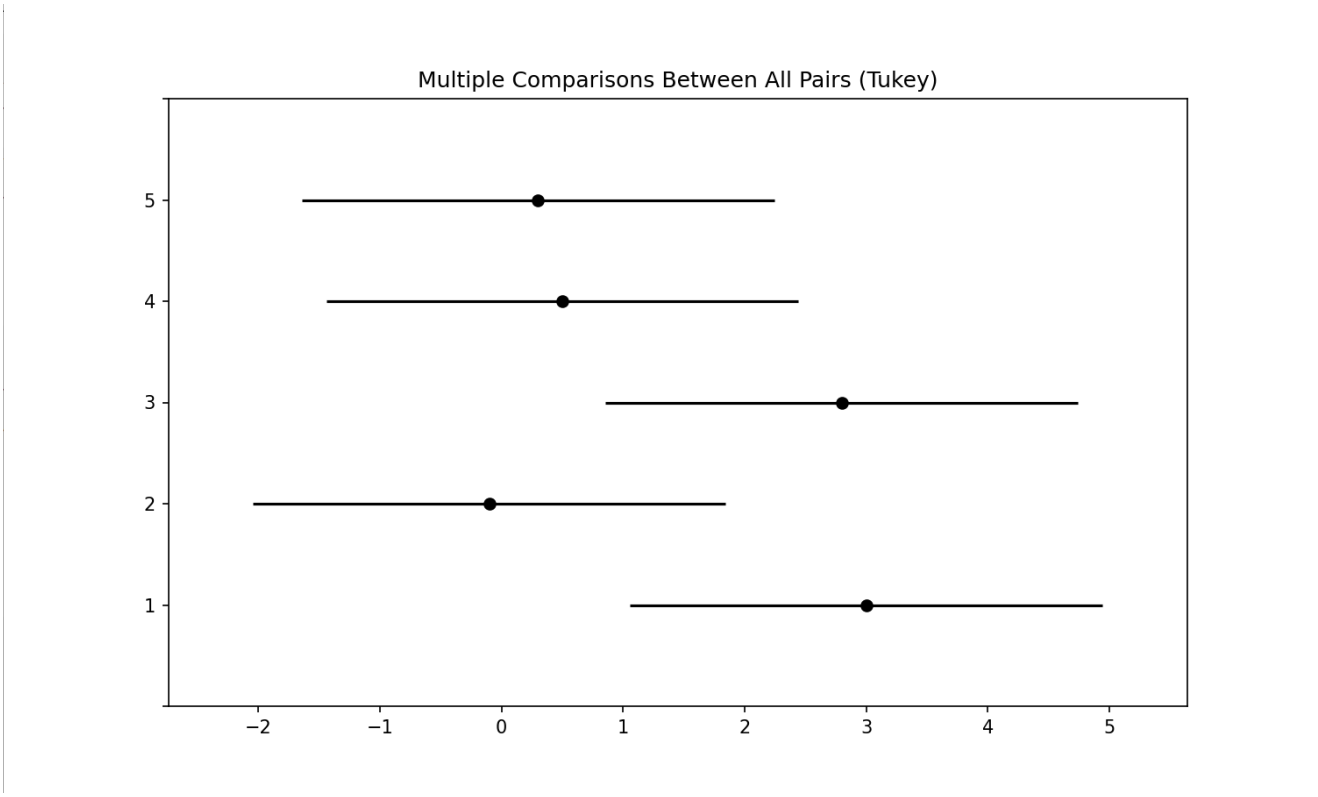
事实上，我们查看了 5 个样本后才决定对 1 和 2 号进行假设检验，这实际上隐含地执行 $C_5^2 = 10$ 次假设检验，因此考虑用 Tukey 方法校正。这里用到的函数是来自 `statsmodels.stats.muticomp` 的 `pairwise_tukeyhsd()` 函数。这个函数接受一个拟合的 ANOVA 回归模型(本质上是一种特殊的线性回归)：

```
# Tukey方法校正
returns = np.hstack([fund_mini.iloc[:,i] for i in range(5)]) # 按列堆叠数组
managers = np.hstack([[i+1]*50 for i in range(5)]) # 建立一个分组标签，值分别为1,2,3,4,5，各50个
tukey = pairwise_tukeyhsd(returns, managers)
print(tukey.summary())
"""
输出
Multiple Comparison of Means - Tukey HSD, FWER=0.05
=================================================
group1 group2 meandiff p-adj   lower  upper  reject
-------------------------------------------------
     1      2     -3.1 0.1862 -6.9865 0.7865  False
     1      3     -0.2 0.9999 -4.0865 3.6865  False
     1      4     -2.5 0.3948 -6.3865 1.3865  False
     1      5     -2.7 0.3152 -6.5865 1.1865  False
     2      3      2.9 0.2453 -0.9865 6.7865  False
     2      4      0.6 0.9932 -3.2865 4.4865  False
     2      5      0.4 0.9986 -3.4865 4.2865  False
     3      4     -2.3  0.482 -6.1865 1.5865  False
     3      5     -2.5 0.3948 -6.3865 1.3865  False
     4      5     -0.2 0.9999 -4.0865 3.6865  False
-------------------------------------------------
"""
```

这里的 `np.hstack` 表示按列堆叠数据，返回一个一维序列，注意此序列不是向量 (因为维度为 1)。观察输出结果，我们发现输出提供校正 p 值和均值差值的置信区间。我们可用 `plot_simultaneous()` 方法绘制对比的置信区间：

```
# 绘制置信区间对比图
fig, ax = plt.subplots(figsize=(8,8))
tukey.plot_simultaneous(ax=ax)
plt.show()
```

图像表示每个基金数据的 95% 置信区。如果置信区间不重叠，我们可认为两组的均值差包含明显差异



13.6.3 错误发现率

现在考虑对 2000 名基金经理进行假设检验，检验其汇报均值是否为 0：

```
# 对2000的数据进行假设检验
fund_pvalues = np.empty(2000)
for i, manager in enumerate(Fund.columns):
    fund_pvalues[i] = ttest_1samp(Fund[manager], 0).pvalue
```

因为样本很多，我们不再考虑控制 FWER，而是转向控制 FDR。`multipletests()` 可用于执行 Benjamini-Hochberg 程序以控制 FDR：

```
# BH程序
fund_qvalues = mult_test(fund_pvalues, method = "fdr_bh")[1] # 第1个数组是校验后的q值数组
print(fund_qvalues[:10])
"""
输出
array([0.08988921, 0.991491  , 0.12211561, 0.92342997, 0.95603587,
```

```
        0.07513802, 0.0767015 , 0.07513802, 0.07513802, 0.07513802])
"""
```

这里输出的数组是**q 值 (q-values)** 的序列。举例说明，$q = 0.1$ 时，表示我们可在设定 FDR 设定为 10% 或更高的情况下否定响应的原假设，但是在 FDR $\leq 10\%$ 时不能拒绝。例如，我们将 FDR 设定为 10%，然后查看能够拒绝零假设的总数：

```
# 查看拒绝零假设的总数
print((fund_qvalues <= 0.1).sum())
"""
输出
146
"""
```

这表明我们认为 146 位经理的表现超过 0，且里面只有 $\approx 10\%$ 的是错误的发现。相比之下，若改用 Bonferroni 方法控制 FWER 为 $\alpha = 0.1$：

```
print((fund_pvalues <= 0.1 / 2000).sum())
```

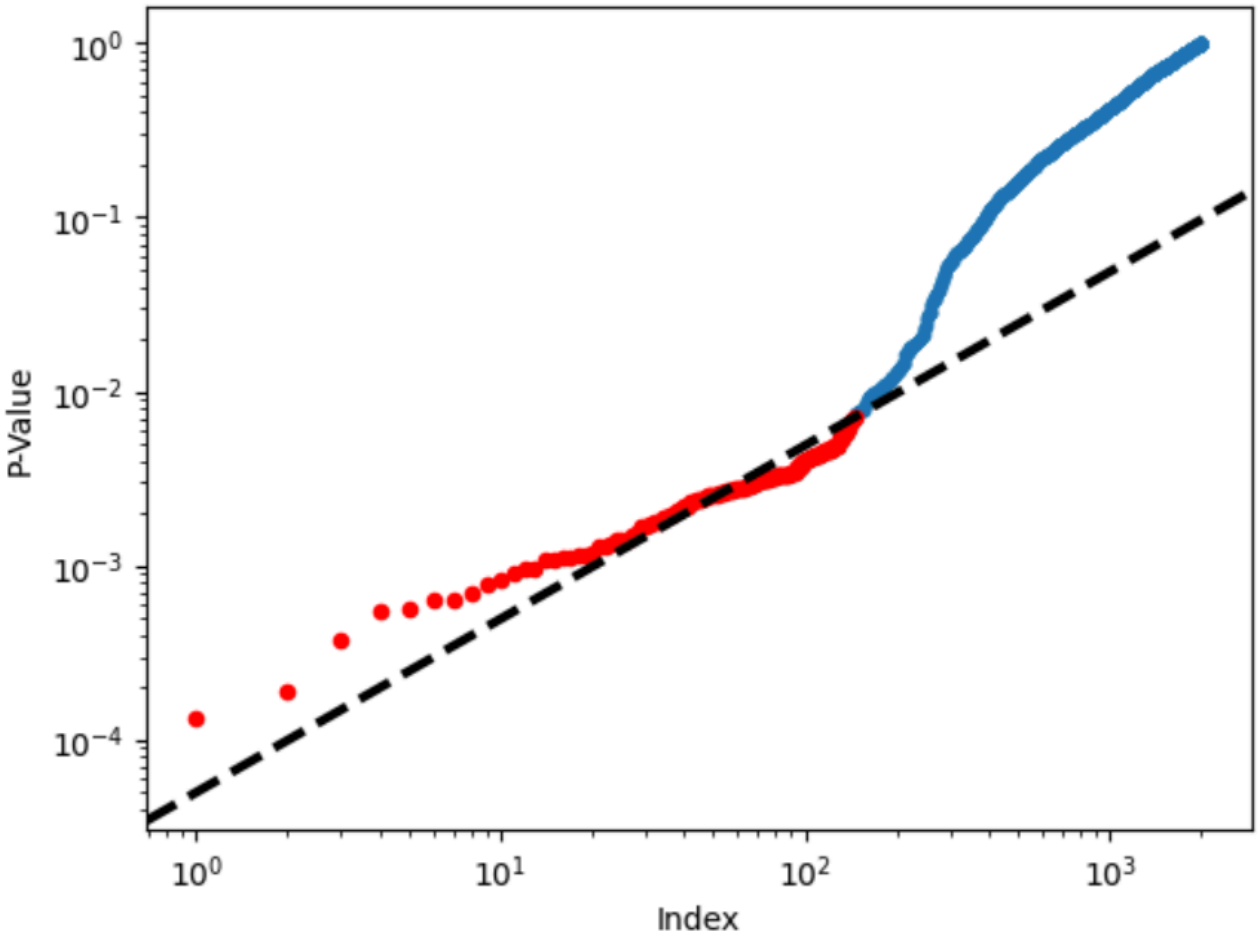
结果显示我们不能拒绝任何零假设

在下面的代码中，我们手动实现了 Benjamini–Hochberg 程序。我们首先对 p 值进行排序，然后确定满足 $p_{(j)} < qj/m$ 的样本并拒绝之：

```
# 手动实现BH程序
sorted_ = np.sort(fund_pvalues) # 对p值排序
m = fund_pvalues.shape[0] # p值的总数
q = 0.1 # FDR控制阈值
sorted_set_ = np.where(sorted_ < q * np.linspace(1, m, m) / m)[0]
if sorted_set_.shape[0] > 0: # 存在满足条件的p值
    selected_ = fund_pvalues < sorted_[sorted_set_].max() # 布尔数组，用于标记
fund_pvalues中小于sorted_[sorted_set_]中最大p值的元素
    sorted_set_ = np.arange(sorted_set_.max()) # 从0到sorted_set_中最大索引的
数组
else: # 不存在满足条件的p值
    selected_ = []
    sorted_set_ = []

# 绘制BH程序图像
fig, ax = plt.subplots()
ax.scatter(np.arange(0, sorted_.shape[0]) + 1,
           sorted_, s=10) # 绘制排序后p点
ax.set_yscale('log')
ax.set_xscale('log')
ax.set_ylabel('P-Value')
ax.set_xlabel('Index')
ax.scatter(sorted_set_+1, sorted_[sorted_set_], c='r', s=20) # 绘制满足条件的p
```

```
值的散点图
ax.axline((0, 0), (1,q/m), c='k', ls='--', linewidth=3) # 绘制阈值直线
plt.show()
```



> 注意，HB 程序寻找到最大(尽可能大)不满足 $p_{(j)} \leq qj/m$ 的点，无法拒绝之后的原假设。虚线表示 qj/m，最大的不满足 qj/m 的点在红色点在红色最后 (事实上，当红色点下落到虚线以下时，显然后续上升的点都不满足此条件)

13.6.4 重采样方法

下面采用 `Khan` 数据集实施重采样方法。首先合并训练数据和测试数据，从而观察 83 名患者的 2308 个基因：

```
# 观察Khan数据
Khan = load_data('Khan') # Khan是一个字典，包含四个dataframe
D = pd.concat([Khan['xtrain'], Khan['xtest']])
D['Y'] = pd.concat([Khan['ytrain'], Khan['ytest']])
print(D['Y'].value_counts()) # 统计不同标签(元素)出现的次数
"""
输出，每种数字对应一种类型的癌症
Y
2    29
4    25
3    18
```

```
1    11
Name: count, dtype: int64
"""
```

我们对数量最多的第二类和第四类癌症的基因平均表达量进行比较，执行双样本 t 检验：

```
# 对2和4进行双样本t检验
D2 = D[lambda df:df['Y'] == 2]
D4 = D[lambda df:df['Y'] == 4]
gene_11 = 'G0011' # 检验G0011基因在两类样本中的表达均值差异
observedT, pvalue = ttest_ind(D2[gene_11],
                              D4[gene_11],
                              equal_var=True)
print(observedT, pvalue) # t统计量值, 对应p值
```

这里使用的是来自 `scipy.stats` 的 `ttest_ind()` 函数，用于双样本的 t 统计量检验，指定 `equal_var=True` 表示假设两个样本的方差相等

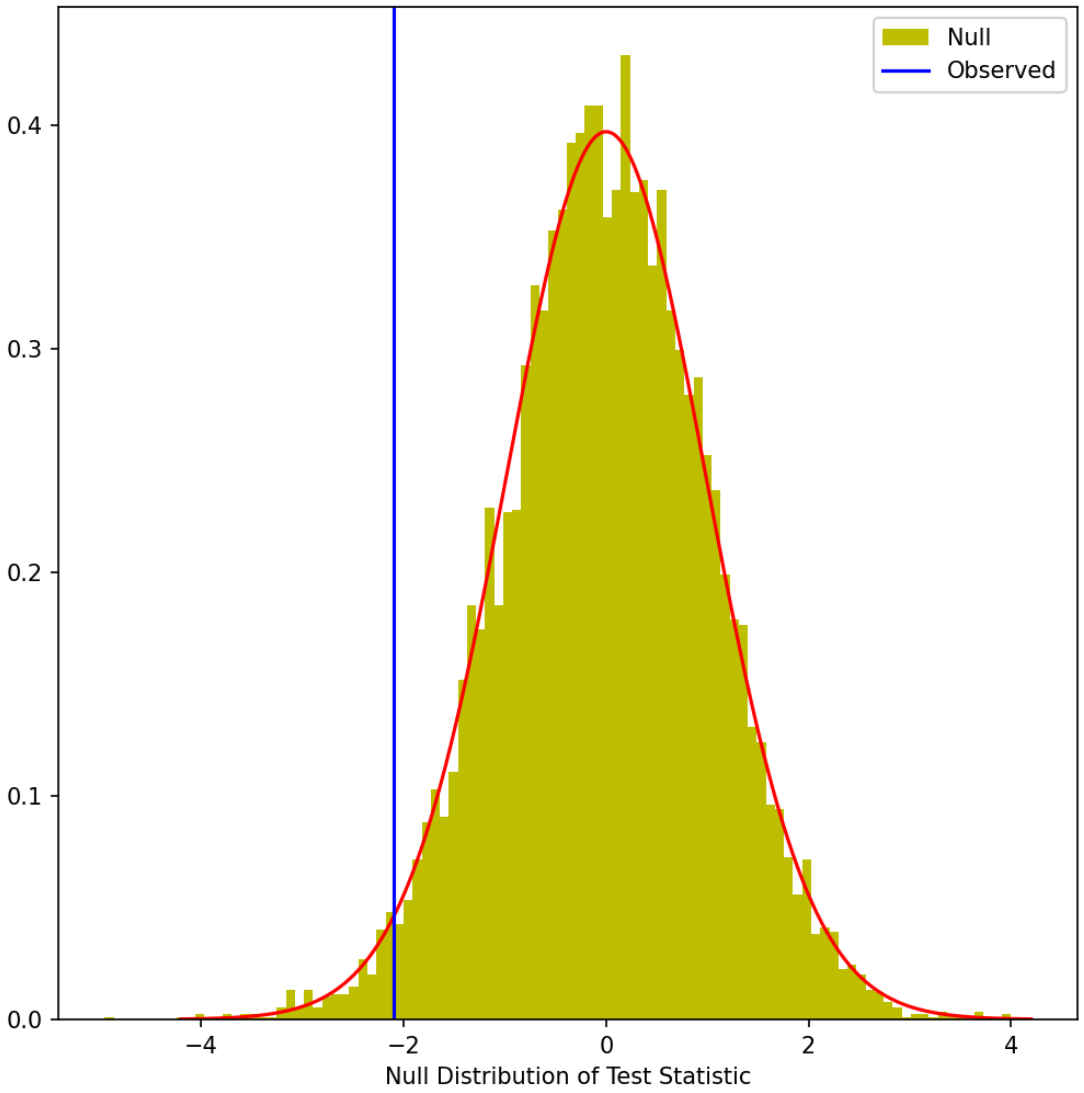
这个 p 值依赖于检验统计量服从 $t_{(52)}$ 的零分布。我们可将 54 名患者随机分成两组，分别有 29 名和 25 名，然后计算新的检验统计量。在 H_0 下，新的检验统计量和原始的检验统计量有相同的分布，重复此过程 $B = 10000$ 次可以近似检验统计量的零分布，然后计算观测到的检验统计量超过原始检验统计量的分数：

```
# 重采样
B = 10000
Tnull = np.empty(B)
D_ = np.hstack([D2[gene_11], D4[gene_11]])
n_ = D2[gene_11].shape[0] # 获取D2中G0011基因表达数据的样本数量, 后续重新划分
D_null = D_.copy()
for b in range(B):
    rng.shuffle(D_null) # 打乱
    ttest_ = ttest_ind(D_null[:n_], # 患者2部分
                       D_null[n_:], # 患者4部分
                       equal_var=True)
    Tnull[b] = ttest_.statistic # 获得统计量
print((np.abs(Tnull) < np.abs(observedT)).mean()) # 计算计算T的均值, 即估计p值
```

绘制关于重采样的直方图，以比较重采样得到的估计值与理论零分布的值关系：

```
# 绘制重采样分布
fig, ax = plt.subplots(figsize=(8,8))
ax.hist(Tnull, # 绘制重抽样直方图
        bins=100,
        density=True, # 设定纵坐标表示概率密度(概率)而不是频数
        facecolor='y',
        label='Null')
xval = np.linspace(-4.2, 4.2, 1001)
```

```
ax.plot(xval, # 绘制理论零分布
        t_dbn.pdf(xval, D_.shape[0]-2),
        c='r')
ax.axvline(observedT,
           c='b',
           label='Observed')
ax.legend()
ax.set_xlabel("Null Distribution of Test Statistic")
plt.show()
```



如上图所示，重采样得到的零分布与理论零分布几乎相同

最后，我们尝试实现重采样估计 FDR 的方法：

```
# 重采样估计FDR, 对统计量T进行采样
m, B = 100, 10000
idx = rng.choice(Khan['xtest'].columns, m, replace=False) # 随机选择基因, 不允
```

```
许重复
T_vals = np.empty(m) # 存储m长的每个基因在原始样本t检验的统计量
Tnull_vals = np.empty((m, B)) # 存储每个基因在每次重采样后得到的t统计量，表示为m×B
大小矩阵

for j in range(m):
    col = idx[j] # 当前检测基因名
    T_vals[j] = ttest_ind(D2[col],
                          D4[col],
                          equal_var=True).statistic # 原数数据得到的t统计量
    D_ = np.hstack([D2[col], D4[col]]) # 堆叠两组数据
    D_null = D_.copy()
    for b in range(B):
        rng.shuffle(D_null) # 打乱后分配
        ttest_ = ttest_ind(D_null[:n_],
                           D_null[n_:],
                           equal_var=True) # 重采样得到的t统计量
        Tnull_vals[j,b] = ttest_.statistic # 更新到重采样矩阵中
```

> 这里因为计算所有 2308 个基因需要花费相当的时间，因此计算的基因数量 $m = 100$

接下来，计算算法中阈值范围 c 内被拒绝的原假设数 R，估计假阳性数 V 和估计 FDR，阈值是用 100 个基因的检验统计量的绝对值选择的：

```
# 对R和V进行估计
cutoffs = np.sort(np.abs(T_vals)) # 对原始t统计量进行排序，这里排布的是绝对值，得
到绝对值数组
FDRs, Rs, Vs = np.empty((3, m))  # 创建3×m的矩阵，每行表示FDR，R和V
for j in range(m):
   R = np.sum(np.abs(T_vals) >= cutoffs[j]) # 计算R
   V = np.sum(np.abs(Tnull_vals) >= cutoffs[j]) / B # 计算V
   Rs[j] = R
   Vs[j] = V
   FDRs[j] = V / R
```

现在对于任意给定 FDR，我们都可以找到被拒绝原假设的基因，例如当 FDR 控制在 0.1 时，我们拒绝原假设 100 个原假设中的 15 个，而这些中大约有 10% 时错误的发现

变量 `idx` 存储了我们随机选择 100 个基因中包含哪些基因，我们可以在其中观察 FDR ≤ 0.1 的基因：

```
print(sorted(idx[np.abs(T_vals) >= cutoffs[FDRs < 0.1].min()]))
"""
输出
['G0097',
 'G0129',
 'G0182',
 'G0714',
```

```
 'G0812',
 'G0941',
 'G0982',
 'G1020',
 'G1022',
 'G1090',
 'G1320',
 'G1634',
 'G1697',
 'G1853',
 'G1854',
 'G1994',
 'G2017',
 'G2115',
 'G2193']
"""
```

如果设定 FDR 为 0.2，我们选择了更多基因：

```
print(sorted(idx[np.abs(T_vals) >= cutoffs[FDRs < 0.2].min()]))
"""
输出
['G0097',
 'G0129',
 'G0158',
 'G0182',
 'G0242',
 'G0552',
 'G0679',
 'G0714',
 'G0751',
 'G0812',
 'G0908',
 'G0941',
 'G0982',
 'G1020',
 'G1022',
 'G1090',
 'G1240',
 'G1244',
 'G1320',
 'G1381',
 'G1514',
 'G1634',
 'G1697',
 'G1768',
 'G1853',
 'G1854',
 'G1907',
```

```
 'G1994',
 'G2017',
 'G2115',
 'G2193']
"""
```

最后，我们绘制随着 FDR 增加选择被拒绝原假设的数量：

```
# 绘制FDR与拒绝原假设数量的曲线
fig, ax = plt.subplots()
ax.plot(Rs, FDRs, 'b', linewidth=3)
ax.set_xlabel("Number of Rejections")
ax.set_ylabel("False Discovery Rate")
plt.show()
```