HW2021.10.26

李俊逸 19336073

目录

Readme

下面代码均可直接运行

```
rm(list = ls())
```

Table 1 results reproduced

考虑观察变量 x_i, y_i, 记 $z_i = y_i - x_1$ 考虑模型 $z_i = \theta + e_i$ 考虑假设检验 $H_0 : \theta = 0$,$H_1 : \theta \neq 0$, 采用 `Wilcoxon` 符号秩检验

考虑统计量

$$T^+ = \sum_{i=1}^{n} R_i \psi_i$$

其中 $\psi_i = \begin{cases} 1 & \text{if } z_i > 0 \\ 0 & \text{if } z_i < 0 \end{cases}$ 在零假设的情况下辖有

$$\begin{aligned} T^* &= \frac{T^+ - E\left(T^+\right)}{\sqrt{\operatorname{var}\left(T^+\right)}} \\ &= \frac{T^+ - n(n+1)/4}{\sqrt{n(n+1)(2n+1)/24}} \end{aligned} \rightarrow N(0,1)$$

其中

$$\begin{aligned} E\left(T^+\right) =& np_1 + n(n-1)p_2 \\ \operatorname{var}\left(T^+\right) =& np_1\left(1-p_1\right) + n(n-1)\left(p_1^2 - 4p_1p_2 + 3p_2 - 2p_2^2\right) \\ &+ n(n-1)(n-2)\left(p_3 + 4p_4 - 4p_2^2\right) \end{aligned}$$

$$
\begin{aligned}
p_1 &= P\left(z_1 > 0\right) \\
p_2 &= P\left(z_1 \geq |z_2|\right) \\
p_3 &= P\left(z_1 \geq |z_2|, z_1 \geq |z_3|\right) \\
p_4 &= P\left(z_1 \geq z_2 \geq |z_3|\right)
\end{aligned}
$$

其中 p_i 可以估计为

$$
\begin{aligned}
\hat{p}_1 &= \frac{1}{n}\sum_{i=1}^{n} I\left\{z_I > 0\right\} \\
\hat{p}_2 &= \frac{1}{n(n-1)}\sum_{i\neq j} I\left\{z_i \geq |z_j|\right\} \\
\hat{p}_3 &= \frac{1}{n(n-1)(n-2)}\sum_{i\neq j\neq k} I\left\{z_i \geq |z_j|, z_i \geq |z_k|\right\} \\
\hat{p}_4 &= \frac{1}{n(n-1)(n-2)}\sum_{i\neq j\neq k} I\left\{z_i \geq z_j \geq |z_k|\right\}
\end{aligned}
$$

原始法构造函数

```
p_1 <- function(z) {
    n <- length(z)
    p1 <- 1/n * sum(z > 0)
    return(p1)
}

p_2 <- function(z) {
    n <- length(z)
    count <- 0
    for (i in 1:n) {
        count <- sum(z[-i] >= abs(z[i])) + count  ## i != j
    }
    return(count/(n * (n - 1)))
}



## 貌似三重循环会比较慢，用空间换复杂度
p_3 <- function(z) {
    library(foreach)
    n <- length(z)
    temp <- matrix(FALSE, n, n)
    for (i in 1:n) {
        temp[i, ] = (z >= abs(z[i]))  ## 获取行 i 为 z >= abs(z[i])
    }
    count <- 0
    pb <- txtProgressBar(style = 3)
    t1 <- Sys.time()
    for (i in 1:n) {
        tempa <- temp[, i][-i]  ## 获取列，只需计算任一不相等且同为 T 的值
        for (j in 1:(n - 1)) {
```

```
            if (tempa[j] == TRUE) {
                count <- sum(tempa[-j] == TRUE) + count  ## 计算了两次
            }
        }
        setTxtProgressBar(pb, i/n)
    }
    print(Sys.time() - t1)
    count <- count/2  # 计算了两次
    p3 <- 1/(n * (n - 1) * (n - 2)) * count
    return(p3)
}



p_4 <- function(z) {
    n <- length(z)
    # temp <- matrix(FALSE,n,n)
    count <- 0
    for (i in 1:n) {
        temp <- which(z >= abs(z[i]))  ## 获取行 k 为 z >= abs(z[k])
        for (j in temp[which(temp != i)]) {
            if (i != j) {
                count <- sum(z[c(-i, -j)] >= z[j]) + count  # 计算 z_i>=z_j 的数量
            }
        }
    }
    p4 <- 1/(n * (n - 1) * (n - 2)) * count
    return(p4)
}
```

发现时间成本较大，重新构造函数

```
p <- function(z) {
    n <- length(z)
    p1 <- 1/n * sum(z > 0)
    p2 <- 0
    p3 <- 0
    p4 <- 0
    for (i in 1:n) {
        # p2 <- sum(z[-i]>=abs(z[i])) + p2 ## i != j
        temp <- sum(abs(z) <= z[i]) - 1
        p2 <- temp + p2
        ## i!=j!=k
```

```
        ## 相当于从 p2,i!=j 且满足条件里面选两组同时为 T 的组合 (因为不除去 i, 为了速度, 故 >1)
        ## 此处判断不能少, 因为 choose(-1,2)=0
        p3 <- 2 * choose(temp, 2) * (temp > 1) + p3
        ## p4 同理
        temp1 <- sum(z >= z[i]) - 1
        if (temp > 0 & temp1 > 0) {
            p4 <- temp1 * temp + p4
        }
    }
    p2 <- p2/(n * (n - 1))
    p3 <- p3/(n * (n - 1) * (n - 2))
    p4 <- p4/(n * (n - 1) * (n - 2))
    c(p1, p2, p3, p4)
}
```

因为文章说明

> A simulation study was conducted to evaluate the performance of the derived sample size of the formula in Eq. (2). The zis are generated from normal population with mean y and variance 1. The pis are estimated by Monte Carlo method based on a sample of size 10,000. The estimated values of pis are used to determine the sample size from the formula in Eq. (2). Then, using the calculated sample size, the true power is simulated based on 10,000 simulations. Table 1 ummarizes the results from the simulation. As can be seen from Table 1, the sample size needed to achieve the desired power is not too large, and the actual power for the calculated sample size is very close to the nominal power, which indicates that the sample size formula works very well.

我们采用 Monte Carlo 方法，构造函数

```
## 蒙特卡洛, 设计多线程实验

theta <- c(seq(0.2, 0.49, 0.01))
table <- rep(0, 9)
names(table) <- c("theta", "p1", "p2", "p3", "p4", "n0.8", "true_power_0.8", "n0.9",
    "true_power_0.9")
library(parallel)

for (i in theta) {
    z <- rnorm(10000, mean = theta, sd = 1)
    pt <- p(z)
    p1 <- pt[1]
    p2 <- pt[2]
    p3 <- pt[3]
    p4 <- pt[4]
```

```
    ## power = 0.8 的情况
    tempn <- floor((qnorm(0.975)/sqrt(12) + qnorm(0.8) * sqrt(p3 + 4 * p4 - 4 * p2^2))^2/((1/4 -
        p2)^2)) + 1
    monte <- function(N) {
        n <- length(z)
        tempz <- sample(z, size = tempn, replace = T)  ## 有放回
        t <- sum(rank((abs(tempz)))[which(tempz > 0)])
        t <- (t - tempn * (tempn + 1)/4)/sqrt(tempn * (tempn + 1) * (2 * tempn +
            1)/24)
        return(abs(t) > qnorm(0.975))
    }
    cl <- makeCluster(16)
    nsim <- 10000
    clusterExport(cl, "z")
    clusterExport(cl, "tempn")
    clusterExport(cl, "p")
    T1 <- do.call(c, parLapply(cl, 1:nsim, monte))
    stopCluster(cl)
    true_power_8 <- sum(T1 == TRUE)/nsim
    n1 <- tempn
    ## power .9 的情况，为了多线程传入参数方便重新构造
    cl <- makeCluster(16)
    tempn <- floor((qnorm(0.975)/sqrt(12) + qnorm(0.9) * sqrt(p3 + 4 * p4 - 4 * p2^2))^2/((1/4 -
        p2)^2)) + 1
    clusterExport(cl, "z")
    clusterExport(cl, "tempn")
    clusterExport(cl, "p")
    T2 <- do.call(c, parLapply(cl, 1:nsim, monte))
    stopCluster(cl)
    true_power_9 <- sum(T2 == TRUE)/nsim
    n2 <- tempn
    out <- c(i, p1, p2, p3, p4, n1, true_power_8, n2, true_power_9)
    table <- rbind(table, out)
}
table <- table[2:nrow(table), ]
```

```
library(knitr)
library(kableExtra)
knitr::kable(table, format = "markdown")
```

	theta	p1	p2	p3	p4	n0.8	true_power_0.8	n0.9	true_power_0.9
out	0.20	0.6367	0.3442419	0.2393889	0.0758254	70	0.7958	92	0.8955
out	0.21	0.6353	0.3443538	0.2393584	0.0759082	70	0.8019	92	0.9005
out	0.22	0.6322	0.3428440	0.2389247	0.0750511	72	0.8007	95	0.8954
out	0.23	0.6360	0.3436698	0.2392902	0.0754761	71	0.7976	93	0.8914
out	0.24	0.6342	0.3443536	0.2396452	0.0758279	70	0.7966	92	0.8981
out	0.25	0.6341	0.3414606	0.2372258	0.0746639	75	0.8027	98	0.8998
out	0.26	0.6277	0.3402995	0.2372184	0.0739277	77	0.8026	101	0.8923
out	0.27	0.6347	0.3436667	0.2388167	0.0756241	71	0.7968	93	0.8962
out	0.28	0.6349	0.3430332	0.2386164	0.0752675	72	0.8016	95	0.8984
out	0.29	0.6285	0.3400486	0.2365729	0.0739545	77	0.8012	101	0.8946
out	0.30	0.6413	0.3464014	0.2409123	0.0767681	67	0.8069	88	0.8978
out	0.31	0.6354	0.3456729	0.2409406	0.0762886	68	0.7999	89	0.9014
out	0.32	0.6438	0.3502106	0.2443813	0.0782236	62	0.8020	81	0.8998
out	0.33	0.6380	0.3433242	0.2384708	0.0754954	72	0.7997	94	0.8970
out	0.34	0.6423	0.3479119	0.2424214	0.0773040	65	0.7997	85	0.8973
out	0.35	0.6460	0.3483195	0.2417282	0.0777815	64	0.7958	84	0.8959
out	0.36	0.6333	0.3419535	0.2376200	0.0748687	74	0.8017	97	0.8969
out	0.37	0.6363	0.3436269	0.2393338	0.0754388	71	0.8002	93	0.8973
out	0.38	0.6309	0.3424803	0.2387947	0.0748622	73	0.8020	96	0.8944
out	0.39	0.6357	0.3435700	0.2391848	0.0754443	71	0.8017	93	0.8944
out	0.40	0.6398	0.3451852	0.2398879	0.0762884	69	0.7997	90	0.8988
out	0.41	0.6325	0.3438972	0.2392629	0.0756340	71	0.8018	93	0.8952
out	0.42	0.6317	0.3418052	0.2374935	0.0748183	74	0.7964	97	0.8960
out	0.43	0.6422	0.3474330	0.2411936	0.0773647	65	0.7988	86	0.8959
out	0.44	0.6407	0.3441508	0.2392636	0.0758010	70	0.7932	92	0.8937
out	0.45	0.6254	0.3389598	0.2361314	0.0733869	79	0.8002	104	0.9010
out	0.46	0.6355	0.3445438	0.2400530	0.0758118	70	0.8069	91	0.8938
out	0.47	0.6384	0.3450285	0.2397526	0.0762205	69	0.7971	90	0.8979
out	0.48	0.6313	0.3424419	0.2381206	0.0750390	73	0.8018	96	0.9006
out	0.49	0.6295	0.3397285	0.2359406	0.0739455	78	0.7999	102	0.8943

Table 3 results reproduced

```
rm(list = ls())
```

同理，考虑两组随机变量 $x_i, i = 1, \cdots, n_1$ ，$y_j, j = 1, \cdots, n_2$ 考虑零假设 $H_0 : \theta = 0$,$H_1 : \theta \neq 0$, 其中有

$x_i = e_i, i = 1, \cdots, n_1$,$y_j = e_{n_1+j} + \theta$ 考虑统计量

$$W = \sum_{i=1}^{n_2} R_i,$$

在零假设情况下，当 $n_1, n_2 \to \infty$ 时候有

$$\begin{aligned} W^* &= \frac{W - E(W)}{\sqrt{\operatorname{var}(W)}} \\ &= \frac{W - \frac{1}{2} n_2 (n_2 + n_1 + 1)}{\sqrt{\frac{1}{12} n_1 n_2 (n_1 + n_2 + 1)}} \to N(0,1) \end{aligned}$$

我们拒绝零假设当满足 $|W^*| \geq z_{\alpha/2}$ 又

$$\begin{aligned} W &= \sum_{i=1}^{n_2} \left(\sum_{j=1}^{n_2} I\{y_i \geq y_j\} + \sum_{j=1}^{n_1} I\{y_i \geq x_j\} \right) \\ &= \frac{n_2 (n_2 + 1)}{2} + \sum_{i=1}^{n_2} \sum_{j=1}^{n_1} I\{y_i \geq x_j\} \end{aligned}$$

则

$$E(W) = \frac{n_1 (n_1 - 1)}{2} + n_1 n_2 p_1$$

$$\operatorname{var}(W) = n_1 n_2 p_1 (1 - p_1) + n_1 n_2 (n_1 - 1)(p_2 - p_1^2) + n_1 n_2 (n_2 - 1)(p_3 - p_1^2)$$

$$\begin{aligned} p_1 &= P(y_1 \geq x_1) \\ p_2 &= P(y_1 \geq x_1 \text{ and } y_1 \geq x_2) \\ p_3 &= P(y_1 \geq x_1 \text{ and } y_2 \geq x_1) \end{aligned}$$

$$\hat{p}_1 = \frac{1}{n_1 n_2} \sum_{i=1}^{n_2} \sum_{j=1}^{n_1} I\{y_i \geq x_j\}$$

$$\hat{p}_2 = \frac{1}{n_1 n_2 (n_1 - 1)} \sum_{i=1}^{n_2} \sum_{j_1 \neq j_2} I\left\{y_i \geq x_{j_1} \text{ and } y_i \geq x_{j_2}\right\}$$

$$\hat{p}_3 = \frac{1}{n_1 n_2 (n_2 - 1)} \sum_{i_1 \neq i_2} \sum_{j=1}^{n_1} I\left\{y_{i_1} \geq x_i \text{ and } y_{i_2} \geq x_j\right\}$$

构造函数有

```
p <- function(x, y) {
    p1 <- 0
    p2 <- 0
    p3 <- 0
    n1 <- length(x)
    n2 <- length(y)
    tempz <- outer(y, x, "-")
```

```
    p1 <- sum(tempz >= 0)/(n1 * n2)
    for (i in 1:n2) {
        p2 <- 2 * choose(sum(tempz[i, ] >= 0), 2) + p2
    }
    for (i in 1:n1) {
        p3 <- 2 * choose(sum(tempz[, i] >= 0), 2) + p3
    }
    p2 <- p2/n1/n2/(n1 - 1)
    p3 <- p3/n1/n2/(n2 - 1)
    return(c(p1, p2, p3))
}
```

当 n_1, n_2 较大时候有均值

$$\mu w = \frac{n_2\left(n_2+1\right)}{2} + n_1 n_2 p_1$$

方差

$$\sigma_W^2 = n_1 n_2 p_1\left(1-p_1\right) + n_1 n_2\left(n_1-1\right)\left(p_2-p_1^2\right) + n_1 n_2\left(n_2-1\right)\left(p_3-p_1^2\right)$$

有

$$\begin{aligned}
1-\beta =& P\left(|W^*| > z_{\alpha/2}\right) \\
\approx & P\left(W^* > z_{\alpha/2}\right) \\
= & P\left(\frac{W - n_2\left(n_2+1\right)/2 - n_1 n_2 p_1}{\sigma_W}\right. \\
& \left. > \frac{z_{\alpha/2}\sqrt{n_1 n_2\left(n_1+n_2+1\right)/12} + n_1 n_2\left(1/2-p_1\right)}{\sigma_W}\right) \\
= & P\left(N(0,1) > \frac{z_{\alpha/2}\sqrt{\kappa(1+\kappa)/12} + \sqrt{n_2}\kappa\left(1/2-p_1\right)}{\sqrt{\kappa^2\left(p_2-p_1^2\right) + \kappa\left(p_3-p_1^2\right)}}\right)
\end{aligned}$$

则可知若 $n_1 = \kappa n_2$ 有

$$n_2 = \frac{\left(z_{\alpha/2}\sqrt{\kappa(\kappa+1)/12} + z_\beta\sqrt{\kappa^2\left(p_2-p_1^2\right) + \kappa\left(p_3-p_1^2\right)}\right)^2}{\kappa^2\left(1/2-p_1\right)^2}$$

又因为作者写道

> A simulation study was conducted to evaluate the above formula for sample size calculation. The xis are generated from normal population with mean 0 and variance 1, yis are generated from normal population with mean y and variance 1. The sample size ratio k is chosen to be 1. The pis are estimated by a Monte Carlo method based on a sample size of 10,000. The estimated values of pis are used to determined the sample size from the formula in Eq. (4). Using the calculated sample size, the true power is simulated based on 10,000 simulations. The results are given in Table 3.

则

```r
theta <- c(seq(0.22, 0.8, 0.02))
table <- rep(0, 8)
names(table) <- c("theta", "p1", "p2", "p3", "n0.8", "true_power_0.8", "n0.9", "true_power_0.9")
library(parallel)

for (i in theta) {
    x <- rnorm(10000, mean = 0, sd = 1)
    y <- rnorm(10000, mean = i, sd = 1)
    pt_ <- p(x, y)
    p1 <- pt_[1]
    p2 <- pt_[2]
    p3 <- pt_[3]
    ## power = 0.8
    tempn <- floor((qnorm(0.975) * sqrt(2/12) + qnorm(0.8) * sqrt((p2 - p1^2)) +
        (p3 - p1^2))^2/(1/2 - p1)^2) + 1
    n21 <- tempn
    nsim <- 10000
    monte <- function(N) {
        tempx <- sample(x, size = tempn, replace = T)
        tempy <- sample(y, size = tempn, replace = T)
        w <- sum(rank(c(tempx, tempy))[(tempn + 1):(2 * tempn)])
        w <- (w - 1/2 * tempn * (2 * tempn + 1))/sqrt(1/12 * tempn^2 * (2 * tempn +
            1))
        return(abs(w) >= qnorm(0.975))
    }
    cl <- makeCluster(16, type = "FORK")  ## 因为是 Linux 可以直接共享数据
    # clusterExport(cl, 'x') clusterExport(cl, 'y') clusterExport(cl,'tempn')
    # clusterExport(cl, 'p')
    T1 <- do.call(c, parLapply(cl, 1:nsim, monte))
    stopCluster(cl)
    true_power_8 <- sum(T1 == TRUE)/nsim
    ## power = 0.9
    tempn <- floor((qnorm(0.975) * sqrt(2/12) + qnorm(0.9) * sqrt((p2 - p1^2)) +
        (p3 - p1^2))^2/(1/2 - p1)^2) + 1
    n22 <- tempn
    cl <- makeCluster(16, type = "FORK")
    T2 <- do.call(c, parLapply(cl, 1:nsim, monte))
    stopCluster(cl)
    true_power_9 <- sum(T2 == TRUE)/nsim
    out <- c(i, p1, p2, p3, n21, true_power_8, n22, true_power_9)
    table <- rbind(table, out)
```

```
}
table <- table[2:nrow(table), ]
```

结果有

```
knitr::kable(table, format = "markdown")
```

	theta	p1	p2	p3	n0.8	true_power_0.8	n0.9	true_power_0.9
out	0.22	0.5609094	0.3983709	0.3943119	341	0.7872	422	0.8612
out	0.24	0.5715078	0.4068681	0.4087271	246	0.7748	304	0.8624
out	0.26	0.5790208	0.4158791	0.4160090	201	0.7920	249	0.8864
out	0.28	0.5755179	0.4108894	0.4134159	220	0.7748	272	0.8624
out	0.30	0.5811039	0.4196862	0.4167485	191	0.7954	237	0.8752
out	0.32	0.5851415	0.4224749	0.4229112	173	0.7766	214	0.8724
out	0.34	0.5900506	0.4279490	0.4280502	155	0.8132	191	0.8704
out	0.36	0.5950402	0.4351112	0.4321503	139	0.7846	172	0.8628
out	0.38	0.6049278	0.4443759	0.4449202	113	0.7888	140	0.8740
out	0.40	0.6190308	0.4597356	0.4613699	88	0.7924	108	0.8674
out	0.42	0.6202764	0.4605531	0.4634183	86	0.7700	105	0.8568
out	0.44	0.6217985	0.4642137	0.4631513	84	0.7910	103	0.8466
out	0.46	0.6289279	0.4729366	0.4709881	75	0.7906	92	0.8558
out	0.48	0.6327258	0.4744012	0.4782130	70	0.7684	86	0.8498
out	0.50	0.6374385	0.4802563	0.4830741	65	0.7958	80	0.8688
out	0.52	0.6446629	0.4895866	0.4908603	59	0.7766	72	0.8464
out	0.54	0.6463032	0.4934728	0.4908599	58	0.8228	71	0.8722
out	0.56	0.6553151	0.5023442	0.5030234	51	0.7398	62	0.8708
out	0.58	0.6568781	0.5018664	0.5072714	50	0.7846	61	0.8574
out	0.60	0.6628146	0.5108895	0.5122871	46	0.7730	56	0.8544
out	0.62	0.6696549	0.5203111	0.5193429	42	0.7856	52	0.8674
out	0.64	0.6780523	0.5307782	0.5290236	38	0.7794	47	0.8756
out	0.66	0.6757496	0.5276962	0.5266033	39	0.7442	48	0.8648
out	0.68	0.6840320	0.5375073	0.5365280	36	0.8194	44	0.8706
out	0.70	0.6922392	0.5485460	0.5459881	33	0.8098	40	0.8720
out	0.72	0.6900820	0.5436426	0.5456089	33	0.7874	41	0.8626
out	0.74	0.7005802	0.5575477	0.5575950	30	0.7766	36	0.8658
out	0.76	0.7041426	0.5616799	0.5621951	29	0.7940	35	0.8530
out	0.78	0.7123293	0.5732673	0.5707959	26	0.7936	32	0.8594
out	0.80	0.7143611	0.5760981	0.5734836	26	0.7620	31	0.8338