HW02

Zach White

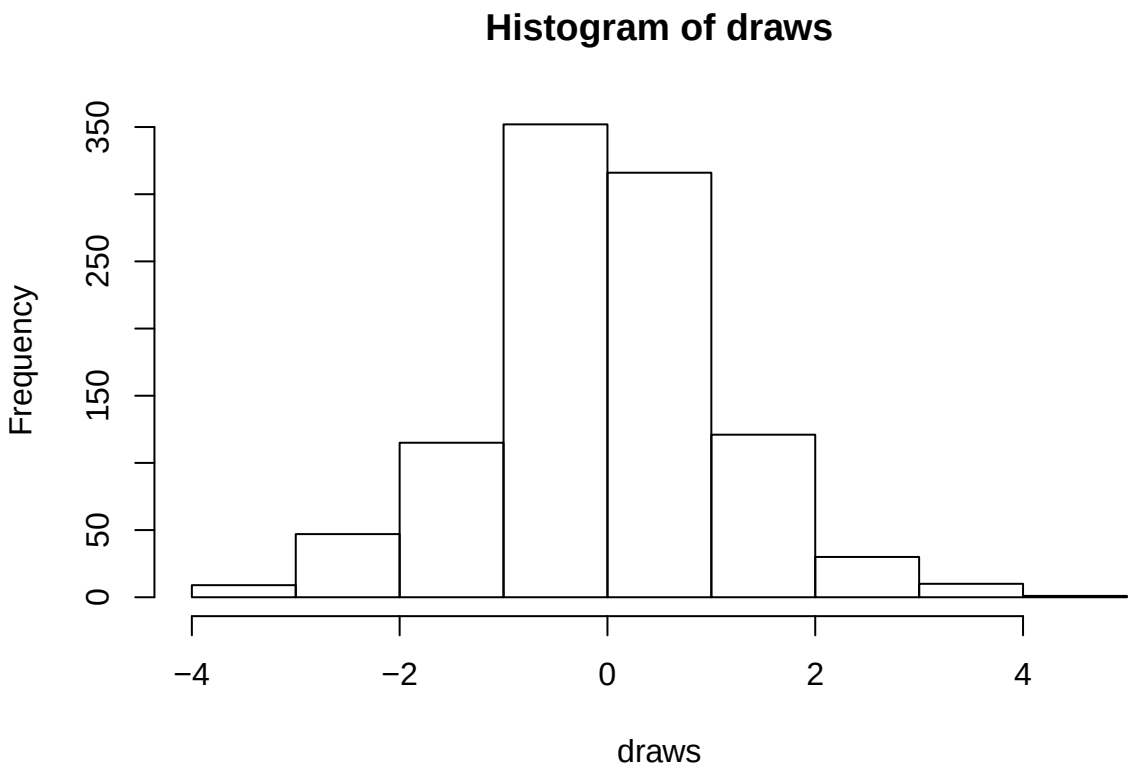
February 10, 2017

Problem 2.29B

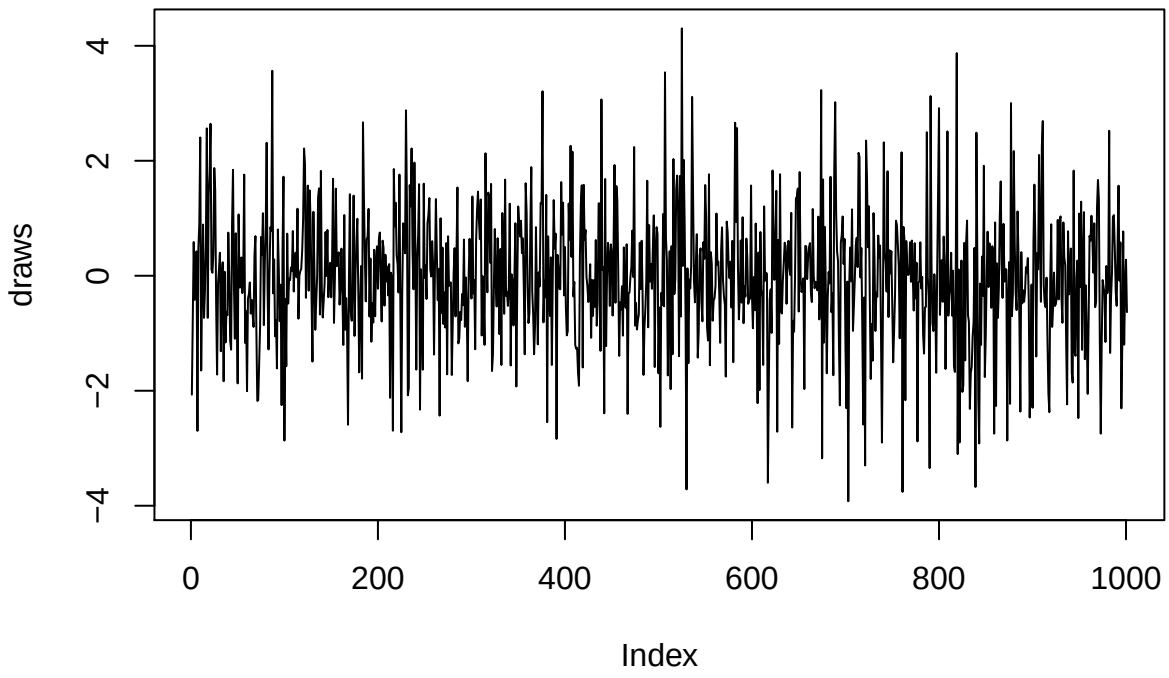
```
nreps = 10000
draws = rep(NA,nreps)
for(i in 1:nreps){
  theta.draw = rcauchy(1,0,1)
  x = rnorm(1,theta.draw,1)
  u = runif(1,0,1)
  post = (1/(pi*(1+theta.draw^2)))*(1/(2*pi))*exp((-(x-theta.draw)^2)/2)
  if(u <= post){draws[i] = theta.draw}
}

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)
```



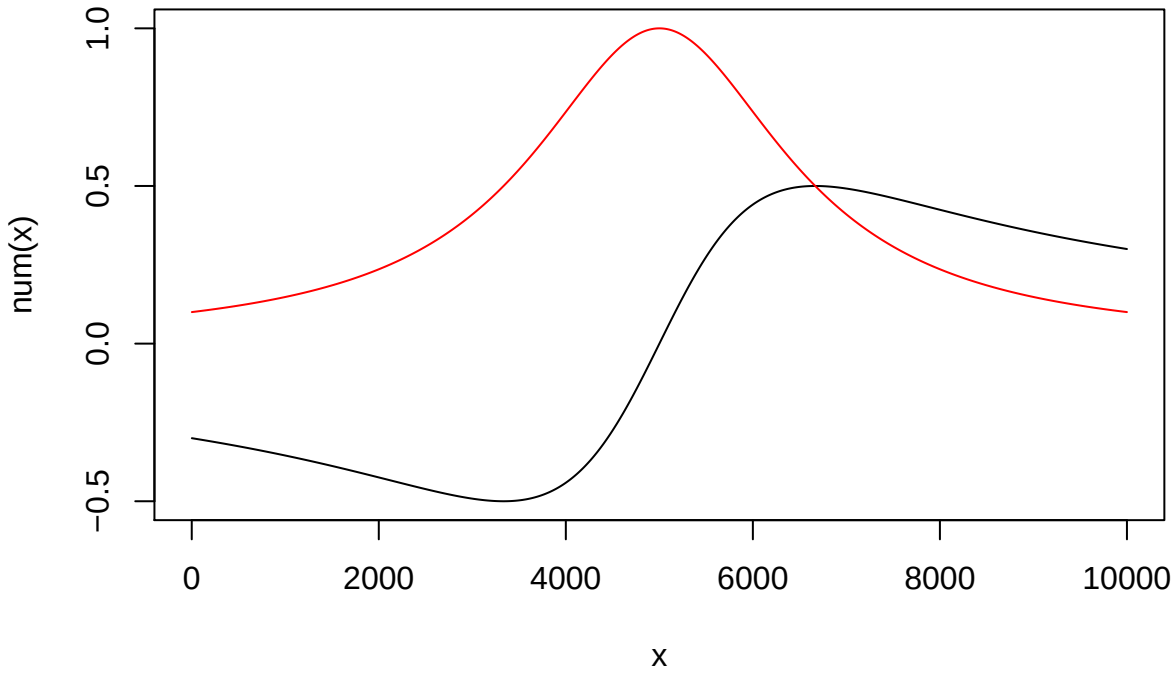

```
plot(draws,type = "l")
```



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")
```

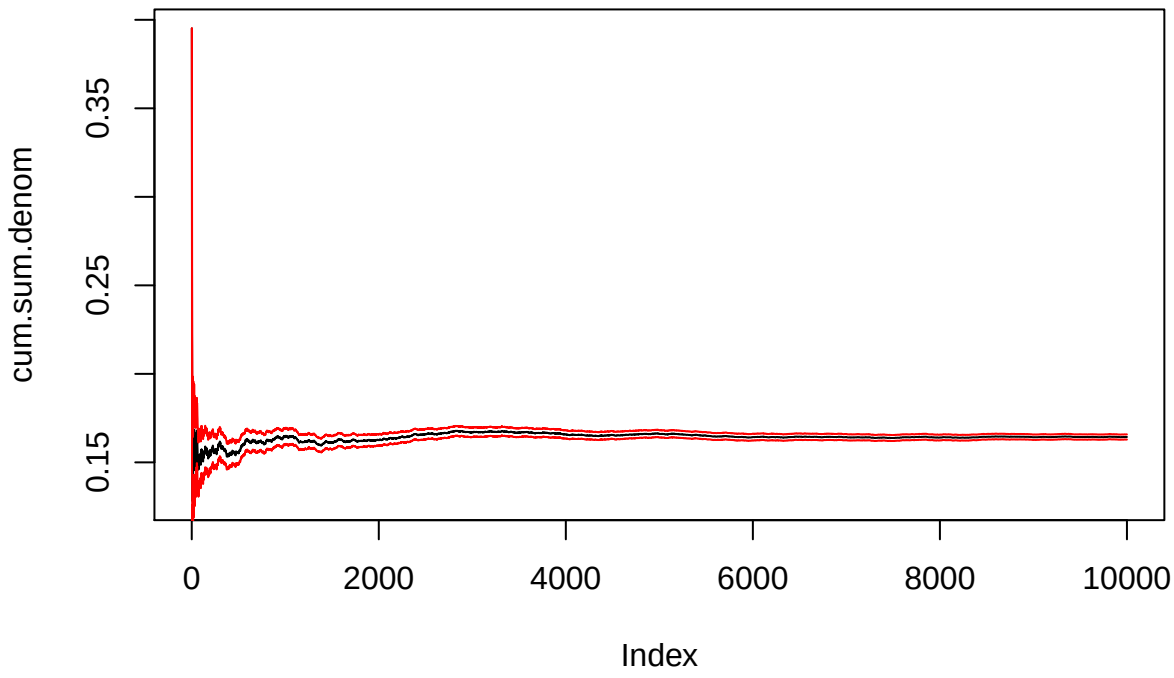


The red line shows the numerator while the black shows the denominator. It is clear that this estimator is the posterior mean.

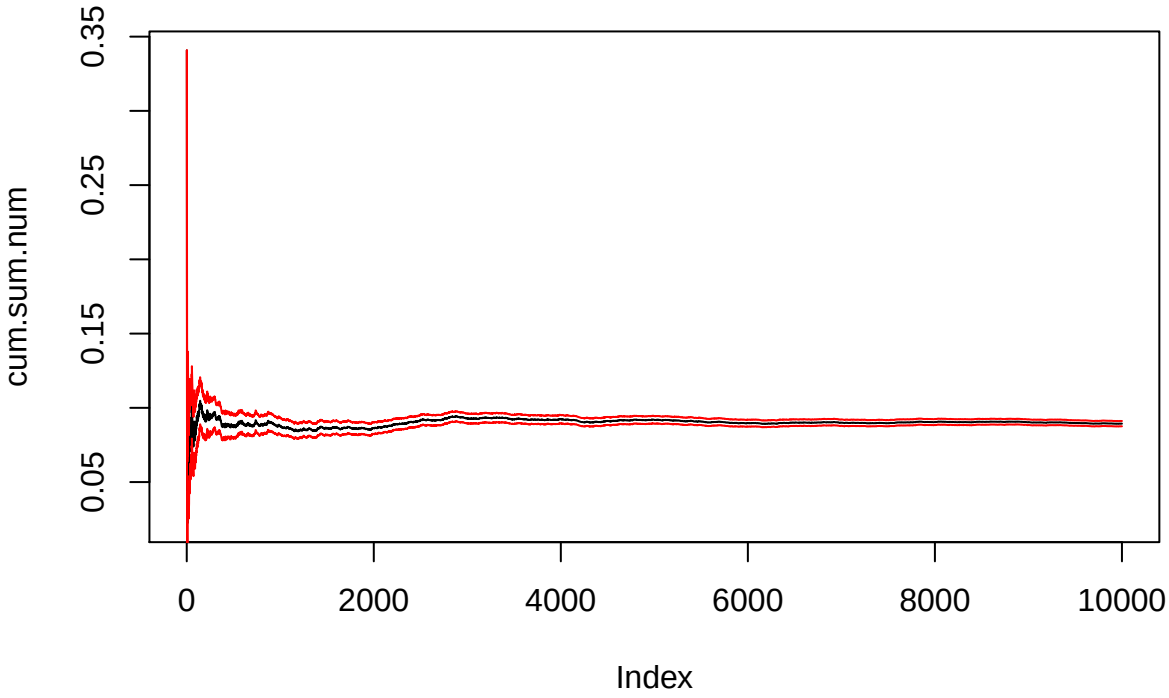
Part B

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

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

Problem 3.2

Part A

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.