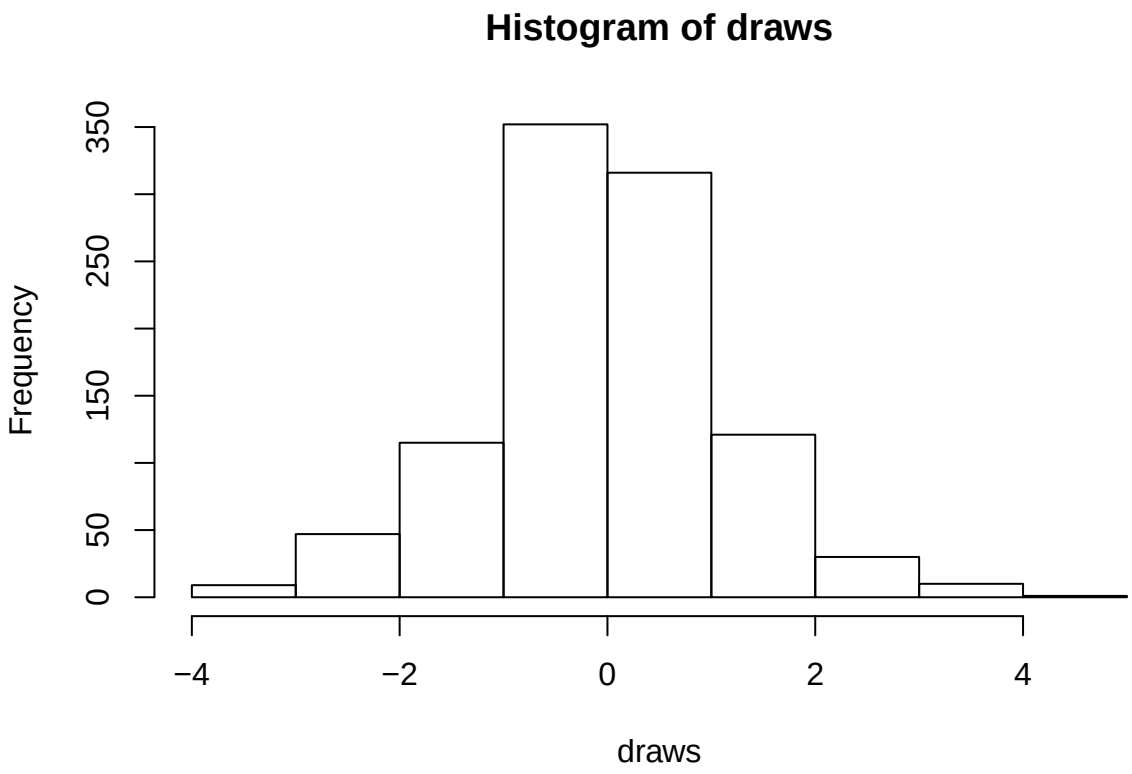
HW02

Zach White

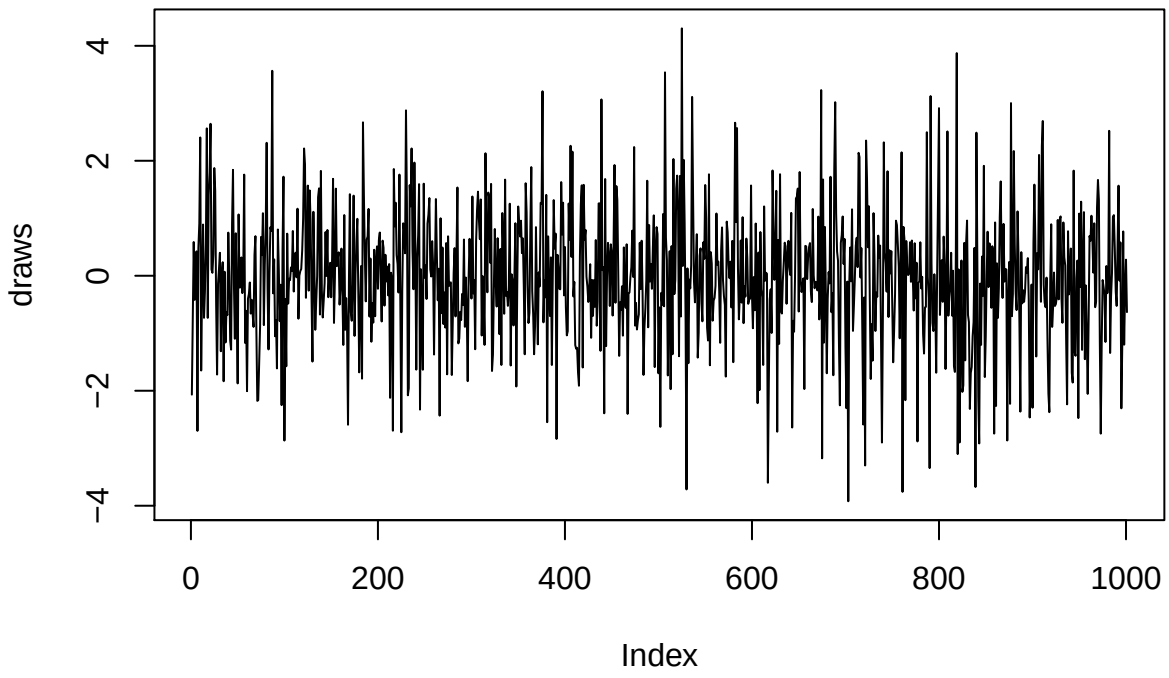
February 10, 2017

Problem 2.29B

```
nreps = 10000
draws = rep(NA,nreps)
for(i in 1:nreps){
  theta.draw = rcauchy(1,0,1)
  x = rnorm(1,theta.draw,1)
  u = runif(1,0,1)
  post = (1/(pi*(1+theta.draw^2)))*(1/(2*pi))*exp((-(x-theta.draw)^2)/2)
  if(u <= post){draws[i] = theta.draw}
}

post = function(x,t){
  1/(1+t*t)*exp(-(x-t)^2/2)
}
draws = NULL
theta.0 = 0
acc = 0
tot = 0
n = 1
while(length(draws) <= 1000){
  data = rnorm(n,theta.0,1)
  x.bar = mean(data)
  M = n * dnorm(x.bar,theta.0,sd = 1)
  prior = rcauchy(1)
  u = runif(1,0,1)
  num = post(x = data, t = prior)
  ratio = num / (M*dcauchy(prior))
  if(u <= ratio){
    draws = c(draws,prior)
    acc+1
  }
  tot = tot + 1
}

hist(draws)
```



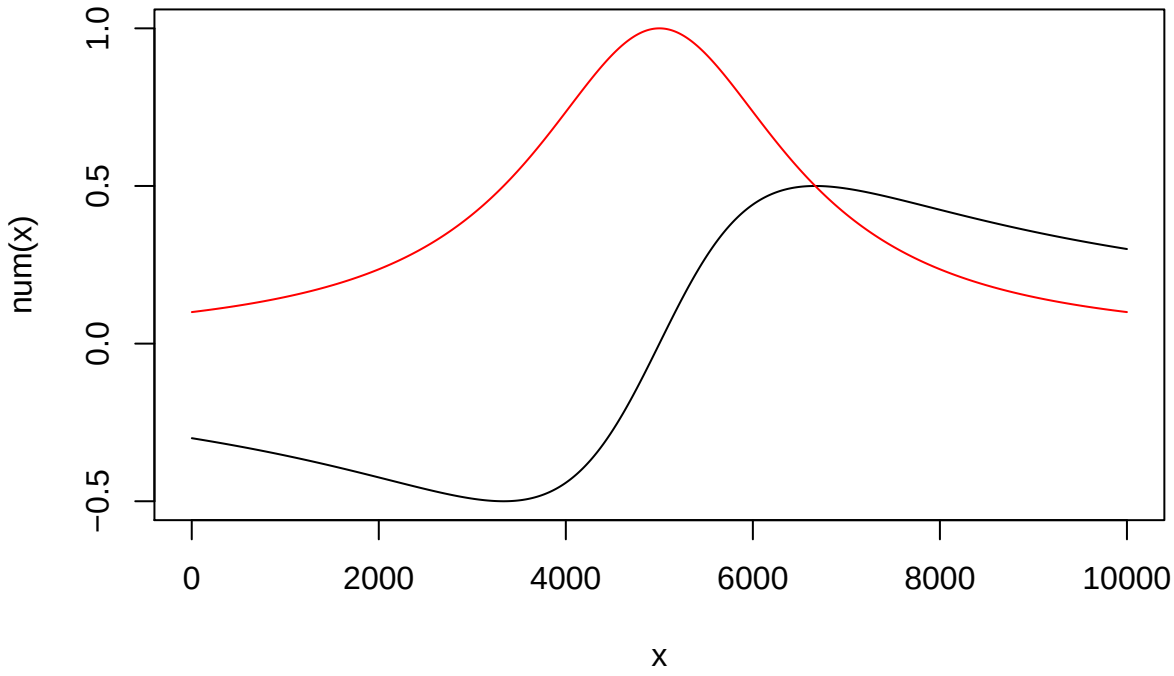

```
plot(draws,type = "l")
```



This code generates draws from the psterior distribution. However, it is important to note that this works under the assumption that we have just 1 data point x. If we have more, then I would need to modify this code a little bit more.

Problem 3.1

```
num = function(theta){(theta/(1+theta^2)) * exp(-(1/2)*(x-theta)^2)}
denom = function(theta){(1/(1+theta^2)) * exp(-(1/2)*(x-theta)^2)}
x = seq(-3,3,length= 10000)
plot(num(x), type = "l", ylim = c(-.5,1), xlab = "x")
lines(denom(x), type = "l", col = "red",xlab = "x")
```

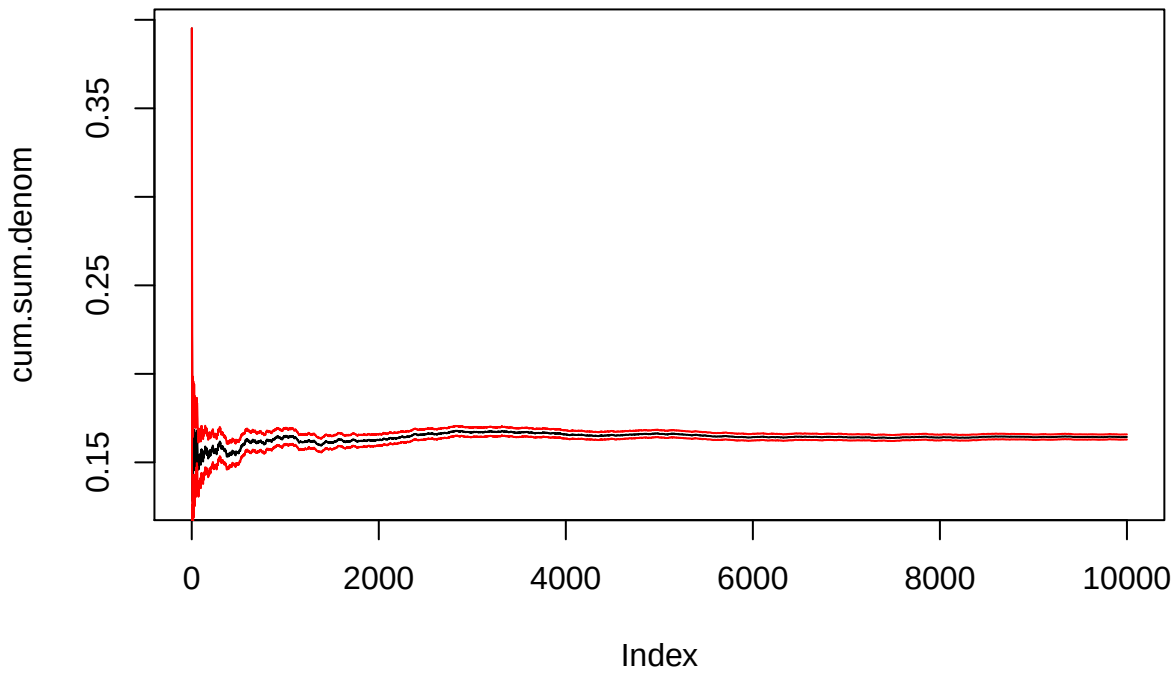


The red line shows the numerator while the black shows the denominator. It is clear that this estimator is the posterior mean.

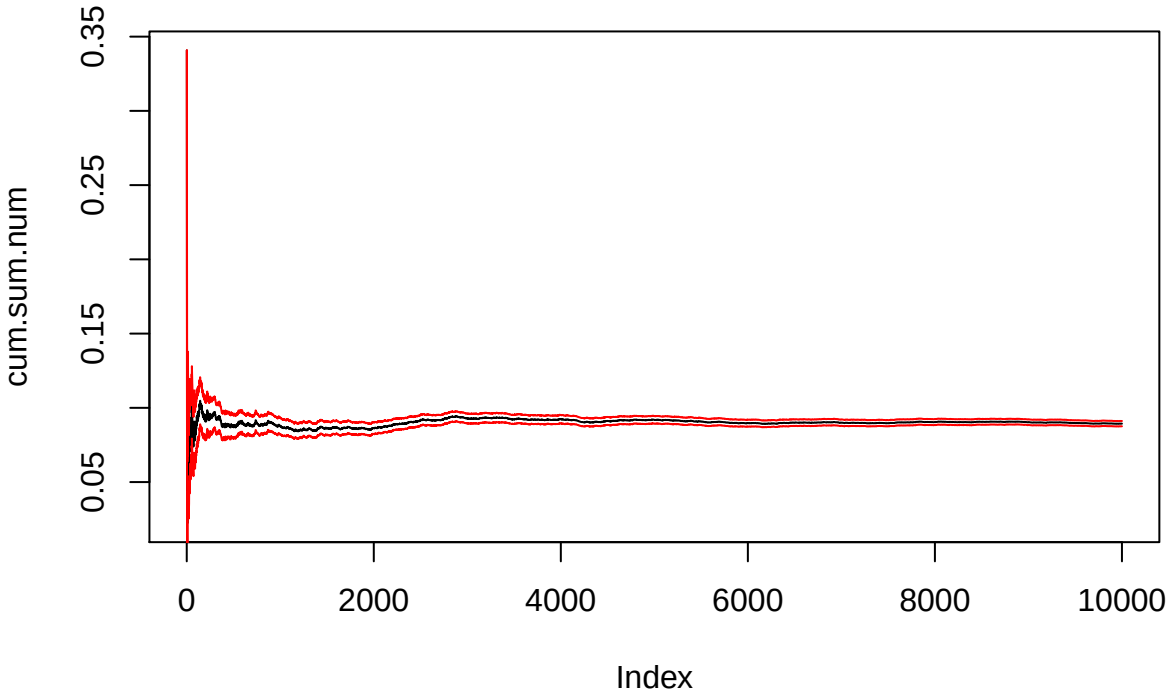
Part B

I now show the convergence of each of these integrands. The appropriate sample size to ensure is relatively easy to calculate, and I will show the closed form solution after some simulations and plots.

```
# Part B
x = 1
nreps = 10000
cauchy.draws = rcauchy(nreps,0,1)
eval.denom = dnorm(cauchy.draws,x)
cum.sum.denom = cumsum(eval.denom) / (1:nreps)
stand.error = sqrt(cumsum((eval.denom - cum.sum.denom)^2))/(1:nreps)
plot(cum.sum.denom, type = "l")
lines(cum.sum.denom + stand.error, col = "red")
lines(cum.sum.denom - stand.error, col = "red")
```




```
eval.num = cauchy.draws * eval.denom
cum.sum.num = cumsum(eval.num) / (1:nreps)
num.stand.error = sqrt(cumsum((eval.num - cum.sum.num)^2))/(1:nreps)
plot(cum.sum.num, type = "l")
lines(cum.sum.num + num.stand.error, col = "red")
lines(cum.sum.num - num.stand.error, col = "red")
```



In this case, the lines represent the standard error, and it clear that the accuracy increases as the number of iterations increases. The following shows the calculation to find the approximate n necessary to get accuracy

$$
\begin{aligned}
2se &<= .001 \\
2(\frac{\hat{\sigma}}{\sqrt{n}}) &<= .001 \\
2000(\hat{\sigma}) &<= \sqrt{n} \\
4e^{-06}\hat{\sigma} &<= n
\end{aligned}
$$

Problem 3.2

Part A

We will continue under the assumption that $x = 1$ and θ_0. For the accept-reject algorithm, we will use a cauchy candidate, as desired. So $g(\theta) = \frac{1}{\pi(1+\theta^2)}$, and we know that by the algorithm, with a $U \sim unif(0,1)$, if $U < \frac{g(Y|x)}{Mg(Y)}$, we accept the proposed $Y \sim g$ We need to find M.

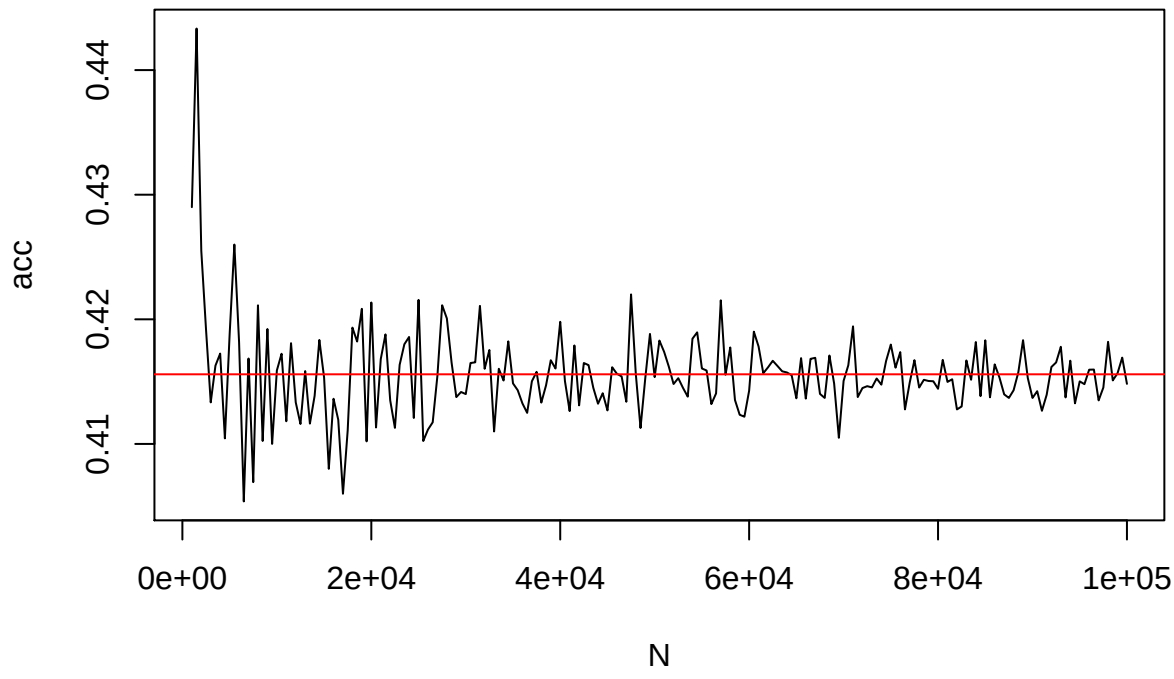
```
post.func = function(t){
  x = 1
  1/(1+t*t)*exp(-(x-t)^2/2)
}
test.dnorm = function(x){

}
```

```
c = integrate(post.func,-Inf,Inf)$value
m = pi/c
```

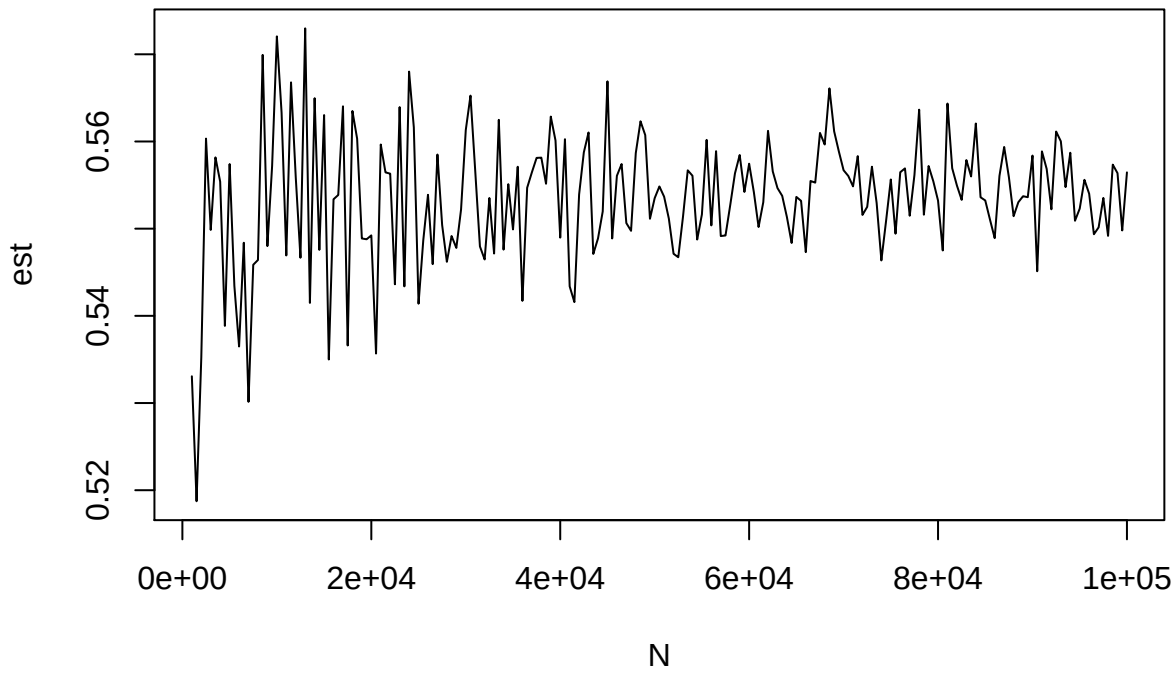
To do this, we recognize that $\frac{g(Y|x)}{Mg(Y)} = \frac{\pi}{CM} \exp -\frac{1}{2}(x-y)^2$, with $CM = C \sup \frac{g(\theta|x)}{g(\theta)} = Csup\frac{\pi}{C} \exp -\frac{1}{2}(x-\theta)^2$, which is π and occurs when $\theta = x$. So clearly $M = \frac{\pi}{C}$, which we can calculate to be 1.3056085. Thus $M = 2.4062287$ given our predetermined parameters, and it is important to note that under these conditions, that the probability of acceptance is $\frac{1}{M} = \frac{C}{\pi} = 0.4156$

I will do this under multiple different sample sizes to compare the acceptance probability

```
N = seq(1000,100000,by = 500)
x = 1
est = rep(NA,length(N))
acc = rep(NA,length(N))
for(i in 1:length(N)){
  n = N[i]
  y = rcauchy(n)
  u = runif(n,0,1)
  acceptance = (u < exp(-(x-y)^2/2))
  acc[i] = mean(acceptance)
  est[i] = mean(y[acceptance])
}
plot(N,acc, type = "l")
abline(h = (1/m), col = "red")
```




```
plot(N,est,type = "l")
```




```
M <- seq(1000,100000,by=1000) # number of simulated values
x <- 3 # observed data
ar.es <- rep(NA,length(M))
fr.acc <- rep(NA,length(M))

for(i in 1:length(M)){
  m <- M[i]
  y <- rcauchy(m) # simulate m standard cauchys
  u <- runif(m,0,1) # simulate m uniform(0,1) RVs
  accept <- (u<exp(-(x-y)^2/2)) # test if we accept or not
  ar.es[i] <- mean(y[accept])
  fr.acc[i] <- mean(accept)
}
```

Part B

This one will be on my own.

Problem 3.3

Part A

Let $Z \sim N(0,1)$. Ultimately, we will want to use monte carlo sums based on indicator variables, but beforew we can do that, we will go over some helpful preliminary results. We want to calculate $\Pr(Z > 2.5)$, which is $\int_{2.5}^{\infty} f(z)dz = 1 - \Phi(2.5) = I(-2.5) = \int_{-\infty}^{-2.5} f(z)dz$. We can do this because of the symmetry o the normal distribution. Here, we will introduce the Monte Carlo sums $\hat{\Phi}(-2.5) = \frac{1}{n}\sum_{i=1}^{n} 1_{z_i \leq -2.5}$. Now this is the summation of Bernoulli random variables, which means that we can leverage to find apropriate n using the exact variance of the estimator, which is $var(\hat{\Phi}) = \frac{\Phi(-2.5)(1-\Phi(-2.5))}{n}$. We will use this to find the required sample size for the accuracy to be less than .001.

$$
\begin{aligned}
2\text{qnorm}(.975)\sqrt{\frac{\Phi(-2.5)(1-\Phi(-2.5))}{n}} &<= .001 \\
\sqrt{\frac{\Phi(-2.5)(1-\Phi(-2.5))}{n}} &<= .000255 \\
\frac{\Phi(-2.5)(1-\Phi(-2.5))}{(.000255)^2} &<= n
\end{aligned}
$$

```
req.n = ceiling(pnorm(-2.5)*(1-pnorm(-2.5)) / (.000255)^2)
```

This means that it would require roughly 95000 samples for that level of accuracy, which is a lot.

```
z = rnorm(req.n)
phi.est = (1/n)*sum(z <= -2.5)
abs.diff = abs(pnorm(-2.5)-phi.est)
```

This has a very small difference .00051. Our sample size estimate in fact is probably too conservative because this is a very smal difference. I will test to see if on average this is too conservative

```
diff = rep(NA,10000)
for(i in 1:10000){
  z = rnorm(req.n)
  phi.est = (1/req.n)*sum(z <= -2.5)
  abs.diff = abs(pnorm(-2.5)-phi.est)
  if(abs.diff <= .001){
    diff[i] = 1
  }
  else{
    diff[i] = 0
}
}
mean(diff)
```

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

According to this, our estimated required sample size is probably too conservative.

Part B

We now do an almost identical process with a gamma distribution.

$$\begin{aligned} \text{qnorm}(.975)\sqrt{\frac{\Pr(X > 5.3)(1 - \Pr(X > 5.3))}{n}} &<= .001 \\ \sqrt{\frac{\Pr(X > 5.3)(1 - \Pr(X > 5.3))}{n}} &<= .00051 \\ \frac{\Pr(X > 5.3)(1 - \Pr(X > 5.3))}{.00051^2} &<= n \end{aligned}$$

```
gam.req.n = ceiling(pgamma(5.3,1,1)*(1-pgamma(5.3,1,1))/(.00051^2))
```

According to this, the required sample size would be approximately 19100.

```
diff = rep(NA,10000)
for(i in 1:10000){
  gam = rgamma(gam.req.n,1,1)
  gam.est = (1/gam.req.n)*sum(gam > 5.3)
  abs.diff = abs((1-pgamma(5.3,1,1))-gam.est)
  if(abs.diff <= .001){
    diff[i] = 1
  }
  else{
    diff[i] = 0
}
}
mean(diff)
```

```
## [1] 0.9494
```

We can see by the mean that we are getting about what we expected and planned.

Problem 3.7

We use both the multinomal and two-binomial model to examine the association between disinfectant and surgical success rates from Joseph Lister's experiment.

```
success = c(34,19)
failure = c(6,16)
list.data = rbind(success,failure)
colnames(list.data) = c("dis","notdis")
n = sum(list.data)
N = 50000
list.data = as.data.frame(list.data)

# I will start with the two-binomial model
n1 = sum(list.data$dis)
n2 = sum(list.data$notdis)
two.binlam = rep(0,N)
for(i in 1:N){
  p = runif(1,0,1)
  k1 = rbinom(1,n1,p)
  k2 = rbinom(1,n2,p)
```

```
  null.data = data.frame(rbind(c(k1,n1-k1),c(k2,n2-k2)))
  two.binlam[i] = likelihood.test(null.data)$statistic
}

# I will now go over the multinomial model
multi.lam = rep(0,N)
for(i in 1:N){
  p = runif(2,0,1)
  vec.p = c(p[1]*p[2],p[1]*(1-p[2]),(1-p[1])*p[2],(1-p[1])*(1-p[2]))
  k = rmultinom(1,n,vec.p)
  null.data = data.frame(rbind(k[1:2],k[3:4]))
  multi.lam = likelihood.test(null.data)$statistic
}
quantile(two.binlam,c(.9,.95,.99))
```

```
##      90%      95%      99%
## 2.769113 3.880865 6.752916
```

```
quantile(multi.lam, c(.9,.95,.99))
```

```
##      90%      95%      99%
## 1.757154 1.757154 1.757154
```

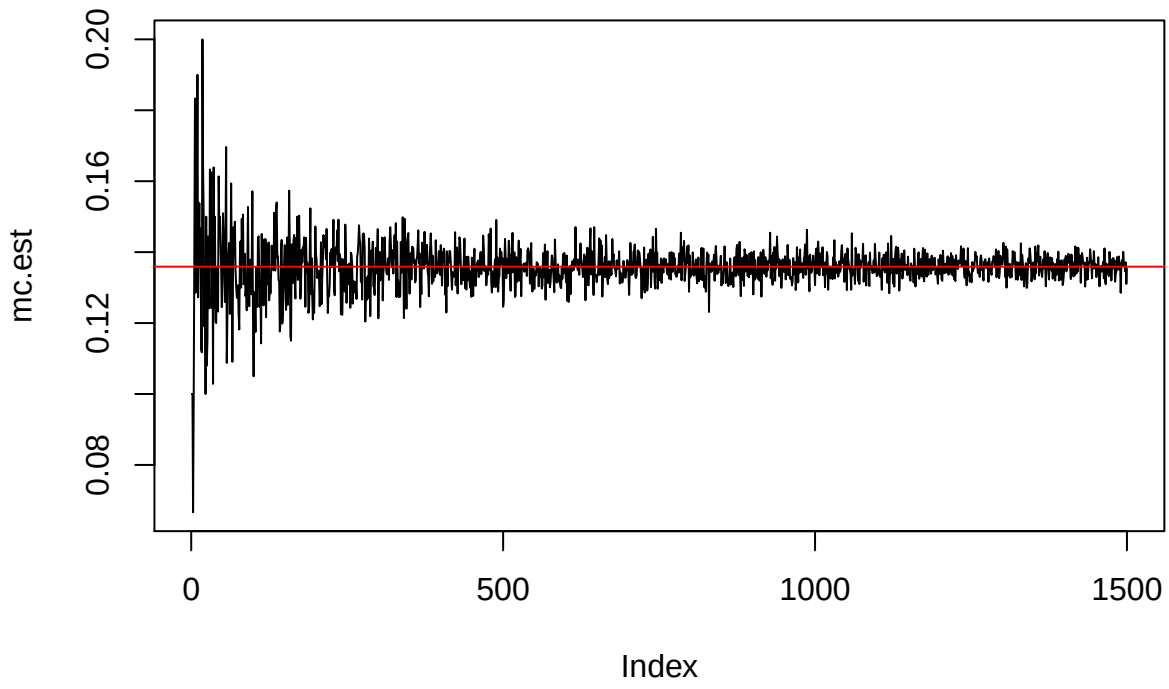
```
chi = c(qchisq(.9,df = 1),qchisq(.95,df = 1),qchisq(.99,df = 1))
```

We can reject the null hypothesis and conclude that the disinfectant does have some bearing on the surgical success of operations.

Problem 3.10

In this problem, we will analyze the performance of the regular Monte Carlo estimator with that of another estimator based on an optimal choice of "instrumental distribution" described in the text. The regular Monte Carlo estimator is $\int_1^2 \frac{e^{-x^2/2}}{\sqrt{2\pi}} = \Phi(2) - \Phi(1)$.

```
N = seq(10,15000, by = 10)
mc.est = rep(NA,length(N))
for(i in 1:length(N)){
  n = N[i]
  x = rnorm(n,0,1)
  mc.est[i] = sum((x>=1) & (x<=2))/n
}
real = pnorm(2,0,1) - pnorm(1,0,1)
plot(mc.est, type = "l")
abline(h = real, col = "red")
```



I now use the other estimator $\frac{\sum_{j=1}^{m} h(x_j) f(x_j)/g(x_j)}{\sum_{j=1}^{m} f(x_j)/g(x_j)} = \frac{\sum_{j=1}^{m} h(x_j)|h(x_j)|^{-1}}{\sum_{j=1}^{m} |h(x_j)|^{-1}}$

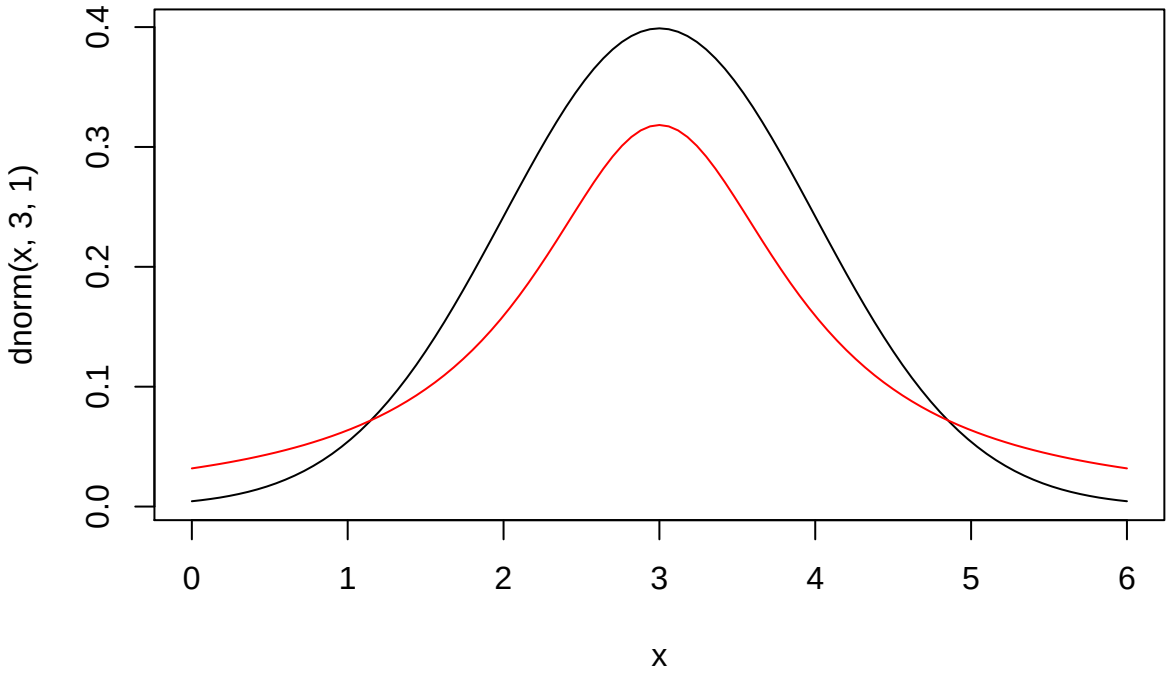
```
h <- function(x){return(exp(-x^2/2)/sqrt(2*pi))}
oth.est = rep(NA,length(N))
for(i in 1:length(N)){
  n = N[i]
  x = rtruncnorm(n,a=1,b=2,0,1)
  oth.est[i] = sum(h(x)*abs(h(x))^(-1)) / sum(abs(h(x))^(-1))
}
```

Problem 3.18

Importance sampling

To analyze the similarities between this method and importance sampling, I will draw normal draws from a cauchy proposal in both situations.

```
curve(dnorm(x,3,1), xlim = c(0,6))
curve(dcauchy(x,3,1), col = "red", add = TRUE)
```

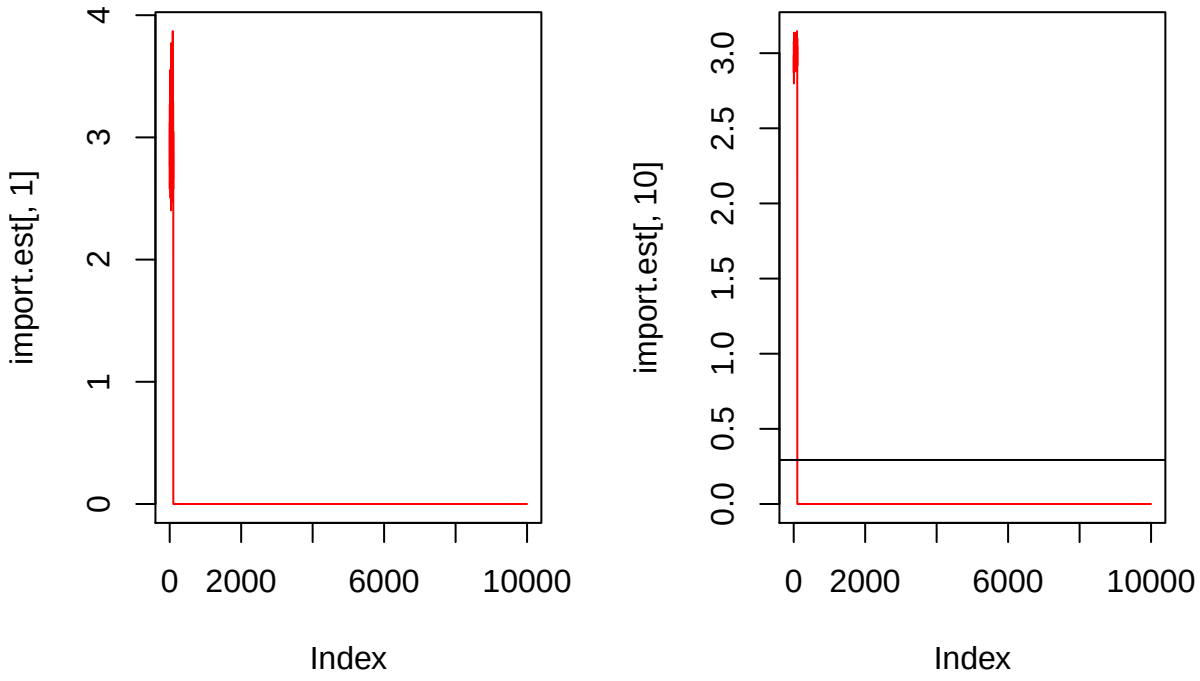



```
nreps = seq(50,1000,by = 100)
import.est = matrix(0,nrow = 10000,ncol = length(nreps))
for(j in 1:100){
  for(i in 1:length(nreps)){
    n = nreps[i]
    y = rcauchy(n,3,1)
    w = dnorm(y,3,1) / dcauchy(y,3,1)
    integrand = y * w
    import.est[j,i] = sum(integrand) / n
  }
}
```

I will be using the same distributions in this case, but I will be using the scheme and estimator discussed in the text.

```
nreps = seq(50,1000,by = 100)
book.est = rep(NA, length(nreps))
  for(j in 1:length(nreps)){
    samps = NULL
    n = nreps[j]
    while(length(samps) < nreps[j]){
    y = rcauchy(n,3,1)
    w = (dnorm(y,3,1) / dcauchy(y,3,1)) / sum((dnorm(y,3,1) / dcauchy(y,3,1)))
    draw = rbinom(n,1,w)
    samps = c(samps,y[draw])
    }
    book.est[j] = mean(samps)
```

```
  }

par(mfrow = c(1,2))
#plot(book.est[1],type = "l" , col = "black")
plot(import.est[,1],type= "l" , col = "red")
abline(h = book.est[1])

#plot(book.est[10],type = "l" , col = "black")
plot(import.est[,10],type= "l" , col = "red")
abline(h= book.est[10])
```



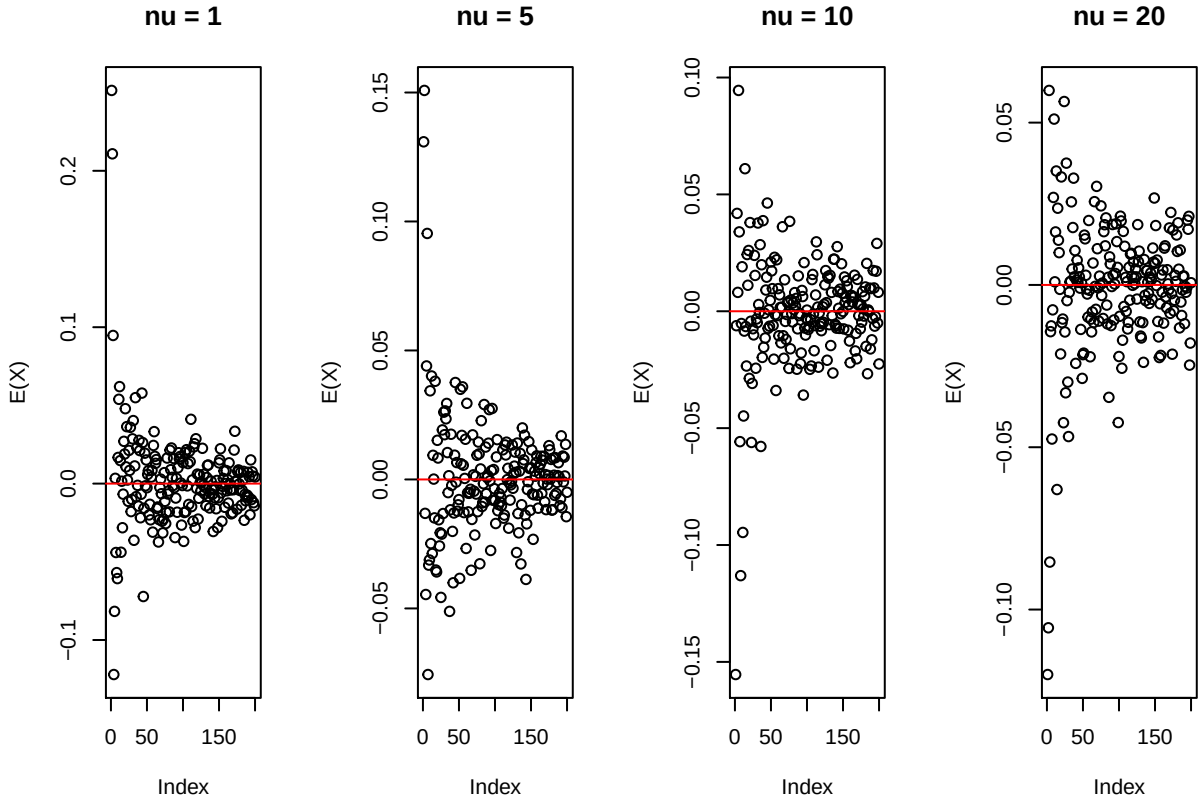

```
#plot(book.est[15],type = "l" , col = "black")
#plot(import.est[,12],type= "l" , col = "red")
#abline(h = book.est[12])
```

It's clear that these values are very similar assymptotically.

Problem 2

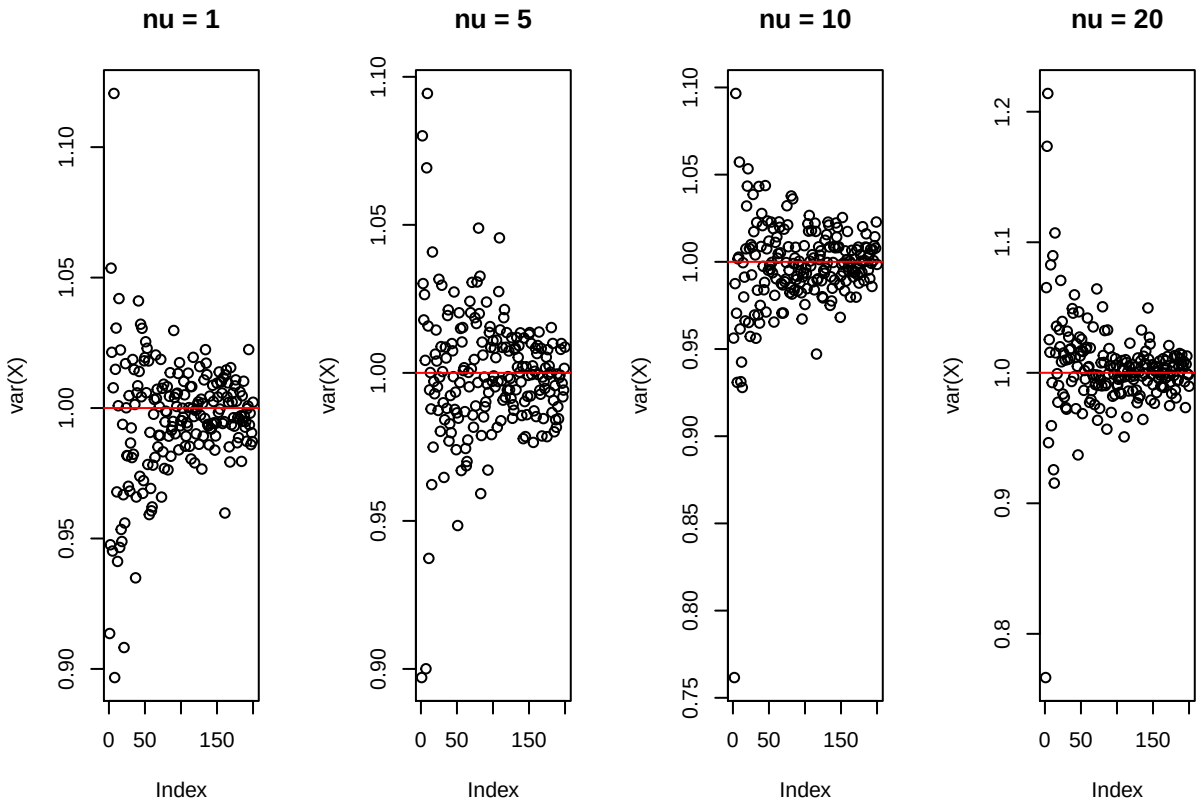
```
N = seq(50,10000, by = 50)
nu = c(1,5,10,20)
est.mean = est.var = matrix(NA,ncol = length(nu),nrow = length(N))

for(i in 1:length(nu)){
```

```
  nu.0 = nu[i]

  for(j in 1:length(N)){
    n = N[j]
    x = rt(n,nu.0)
    est.mean[j,i] = sum(x *dnorm(x,0,1) / dt(x,nu.0)) / n
    est.var[j,i] = sum((x-est.mean[j,i])^2 * dnorm(x,0,1) / dt(x,nu.0)) / n
  }
}

par(mfrow = c(1,4))
plot(est.mean[,1], main = paste(expression(nu),"= 1"), ylab = "E(X)")
abline(h = 0, col = "red")
plot(est.mean[,2], main = paste(expression(nu),"= 5"), ylab = "E(X)")
abline(h = 0, col = "red")
plot(est.mean[,3], main = paste(expression(nu),"= 10"), ylab = "E(X)")
abline(h = 0, col = "red")
plot(est.mean[,4], main = paste(expression(nu),"= 20"), ylab = "E(X)")
abline(h = 0, col = "red")
```




```
plot(est.var[,1], main = paste(expression(nu),"= 1"), ylab = "var(X)")
abline(h = 1, col = "red")
plot(est.var[,2], main = paste(expression(nu),"= 5"), ylab = "var(X)")
abline(h = 1, col = "red")
plot(est.var[,3], main = paste(expression(nu),"= 10"), ylab = "var(X)")
abline(h = 1, col = "red")
```

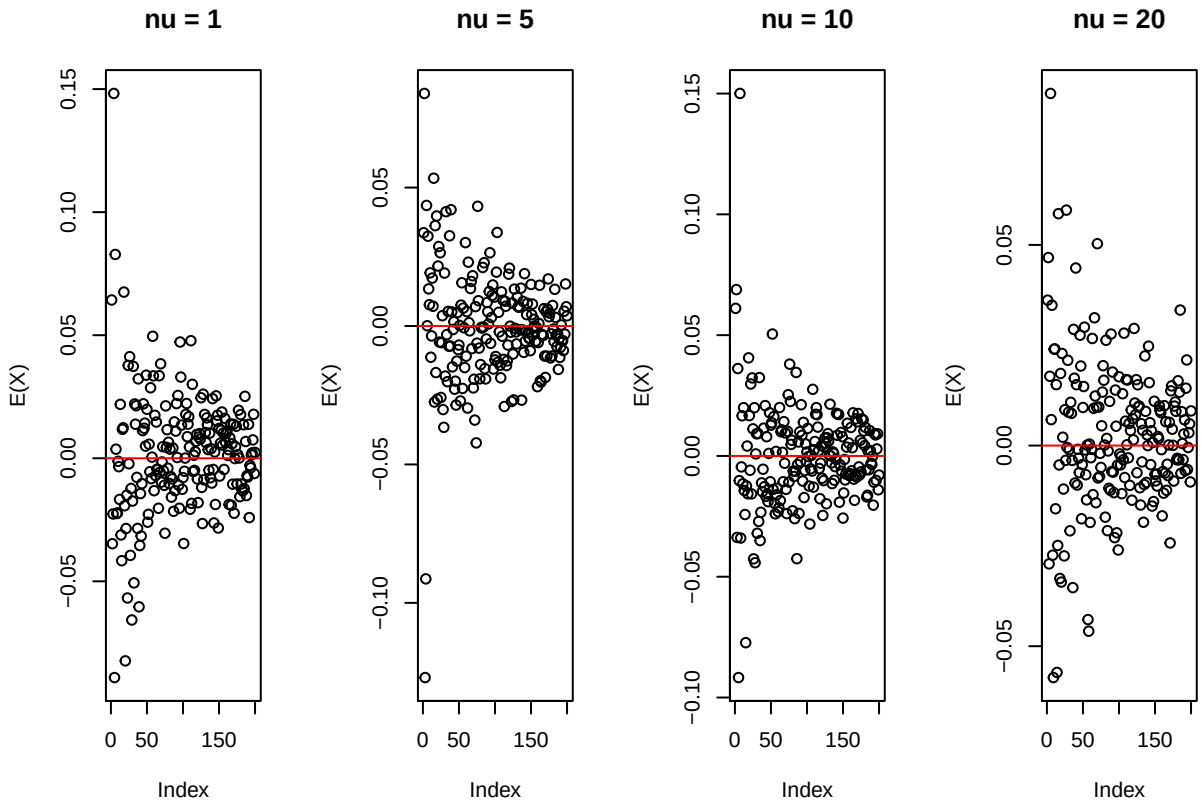
```
plot(est.var[,4], main = paste(expression(nu),"= 20"), ylab = "var(X)")
abline(h = 1, col = "red")
```



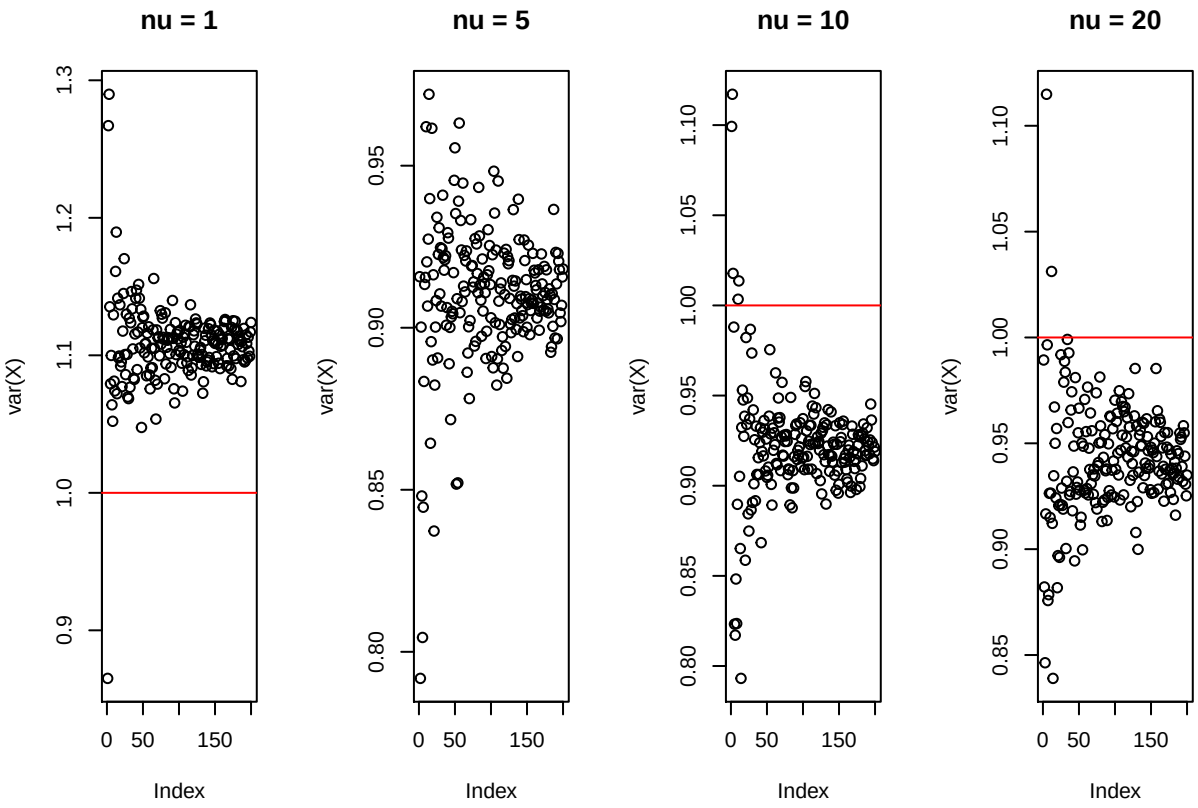
I now repeat the previous exercise with a t-distribution as the target and drawing from a normal distribution.

```
N = seq(50,10000, by = 50)
nu = c(1,5,10,20)
est.mean = est.var = matrix(NA,ncol = length(nu),nrow = length(N))

for(i in 1:length(nu)){
  nu.0 = nu[i]

  for(j in 1:length(N)){
    n = N[j]
    x = rnorm(n,0,1)
    est.mean[j,i] = sum(x *dnorm(x,0,1) / dt(x,nu.0)) / n
    est.var[j,i] = sum((x-est.mean[j,i])^2 * dnorm(x,0,1) / dt(x,nu.0)) / n
  }
}

par(mfrow = c(1,4))
plot(est.mean[,1], main = paste(expression(nu),"= 1"), ylab = "E(X)")
abline(h = 0, col = "red")
plot(est.mean[,2], main = paste(expression(nu),"= 5"), ylab = "E(X)")
abline(h = 0, col = "red")
plot(est.mean[,3], main = paste(expression(nu),"= 10"), ylab = "E(X)")
abline(h = 0, col = "red")
```

```r
plot(est.mean[,4], main = paste(expression(nu),"= 20"), ylab = "E(X)")
abline(h = 0, col = "red")
```




```r
plot(est.var[,1], main = paste(expression(nu),"= 1"), ylab = "var(X)")
abline(h = 1, col = "red")
plot(est.var[,2], main = paste(expression(nu),"= 5"), ylab = "var(X)")
abline(h = 1, col = "red")
plot(est.var[,3], main = paste(expression(nu),"= 10"), ylab = "var(X)")
abline(h = 1, col = "red")
plot(est.var[,4], main = paste(expression(nu),"= 20"), ylab = "var(X)")
abline(h = 1, col = "red")
```



This doesn't work well because the t-istribution is heavier tailed than the normal distribution. We generaly want to draw from a heavier tailed distribution.