生物统计第二次编程作业——各种检验方法

统计81 于越 *

May 3, 2021

Contents

1 报告简介 **3**

2 Task1-two sample independent test **3**

2.1 chi-square test . . . 3

2.1.1 原理解释 . . . 3

2.1.2 具体实验1：作用于模拟数据 . . . 4

2.1.3 具体实验2：作用于实际数据 . . . 7

2.1.4 具体实验3：关于卡方检验的评价——power function . . . 8

2.2 Fisher exact test . . . 10

2.2.1 原理解释 . . . 10

2.2.2 具体实验1：作用于模拟数据 . . . 11

2.2.3 具体实验2：作用于实际数据 . . . 13

2.2.4 具体实验3：关于Fisher精确检验的评价——power function . . . 13

3 Task2-Two sample dependent test(Mcnemar test) **15**

3.1 原理解释 . . . 15

3.2 具体实验1：实际数据(正态理论检验) . . . 16

3.3 具体实验2：实际数据(精确检验) . . . 17

3.4 检验方法评价 . . . 18

4 Task3——非参方法(Wilcoxon rank sum test) **18**

4.1 原理解释 . . . 18

4.2 具体实验1：作用于模拟数据 . . . 19

4.3 具体实验2：作用于模拟正态分布比较wilcoxon秩和检验与t检验的效果 . . . 20

4.4 具体实验3：通过power function比较wilcoxon秩和检验与t检验的效果 . . . 21

*学号：2183210516

4.5 具体实验4：对于大样本的情况采用置换检验(permutation test) 24
4.5.1 原理解释 24
4.5.2 permutation test作用于实际数据 24

5 报告总结 **25**

A 相关代码 **26**

1 报告简介

生物统计课程的第五章与第六章主要讲述了对于各种不同类型数据的检验方法，主要涉及非参检验与类别数据检验两部分。本次作业就主要涉及这两个方面。在类别数据的检验方面，本篇报告主要研究类别数据中对于两个变量的关系的两样本独立性检验以及两样本非独立性检验；对于两样本独立性检验方法，代表是卡方检验与Fisher精确检验(主要是样本量不大的情况下)，而对于两样本非独立检验方法，代表是McNemar Test，主要针对两样本之间成对出现情况或者是显然的两样本不独立的情况。针对非参检验的方法，主要涉及Wilcoxon秩和检验，同时对于模拟数据和实际数据，都与常用的t检验方法进行了比较并进行评价。所有的检验方法都是在自行生成的模拟数据与实际数据上经过了作用。

2 Task1-two sample independent test

2.1 chi-square test

2.1.1 原理解释

对于卡方检验，我采用的都是列联表数据进行检验，研究是否一个因素的不同水平会对某一结果(是否得病等等)产生显著影响。假定我们将实验结果分为case组与control组，可以得到的列联表如下所示：

Table 1: 一般列联表形式

ElementA	ElementB	ElementB	
Status	level1	level2	Total
Case	x_1	$n_1 - x_1$	n_1
Control	x_2	$n_2 - x_2$	n_2
Total	$x_1 + x_2$	$n_1 + n_2 - x_1 - x_2$	$n_1 + n_2$

我们假设的H_0为因素的不同水平不会对不同组实验结果产生显著影响(即$p_1 = p_2$)，H_1为因素的不同水平确实会对不同组实验结果产生显著影响(即$p_1 \neq p_2$)。通过列联表我们可以获得对于比率p的最好估计为

$$\hat{p} = \frac{n_1\hat{p_1} + n_2\hat{p_2}}{n_1 + n_2} = \frac{x_1 + x_2}{n_1 + n_2}. \tag{1}$$

对此我们可以求出列联表中各个位置的期望值E_{ij}(i表示行，j表示列).例如

$$E_{11} = n_1\hat{p} = \frac{n_1(x_1 + x_2)}{n_1 + n_2}, \qquad E_{21} = n_2\hat{p} = \frac{n_2(x_1 + x_2)}{n_1 + n_2} \tag{2}$$

我们用E_{ij}表示列联表(i,j)位置的估计值，用O_{ij}表示列联表(i,j)位置的观测值。故可以构造统计量为

$$X^2 = \frac{(|O_{11} - E_{11}| - 0.5)^2}{E_{11}} + \frac{(|O_{12} - E_{12}| - 0.5)^2}{E_{12}} + \frac{(|O_{21} - E_{21}| - 0.5)^2}{E_{21}} + \frac{(|O_{22} - E_{22}| - 0.5)^2}{E_{22}} \tag{3}$$

可以证明该统计量服从chi-square分布，即$X^2 \sim \chi^2(1)$.

注：事实上可以证明对于m*n阶列联表，构造的这种统计量$X^2 \sim \chi^2((m-1)*(n-1))$

故对于显著水平α，有

$$\text{拒绝} H_0 : X^2 > \chi^2_{1,1-\alpha} \qquad \text{接受} H_0 : X^2 < \chi^2_{1,1-\alpha} \tag{4}$$

注：卡方检验对于列联表的检验应当在列联表各个位置的期望值$E_{ij} \geq 5$时使用。

2.1.2 具体实验1：作用于模拟数据

为保证生成数据的独立性，我先随机取总实验对象数为1000，用二项分布生成随机数的方式将1000个总实验对象分为两类，代表将列联表分为上下两行。接着对于上下两行的实验对象个数再分别进行二项分布生成随机数的抽样，由此共计得到4个数据，分别对应于列联表的4个位置。

在这里，我先令p=0.3,将全体N=1000个实验对象分为两部分。首先进行检验$p_1 \neq p_2$的实验。对于第一部分数据，采用$p_1 = 0.4$进行抽样，将第一部分数据再分为两部分，分别对应列联表的上一行两个数据；而对于第二部分数据，采用$p_2 = 0.6$进行抽样，将第二部分数据再分为两部分，分别对应列联表的下一行两个数据。在这里，进行1次数据生成，得到的生成数据结果以及计算对应的期望值如下所示：

```
> n11
[1] 130
> n12
[1] 176
> n21
[1] 404
> n22
[1] 290
```

(a) 自行模拟列联表数据生成

```
> E11
[1] 163.404
> E12
[1] 142.596
> E21
[1] 370.596
> E22
[1] 323.404
```

(b) 对应列联表数据期望值

Figure 1: 模拟列联表数据

故可以得到这样一张列联表：

Table 2: 一般列联表形式

ElementA	ElementB	ElementB	
Status	level1	level2	Total
Case	130	176	306
Control	404	290	694
Total	534	466	1000

而对应的列联表期望值如下表所示：

Table 3: 一般列联表形式

ElementA	ElementB	ElementB	
Status	level1	level2	Total
Case	163.4	142.6	306
Control	370.6	323.4	694
Total	534	466	1000

采用上面的计算公式，取显著水平$\alpha = 0.05$，可以计算出统计量与对应的卡方分位数如下所示：

```
> X_2
[1] 20.48752
> quanchi
[1] 3.841459
```

Figure 2: 卡方检验结果

可以看到统计量$X^2 = 20.4875$，而分位数$\chi^2_{1,1-\alpha} = 3.8414$，显然有$X^2 > \chi^2_{1,1-\alpha}$成立，故应当拒绝原假设，认为两组数据的抽样概率$p_1 \neq p_2$，即在此时，因素的影响对于不同实验组的实验结果影响显著。这与我们最初模拟数据时的采用的$p_1 = 0.4 \neq p_2 = 0.6$相比也是相符的。

再进行$p_1 = p_2$情况的实验。对于N=1000情况下的分成两部分的数据，我们采用$p_1 = p_2 = 0.5$的概率进行二项分布的抽样。此时得到的生成数据结果以及计算对应的期望值如下所示：

```
> n11
[1] 136
> n12
[1] 159
> n21
[1] 350
> n22
[1] 355
```

(a) 自行模拟列联表数据生成

```
> E11
[1] 143.37
> E12
[1] 151.63
> E21
[1] 342.63
> E22
[1] 362.37
```

(b) 对应列联表数据期望值

Figure 3: 模拟列联表数据

故可以得到这样一张列联表：

Table 4: 一般列联表形式

ElementA	ElementB	ElementB	
Status	level1	level2	Total
Case	136	159	295
Control	350	355	705
Total	486	514	1000

而对应的列联表期望值如下表所示：

Table 5: 一般列联表形式

ElementA	ElementB	ElementB	
Status	level1	level2	Total
Case	143.4	151.6	295
Control	342.6	362.4	705
Total	486	514	1000

取显著水平$\alpha = 0.05$，可以计算出统计量与对应的卡方分位数如下所示：

```
> X_2
[1] 0.908454
> quanchi
[1] 3.841459
```

Figure 4: 卡方检验结果

此时的统计量$X^2 = 0.9085$,而此时的分位数$\chi^2_{1,1-\alpha} = 3.8414$，显然有$X^2 < \chi^2_{1,1-\alpha}$成立，故不拒绝原假设，认为两组数据抽样概率$p_1 = p_2$即在此时，因素的影响对于不同实验组的实验结果影响不显著。这与我们最初模拟数据时的采用的$p_1 = p_2 = 0.5$相比也是相符的。

2.1.3 具体实验2：作用于实际数据

在此处采用的实际数据来源于R软件中的vcd包自带数据集，该数据集主要描述了两个实验组分别服用治疗药剂与安慰剂后治疗关节炎的情况。具体的列联表如下所示：

```
Total Observations in Table:  84


                    | Arthritis$Improved
Arthritis$Treatment |       None |       Some |     Marked | Row Total |
--------------------|------------|------------|------------|-----------|
            Placebo |         29 |          7 |          7 |        43 |
                    |      2.616 |      0.004 |      3.752 |           |
                    |      0.674 |      0.163 |      0.163 |     0.512 |
                    |      0.690 |      0.500 |      0.250 |           |
                    |      0.345 |      0.083 |      0.083 |           |
--------------------|------------|------------|------------|-----------|
            Treated |         13 |          7 |         21 |        41 |
                    |      2.744 |      0.004 |      3.935 |           |
                    |      0.317 |      0.171 |      0.512 |     0.488 |
                    |      0.310 |      0.500 |      0.750 |           |
                    |      0.155 |      0.083 |      0.250 |           |
--------------------|------------|------------|------------|-----------|
       Column Total |         42 |         14 |         28 |        84 |
                    |      0.500 |      0.167 |      0.333 |           |
--------------------|------------|------------|------------|-----------|
```

Figure 5: 生成的实际数据列联表

此时可以计算得到数据集的期望值列联表如下所示：

Table 6: 一般列联表形式

Treatment	Improved	Improved	Improved	
Status	None	Some	Marked	Row Total
Placebo	21.5	7.2	14.3	43
Treated	20.5	6.8	13.7	41
Column Total	42	14	28	84

取显著水平$\alpha = 0.05$，可以计算出统计量与对应的卡方分位数如下所示：

```
> X_2
[1] 11.3755
> quanchi
[1] 5.991465
```

Figure 6: 卡方检验结果

此时的统计量$X^2 = 11.3755$，而分位数$\chi^2_{2,1-\alpha} = 5.9915$，显然有$X^2 > \chi^2_{2,1-\alpha}$成立，故应当拒绝原假设，认为两组数据的抽样概率$p_1 \neq p_2$。由此可以说明，服用关节炎治疗药物与服用安慰剂对于治疗关节炎的效果有着显著影响。

2.1.4 具体实验3：关于卡方检验的评价——power function

要具体评价一种检验方法，可以采用power function也就是势函数进行评价。关于势函数的定义如下：

设检验问题

$$H_0 : \theta \in \Theta_0 \qquad vs \qquad H_1 : \theta \in \Theta_1 \tag{5}$$

的拒绝域为W，则样本观测值X落在拒绝域W内的概率称为该检验的势函数，记为

$$g(\theta) = P_\theta(X \in W), \quad \theta \in \Theta = \Theta_0 \cup \Theta_1 \tag{6}$$

这是定义在参数空间上的函数。

势函数的意义是参数落在参数空间上时拒绝原假设的概率。在这里，我们也做出势函数的图像，横坐标为检验中对应的两个参数的差值，纵坐标通过多次实验的结果模拟得出拒绝次数而估计势函数的值。例如在该实验中，我对于每两个参数采取1000次实验，计算1000次实验中拒绝原假设的实验次数记为n，再用n/1000估计这一此时势函数的值。

第一次实验：固定$p_1 = 0.5$的情况下，p_2的值从0开始每次增加0.01到1，共计完成100组对应的参数差值测试，每组测试中进行1000次测试，通过卡方检验方法确定每组测试中拒绝原假设的次数n，再利用n/1000估计该组测试的对应势函数的差值的势函数。此时我们可以得到的势函数如下所示：

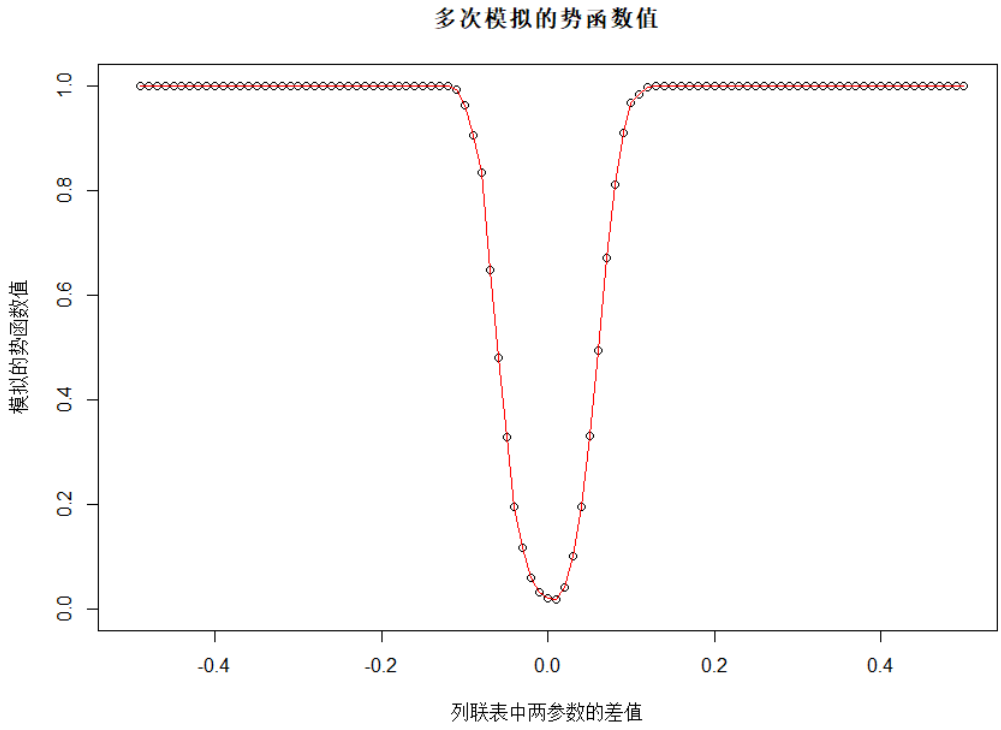

Figure 7: 卡方检验势函数评价$p_1 = 0.5$为基准

可以看到该函数呈中间下凹的趋势。当$p_2 - p_1 \leq -0.1$或者$p_2 - p_1 \geq 0.1$的情况下，做1000次实验几乎1000次全部拒绝原假设，而当$|p_2 - p_1| < 0.1$时，卡方检验的势函数值小于1，尤其是当$p_2 - p_1 = 0$时，势函数值几乎为0，这说明此时要接受原假设。总体来说，这个图像和我们预想的情况是基本符合的，当两参数差值$p_2 - p_1$较大时要拒绝原假设，而两参数差值较小时接受原假设。且该势函数由于横坐标的对称性也几乎是对称的。

但是值得一提的是，由于抽样方法的原因，当$|p_2 - p_1| < 0.1$时，有些应该拒绝原假设的点对应的势函数的值有些偏小了。如何减少这一参数差值的阈值是该实验需要改进的地方。

第二次实验：固定$p_1 = 0$的情况下，p_2的值从0.01开始每次增加0.01到1，共计完成100组对应的参数差值测试，每组测试中进行1000次测试，通过卡方检验方法确定每组测试中拒绝原假设的次数n，再利用n/1000估计该组测试的对应势函数的差值的势函数。此时我们可以得到的势函数如下所示：

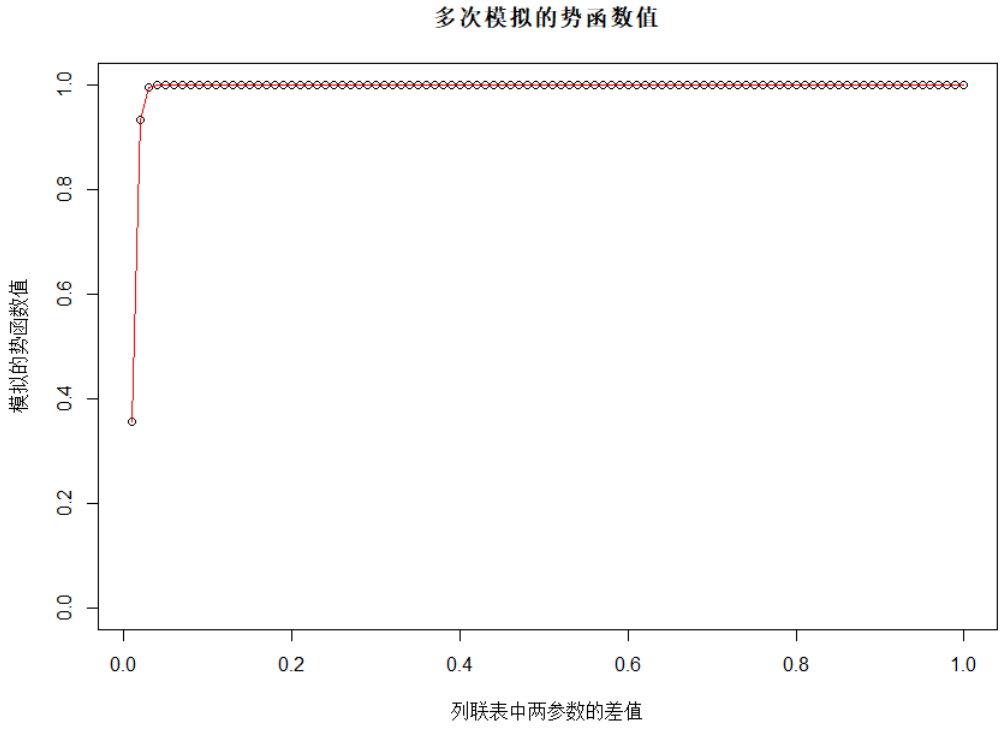

Figure 8: 卡方检验势函数评价$p_1 = 0$为基准

可以看到此时的差值是从0到1分布的。在这里我剔除了$p_1 = p_2 = 0$的情况，因为此时的两组数据完全是0-1抽样的情况，因此我从$p_2 = 0.01$开始。发现此时当两参数的差值为0.05左右时，势函数的值就已经接近于1了，说明对于1000次实验完全拒绝了原假设。这次的实验结果阈值比之前的少一些，说明实验的结果更好，而且参数差值关于势函数的图像大体呈单调上升的趋势，这也与我们预想的相符。

总体而言，对于两样本的独立性检验，在样本量较大(计算求得的期望值均不小

于5的情况下)，卡方检验方法是一个方便而实用的方法。

2.2 Fisher exact test

2.2.1 原理解释

对于两样本独立性检验中的列联表，我们通常采用卡方检验的方式。但是对于小样本的情况，卡方检验很有可能因为样本量过小而导致检验不精确的情况。这是因为卡方检验是一种主要应用于大样本的检验，卡方分布作为三大分布的一种，主要适用于数据量较大的情况。因此对于小样本的情况(当计算的期望值中有期望值¡5时)，我们通常采用Fisher精确检验。

关于Fisher精确检验，我们通常采用超几何分布进行类比。一般的列联表如下所示：

Table 7: 一般列联表形式

ElementA	ElementB	ElementB	
Status	level1	level2	Total
Case	a	b	N_1=a+b
Control	c	d	N_2=c+d
Total	M_1=a+c	M_2=b+d	n

类比超几何分布的情形。超几何分布主要是关于无放回情形下的次品抽样问题。设共有M个样品，其中有n件次品。我们无放回地进行N次抽样，计算在N次抽样中抽到a件次品的概率。可得概率为

$$Pr(X=a)=\frac{C_n^a C_{M-n}^{N-a}}{C_M^N}=\frac{n!N!(M-n)!(M-N)!}{a!M!(n-a)!(N-a)!(M-n-N+a)!} \tag{7}$$

因而在列联表中，我们用行或列的总和作为次品个数，可以得到类似的公式为

$$Pr(a,b,c,d)=\frac{(a+b)!(c+d)!(a+c)!(b+d)!}{n!a!b!c!d!} \tag{8}$$

而对于Fisher精确检验，我们需要将左上角位置a从0到$min(M_1,N_1)$依次进行排列，计算出对应的$Pr(X=a)$值。这是为了让我们计算p值以提供方便。我们记$k=min(M_1,N_1)$，对于列联表左上角的a值，对应的不同检验情况下的p值如下所示：

condition1:

$$H_0:p_1=p_2 \qquad vs \qquad H_1:p_1\neq p_2 \tag{9}$$

此时

$$p-value=2min[Pr(0)+Pr(1)+\cdots+Pr(a),Pr(a)+\cdots+Pr(k),0.5] \tag{10}$$

condition2:

$$H_0 : p_1 = p_2 \qquad vs \qquad H_1 : p_1 < p_2 \tag{11}$$

此时

$$p - value = Pr(0) + Pr(1) + \cdots + Pr(a) \tag{12}$$

condition3:

$$H_0 : p_1 = p_2 \qquad vs \qquad H_1 : p_1 > p_2 \tag{13}$$

此时

$$p - value = Pr(a) + Pr(a+1) + \cdots + Pr(k) \tag{14}$$

即可做出对应的对于检验的判断。

2.2.2 具体实验1：作用于模拟数据

对于两样本的独立性检验生成数据我们还采用和卡方检验相同的方法。为了保证计算出的期望值有¡5的数值，同时也是响应小样本的需求，我们将一次实验的实验对象改为N=50.在这里，我先令p=0.2，采用第一次的二项分布抽样将数据分为上下两组，然后对两组数据再模拟p_1与p_2的值。

首先进行检验$p_1 \neq p_2$是实验。对于第一部分数据，采用$p_1 = 0.1$进行抽样，将第一部分数据再分为两部分，分别对应列联表的上一行两个数据；而对于第二部分数据，采用$p_2 = 0.5$进行抽样，将第二部分数据再分为两部分，分别对应列联表的下一行两个数据。在这里，进行1次数据生成，得到的生成数据结果如下所示：

```
> n11
[1] 1
> n12
[1] 11
> n21
[1] 18
> n22
[1] 20
```

Figure 9: fisher精确检验自行生成的列联表数据

故此时我们得到的列联表如下所示：

Table 8: 一般列联表形式

ElementA	ElementB	ElementB	
Status	level1	level2	Total
Case	1	11	12
Control	18	20	38
Total	19	31	50

可以得到的p值如下所示：

```
> p_val
[1] 0.02882866
```

Figure 10: fisher精确检验对于模拟数据实验的p值

我们一般取显著水平$\alpha = 0.05$，故p值为0.0288¡0.5，因此我们可以拒绝原假设，认为两组数据的抽样概率不相等。这与我们设定的初始数据$p_1 = 0.1 \neq p_2 = 0.5$也是相符合的。

再检验$p_1 = p_2$的情形。这时对于两部分的数据，我们均采用$p_1 = p_2 = 0.4$进行抽样，将原来的两组数据给分散为列联表中对应的4个数据。此时的生成数据如下所示：

```
> n11
[1] 4
> n12
[1] 5
> n21
[1] 16
> n22
[1] 25
```

Figure 11: fisher精确检验自行生成的列联表数据

故此时我们得到的列联表如下所示：

Table 9: 一般列联表形式

ElementA	ElementB	ElementB	
Status	level1	level2	Total
Case	4	5	9
Control	16	25	41
Total	20	30	50

可以得到的p值如下所示：

```
> p_val
[1] 1
```

Figure 12: fisher精确检验对于模拟数据实验的p值

可以看到此时的p=1，要明显大于我们所给的显著水平$\alpha = 0.05$.故我们不拒绝原假设，认为两次抽样的概率$p_1 = p_2$.而事实上我们采用的抽样概率为$p_1 = p_2 = 0.4$，这显然是和实验结果相符。

2.2.3 具体实验2：作用于实际数据

此处采用的实际数据来源于生物统计基础教材中第388页的例子，该数据集主要描述了实验者食用盐含量的高低对于是否得心脏病的影响。具体的列联表如下所示：

Table 10: 一般列联表形式

ElementA	ElementB	ElementB	
Status	level1	level2	Total
Case	2	23	25
Control	5	30	35
Total	7	53	60

可以得到的p值如下所示：

```
> p_val
[1] 0.7493036
```

Figure 13: fisher精确检验对于实际数据实验的p值

可以看到此时的p=0.749，要明显大于我们所给的显著水平$\alpha = 0.05$.故我们不拒绝原假设，可以认为摄入盐含量的多少对于是否的心脏病的影响不大。

2.2.4 具体实验3：关于Fisher精确检验的评价——power function

类比于卡方检验时的情况，我们也用power function对检验方法进行评价。我们得出的势函数的横坐标仍旧对应检验中的两个参数的差值，而纵坐标为通过多次实验的结果模拟得出的拒绝次数除以总实验次数而估计出的势函数的值。对此，我们仍旧固定p_1为不同的值进行实验。

第一次实验：固定$p_1 = 0.5$的情况下，p_2的值从0开始每次增加0.01到1，共计完成100组对应的参数差值测试，每组测试中进行1000次测试，通过Fisher精确检验方法确定每组测试中拒绝原假设的次数n，再利用n/1000估计该组测试的对应势函数的差值的势函数。此时我们可以得到的势函数如下所示：

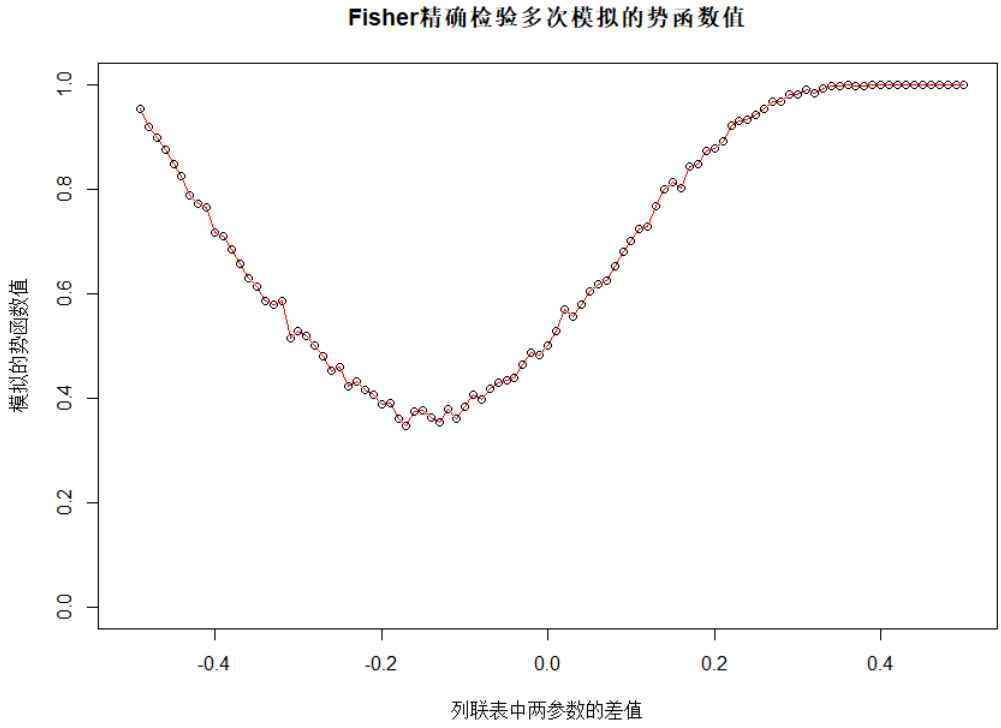

Figure 14: 卡方检验势函数评价$p_1 = 0.5$为基准

可以看到此时的势函数也同样呈现先向下凹陷再重回1的走势。当参数的差值较大时，势函数的值在1附近波动，这和参数差值较大时拒绝原假设的结论相符。而当参数差值较小时，势函数的值会相应缩小，表面此时不需要过分拒绝原假设。但是值得一提的时，此时做出的势函数图像仍旧有两个较大的缺陷：

首先，势函数的最小值点并不是在差值$p_2 - p_1 = 0$处取到的，而是存在了一定程度上的偏移。这和我们预期的并不是特别相符。

第二，势函数的最小值并不是十分接近于0.理论上，对于检验方法，当二者的参数差值十分小时，拒绝原假设的次数应该也十分少甚至于接近于0.但在此时的势函数中，模拟的势函数最小值并不是特别接近于0，而是在0.4左右。

个人认为会出现以上两大缺陷的原因主要还是在于Fisher精确检验应对的主要是小样本的情形，而小样本的情形受制于样本量本身的关系，在抽样时会出现各种各样的些微的差异，导致了拒绝原假设的情况并没有特别理想。因而此时的势函数图像做出来也没有特别地与预想情况相符。

但不能否认的是，这是由于小样本检验这一情形的本质情况导致的。在应对小样本检验的类别数据检验时，Fisher精确检验还是十分值得一用的一个方法，因为统计量的计算并不复杂且判断方式简单。

第二次实验：固定$p_1 = 0$的情况下，p_2的值从0开始每次增加0.01到1，共计完成100组对应的参数差值测试，每组测试中进行1000次测试，通过Fisher精确检验方法确定每组测试中拒绝原假设的次数n，再利用n/1000估计该组测试的对应势函数的差值的势函

数。此时我们可以得到的势函数如下所示：

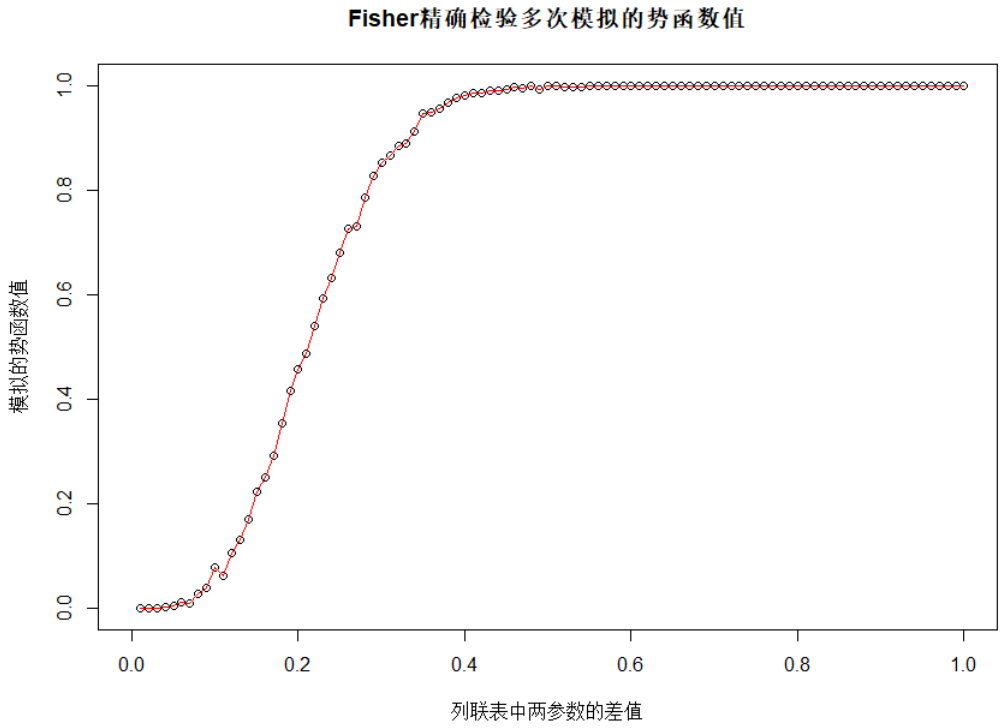

Figure 15: 卡方检验势函数评价$p_1 = 0$为基准

这时的势函数呈明显的单调上升趋势，这与这一情形下我们的预想情况是相符合的。在差值为0时的势函数值也近乎接近于0，这说明当差值不大时进行1000次实验完全接受了原假设。而随着两组数据的参数差值的增大，对应的势函数值也相应增大。而当两个参数差值为0.4左右时，势函数值就接近于1了，说明此时的1000次实验近乎是完全拒绝了原假设。这时的势函数能够很好地描述参数差值与检验中拒绝实验次数的关系。这一次实验得到的结果说明Fisher精确检验对于小样本数据的检验是合适的。

3 Task2-Two sample dependent test(Mcnemar test)

和TASK1中不同，此时探讨的是两样本的类别数据检验在不独立的情况下。此时我们主要采用Mcnemar Test的检验方法。

3.1 原理解释

两样本的类别数据在独立的情形下，主要用于检验数据对是否协调的情况。例如对于两种不同的药剂比较治疗情况，作用于多对病人从而比较病人的后续情况。此时由于病人的情况是成对出现的，因此这时的两样本之间不存在独立的情况，不能采用两样本独立性检验方法。因而引入Mcnemar Test.

仍旧以两种特效药对于两对病人的治疗效果为例。此时我们再引入协调对与不协调对的概念。协调对指对于一对病人采取两种特效药效果一样好的实验对，而不协调对为对于一对病人采取两种特效药效果不一样的实验对。而A类型不协调对为A药出现而B药未出现情况的实验对，B类型的依此类推。我们设定p为不协调对为A类型不协调对的概率。因而当p=0.5时，我们可以认为两种药的药效差不多；当p¡0.5时，我们认为A药的药效要好于B药的药效，p¿0.5时，对应地，B药的药效要好于A药的。因此此时的假设检验为

$$H_0 : p = \frac{1}{2} \qquad vs \qquad H_1 : p \neq \frac{1}{2} \tag{15}$$

我们用n_D表示总共的不协调对个数，n_A表示A类不协调对个数。在原假设H_0我们可以求出对应的不协调对期望和方差为$E(n_A) = \frac{n_D}{2}$，$Var(n_A) = \frac{n_D}{4}$.(此处类比二项分布的期望为np,方差为np(1-p)进行求解)。

因此我们可以构造统计量

$$X^2 = \frac{(|n_A - \frac{n_D}{2}| - 0.5)^2}{\frac{n_D}{4}} = \frac{(|n_A - n_B| - 1)^2}{n_A + n_B} \tag{16}$$

此时的X^2是服从自由度为1的卡方分布(来源于正态分布，故也称正态理论检验)，因而对于显著水平α，有以下检验结果：

$$\text{拒绝}H_0 : X^2 > \chi^2_{1,1-\alpha} \qquad \text{接受}H_0 : X^2 < \chi^2_{1,1-\alpha} \tag{17}$$

此时的p值为$p - value = P(X^2 \leq \chi^2_{1,1-\alpha})$

这主要是对于$Var(n_A) = \frac{n_D}{5} \geq 5$即$n_D \geq 20$的情况。否则应采用精确检验，此时的p值计算公式如下：

condition1:

$$n_A < \frac{n_D}{2} : p = 2\sum_{k=0}^{n_A} C^k_{n_D}(\frac{1}{2})^{n_D} \tag{18}$$

condition2:

$$n_A > \frac{n_D}{2} : p = 2\sum_{k=n_A}^{n_D} C^k_{n_D}(\frac{1}{2})^{n_D} \tag{19}$$

condition3:

$$n_A = \frac{n_D}{2} : p = 1 \tag{20}$$

以此判断是否拒绝检验。

3.2 具体实验1：实际数据(正态理论检验)

此时采用的数据集来自于生物统计教材第396页，该数据集主要描述了两种不同的药A药与B药对于621对病人的治疗情况，以此比较两种药的药效结果。相关的实验数据列联表如下所示：

Table 11: mcnemar test数据列联表形式

Element	Outcome	Outcome	
Treatment	Survive for 5 years	Die within 5 years	Total
A	526	95	621
B	515	106	621
Total	1041	201	1242

将这一数据细化到不协调对的情况，结果如下所示：

Table 12: mcnemar test不协调对列联表形式

Element	OutcomeB	OutcomeB	
OutcomeA	Survive for 5 years	Die within 5 years	Total
Survive for 5 years	510	16	526
Die within 5 years	5	90	95
Total	515	106	621

采用Mcnemar Test获得的结果如下所示：

```
> X_2
[1] 4.761905
> quanchi
[1] 3.841459
```

Figure 16: Mcnemar Test检验结果

可以看到此时的检验统计量大于分位数，故我们应当拒绝原假设，认为两种药的药效不一样。

3.3 具体实验2：实际数据(精确检验)

此时采用的数据集来自于生物统计教材第401页，该数据主要描述了计算机与专家判断病人病情的差异情况，以此比较计算机与专家判断病人病情的结论是否相同。相关的实验数据列联表如下所示：

Table 13: mcnemar test数据列联表形式

Element	Trained observer	Trained observer	
Computer	+	-	Total
+	10	10	20
-	4	16	20
Total	14	26	40

将这一数据细化到不协调对的情况，结果如下所示：

Table 14: mcnemar test不协调对列联表形式

Element	Trained observer	Trained observer	
Computer	+	-	Total
+	3	7	10
-	1	9	10
Total	4	16	20

采用Mcnemar Test求得p值结果如下所示：

```
> p_val
[1] 0.0703125
```

Figure 17: Mcnemar Test检验结果

发现此时的p值为0.07略大于显著水平$\alpha = 0.05$，故我们认为不拒绝原假设，即两种检验病人的方式并没有显著的差异。

3.4 检验方法评价

Mcnemar Test作为一种作用于两样本类别数据的非独立性的检验，在一定情况下可以反映数据背后蕴含的信息，进而提供出有效的结果。但正如之前在卡方检验中也提到的问题，对于小样本的情况想要达到好的效果是困难的，尽管精确检验已经是比较好的方法。因此在这方面，这一检验方法还有改进的余地。

4 Task3——非参方法(Wilcoxon rank sum test)

Wilcoxon秩和检验是一种常见的非参方法。非参方法主要应用于总体分布假定较弱的前提下，进行统计推断。其中的一大典型方法是Wilcoxon秩和检验，可以对应于统计检验中的两样本的情况。

4.1 原理解释

由于给出的两样本中我们无法给出具体的分布，因而我们需要采用秩的方法去构造统计量。对于给出的数据，我们对数据进行从小到大的排序，进而进行秩的计算。对于不同水平下的数据集，我们都可以给出秩的排序，进而求出一个平均的秩值。

接着，对于两样本中的一组数据，基于其所计算出的总体数据的秩的值，我们采用在一个水平下的样本量*对应的秩的方法计算出这一组数据的秩和为R_1.我们不妨令第一

组数据的个数为n_1,第二组数据的个数为n_2，进而有$n_1+n_2=n$为总数据的个数。对于计算出的秩和R_1与R_2，应当满足$R_1+R_2=\frac{n(n+1)}{2}$.

而对于秩和R_1，可以得到秩和的期望值和方差分别为

$$E(R_1)=\mu=\frac{n_1(n_1+n_2+1)}{2} \qquad Var(R_1)=\sigma^2=\frac{n_1n_2(n_1+n_2+1)}{12} \tag{21}$$

由此，我们可以构造统计量T(此处仅考虑数据不打结的情形)：

condition1:当秩和$R_1\neq\frac{n_1(n_1+n_2+1)}{2}$时，有

$$T=\frac{|R_1-\frac{n_1(n_1+n_2+1)}{2}|-0.5}{\sqrt{(\frac{n_1n_2}{12})(n_1+n_2+1)}} \tag{22}$$

condition2:当秩和$R_1=\frac{n_1(n_1+n_2+1)}{2}$时，有$T=0$.

由此，对于数据量较大的情况，可以假定统计量T服从标准正态分布Z。进而对于显著水平α，有

$$\text{拒绝}H_0:T>Z_{1-\frac{\alpha}{2}} \qquad \text{接受}H_0:T<Z_{1-\frac{\alpha}{2}} \tag{23}$$

或者采用求解p值的方法

$$p=2[1-\Phi(T)] \tag{24}$$

与显著水平进行比较。

注：由于是来自于正态近似分布的检验，要求$n_1\geq 10, n_2\geq 10$才能采用这样的检验方法。否则需要列出所有检验可能性进行精确检验。

4.2 具体实验1：作用于模拟数据

在这里，我自行生成了55个数据，并且采用二项分布抽样的方式，令$p=0.45$将数据集分为2类。接着对于每一类数据，我将数据分成8个水平，每个水平都有一些实验数据值，具体如下所示：

```
> res1 <- as.vector(groupdivide(part1,8))
[1] 19 21
[1]  3 31  1  7 23 26  6
[1] 28 13  5 14  2
[1] 8
[1] 15 16  9 17 18 32
[1] 29 24 12 11 22
[1] 25  4 27 10 30
[1] 2
integer(0)
> res2 <- as.vector(groupdivide(part2,8))
[1] 49 54 43 34 47 48 41
[1] 36
[1] 40 39 50 55 52
[1] 33 46 37 38 44 53 51 45
[1] 42
[1] 27
numeric(0)
```

(a) 自行模拟生成的不同水平数据

```
> res
[1]  9  8 10  9  7  6  5  1
> res1
[1] 2 7 5 1 6 5 5 1
> res2
[1] 7 1 5 8 1 1 0 0
```

(b) 自行模拟生成的数据个数统计

Figure 18: wilcoxon秩和检验模拟数据

由此可得实际的不同水平下的数据个数统计以及秩的计算值如下表所示：

Table 15: wilcoxon秩和检验自行生成数据结果

Level	Group1	Group2	Combined Sample	Range of Ranks	Average Rank
level 1	2	7	9	1-9	5
level 2	7	1	8	10-17	13.5
level 3	5	5	10	18-27	22.5
level 4	1	8	9	28-36	32
level 5	6	1	7	37-43	40
level 6	5	1	6	44-49	46.5
level 7	5	0	5	50-54	52
level 8	1	0	0	55	55
Total	32	23	55		

进而通过wilcoxon秩和检验得到的检验结果如下所示：

```
> wilcox
[1] 2.388833
> quannorm
[1] 1.959964
> p_val
[1] 0.016902
```

Figure 19: wilcoxon秩和检验检验结果

从检验结果中可以得到，此时的统计量值为2.389¿对应的正态分位数1.96，同时p值为0.017小于显著水平0.05，因此应当拒绝原假设，认为这两组数据之间存在着水平上的差异。

4.3 具体实验2：作用于模拟正态分布比较wilcoxon秩和检验与t检验的效果

在这一实验中，我比较了对于模拟出来的两组正态分布数据，分别采用wilcoxon秩和检验与t检验的情况下的检验结果，进而比较两种检验方法。在这里，我分别生成了标准正态分布与均值为1，方差为1的两组正态分布数据，每组数据有20个。此时的具体检验结果如下所示：

(a) 采用wilcoxon秩和检验结果

(b) 采用t检验结果

Figure 20: wilcoxon秩和检验与t检验结果比较

我们可以看出采用两种方法对于这样的两组正态分布数据都是显著拒绝原假设，认为这两组数据来自截然不同的两个分布的，因此从这里我们很难将两种方法进行比较。我们还是需要借助势函数来比较两种检验方法。

4.4 具体实验3：通过power function比较wilcoxon秩和检验与t检验的效果

在这里仍旧和之前平均卡方检验和Fisher精确检验时的方法一样，借助power function的手段。在这里，我仍旧取两组正态分布数据的中心位置(均值)的差为power function函数的横坐标值，而用每次抽样时的数据进行1000次实验，1000次中的拒绝次数记为n，用n/1000的值去估计势函数的值表现在函数上。在这里，我都采用标准正态分布为第一组数据的固定分布，而第二组数据的分布的均值在区间(-1,1)之间进行变动，故函数的横坐标也是在(-1,1)之间进行变动。对于wilcoxon秩和检验与t检验实验，可得结果如下所示(红线为wilcoxon秩和检验的势函数，蓝线为t检验的势函数)：

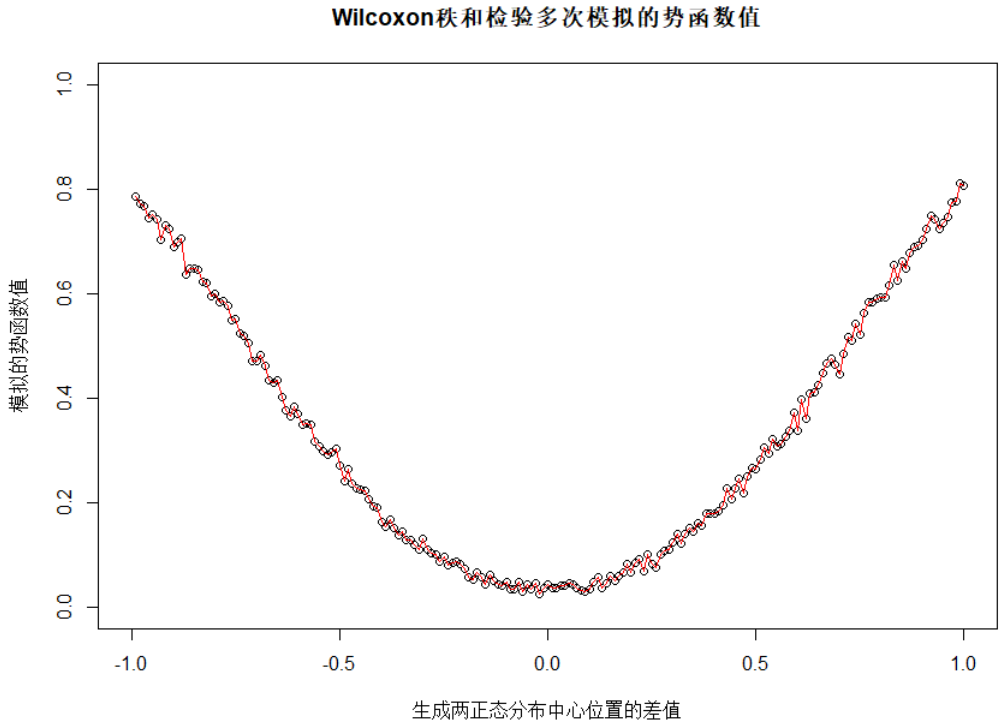

Figure 21: wilcoxon秩和检验势函数

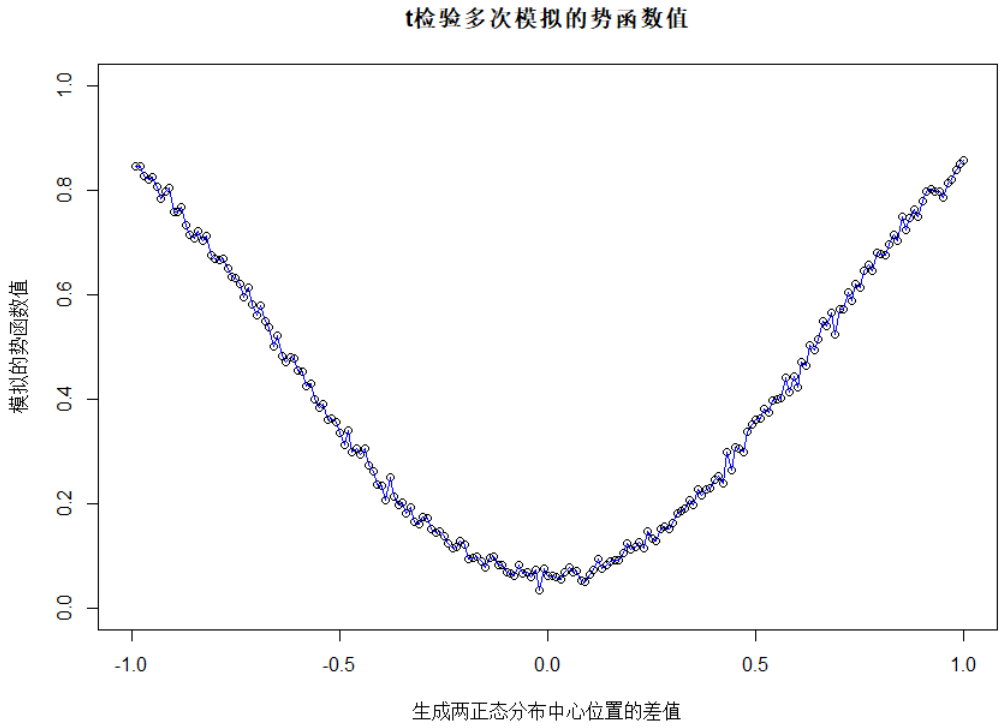

Figure 22: t检验势函数

将两张图合成进行比较，结果如下所示：

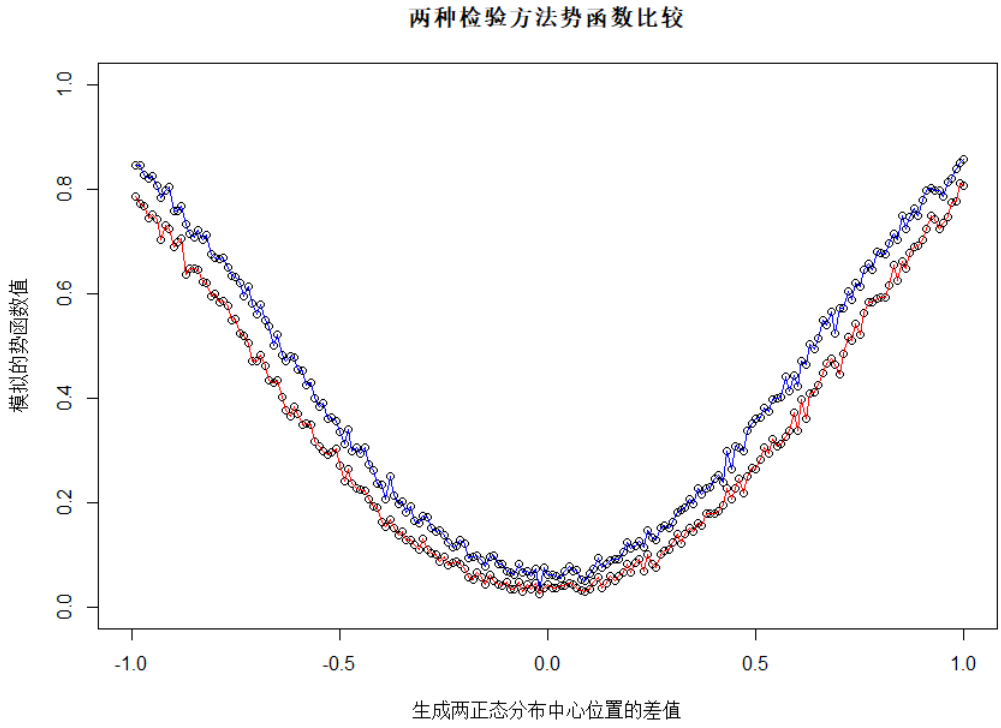

Figure 23: wilcoxon秩和检验与t检验势函数比较

通过观察可以发现，两条势函数的分布都是对称的，且都在两个正态分布的均值差为0时势函数的对应值接近于0，说明此时应当接受原假设，认为两组数据来自同一个分布。而在均值差较大时，势函数的值也对应增大，说明此时1000组数据中拒绝原假设的实验次数增多。从这点上来说，两个检验的势函数图像都是合理的，与预想的情况相符。

值得注意的是，t分布的势函数几乎完全位于wilcoxon秩和检验的势函数的上方。这样可以说明以下两点：

第一，在中心位置的差值为0时，wilcoxon秩和检验的势函数更为接近于0，说明在中心位置的差值较小时，采用wilcoxon秩和检验的效果要略好于t检验的效果，此时采用wilcoxon秩和检验接受原假设的程度更大。

第二，在中心位置的差值较大时，t检验的势函数要大于wilcoxon秩和检验的势函数。这说明在中心位置的差值较大时，采用t检验的效果要好于wilcoxon秩和检验的效果，因为此时采用t检验拒绝原假设的程度更大，t检验的判断更为准确。

总之wilcoxon秩和检验和t检验各有优劣。二者都是一种大样本的检验方法，在思想上有着类似之处。我们应当随机应变，对于总体分布更为明显的情况，采用t检验方法，而对于总体分布假定较弱的情况下，采用wilcoxon秩和检验的方法。

4.5 具体实验4：对于大样本的情况采用置换检验(permutation test)

4.5.1 原理解释

置换检验是用于检验两组数据X和Y的累积概率分布函数F_X与F_Y是否相同的检验方法。对应的假设检验为

$$H_0 : F_X = F_Y \qquad vs \qquad H_1 : F_Y(x) = F_Y(x-\delta)(\delta \neq 0) \tag{25}$$

置换检验的思想在于列出所有可能的分布情况，进而计算出检验所需要的精确的p值。对于已经给出的数据，仍旧进行排序，计算出对应的秩。对于观察值和整个数据集的都计算秩和，分别记为$R_{1,obs}$与$R_{1,perm}$.故我们可以计算出p值为

$$p_value_{perm} = 2min[Pr(R_{1,perm} \leq R_{1,obs}), Pr(R_{1,perm} \geq R_{1,obs}), 0.5] \tag{26}$$

从而进行是否拒绝原假设的判断。

4.5.2 permutation test作用于实际数据

在这里，我选择采用实际数据进行实验，数据来源于生物统计教材第371页练习9.75.该数据集主要用于高加索人种与非裔美国人两大人种的Plasma aldosterone这一指标是否存在差异。具体的数据表如下所示：

Table 16: permutation test具体数据表

Aldosterone Group	Caucasian	African American
0-199	12	28
200-399	17	10
400-599	5	5
≥ 600	9	3

在这里，我对于每一个水平都进行了对应数据的模拟，并且对于所有的排列情况抽样了100000次进行模拟进而求出p值。在这个问题中，要试图列出所有的排列情况显然是不现实的，因为如果列出所有排列情况需要C_{99}^{53}次，这一数字已经达到$3 * 10^{28}$，这需要庞大的计算开销，因此我们抽取100000次排列进行模拟，求出一个近似的p值结果如下所示：

```
> p_value
[1] 2e-05
```

Figure 24: 置换检验的p值

可以看出此时的p值为$2 * 10^{-5}$，故我们应当拒绝原假设，认为此时的两个人种的差别导致了他们在Plasma Aldosterone这一生物数据指标的差别。

从上述实验中，我们可以发现置换检验和wilcoxon秩和检验的思想类似，二者在特定情况下是可以进行替代的。

5 报告总结

这次编程作业主要探讨了几种数据的检验方式，本人将这几种检验方式作用于自行模拟生成的数据以及实际的生物数据，并且比较了这几种检验方式的效果，给出了检验方式的评价。在这里，我们需要对于不同的情况进行检验方法的随机应变。当数据量不大时，考虑采用精确检验的方法；在应用检验方法前，判断此时的数据是否是独立生成的；而对于总体分布假定不够明显的情况，考虑采用非参检验的方法。事实上，检验方法的发展也是与时俱进的，这也督促我们在统计方面寻找更为可靠的检验方法。

A 相关代码

生成两样本独立的列联表的方法：卡方检验作用于模拟数据

```
()
prow <- 0.3
qrow <- 1-prow
p1 <- 0.4
q1 <- 1-p1
p2 <- 0.6
q2 <- 1-p2
N <- 1000
res1 <- rep(0,N)
res1 <- as.vector(rbinom(N,1,prow))
m1 <- 0
m2 <- 0
for(i in 1:N){
        if(res1[i]==1)
        m1 <- m1+1
        else
        m2 <- m2+1
}
res1row <- rep(0,m1)
res1row <- as.vector(rbinom(m1,1,p1))
m11 <- 0
m12 <- 0
for(i in 1:m1){
        if(res1row[i]==1)
        m11 <- m11+1
        else
        m12 <- m12+1
}
res2row <- rep(0,m2)
res2row <- as.vector(rbinom(m2,1,p2))
m21 <- 0
m22 <- 0
for(i in 1:m2){
        if(res2row[i]==1)
```

```
            m21 <- m21+1
            else
            m22 <- m22+1
}
m11
m12
m21
m22

n1 <- m11+m21
n2 <- m12+m22
E11 <- (n1*m1)/N
E12 <- (n2*m1)/N
E21 <- (n1*m2)/N
E22 <- (n2*m2)/N
X_2 <- ((abs(m11-E11)-0.5)^2)/E11+((abs(m12-E12)
    -0.5)^2)/
E12+((abs(m21-E21)-0.5)^2)/E21+((abs(m22-E22)
    -0.5)^2)/E22
quanchi <- qchisq(0.95,1,ncp=0)卡方检验作用于实际数
    据:



n11 <- 29
n12 <- 7
n13 <- 7
n21 <- 13
n22 <- 7
n23 <- 21
n1 <- n11+n12+n13
n2 <- n21+n22+n23
m1 <- n11+n21
m2 <- n12+n22
m3 <- n13+n23
N <- n1+n2
```

```
E11 <- (m1*n1)/N
E12 <- (m2*n1)/N
E13 <- (m3*n1)/N
E21 <- (m1*n2)/N
E22 <- (m2*n2)/N
E23 <- (m3*n2)/N

X_2 <- ((abs(n11-E11)-0.5)^2)/E11+((abs(n12-E12)
    -0.5)^2)/E12+
((abs(n13-E13)-0.5)^2)/E13+((abs(n21-E21)-0.5)
    ^2)/E21+
((abs(n22-E22)-0.5)^2)/E22+((abs(n23-E23)-0.5)
    ^2)/E23
quanchi <- qchisq(0.95,2,ncp=0)生成实际列联表:
```

```
library(vcd)
head(Arthritis)

table(Arthritis$Treatment,Arthritis$Improved)
with(Arthritis,table(Treatment,Improved))
mytable <- xtabs(~Treatment+Improved,data =
    Arthritis)
with(Arthritis,xtabs(~Treatment+Improved,data =
    Arthritis))

margin.table(mytable,2) # sum by row
prop.table(mytable,2)  #proportion by column
prop.table(mytable)  #proportion by total

addmargins(mytable)
addmargins(mytable,1)
addmargins(prop.table(mytable,2),1)
```

```
library(gmodels)
CrossTable(Arthritis$Treatment, Arthritis$
    Improved) ##SAS format
```

fisher exact 作用于模拟数据test:

```
prow <- 0.2
qrow <- 1-prow
p1 <- 0.1
q1 <- 1-p1
p2 <- 0.2
q2 <- 1-p2
N <- 50
res1 <- rep(0,N)
res1 <- as.vector(rbinom(N,1,prow))
n1 <- 0
n2 <- 0
for(i in 1:N){
          if(res1[i]==1)
          n1 <- n1+1
          else
          n2 <- n2+1
}
res1row <- rep(0,n1)
res1row <- as.vector(rbinom(n1,1,p1))
n11 <- 0
n12 <- 0
for(i in 1:n1){
          if(res1row[i]==1)
          n11 <- n11+1
          else
          n12 <- n12+1
}
res2row <- rep(0,n2)
res2row <- as.vector(rbinom(n2,1,p2))
```

```
n21 <- 0
n22 <- 0
for(i in 1:n2){
        if(res2row[i]==1)
        n21 <- n21+1
        else
        n22 <- n22+1
}
n11
n12
n21
n22
m1 <- n11+n21
m2 <- n12+n22

nmax <- min(m1,m2,n1,n2)
minob <- min(n11,n12,n21,n22)
pr_a <- rep(0,nmax)
for(a in 0:nmax){
        pr_a[a+1] <- (factorial(n1)*factorial(n2
            )*factorial(m1)
        *factorial(m2))/(factorial(N)*factorial(
            a)*factorial(n1-a)
        *factorial(m1-a)*factorial(m2-n1+a))
}
p_low <- 0
p_high <- 0
for(i in 0:minob){
        p_low <- p_low+pr_a[i+1]
}
for(i in minob:nmax){
        p_high <- p_high+pr_a[i+1]
}
p_val <- 2*min(p_low,p_high,0.5)
p_val
```

```
Fisher exact 作用于真实数据：test
n11 <- 2
n12 <- 23
n21 <- 5
n22 <- 30
m1 <- n11+n21
m2 <- n12+n22
n1 <- n11+n12
n2 <- n21+n22
N <- n11+n12+n21+n22

nmax <- min(m1,m2,n1,n2)
minob <- min(n11,n12,n21,n22)
pr_a <- rep(0,nmax)
for(a in 0:nmax){
        pr_a[a+1] <- (factorial(n1)*factorial(n2
            )*factorial(m1)
        *factorial(m2))/(factorial(N)*factorial(
            a)*factorial(n1-a)
        *factorial(m1-a)*factorial(m2-n1+a))
}
p_low <- 0
p_high <- 0
for(i in 0:minob){
        p_low <- p_low+pr_a[i+1]
}
for(i in minob:nmax){
        p_high <- p_high+pr_a[i+1]
}
p_val <- 2*min(p_low,p_high,0.5)
p_val
```

```
Mcnemar test:
N <- 500
p <- 0.95
res1 <- rep(0,N)
res2 <- rep(0,N)
res1 <- as.vector(rbinom(N,1,p))
res2 <- as.vector(rbinom(N,1,p))
m11 <- 0
m12 <- 0
m21 <- 0
m22 <- 0
for(i in 1:N){
        if(res1[i]==1)
        m11 <- m11+1
        else
        m12 <- m12+1
}
for(i in 1:N){
        if(res2[i]==1)
        m21 <- m21+1
        else
        m22 <- m22+1
}
n1 <- m11
n2 <- m12
m1 <- m21
m2 <- m22
n11 <- 0
n12 <- 0
n21 <- 0
n22 <- 0
for(i in 1:N){
        if(res1[i]==1&&res2[i]==1)
        n11 <- n11+1
        else if(res1[i]==1&&res2[i]==0)
```

```
            n12 <- n12+1
            else if(res2[i]==1&&res1[i]==0)
            n21 <- n21+1
            else
            n22 <- n22+1
}
n_D <- n12+n21
n_A <- n21
n_B <- n12
p_vallow <- 0
p_valhigh <- 0
X_2 <- ((abs(n_A-n_B)-1)^2)/(n_A+n_B)
if(n_D<20)
{
        if(n_A<0.5*n_D)
        {
                for(k in 0:n_A)
                {
                        p_vallow <- p_vallow+
                            choose(n_D,k)*((0.5)
                            ^(n_D))
                }
                p_val <- 2*p_vallow
        }
        else if(n_A>0.5*n_D)
        {
                for(k in n_A:n_D)
                {
                        p_valhigh <- p_valhigh+
                            choose(n_D,k)*((0.5)
                            ^(n_D))
                }
                p_val <- 2*p_valhigh
        }
        else
        p_val <- 1
}
```

```
quanchi <- qchisq(0.95,1,ncp=0)
```

Mcnemar 作用于真实数据：Test1

```
m11 <- 526
m12 <- 95
m21 <- 515
m22 <- 106
n11 <- 510
n12 <- 16
n21 <- 5
n22 <- 90
n_D <- n12+n21
n_A <- n21
n_B <- n12
p_vallow <- 0
p_valhigh <- 0
X_2 <- ((abs(n_A-n_B)-1)^2)/(n_A+n_B)
if(n_D<20)
{
        if(n_A<0.5*n_D)
        {
                for(k in 0:n_A)
                {
                        p_vallow <- p_vallow+
                            choose(n_D,k)*((0.5)
                            ^(n_D))
                }
                p_val <- 2*p_vallow
        }
        else if(n_A>0.5*n_D)
        {
                for(k in n_A:n_D)
```

```
                {
                        p_valhigh <- p_valhigh+
                            choose(n_D,k)*((0.5)
                            ^(n_D))
                }
                p_val <- 2*p_valhigh
        }
        else
        p_val <- 1
}

quanchi <- qchisq(0.95,1,ncp=0)
```

Mcnemar 作用于真实数据：（精确检验）Test2

```
m11 <- 10
m12 <- 10
m21 <- 4
m22 <- 16
n11 <- 3
n12 <- 7
n21 <- 1
n22 <- 9
n_D <- n12+n21
n_A <- n21
n_B <- n12
p_vallow <- 0
p_valhigh <- 0
X_2 <- ((abs(n_A-n_B)-1)^2)/(n_A+n_B)
if(n_D<20)
{
        if(n_A<0.5*n_D)
        {
                for(k in 0:n_A)
```

```
                {
                        p_vallow <- p_vallow+
                            choose(n_D,k)*((0.5)
                            ^(n_D))
                }
                p_val <- 2*p_vallow
        }
        else if(n_A>0.5*n_D)
        {
                for(k in n_A:n_D)
                {
                        p_valhigh <- p_valhigh+
                            choose(n_D,k)*((0.5)
                            ^(n_D))
                }
                p_val <- 2*p_valhigh
        }
        else
        p_val <- 1
}
```

quanchi <- **qchisq**(0.95,1,ncp=0)秩和检验作用于模拟数据():

```
Wilcoxon1
Total <- c(1:55)
p_div <- 0.45#数据分为两类
divi <- as.vector(rbinom(length(Total),1,p_div))
n1 <- 0
n2 <- 0
for(i in 1:length(divi)){
        if(divi[i]==1)
        n1 <- n1+1
```

```
        else
        n2 <- n2+1
}
part1 <- c(1:n1)
part2 <- c(n1+1:n2)
n <- n1+n2#每一类数据进行计数

groupdivide <- function(arrays,n){#模拟数据的生成
        k=1  #这里是为了计数引入的k
        numof <- rep(0,n)
        while (k<=n&&length(arrays)>0){
                if(length(arrays)>=8)
                m <- ceiling(runif(1,0,8))
                else
                m <- ceiling(runif(1,0,length(
                    arrays)))

                numof[k] <- m

                fz <- sample(arrays,m,replace =
                    FALSE,prob = NULL)
                for (i in 1:m) {
                        arrays <- arrays[-which(
                            arrays==fz[i])]
                        #将抽中的从原列表中删除
                }
                k=k+1
                print(fz)
        }
        print(arrays)
        return(numof)
}
res1 <- as.vector(groupdivide(part1,8))
res2 <- as.vector(groupdivide(part2,8))
res <- res1+res2
rankst <- rep(0,length(res))#计算秩的开始
rankst[1] <- 1
```

```
for(i in 2:length(rankst)){
        if(res[i]!=0)
        rankst[i] <-sum(res[1:i-1])+1
        else
        rankst[i] <- 0
}
rankend <- rep(0,length(res))#计算秩的末尾
for(i in 1:length(rankend)){
        if(res[i]!=0)
        rankend[i] <- sum(res[1:i])
        else
        rankend[i] <- 0
}
Rank <- rep(0,length(rankst))#计算秩向量
for(i in 1:length(Rank)){
        Rank[i] <- (rankst[i]+rankend[i])/2
}
R1 <- 0
for(i in 1:length(res1)){
        R1 <- R1+res1[i]*Rank[i]
}#求秩和
mu <- (n1*(n+1))/2
sigma <- sqrt((n1*n2*(n+1))/12)
wilcox <- (abs(R1-mu)-0.5)/sigma#计算统计量
p_val <- 2*(1-pnorm(wilcox,0,1))
alpha <- 0.05
quannorm <- qnorm(1-alpha/2,0,1)秩和与检验作用于模
   拟数据（）（模拟正态分布情形）：
```

```
Wilcoxont2
n1 <- 20
n2 <- 20
n <- n1+n2
vec1 <- as.vector(rnorm(n1,0,1))
```

```
vec2 <- as.vector(rnorm(n2,1,1))
m11 <- 0
m12 <- 0
m13 <- 0
m14 <- 0
m15 <- 0
m16 <- 0
m17 <- 0
m21 <- 0
m22 <- 0
m23 <- 0
m24 <- 0
m25 <- 0
m26 <- 0
m27 <- 0
for(i in 1:length(vec1)){
        if(vec1[i]>=-3 && vec1[i]< -2)
        m11 <- m11+1
        else if(vec1[i]>=-2 && vec1[i]< -1)
        m12 <- m12+1
        else if(vec1[i]>=-1 && vec1[i]<0)
        m13 <- m13+1
        else if(vec1[i]>=0 && vec1[i]<1)
        m14 <- m14+1
        else if(vec1[i]>=1 && vec1[i]<2)
        m15 <- m15+1
        else if(vec1[i]>=2 && vec1[i]<3)
        m16 <- m16+1
        else
        m17 <- m17+1
        if(vec2[i]>=-3 && vec2[i]< -2)
        m21 <- m21+1
        else if(vec2[i]>=-2 && vec2[i]< -1)
        m22 <- m22+1
        else if(vec2[i]>=-1 && vec2[i]<0)
        m23 <- m23+1
        else if(vec2[i]>=0 && vec2[i]<1)
```

```
            m24 <- m24+1
            else if(vec2[i]>=1 && vec2[i]<2)
            m25 <- m25+1
            else if(vec2[i]>=2 && vec2[i]<3)
            m26 <- m26+1
            else
            m27 <- m27+1
}
res1 <- as.vector(c(m11,m12,m13,m14,m15,m16,m17)
    )
res2 <- as.vector(c(m21,m22,m23,m24,m25,m26,m27)
    )
res <- res1+res2
rankst <- rep(0,length(res))#计算秩的开始
rankst[1] <- 1
for(i in 2:length(rankst)){
            if(res[i]!=0)
            rankst[i] <-sum(res[1:i-1])+1
            else
            rankst[i] <- 0
}
rankend <- rep(0,length(res))#计算秩的末尾
for(i in 1:length(rankend)){
            if(res[i]!=0)
            rankend[i] <- sum(res[1:i])
            else
            rankend[i] <- 0
}
Rank <- rep(0,length(rankst))#计算秩向量
for(i in 1:length(Rank)){
            Rank[i] <- (rankst[i]+rankend[i])/2
}
R1 <- 0
for(i in 1:length(res1)){
            R1 <- R1+res1[i]*Rank[i]
}#求秩和
mu <- (n1*(n+1))/2
```

```
sigma <- sqrt((n1*n2*(n+1))/12)
wilcox <- (abs(R1-mu)-0.5)/sigma#计算统计量
p_val <- 2*(1-pnorm(wilcox,0,1))
alpha <- 0.05
quannorm <- qnorm(1-alpha/2,0,1)

ave1 <- mean(vec1)
ave2 <- mean(vec2)
s_x1 <- 0
s_x2 <- 0
for(i in 1:length(vec1)){
        s_x1 <- s_x1+(vec1[i]-ave1)^2
        s_x2 <- s_x2+(vec2[i]-ave2)^2
}
s_w <- (s_x1+s_x2)/(n-2)
t_sta <- (ave2-ave1)/(s_w*sqrt(1/n1+1/n2))
quan_t <- qt(1-alpha/2,n-2)
```

```
permutation test（教材）:9.75
cau_1 <- runif(12,1,199)
cau_2 <- runif(17,200,399)
cau_3 <- runif(15,400,599)
cau_4 <- runif(9,600,800)
aa_1 <- runif(28,1,199)
aa_2 <- runif(10,200,399)
aa_3 <- runif(5,400,599)
aa_4 <- runif(3,600,800)
tot_vec <- c(cau_1,cau_2,cau_3,cau_4,aa_1,aa_2,
   aa_3,aa_4)
rankpro <- rank(tot_vec)
r_stand <- sum(rankpro[1:53])
arch <- numeric(100000)
for(i in 1:100000){
```

```
        res <- sample(rankpro,53,replace = FALSE
            )
        arch[i] <- sum(res)
}
arch_low <- arch[arch<=r_stand]
arch_high <- arch[arch>=r_stand]
len_1 <- length(arch_low)
len_2 <- length(arch_high)
p_value <- 2*min(len_1,len_2,50000)/100000
p_value势函数:(
```

```
chi-square ) test
prow <- 0.5
qrow <- 1-prow
N <- 1000
tri <- 1000
res1 <- rep(0,N)
res1 <- as.vector(rbinom(N,tri,prow))
res2 <- as.vector(rep(tri,N)-res1)
Numofrej <- rep(0,100)
Numofacc <- rep(0,100)
for(i in 1:100){
        p1 <- 0.5
        q1 <- 1-p1
        p2 <- 0+0.01*i
        q2 <- 1-p2
        n11 <- as.vector(rbinom(N,res1,p1))
        n12 <- as.vector(res1-rbinom(N,res1,p1))
        n21 <- as.vector(rbinom(N,res2,p2))
        n22 <- as.vector(res2-rbinom(N,res2,p2))
        res3 <- as.vector(n11+n21)
        res4 <- as.vector(n12+n22)
        E11 <- as.vector((res1*res3)/N)
```

```
            E12 <- as.vector((res1*res4)/N)
            E21 <- as.vector((res2*res3)/N)
            E22 <- as.vector((res2*res4)/N)
            X_2 <- as.vector(((abs(n11-E11)-0.5)^2)/
                E11+
            ((abs(n12-E12)-0.5)^2)/E12+((abs(n21-E21
                )-0.5)^2)/E21
            +((abs(n22-E22)-0.5)^2)/E22)
            quanchi <- qchisq(0.95,1,ncp=0)
            for(j in 1:length(X_2)){
                    if(X_2[j]>quanchi)
                    Numofrej[i] <- Numofrej[i]+1
                    else
                    Numofacc[i] <- Numofacc[i]+1
            }
}
prej <- rep(0,100)
prej <- as.vector(Numofrej/1000)
x <- seq(from=-0.49,to=0.5,by=0.01)
plot(x,prej,xlim=c(-0.5,0.5),ylim=c(0,1),
main = "chi_检验多次模拟的势函数值square",
xlab="列联表中两参数的差值",ylab="模拟的势函数值")
lines(x,prej,lty=1,lwd=1,col="red")势函数（



:fisher exact ）test
prow <- 0.5
qrow <- 1-prow
N <- 50
tri <- 1000
res1 <- rep(0,tri)
res1 <- as.vector(rbinom(tri,N,prow))
res2 <- as.vector(rep(N,tri)-res1)
Numofrej <- rep(0,100)
```

```
Numofacc <- rep(0,100)
resmax <- rep(0,tri)
minob <- rep(0,tri)
p_lowr <- rep(0,tri)
p_highr <- rep(0,tri)
p_valr <- rep(0,tri)
Numofrej <- rep(0,100)
Numofacc <- rep(0,100)
for(i in 1:100){
        p1 <- 0
        q1 <- 1-p1
        p2 <- 0+0.01*i
        q2 <- 1-p2
        n11 <- as.vector(rbinom(tri,res1,p1))
        n12 <- as.vector(res1-n11)
        n21 <- as.vector(rbinom(tri,res2,p2))
        n22 <- as.vector(res2-n21)
        res3 <- as.vector(n11+n21)
        res4 <- as.vector(n12+n22)
        E11 <- as.vector((res1*res3)/N)
        E12 <- as.vector((res1*res4)/N)
        E21 <- as.vector((res2*res3)/N)
        E22 <- as.vector((res2*res4)/N)
        for(j in 1:tri){
                resmax[j] <- min(res1[j],res2[j
                    ],res3[j],res4[j])
                minob[j] <- min(n11[j],n12[j],
                    n21[j],n22[j])
        }
        for(j in 1:tri){
                pr_a <- rep(0,resmax[j]+1)
                for(a in 0:resmax[j]){
                        pr_a[a+1] <- (factorial(
                            res1[j])*factorial(
                            res2[j])*
                        factorial(res3[j])*
                            factorial(res4[j]))/
```

```
                        (factorial(N)*factorial(
                            a)*factorial(res1[j]-
                            a)*
                        factorial(res3[j]-a)*
                            factorial(res4[j]-
                            res1[j]+a))
                    }
                    for(a in 0:minob[j]){
                        p_lowr[j] <- p_lowr[j]+
                            pr_a[a+1]
                    }
                    for(a in minob[j]:resmax[j]){
                        p_highr[j] <- p_highr[j
                            ]+pr_a[a+1]
                    }
                    p_valr[j] <- 2*min(p_lowr[j],p_
                        highr[j],0.5)
                }
                for(j in 1:length(p_valr)){
                    if(is.na(p_valr[j])==TRUE)
                    p_valr[j] <- 0
                }
                for(j in 1:length(p_valr)){
                    if(p_valr[j]<0.05){
                        Numofrej[i] <- Numofrej[
                            i]+1
                    }
                    else{
                        Numofacc[i] <- Numofacc[
                            i]+1
                    }
                }
                p_lowr <- rep(0,tri)
                p_highr <- rep(0,tri)
                p_valr <- rep(0,tri)
            }
            prej <- rep(0,100)
```

```
prej <- as.vector(Numofrej/tri)
x <- seq(from=-0.49,to=0.5,by=0.01)
plot(x,prej,xlim=c(-0.5,0.5),ylim=c(0,1),main =
"精确检验多次模拟的势函数值Fisher",xlab="列联表中两参
    数的差值",
ylab="模拟的势函数值")
lines(x,prej,lty=1,lwd=1,col="red")秩和与检验势函
    数比较:
```

```
Wilcoxont
n1 <- 20
n2 <- 20
n <- n1+n2
tri <- 1000
vec1 <- matrix(0,tri,n1)
vec2 <- matrix(0,tri,n2)
m11 <- rep(0,tri)
m12 <- rep(0,tri)
m13 <- rep(0,tri)
m14 <- rep(0,tri)
m15 <- rep(0,tri)
m16 <- rep(0,tri)
m17 <- rep(0,tri)
m18 <- rep(0,tri)
m21 <- rep(0,tri)
m22 <- rep(0,tri)
m23 <- rep(0,tri)
m24 <- rep(0,tri)
m25 <- rep(0,tri)
m26 <- rep(0,tri)
m27 <- rep(0,tri)
m28 <- rep(0,tri)
res1 <- matrix(0,tri,8)
```

```
res2 <- matrix(0,tri,8)
res <- matrix(0,tri,8)
rankst <- matrix(0,tri,8)#计算秩的开始
rankend <- matrix(0,tri,8)#计算秩的末尾
Rank <- matrix(0,tri,8)#计算秩向量
R1 <- rep(0,tri)
wilcox <- rep(0,tri)
p_val <- rep(0,tri)
Numofrej <- rep(0,200)
Numofacc <- rep(0,200)
prej <- rep(0,200)
ave1 <- rep(0,tri)
ave2 <- rep(0,tri)
s_x1 <- rep(0,tri)
s_x2 <- rep(0,tri)
s_w <- rep(0,tri)
t_sta <- rep(0,tri)
Numoftrej <- rep(0,200)
trej <- rep(0,200)
for(i in 1:200){
        for(j in 1:tri){
                vec1[j,] <- as.vector(rnorm(n1
                   ,0,1))
                vec2[j,] <- as.vector(rnorm(n2
                   ,-1+0.01*i,1))
        }
        for(j in 1:tri){
                for(k in 1:ncol(vec1)){
                        if(vec1[j,k]>=-4 && vec1
                           [j,k]< -3)
                        m11[j] <- m11[j]+1
                        else if(vec1[j,k]>=-3 &&
                             vec1[j,k]< -2)
                        m12[j] <- m12[j]+1
                        else if(vec1[j,k]>=-2 &&
                             vec1[j,k]< -1)
                        m13[j] <- m13[j]+1
```

```
else if(vec1[j,k]>=-1 &&
    vec1[j,k]<0)
m14[j] <- m14[j]+1
else if(vec1[j,k]>=0 &&
   vec1[j,k]<1)
m15[j] <- m15[j]+1
else if(vec1[j,k]>=1 &&
   vec1[j,k]<2)
m16[j] <- m16[j]+1
else if(vec1[j,k]>=2 &&
   vec1[j,k]<3)
m17[j] <- m17[j]+1
else
m18[j] <- m18[j]+1
if(vec2[j,k]>=-4 && vec2
   [j,k]< -3)
m21[j] <- m21[j]+1
else if(vec2[j,k]>=-3 &&
    vec2[j,k]< -2)
m22[j] <- m22[j]+1
else if(vec2[j,k]>=-2 &&
    vec2[j,k]< -1)
m23[j] <- m23[j]+1
else if(vec2[j,k]>=-1 &&
    vec2[j,k]<0)
m24[j] <- m24[j]+1
else if(vec2[j,k]>=0 &&
   vec2[j,k]<1)
m25[j] <- m25[j]+1
else if(vec2[j,k]>=1 &&
   vec2[j,k]<2)
m26[j] <- m26[j]+1
else if(vec2[j,k]>=2 &&
   vec2[j,k]<3)
m27[j] <- m27[j]+1
else
m28[j] <- m28[j]+1
```

```
        }
    }
    for(j in 1:tri){
        res1[j,] <- as.vector(c(m11[j],
            m12[j],m13[j],m14[j],m15[j]
        ,m16[j],m17[j],m18[j]))
        res2[j,] <- as.vector(c(m21[j],
            m22[j],m23[j],m24[j],m25[j]
        ,m26[j],m27[j],m28[j]))
        res[j,] <- res1[j,]+res2[j,]
        rankst[j,1] <- 1
        for(k in 2:ncol(res)){
            if(res[j,k]!=0)
            rankst[j,k] <-sum(res[j
                ,1:k-1])+1
            else
            rankst[j,k] <- 0
        }
        for(k in 1:ncol(res)){
            if(res[j,k]!=0)
            rankend[j,k] <- sum(res[
                j,1:k])
            else
            rankend[j,k] <- 0
        }
        for(k in 1:ncol(Rank)){
            Rank[j,k] <- (rankst[j,k
                ]+rankend[j,k])/2
        }
        for(k in 1:ncol(res1)){
            R1[j] <- R1[j]+res1[j,k]
                *Rank[j,k]
        }#求秩和
        mu <- (n1*(n+1))/2
        sigma <- sqrt((n1*n2*(n+1))/12)
        wilcox[j] <- (abs(R1[j]-mu)-0.5)
            /sigma#计算统计
```

```
            量
        p_val[j] <- 2*(1-pnorm(wilcox[j
            ],0,1))
}
alpha <- 0.05
quannorm <- qnorm(1-alpha/2,0,1)
for(j in 1:tri){
        if(p_val[j]<0.05){
                Numofrej[i] <- Numofrej[
                    i]+1
        }
        else{
                Numofacc[i] <- Numofacc[
                    i]+1
        }
}
m11 <- rep(0,tri)
m12 <- rep(0,tri)
m13 <- rep(0,tri)
m14 <- rep(0,tri)
m15 <- rep(0,tri)
m16 <- rep(0,tri)
m17 <- rep(0,tri)
m18 <- rep(0,tri)
m21 <- rep(0,tri)
m22 <- rep(0,tri)
m23 <- rep(0,tri)
m24 <- rep(0,tri)
m25 <- rep(0,tri)
m26 <- rep(0,tri)
m27 <- rep(0,tri)
m28 <- rep(0,tri)
R1 <- rep(0,tri)
wilcox <- rep(0,tri)
p_val <- rep(0,tri)

for(j in 1:tri){
```

```
                ave1[j] <- mean(vec1[j,])
                ave2[j] <- mean(vec2[j,])
                for(k in 1:ncol(vec1)){
                        s_x1[j] <- s_x1[j]+(vec1
                            [j,k]-ave1[j])^2
                        s_x2[j] <- s_x2[j]+(vec2
                            [j,k]-ave2[j])^2
                }
                s_w[j] <- (s_x1[j]+s_x2[j])/(n
                    -2)
                t_sta[j] <- (ave2[j]-ave1[j])/(s
                    _w[j]*sqrt(1/n1+1/n2))
        }
        quan_t <- qt(1-alpha/2,n-2)
        for(j in 1:tri){
                if(abs(t_sta[j])>quan_t){
                        Numoftrej[i] <-
                            Numoftrej[i]+1
                }
        }
        ave1 <- rep(0,tri)
        ave2 <- rep(0,tri)
        s_x1 <- rep(0,tri)
        s_x2 <- rep(0,tri)
        s_w <- rep(0,tri)
        t_sta <- rep(0,tri)
}
prej <- Numofrej/1000
trej <- Numoftrej/1000
x <- seq(from=-0.99,to=1,by=0.01)
plot(x,prej,xlim=c(-1,1),ylim=c(0,1),main = "秩和
    检验多次模拟的势函数值Wilcoxon",xlab="生成两正态分
    布中心位置的差值"
,ylab="模拟的势函数值")
lines(x,prej,lty=1,lwd=1,col="red")

xt <- seq(from=-0.99,to=1,by=0.01)
```

```
plot(xt,trej,xlim=c(-1,1),ylim=c(0,1),main = "检
    验多次模拟的势函数值t"
,xlab="生成两正态分布中心位置的差值",ylab="模拟的势函
    数值")
lines(xt,trej,lty=1,lwd=1,col="blue")

plot(x,prej,xlim=c(-1,1),ylim=c(0,1),main = "两种
    检验方法势函数比较",
xlab="生成两正态分布中心位置的差值",ylab="模拟的势函数
    值")
points(xt,trej)
lines(x,prej,lty=1,lwd=1,col="red")
lines(x,trej,lty=1,lwd=1,col="blue")
```