hw03

曾舸舵

2022/4/1

6.3

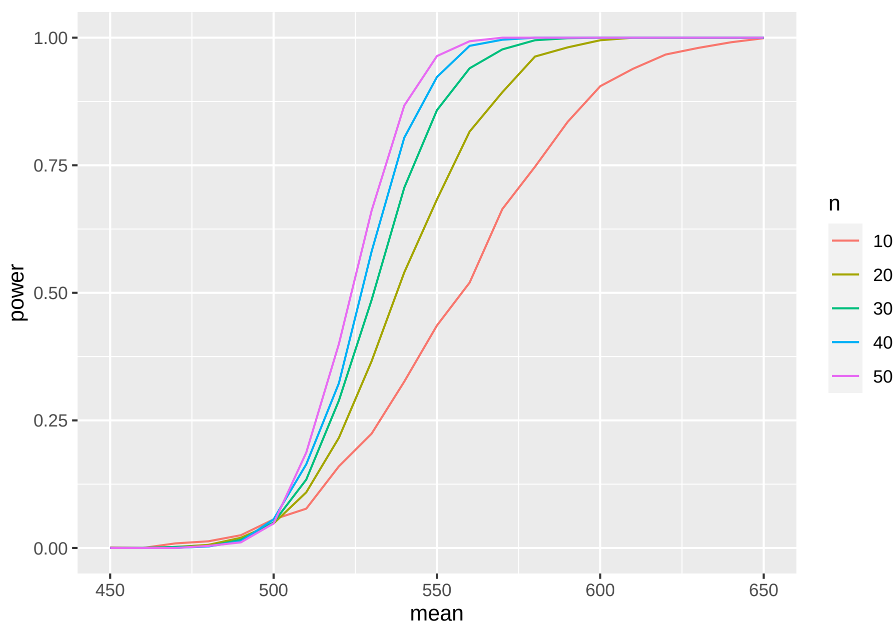
```r
example_7_9 <- function(n) {
  m <- 1000
  mu0 <- 500
  sigma <- 100
  mu <- c(seq(450, 650, 10)) #alternatives
  M <- length(mu)
  power <- numeric(M)
  for (i in 1:M) {
    mu1 <- mu[i]
    pvalues <- replicate(m, expr = {
    #simulate under alternative mu1
    x <- rnorm(n, mean = mu1, sd = sigma)
    ttest <- t.test(x, alternative = "greater", mu = mu0)
    ttest$p.value } )
  power[i] <- mean(pvalues <= .05)
  }
  return(tibble(mean = mu, power = power))
}

n_ <- seq(10, 50, 10)
```

```
by_n <- tibble(n = factor(n_), data = n_ %>% map(example_7_9)) %>%
  unnest()
```

```
## Warning: `cols` is now required when using unnest().
## Please use `cols = c(data)`
```

```
by_n %>%
  ggplot(aes(x=mean, y=power, color=n)) +
  geom_line()
```



随着均值增加，数据量越大，功效越强 # 6.4

```
hw_6_4 <- function(n=1e2, m=1e4){
  alpha <- .025
  d <- replicate(n, mean(rlnorm(m)))
  ds <- sort(d)
  m <- d[[ceiling(alpha * n)]]
  M <- d[[ceiling((1-alpha) * n)]]
  return(c(m, M))
```

```
}

hw_6_4()
```

```
## [1] 1.645017 1.642579
```

6.5

```
exercise_6_5 <- function(seed=123){
set.seed(seed)
n<-20
c<-qt(0.975,n-1) # 0.975 quantile of t-distribution
m <- 1000
cv.t<-sapply(1:m,FUN= function(o){
x<-rchisq(n,2)
m<-mean(x) # estimate of mean
se<-sqrt(var(x)) # estimate of standard error
as.numeric((m-c*se/sqrt(n)<2)&(m+c*se/sqrt(n)>2)) # ci
})
level1 <- mean(cv.t) # mean of  Monte Carlo experiment


alpha <- .05
UCL <- replicate(1000, expr = {
x <- rchisq(n,2)
(n-1) * var(x) / qchisq(alpha, df = n-1)
} )
level2 <- sum(UCL > 4)/m
return(data.frame(level1,level2))
}

exercise_6_5(1012)
```

```
##   level1 level2
## 1  0.908  0.794
```

我们可以看到结果远小于 0.95，因此 t-区间更稳健 # 6.8

```
exercise_6_8 <- function(){
count5test <- function(x,y){
  X <- x - mean(x)
  Y <- y - mean(y)
  outx <- sum(X > max(Y)) + sum(X < min(Y))
  outy <- sum(Y > max(X)) + sum(Y < min(X))
  return(as.integer(max(c(outx,outy)) > 5))
}
n <- c(20,200,1000)# 分别对应小样本、中样本和大样本
mu1 <- mu2 <- 0
sigma1 <- 1
sigma2 <- 1.5
m <- 10000
power1 <- power2 <- numeric(length(n))
set.seed(1234)
for(i in 1:length(n)){
  power1[i] <- mean(replicate(m,expr = {
    x <- rnorm(n[i],mu1,sigma1)
    y <- rnorm(n[i],mu2,sigma2)
    x <- x - mean(x)
    y <- y - mean(y)
    count5test(x,y)
  }))
  pvalues <- replicate(m,expr={
    x <- rnorm(n[i],mu1,sigma1)
    y <- rnorm(n[i],mu2,sigma2)
    Ftest <- var.test(x, y, ratio = 1,
                      alternative = c("two.sided"),
                      conf.level = 0.945, ...)
```

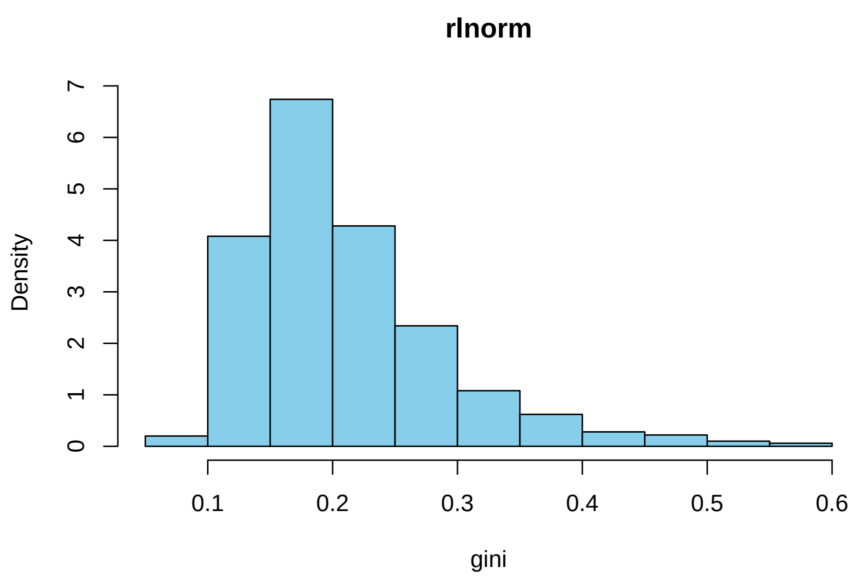
```
    Ftest$p.value})
  power2[i] <- mean(pvalues<=0.055)
}
return(data.frame(power1,power2))
}
exercise_6_8()
```

```
##   power1 power2
## 1 0.3128 0.4118
## 2 0.9475 0.9999
## 3 0.9980 1.0000
```

6.9

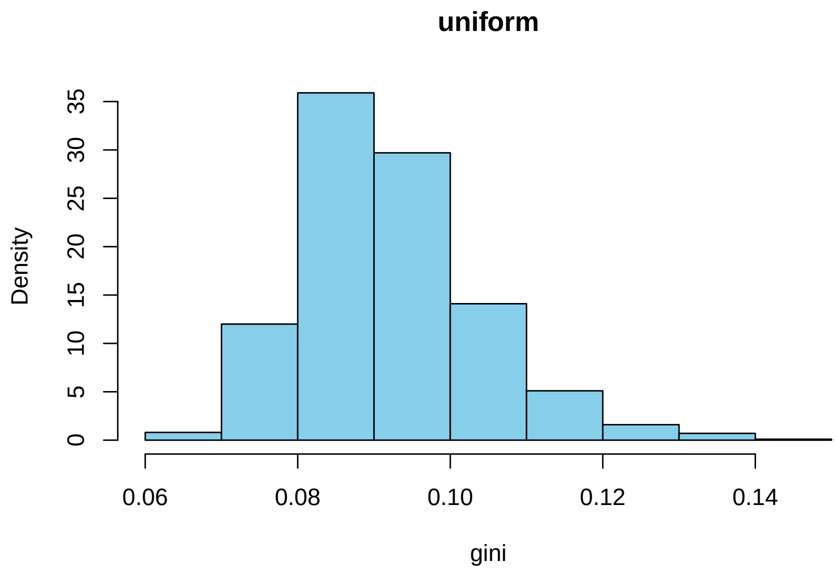
```
exercise_6_9 <- function(distribution=c('rlnorm','uniform','Bernoulli')){
n <- 20
size <- 1000
ginifun <- function()
{
  if (distribution == 'rlnorm')x <- sort(rlnorm(n))
  else if(distribution == 'uniform') x <- sort(runif(n,0,1))
  else x <- sort(rbinom(n,size = 100,prob = .1))
  m=mean(x)
  sum=0
  for (k in n) {
    t=(2*k-n-1)*x[k]
    sum=sum+t
  }
  gini=sum/(n^2*m)
}
res <- replicate(size,expr = ginifun())
hist(as.numeric(res), prob = TRUE, main = distribution,xlab='gini',col='skyblue')
```

```
help(hist)
return(data.frame(mean = mean(res),median = median(res),quantile = quantile(res,seq(.1,
}
exercise_6_9(distribution = 'rlnorm')
```



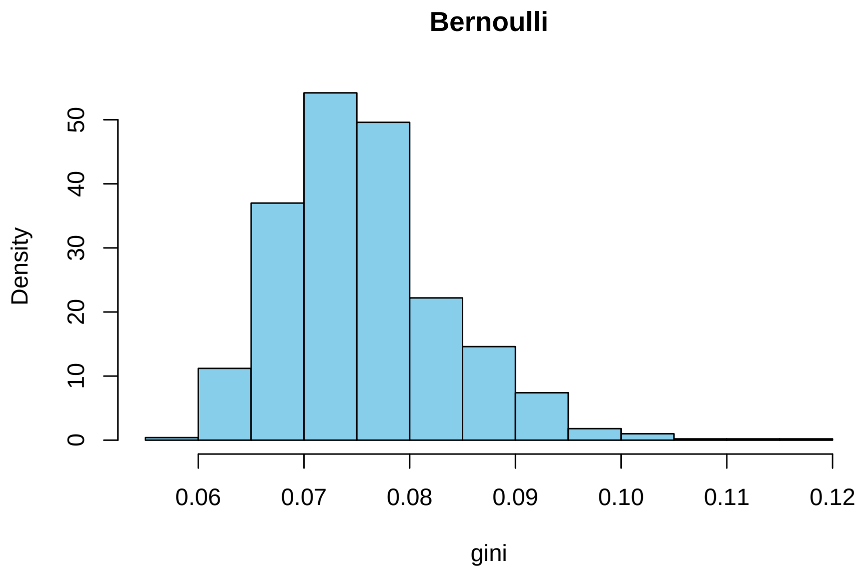

```
##          mean    median  quantile
## 10% 0.2098996 0.1916062 0.1301766
## 20% 0.2098996 0.1916062 0.1476594
## 30% 0.2098996 0.1916062 0.1620432
## 40% 0.2098996 0.1916062 0.1748635
## 50% 0.2098996 0.1916062 0.1916062
## 60% 0.2098996 0.1916062 0.2089830
## 70% 0.2098996 0.1916062 0.2306749
## 80% 0.2098996 0.1916062 0.2619465
## 90% 0.2098996 0.1916062 0.3119318
```

```r
exercise_6_9(distribution = 'uniform')
```




```
##            mean     median   quantile
## 10% 0.09227028 0.09059201 0.07865068
## 20% 0.09227028 0.09059201 0.08263439
## 30% 0.09227028 0.09059201 0.08544153
## 40% 0.09227028 0.09059201 0.08794833
## 50% 0.09227028 0.09059201 0.09059201
## 60% 0.09227028 0.09059201 0.09351204
## 70% 0.09227028 0.09059201 0.09695004
## 80% 0.09227028 0.09059201 0.10066065
## 90% 0.09227028 0.09059201 0.10790021
```

```
exercise_6_9(distribution = 'Bernoulli')
```




```
##            mean     median   quantile
## 10% 0.07577415 0.07468792 0.06658879
## 20% 0.07577415 0.07468792 0.06917476
## 30% 0.07577415 0.07468792 0.07114537
## 40% 0.07577415 0.07468792 0.07307692
## 50% 0.07577415 0.07468792 0.07468792
## 60% 0.07577415 0.07468792 0.07671604
## 70% 0.07577415 0.07468792 0.07878049
## 80% 0.07577415 0.07468792 0.08159661
## 90% 0.07577415 0.07468792 0.08636364
```

6A

```
exercise_6_A <- function(seed){
set.seed(123)
num<-c(50,100,200,500,1000) # Estimate the Type-I error for different sizes.
m<-10000

er<-NULL
for (n in num){
  cv<-qt(0.975,n-1)
  er1<-mean(sapply(1:m,FUN = function(o){
  x<-rchisq(n,1)
  m<-mean(x)
  se<-sqrt(var(x))
  abs((m-1)*sqrt(n)/se)>=cv
}))  # 估计卡方分布的第一类错误
  er2<-mean(sapply(1:m,FUN = function(o){
  x<-runif(n,0,2)
  m<-mean(x)
  se<-sqrt(var(x))
  abs((m-1)*sqrt(n)/se)>=cv
}))   # 估计均匀分布的第一类错误
  er3<-mean(sapply(1:m,FUN = function(o){
  x<-rexp(n,1)
  m<-mean(x)
  se<-sqrt(var(x))
  abs((m-1)*sqrt(n)/se)>=cv
}))  # 估计指数分布的第一类错误
er<-cbind(er,c(er1,er2,er3))
}
colnames(er)<-num
rownames(er)<-c("chi(1)","U(0,2)","exp(1)")
return(er)
}
exercise_6_A(1012)
```

```
##            50    100    200    500   1000
## chi(1) 0.0783 0.0657 0.0584 0.0496 0.0535
## U(0,2) 0.0492 0.0495 0.0460 0.0499 0.0493
## exp(1) 0.0655 0.0644 0.0515 0.0492 0.0518
```

6B

```
exercise_6_B <- function(seed){
seed <- set.seed(123)
x <- rnorm(20,2,10)
sigma <- rnorm(20,5,50)
y <- 3*x+sigma
cor(x,y)
pearson <- cor.test(x,y)
kendall <- cor.test(x,y,method = 'kendall')
spearman <- cor.test(x,y,method = 'spearman')
data.frame(x,y)
return(list(pearson=pearson,kendall=kendall,spearman=spearman))
}
exercise_6_B()
```

```
## $pearson
##
##  Pearson's product-moment correlation
##
## data:  x and y
## t = 2.6052, df = 18, p-value = 0.0179
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.1050841 0.7842035
```

```
## sample estimates:
##       cor
## 0.5232718
##
##
## $kendall
##
##  Kendall's rank correlation tau
##
## data:  x and y
## T = 136, p-value = 0.007346
## alternative hypothesis: true tau is not equal to 0
## sample estimates:
##       tau
## 0.4315789
##
##
## $spearman
##
##  Spearman's rank correlation rho
##
## data:  x and y
## S = 532, p-value = 0.00608
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
## rho
## 0.6
```