3701_Assignment2_JinYao

JinYao

2019/7/7

R Markdown

Problem 1

```
pro.est = function(p, n){
  esti = numeric(n)
  for(times in 1 : n){
  reps = 1e4;ber.row = c(); cum.row = c()
  for(i in 1:reps){
    u = runif(7)
    for(j in 1:7){
      ##give a binomial estimate for every 7 attepmts
      ber.row[j] = ifelse(u[j]>p, 0, 1)
    }
    cum.row[i] = sum(ber.row)
  }
  est = c(); sum = 0
  for(k in 1:reps){
    est[k] = ifelse(cum.row[k] > 3, 1, 0)
    if(est[k] > 0){
      sum = sum + 1
    }
  }
  esti[times] = sum/reps
  }
  return(esti)
}
x1 = pro.est(0.3, 100)
x2 = pro.est(0.5, 100)
x3 = pro.est(0.7, 100)
probability = c(0.3,0.5,0.7)
mean = c(mean(x1),mean(x2),mean(x3))
sd = c(sd(x1),sd(x2),sd(x3))
data.frame(probability,mean,sd)
```

```
##   probability     mean          sd
## 1         0.3 0.126719 0.003207085
## 2         0.5 0.499617 0.005899205
## 3         0.7 0.874252 0.003252861
```

Problem 2

i

```
norm.gen = function(n, mu=0, sigma=1){
  odd = (n %% 2) != 0
  if(odd==TRUE){
   n = n+1
   }
   u1 = runif(n/2)
   u2 = runif(n/2)
   t = sqrt(-2*log(u1))
   z1 = t*cos(2*pi*u2)
   z2 = t*sin(2*pi*u2)
   z.list = c(z1, z2)
  if(odd){
   z.list = z.list[-n]
   }
   x.list = mu + sigma*z.list
   return(x.list)
}

chi.gen = function(n, df){
  chi.mat = matrix(norm.gen(n = n*df)^2, nrow = n, ncol = df)
  chi.list = apply(chi.mat, 1, sum)
  return(chi.list)
}
est = chi.gen(1000, 10)
mean(est)
```

```
## [1] 10.07333
```

```
sd(est)^2
```

```
## [1] 20.22516
```

ii

```
gamma.gen = function(k,n,lambda){
  x.list = c(); est.lam = c()
  for(j in 1:k){
  for(i in 1:n){
    u = runif(1)
    x.list[i] = -lambda*log(u)
  }
  #est.lam = sum(x.list)
    est.lam[j] = sum(x.list)
  }
  return(est.lam)
}
est.exp = gamma.gen(k = 200,n = 100,lambda = 2)
data.frame("gamma:", "mean",mean(est.exp),"sd", sd(est.exp))
```

```
##   X.gamma.. X.mean. mean.est.exp. X.sd. sd.est.exp.
## 1    gamma:    mean       199.697    sd    20.12888
```

```
est.chi = chi.gen(n = 200, df = 200)
data.frame("chi:", "mean",mean(est.chi),"sd", sd(est.chi))
```

```
##   X.chi.. X.mean. mean.est.chi. X.sd. sd.est.chi.
## 1    chi:    mean      202.4761    sd    20.33967
```

Problem 3

i.

$\Phi(b) - \Phi(a) > 0, \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} > 0$

$f(x) = \frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}}{\Phi(b)-\Phi(a)}$

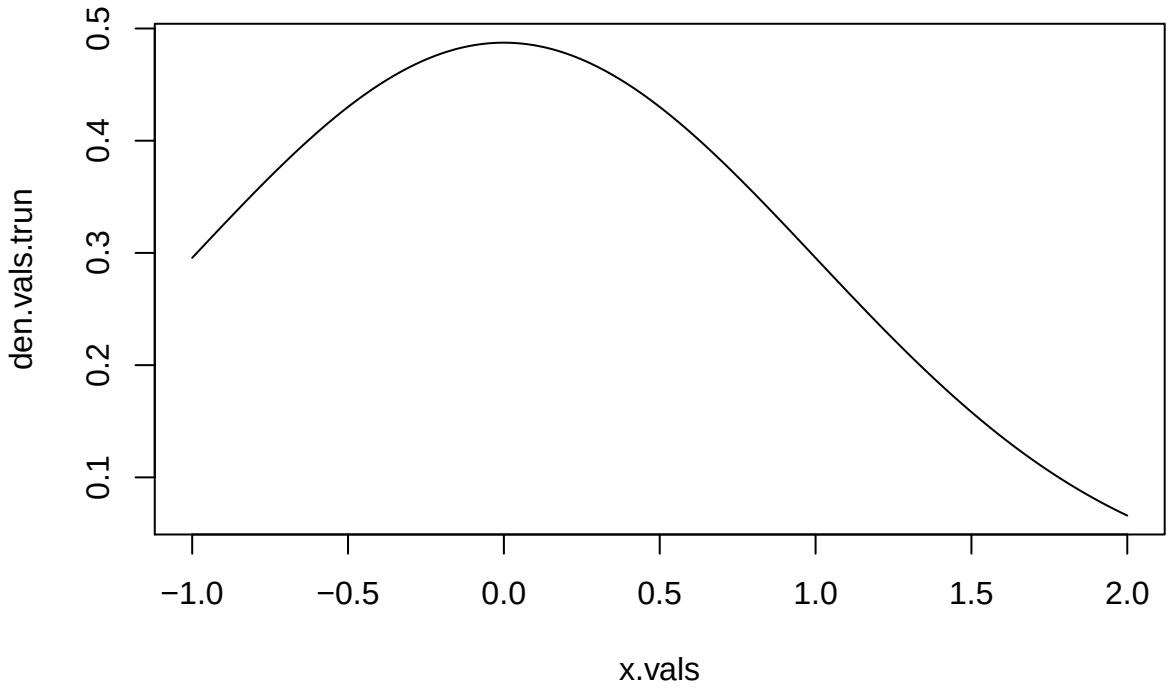
$F(x) = \int_a^b f(x)dx = \frac{\int_a^b \frac{1}{2\pi} e^{\frac{x^2}{2}} dx}{\int_{-\infty}^b \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt - \int_{-\infty}^a \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt} = \frac{\int_{-\infty}^b f(x)dx - \int_{-\infty}^a f(x)dx}{\int_{-\infty}^b f(t)dt - \int_{-\infty}^a f(t)dt} = 1$

ii.

$E(x) = \int_a^b x \frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}}{\Phi(b)-\Phi(a)} dx = \frac{1}{\Phi(b)-\Phi(a)} \int_a^b x \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$ the probability of the standard normal distribution is symmetric, so E(x)=0

iii

```
x.vals = seq(from = -1, to = 2, length.out = 1e3)
den.vals = dnorm(x = x.vals, mean = 0, sd = 1)
den.vals.trun = den.vals/(pnorm(2,0,1) - pnorm(-1,0,1))
plot(x.vals, den.vals.trun, type = "l")
```

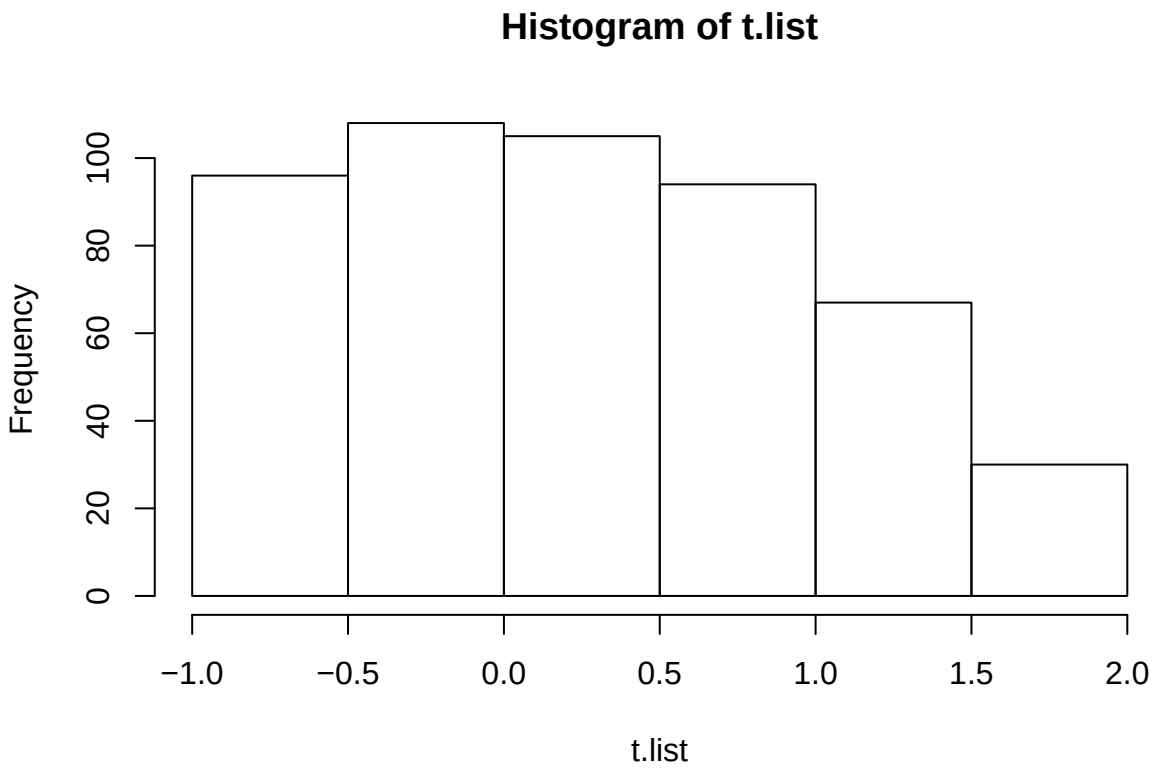


iv

```
norm.gen=function(n, mu=0, sigma=1) {
  odd=(n %% 2)!=0
  if(odd) {
  n=n+1 }
  u1=runif(n/2)
  u2=runif(n/2)
  t=sqrt(-2*log(u1))
  z.list=c(t*cos(2*pi*u2), t*sin(2*pi*u2) )
  if(odd) {
  z.list=z.list[-n] }
  x.list=mu + sigma*z.list
  return(x.list)
}

norm.t.gen = function(n, a, b){
  num = 0
  t.list = c()
  while(num < n){
  norm = norm.gen(n=n - num)
  limit = which(norm <= b & norm >= a)
  t.list = append(t.list, norm[limit])
  num = length(t.list)
  }
  return(t.list)
}
```

v

```
t.list = norm.t.gen(n = 500, a = -1, b = 2)
hist(t.list)
```



vi

```
set.seed(3701)
reps = 1e4
t = 0.23
alpha = 0.05
n = 500
tperc = qt(1 - alpha/2, df = n - 1)
captured = numeric(reps)
for(i in 1:reps){
  t.list = norm.t.gen(n = n, a = -1, b = 2)
  tbar = mean(t.list)
  s = sd(t.list)
  moe = tperc*s/sqrt(n)
  left.pt = tbar - moe
  right.pt = tbar + moe
  captured[i] = 1*(left.pt <= t)*(right.pt >= t)
}
mean(captured)
```

```
## [1] 0.9514
```

Problem 4

i

```
###type I
pbinom(3,10,0.5)
```

```
## [1] 0.171875
```

```
###power
pbinom(3,10,0.25)
```

```
## [1] 0.7758751
```

ii

```
binom.gen = function(resp = 1e4,n,p){
  u.mat = matrix(runif(reps*n),reps,n)
  ber.mat = ifelse(u.mat<p,1,0)
  return((apply(ber.mat,1,mean)))
}
est.bin.ci = function(n,p1,p2){
  bin.gen = binom.gen(n=n,p=p1)
  est.pow = mean(bin.gen<=p2)
  lower.pt = est.pow-1/sqrt(10^4)
  upper.pt = est.pow+1/sqrt(10^4)
  show = data.frame("confidence interval:",lower.pt, upper.pt )
  return(show)
}
est.bin.ci(n = 10,p1=0.25,p2=0.3)
```

```
##   X.confidence.interval.. lower.pt upper.pt
## 1    confidence interval:   0.7658   0.7858
```

we find that the power is included in this confidence interval

iii

```
est.bin.ci(n=10,p1=0.25,p2=0.3)
```

```
##   X.confidence.interval.. lower.pt upper.pt
## 1    confidence interval:   0.7666   0.7866
```

```
est.bin.ci(n=20,p1=0.25,p2=0.3)
```

```
##   X.confidence.interval.. lower.pt upper.pt
## 1    confidence interval:   0.7695   0.7895
```

```
est.bin.ci(n=30,p1=0.25,p2=0.3)
```

```
##   X.confidence.interval.. lower.pt upper.pt
## 1    confidence interval:   0.7952   0.8152
```

```
est.bin.ci(n=40,p1=0.25,p2=0.3)
```

```
##   X.confidence.interval.. lower.pt upper.pt
## 1    confidence interval:   0.8094   0.8294
```

```
est.bin.ci(n=50,p1=0.25,p2=0.3)
```

```
##   X.confidence.interval.. lower.pt upper.pt
## 1    confidence interval:   0.8233   0.8433
```

```
### the outcome shows that the estimated power increases as the number of the n increases, and the powe
```

Problem 5

a.

$E(v) = \int_{-\infty}^{+\infty} \frac{1}{\lambda} e^{-\frac{v}{\lambda}} v dv$ let $\frac{v}{\lambda} = x$ $E(v) = \lambda \int_0^{+\infty} x e^{-x} dx = \lambda$ so, $E(x) = E(yv) = E(y)E(v) = \theta\lambda$ When theta $= 1/4$, lambda $= 3$, we can get E(x) $= 3/4$ $Var(x) = var(yv) = E(Y^2V^2) - E(YV)^2, E(Y^2V^2) = E(Y^2)E(V^2)$ $E(V^2) = \int_0^{+\infty} \frac{1}{\lambda} e^{\frac{v}{\lambda}} v^2 dv = \lambda^2 \int_0^{+\infty} e^{-x} x^2 dx = 2\lambda^2$ so, $var(x) = var(yv) = 2\lambda^2\theta - \theta^2\lambda^2 = \lambda^2\theta(2-\theta)$ When theta $= 1/4$, lambda $= 3$, we can get v(x) $= 3.9375$

b.

```
p = 1/4; lambda = 3; reps = 1e5
x1.list = runif(reps);x2.list = runif(reps)
ber.list = c(); exp.list = c(); x.list = c()
ber.list = ifelse(x1.list>p,0,1)
exp.list = -lambda*log(x2.list)
for(i in 1:reps){
  y = ber.list[i]
  v = exp.list[i]
  x.list[i] = y*v
}
x.mean = mean(x.list)
s = sd(x.list)
v = s^2
c(x.mean,v)
```

```
## [1] 0.7509024 3.9389443
```

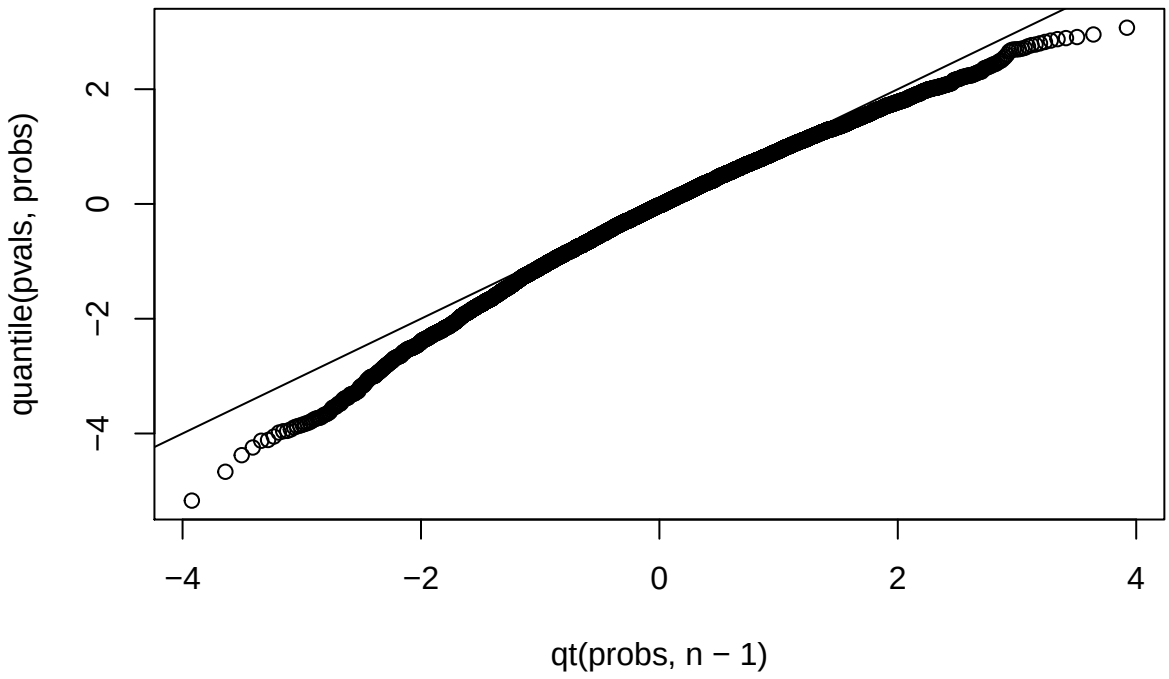
```
###
```

We can conclude that the outcomes are very close to the theorical value.

c.

```
est.pval = function(p = 1/2,lambda = 2,reps = 1e4,n){
  ber.list = c(); exp.list = c(); x.list = c(); t.stat = c()
  pvalue.list = numeric(reps)
  for(i in 1:reps){
  x1.list = runif(n);x2.list = runif(n)
```

```
  ber.list = ifelse(x1.list>p,0,1)
  exp.list = -lambda*log(x2.list)
  for(j in 1:n){
  y = ber.list[j]
  v = exp.list[j]
  x.list[j] = y*v
  }
  xbar = mean(x.list)
  s = sd(x.list)
  t.stat[i] = sqrt(n)*(xbar-1)/s
  #pvalue.list[i] = 2*pt(-abs(t.stat[i]),n-1)
}
  return(t.stat)
}

reps = 1e4
pvals = est.pval(n = 200)
probs = ppoints(reps)
plot(qt(probs, n - 1), quantile(pvals, probs))
abline(0,1)
```

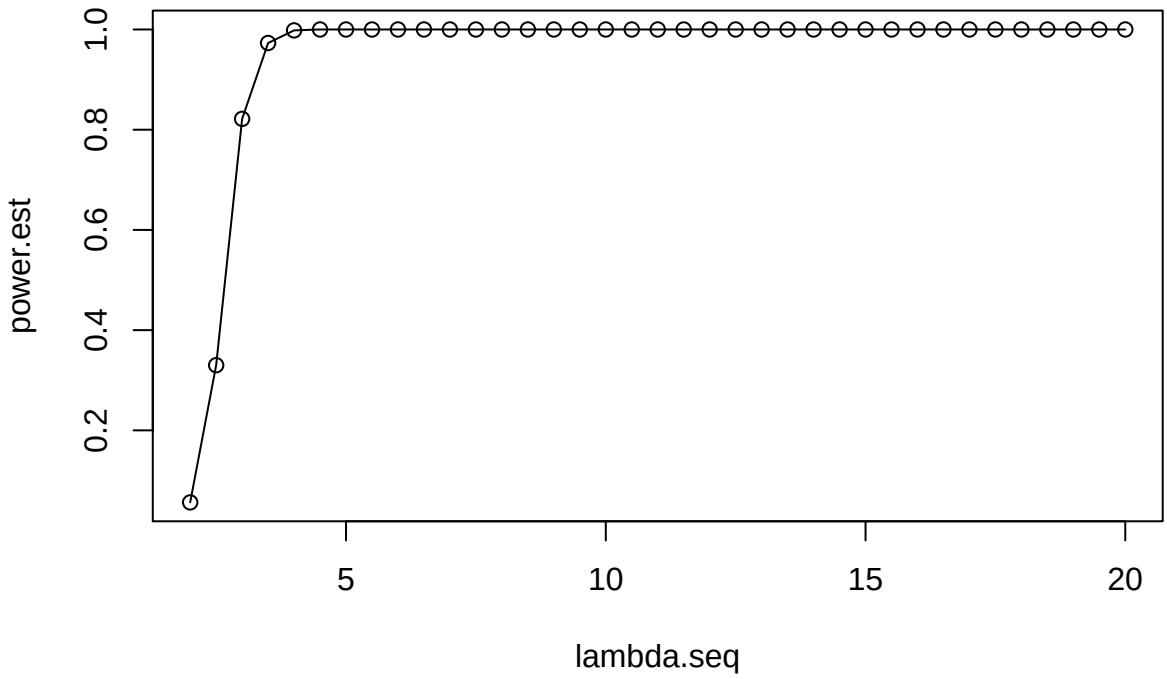


It is very obvious that it does not fit well. We can try to increase our sample size.

d.

```
n = 200;theta = 1/2; lambda.seq = seq(from = 2, to = 20, by = 0.5); reps = 1e4
ber.list = c(); exp.list = c(); x.list = c()
power.est = numeric(length(lambda.seq))
for(i in 1:length(lambda.seq)){
  ber.list = 1*(runif(n*reps) <= theta)
  exp.list = -lambda.seq[i]*log(runif(n*reps))
```

```
  x.mat = matrix(ber.list*exp.list,reps,n)
  xbar= apply(x.mat, 1, mean)
  s = apply(x.mat, 1, sd)
  t = sqrt(n)*(xbar - 1)/s
  power.est[i] = mean(abs(t) >= qt(1 - 0.05/2, df = n-1))
}
plot(lambda.seq,power.est,type="o")
```



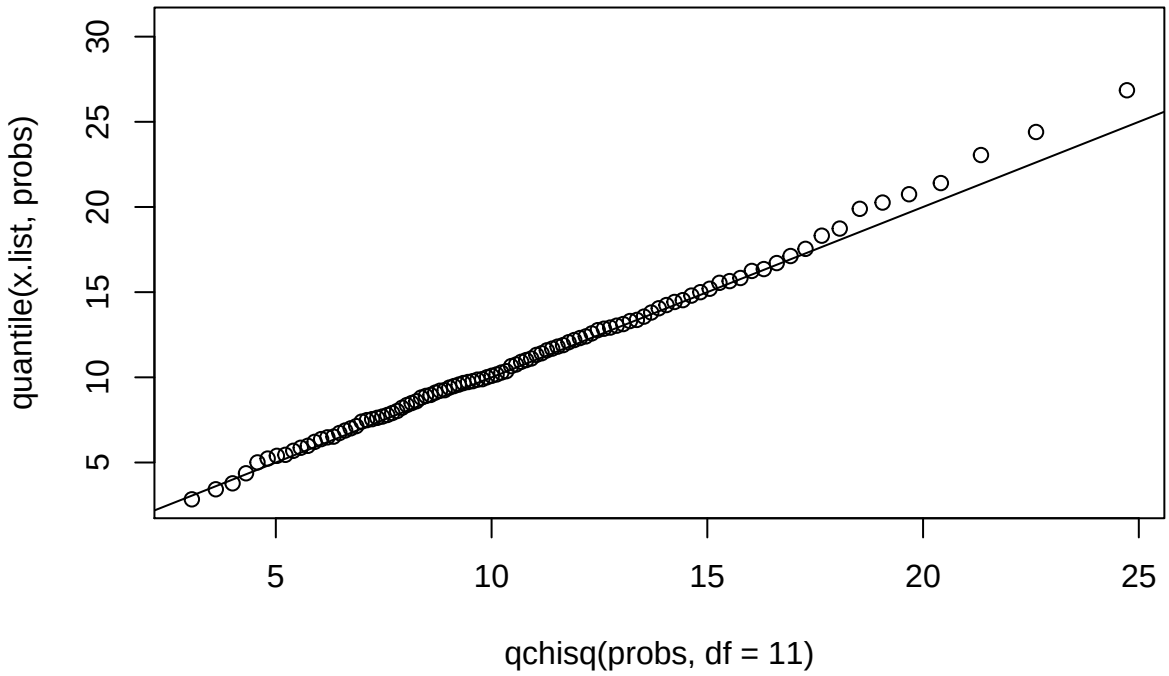
It is obvious that when alpha is at about 5 that the power was led to almost 1, as definitely the power increases as lambda increases.

Problem 6

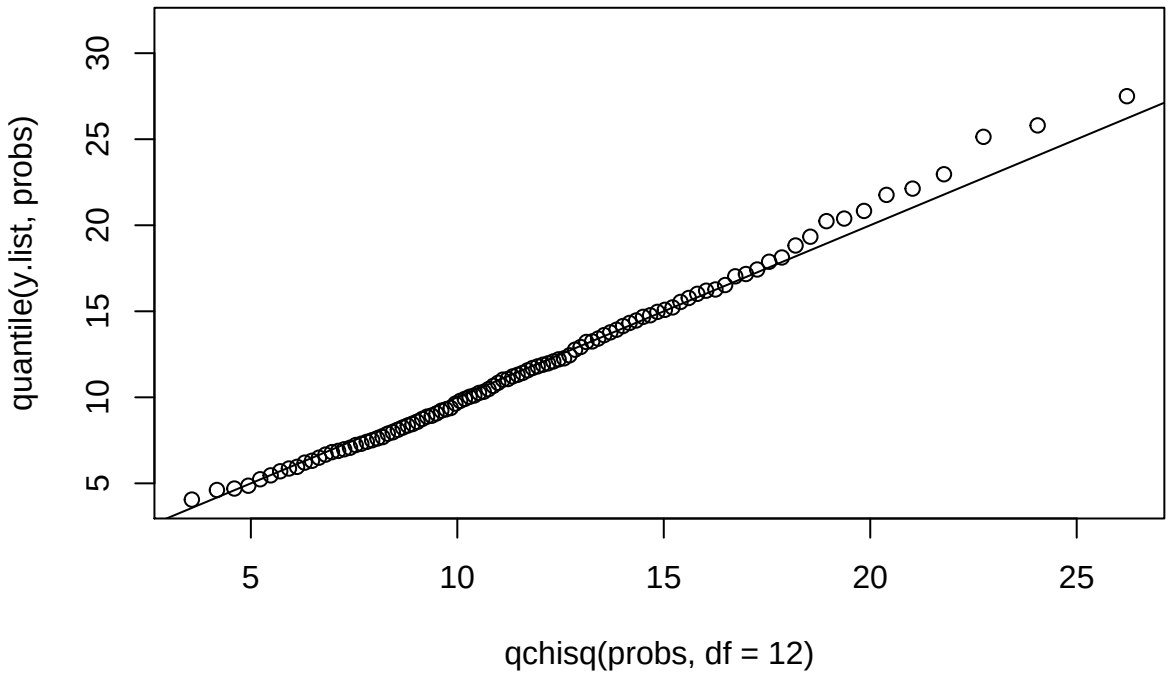
a.

```
norm.gen = function(n, mu=0, sigma=1){
  odd = (n %% 2) != 0
  if(odd==TRUE){
   n = n+1
   }
   u1 = runif(n/2)
   u2 = runif(n/2)
   t = sqrt(-2*log(u1))
   z1 = t*cos(2*pi*u2)
   z2 = t*sin(2*pi*u2)
   z.list = c(z1, z2)
  if(odd){
   z.list = z.list[-n]
   }
   x.list = mu + sigma*z.list
```

```
   return(x.list)
}

chi.gen = function(n, df){
  chi.mat = matrix(norm.gen(n = n*df)^2, nrow = n, ncol = df)
  chi.list = apply(chi.mat, 1, sum)
  return(chi.list)
}

n = 500
x.list = chi.gen(n = n, df = 11)
probs = seq(from = 0.01, to = 1, by = 0.01)
plot(qchisq(probs, df = 11), quantile(x.list, probs))
abline(0, 1)
```




```
y.list = chi.gen(n = n, df = 12)
probs = seq(from = 0.01, to = 1, by = 0.01)
plot(qchisq(probs, df = 12), quantile(y.list, probs))
abline(0, 1)
```

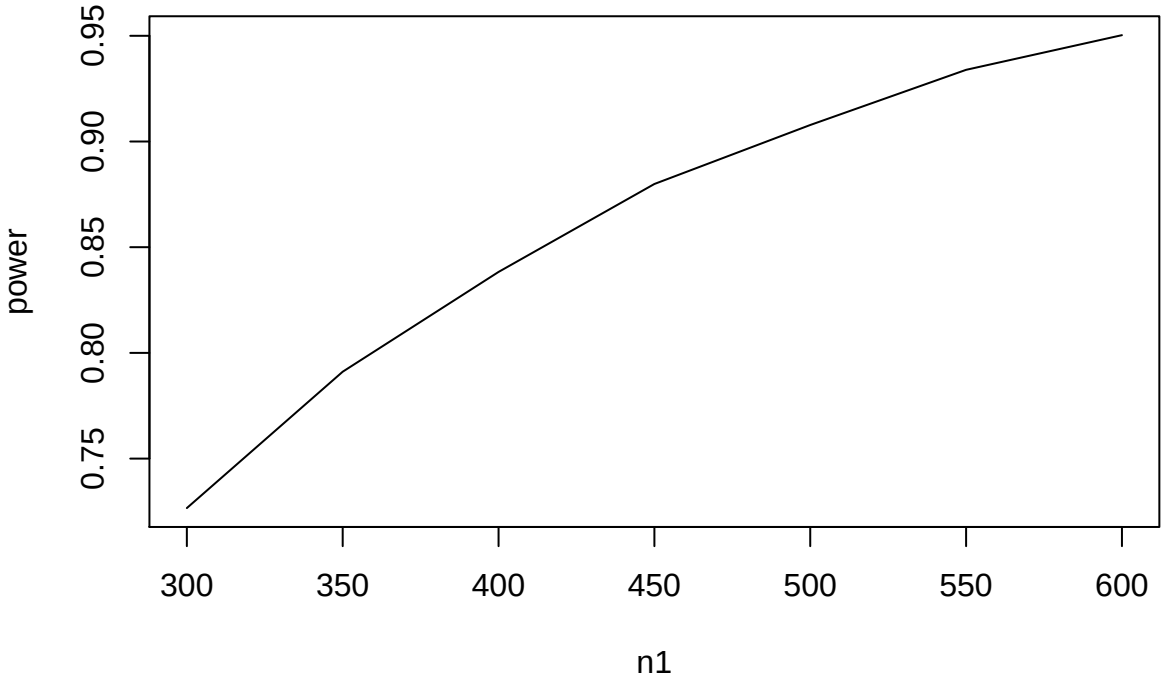


It seems that the most of the points are on the line. But it is not so good.

b.

```
norm.gen = function(n, mu=0, sigma=1){
  odd = (n %% 2) != 0
  if(odd==TRUE){
   n = n+1
   }
   u1 = runif(n/2)
   u2 = runif(n/2)
   t = sqrt(-2*log(u1))
   z1 = t*cos(2*pi*u2)
   z2 = t*sin(2*pi*u2)
   z.list = c(z1, z2)
  if(odd){
   z.list = z.list[-n]
   }
   x.list = mu + sigma*z.list
   return(x.list)
}

chi.gen = function(n, df){
  chi.mat = matrix(norm.gen(n = n*df)^2, nrow = n, ncol = df)
  chi.list = apply(chi.mat, 1, sum)
  return(chi.list)
}

pval.dist.sim = function(n1, n2, reps=1e4){
  pval.list = numeric(reps)
  for(r in 1:reps){
```

```
    x.list = chi.gen(n = n1,df = 11)
    y.list = chi.gen(n = n2,df = 12)
    xbar = mean(x.list)
    ybar = mean(y.list)
    residuals = c(x.list - xbar, y.list - ybar)
    sp.sq = sum(residuals^2)/(n1+n2-2)
    t = (xbar - ybar)/sqrt(sp.sq*(1/n1+1/n2))
    pval.list[r] = 2*pt(-abs(t),n1+n2-2)
  }
  return(pval.list)
}
n.list = seq(from = 300, to = 600, by = 50)
est.power = numeric(length(n.list))
for(j in 1:length(n.list)){
  pvals = pval.dist.sim(n1 = n.list[j],n2 = n.list[j])
  est.power[j]=mean(pvals<0.05)
}
plot(n.list,est.power,t="l",xlab="n1",ylab="power")
```



At around about 500-520, it may get the 90%.

Problem 7

a.

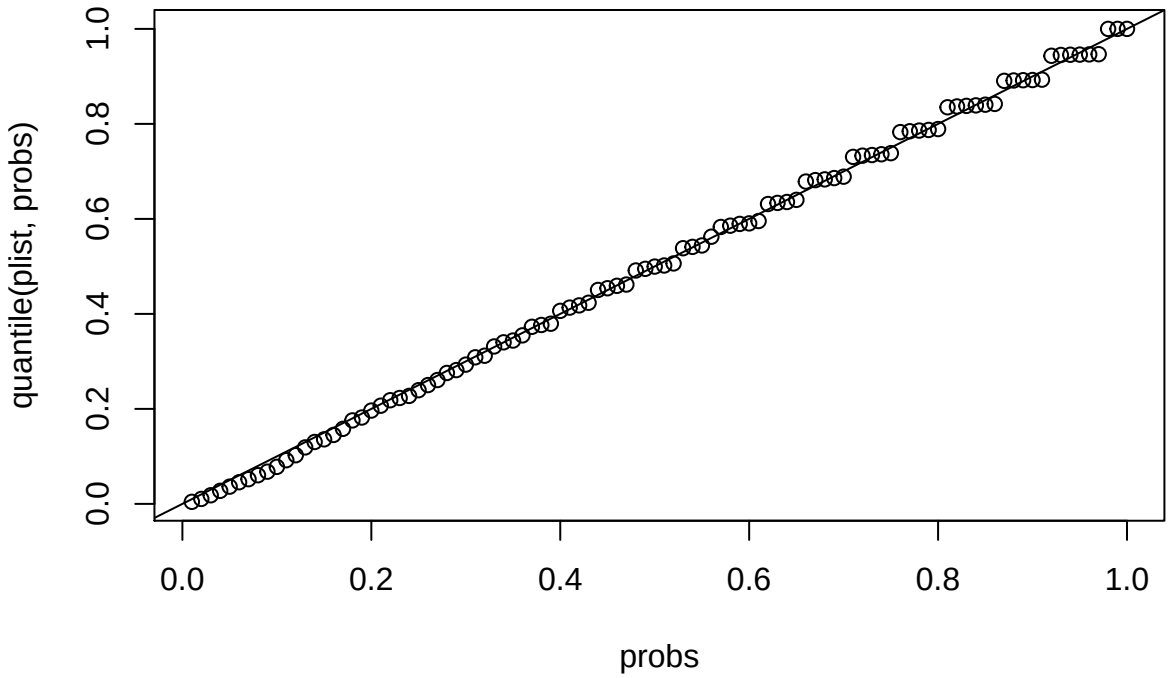
```
propHypothesisTest = function(n,theta1,theta2,reps){

    theta.est = numeric(reps);theta.est1 = numeric(reps); theta.est2 = numeric(reps)
    t.list = numeric(reps)
    pval.list = numeric(reps)
```

```
    u.mat1 = matrix(runif(reps*n),reps,n)
    ber.mat1 = ifelse(u.mat1<theta1,1,0)
    bin.mat1 = apply(ber.mat1,1,sum)
    u.mat2 = matrix(runif(reps*n),reps,n)
    ber.mat2 = ifelse(u.mat2<theta2,1,0)
    bin.mat2 = apply(ber.mat2,1,sum)

    for(i in 1:reps){
    theta.est1[i] = bin.mat1[i]/n
    theta.est2[i] = bin.mat2[i]/n
    theta.est[i] = (bin.mat1[i]+bin.mat2[i])/(2*n)
    t.list[i] = (theta.est1[i]-theta.est2[i])/sqrt(2*theta.est[i]*(1-theta.est[i])/n)
    pval.list[i] = 2*pnorm(-abs(t.list[i]),mean = 0, sd = 1)
    }
    return(pval.list)

}
mean(propHypothesisTest(500,0.68,0.68,1e4)<0.03)
```

```
## [1] 0.0301
```

b.

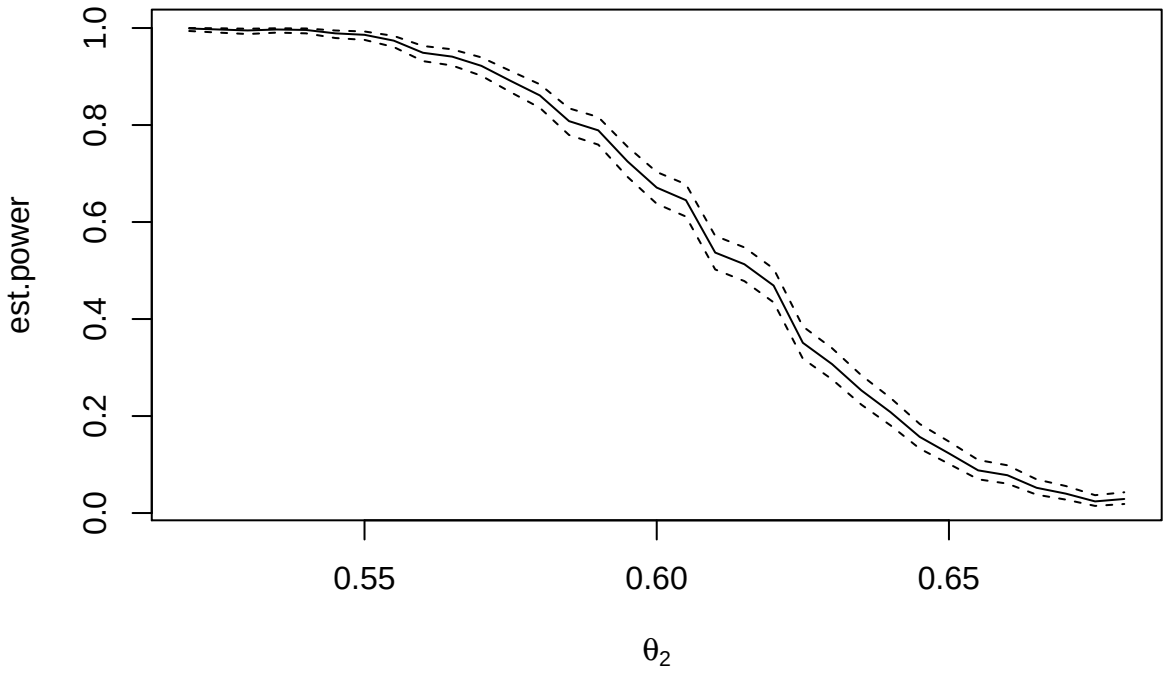
```
plist = propHypothesisTest(n = 500, theta1 = 0.68, theta2 = 0.68, reps = 1000)
probs = seq(from = 0.01, to = 1, by = 0.01)
plot(probs, quantile(plist,probs))
abline(0,1)
```



We can find that it is not very suitable, if we increase our sample size, it may help.

c.

```
theta.list = seq(from = 0.52, to = 0.68, by = 0.005)
est.power = numeric(length(theta.list))
LB.list = numeric(length(theta.list))
UB.list = numeric(length(theta.list))
for(j in 1:length(theta.list))
{
  pvals = propHypothesisTest(n = 500, theta1 = 0.68, theta2 = theta.list[j],reps = 1000)
  est.power[j] = mean(pvals < 0.03)
  bounds = binom.test(x=sum(pvals<0.03),n = 1000, conf.level = 0.97)$conf.int[1:2]
  LB.list[j] = bounds[1]
  UB.list[j] = bounds[2]
}
plot(theta.list, est.power, t="l", xlab = expression(theta[2]),ylab = "Power"))
lines(theta.list,LB.list, lty = 2)
lines(theta.list,UB.list, lty = 2)
```



Problem 8

a.

```
gen.x.data = function(n, mu1 = 68, sigma = sqrt(2), rho = 1){
  a.list = rnorm(n = n , mean = 0, sd = sqrt(rho)*sigma)
  x.list = mu1 + a.list + rnorm(n = n, mean = 0, sd = 1)
  return(list(x.list = x.list))
}
est.var.x = gen.x.data(n = 50000)
t.test((est.var.x$x.list - 68)^2, conf.level = 0.99)$conf.int[1:2]
```

```
## [1] 2.94479 3.04221
```

So, we can see that 3 was captured in the CI.

b.

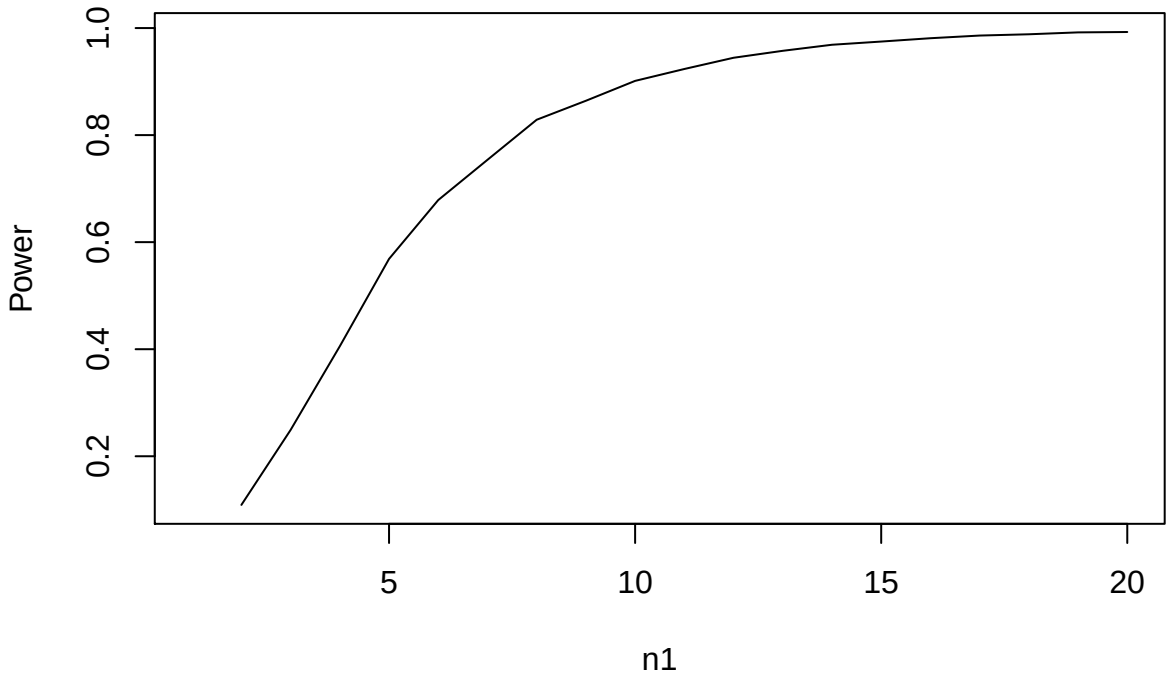
```
gen.y.data = function(n, mu2 = 70, sigma = sqrt(2), rho = 1){
  a.list = rnorm(n = n , mean = 0, sd = sqrt(rho)*sigma)
  y.list = mu2 + a.list + rt(n = n, 4)
  return(list(y.list = y.list))
}
est.var.y = gen.y.data(n = 50000)
t.test((est.var.y$y.list - 70)^2, conf.level = 0.99)$conf.int[1:2]
```

```
## [1] 3.948533 4.163711
```

So, as we can see, 4 is captured in the CI.

c.

```
gen.paired.data = function(n, mu1 = 68,mu2 = 70, sigma = sqrt(2), rho = 1){
  a.list = rnorm(n = n , mean = 0, sd = 1)
  x.list = mu1 + a.list + rnorm(n = n, mean = 0, sd = 1)
  y.list = mu2 + a.list + rt(n = n, 4)
  return(list(x.list=x.list, y.list=y.list))
}

gen.paired.t.test.pvals = function(n, mu1 = 68,mu2 = 70, sigma = sqrt(2), rho = 1,reps = 10000 ){
  pvals.list = numeric(reps)
  for(r in 1:reps){
    pdat = gen.paired.data(n = n, mu1 = mu1, mu2 = mu2, sigma = sqrt(2), rho = 1)
    z.list = pdat$x.list - pdat$y.list
    t.stat = (mean(z.list) - 0)/(sd(z.list)/sqrt(n))
    pvals.list[r] = 2*pt(-abs(t.stat),n-1)
  }
  return(pvals.list)
}
reps = 10000
n.list = seq(from = 1, to = 20)
est.power = numeric(length(n.list))
LB.list = numeric(length(n.list))
UB.list = numeric(length(n.list))
for(j in 1:length(n.list)){
  pvals = gen.paired.t.test.pvals(n = n.list[j],mu1 = 68,mu2 = 70, sigma = sqrt(2),rho = 1)
  est.power[j] = mean(pvals<0.05)
}
plot(n.list, est.power, t="l", xlab = "n1",ylab = "Power")
```




```
# lines(n.list, LB.list, lty=2)
# lines(n.list, UB.list, lty=2)
```

We can approximately find that when n is about 11