Statistics 700 Homework 3 (alpha)

Based on solution by Po-Heng Chen

October 15, 2017

ESS for Importance Sampling

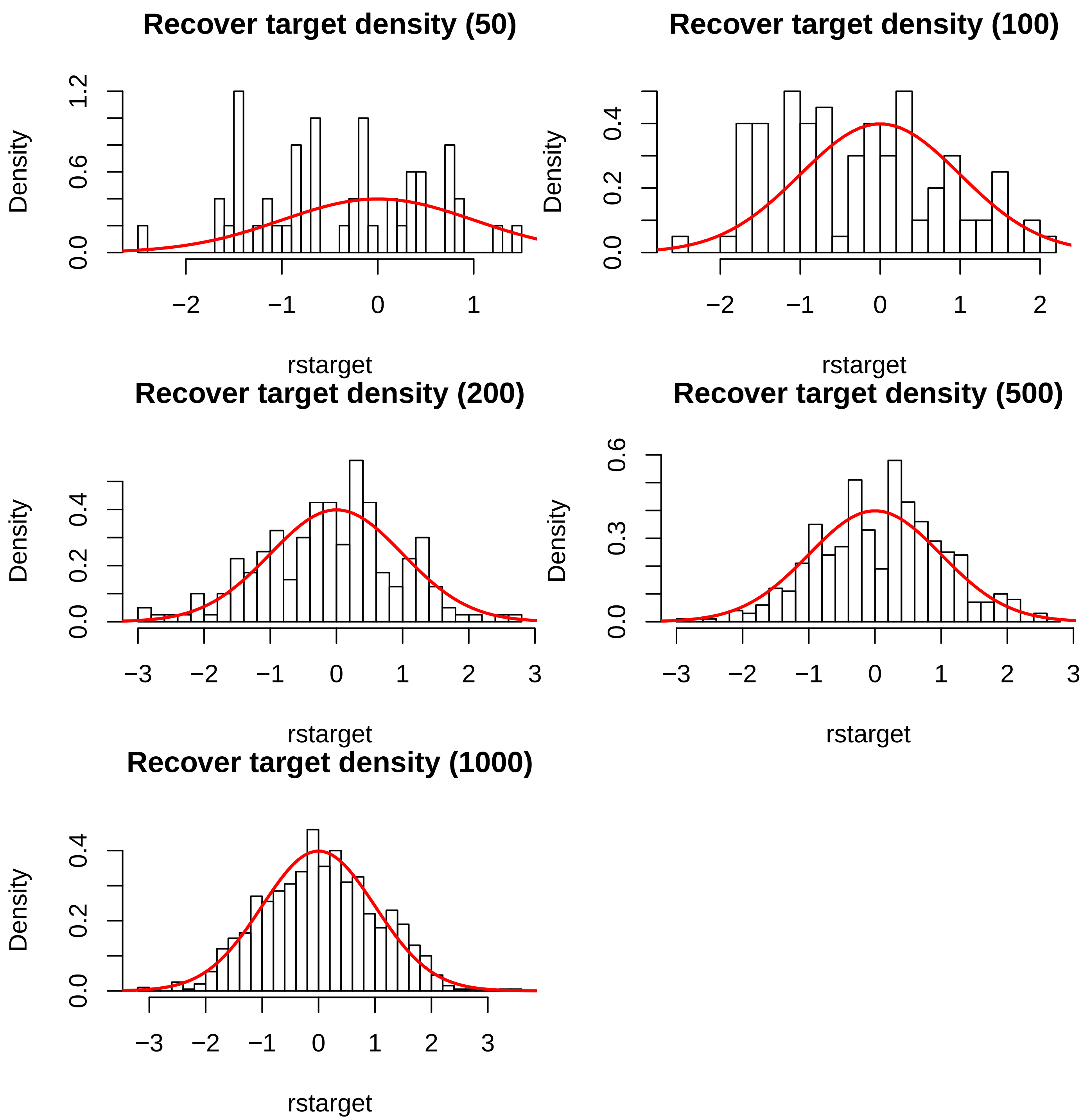
1. Implement an importance sampler and calculate ESS(m) for m = 50, 100, 200, 500, 1000.

Suppose $\pi(x)$ is the target distribution where $\pi(x) \propto N(0, 1)$, and $g(x)$ is the importance function (proposal distribution) where $g(x) \propto$ Student's t distribution with 2 degrees of freedom.

Let's first try to build the importance sampler with sample size $m = 50$, 100, 200, 500, 1000.

```
for( m in c(50, 100, 200, 500, 1000)){
    set.seed(700)
    a = 10
    target = function(x) {dnorm(x, mean = 0, sd = 1)}
    n = m
    # importance function: student t with 2 df
    rs = rt(n, df = 2)
    g = dt(rs, df = 2)
    q = target(rs)
    w = q / g
    rstarget = sample(rs, size = n, prob = w, replace = TRUE)
    hist(rstarget, 30, freq = FALSE,
         main = paste0('Recover target density (',m,')'))
    x = seq(-a, a, length.out = 200)
    c = sum(target(x) * (x[2] - x[1]))
    lines(x, target(x)/c, col = 'red', lwd = 2)
    ess = m / (1 + var(w)/(mean(w))^2)
    print(paste0('ESS(', m, ') = ', ess))
}
```

```
## [1] "ESS(50) = 46.8008333894794"

## [1] "ESS(100) = 89.6052735109369"

## [1] "ESS(200) = 176.12806971714"

## [1] "ESS(500) = 432.447278250017"

## [1] "ESS(1000) = 863.719473712893"
```



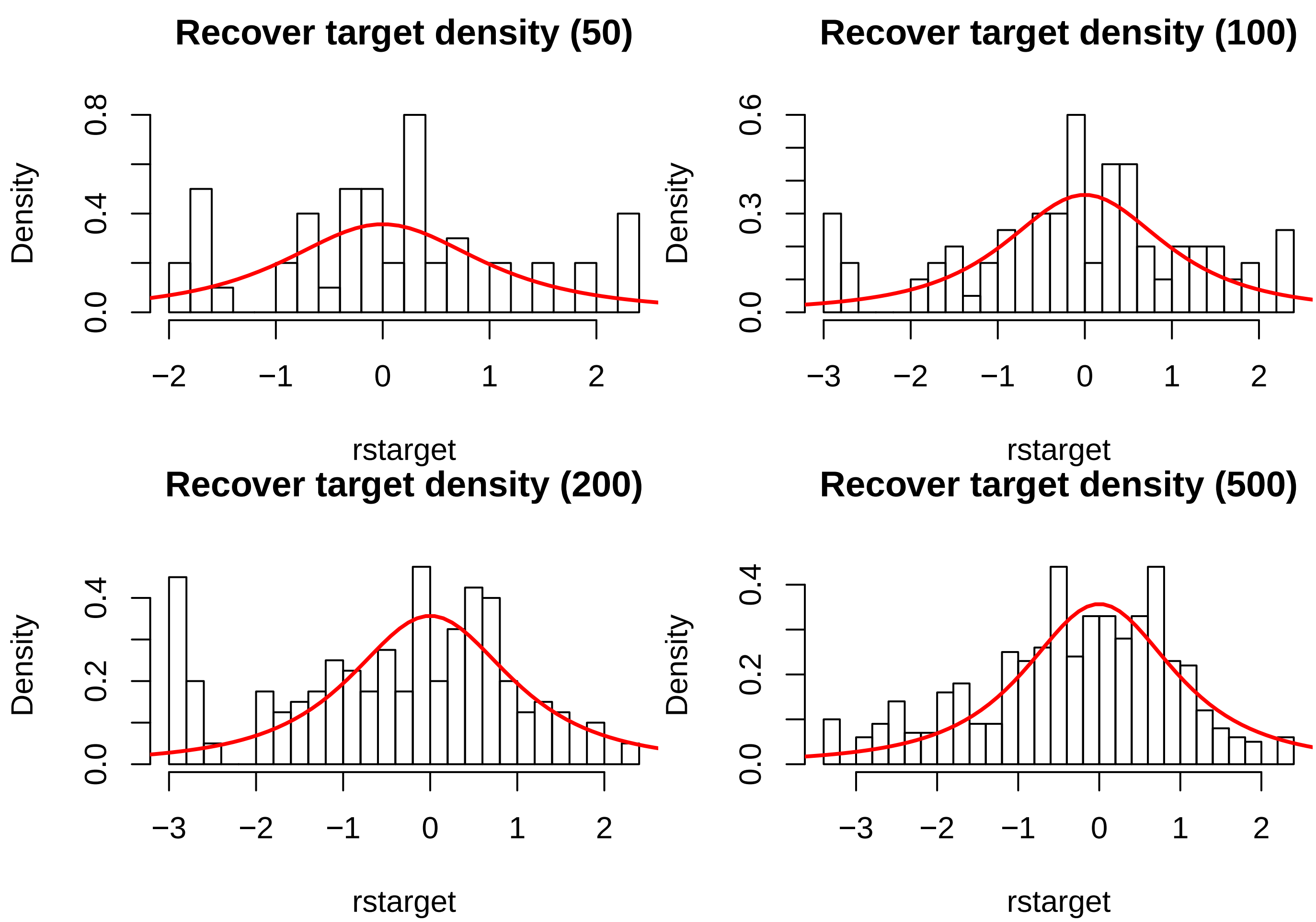
2. Implement an importance sampler and calculate ESS(m) for m = 50, 100, 200, 500, 1000.

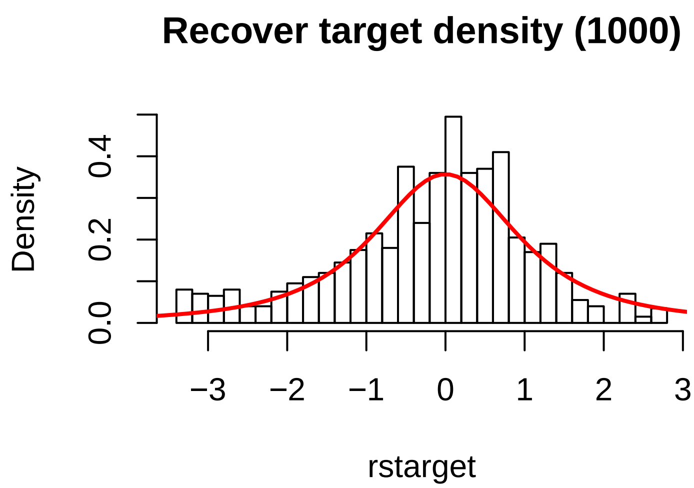
Since the target function and the importance function are switched, we hence switch target/importance functions in the above code.

```
for(m in c(50, 100, 200, 500, 1000)){
    set.seed(700)
    a = 10
    target = function(x) {dt(x, df = 2)}
```

```
    n = m
    # importance function: standard normal density
    rs = rnorm(n, mean = 0, sd = 1)
    g = dnorm(rs, mean = 0, sd = 1)
    q = target(rs)
    w = q / g
    rstarget = sample(rs, size = n, prob = w, replace = TRUE)
    hist(rstarget, 30, freq = FALSE,
         main = paste0('Recover target density (',m,')'))
    x = seq(-a, a, length.out = 200)
    c = sum(target(x) * (x[2] - x[1]))
    lines(x, target(x)/c, col = 'red', lwd = 2)
    ess = m / (1 + var(w)/(mean(w))^2)
    print(paste0('ESS(', m, ') = ', ess))
}
```

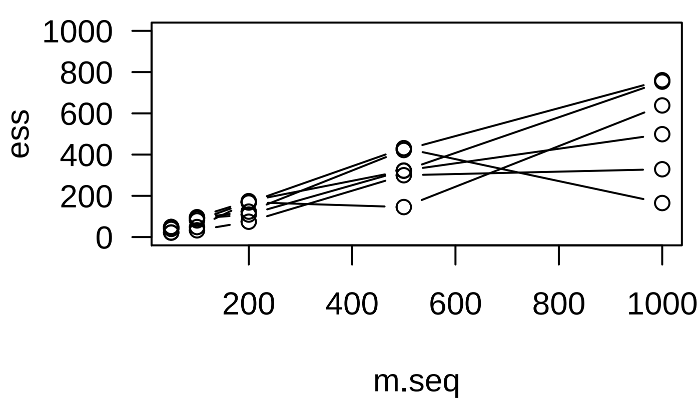
```
## [1] "ESS(50) = 48.3337168262753"

## [1] "ESS(100) = 77.2077131716916"

## [1] "ESS(200) = 155.836641958942"

## [1] "ESS(500) = 341.06417912612"

## [1] "ESS(1000) = 733.985499433072"
```





However, ESS in this case is extremely unstable, since it has infinite variance.

```
ess <- c()
m.seq <- c(50, 100, 200, 500, 1000)
replicant <- function(){
  for(i in 1:length(m.seq)){
    m <- m.seq[i]
    a = 10
    target = function(x) {dt(x, df = 2)}
    n = m
    # importance function: student t with 2 df
    rs = rnorm(n, mean = 0, sd = 1)
    g = dnorm(rs, mean = 0, sd = 1)
    q = target(rs)
    w = q / g
    ess[i] = m / (1 + var(w)/(mean(w))^2)
  }
  ess
}
ess <- replicant()
plot(m.seq, ess, type = 'b', ylim = c(0, 1000), las = 1)
for (i in 1:5) {
  ess <- replicant()
  lines(m.seq, ess, type = 'b')
}
```



3. What do you find by comparing the results above?

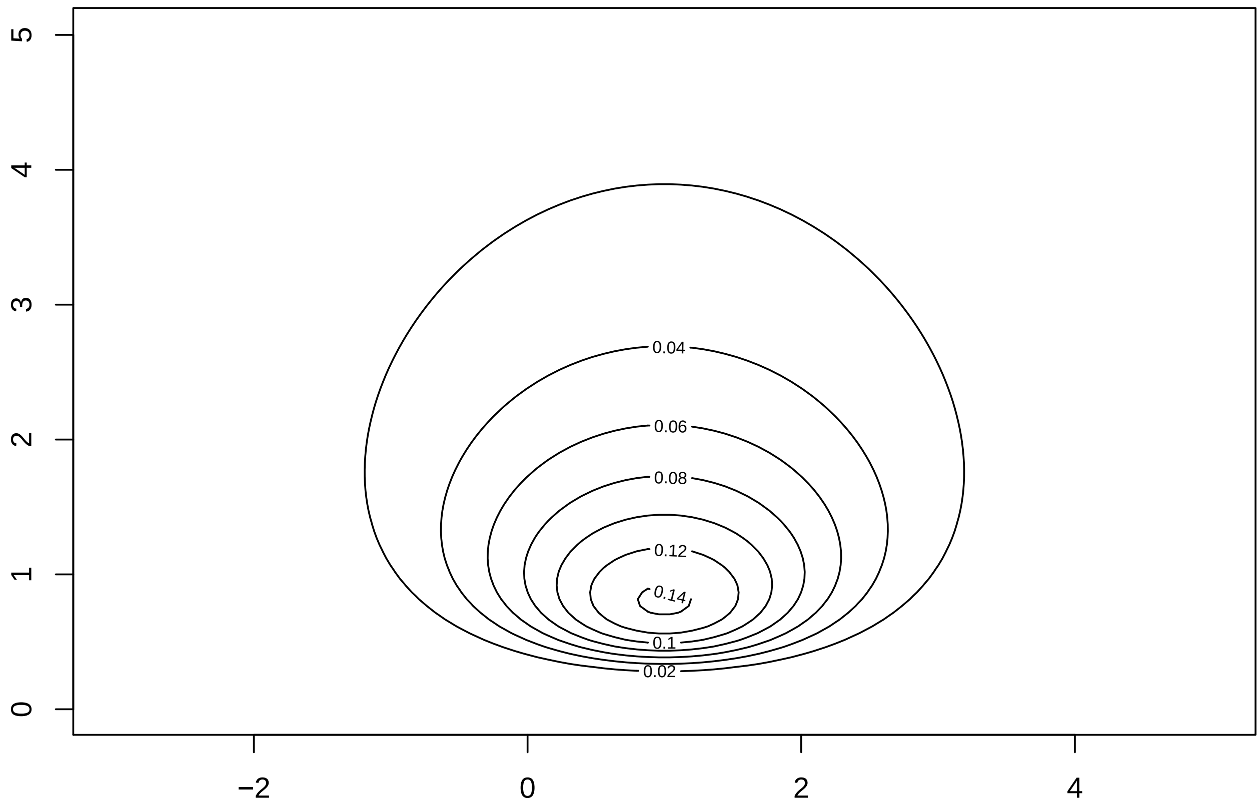
Usingn a heavy tailed distribution as proposal distribution for a (sub-)Gaussian distribution produces weights with infinte variance, and leads to unstable estimates.

4.

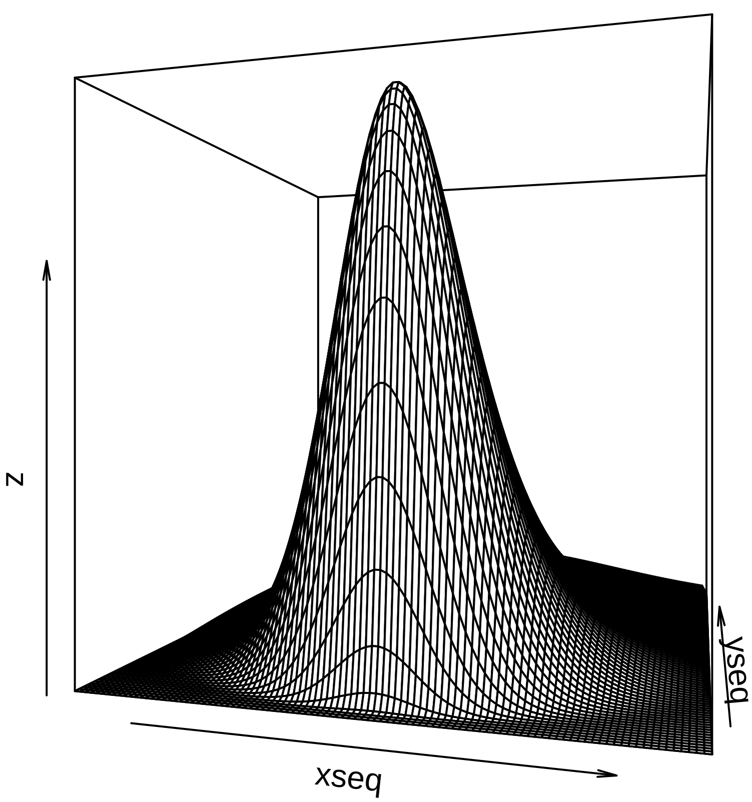
(a) Contour Plot

It's important that we truncate it!

```
target <- function(para){
    mu = para[1]
    var = para[2]
    if ((mu < 5) && (mu > -3) && (var > 0.01) && (var < 50)) {
      return(var^(-5/2)*exp(-((mu-1)^2 + 4)/(2*var)))
    } else {
      return(0)
    }
}
xseq = seq(-3, 5, length.out = 100)
yseq = seq(0.01, 5, length.out = 100)
z = matrix(0, length(xseq), length(yseq))
for(i in 1:length(xseq)){
    for(j in 1:length(yseq)){
        z[i, j] = target(c(xseq[i], yseq[j]))
    }
}
contour(xseq, yseq, z)
```



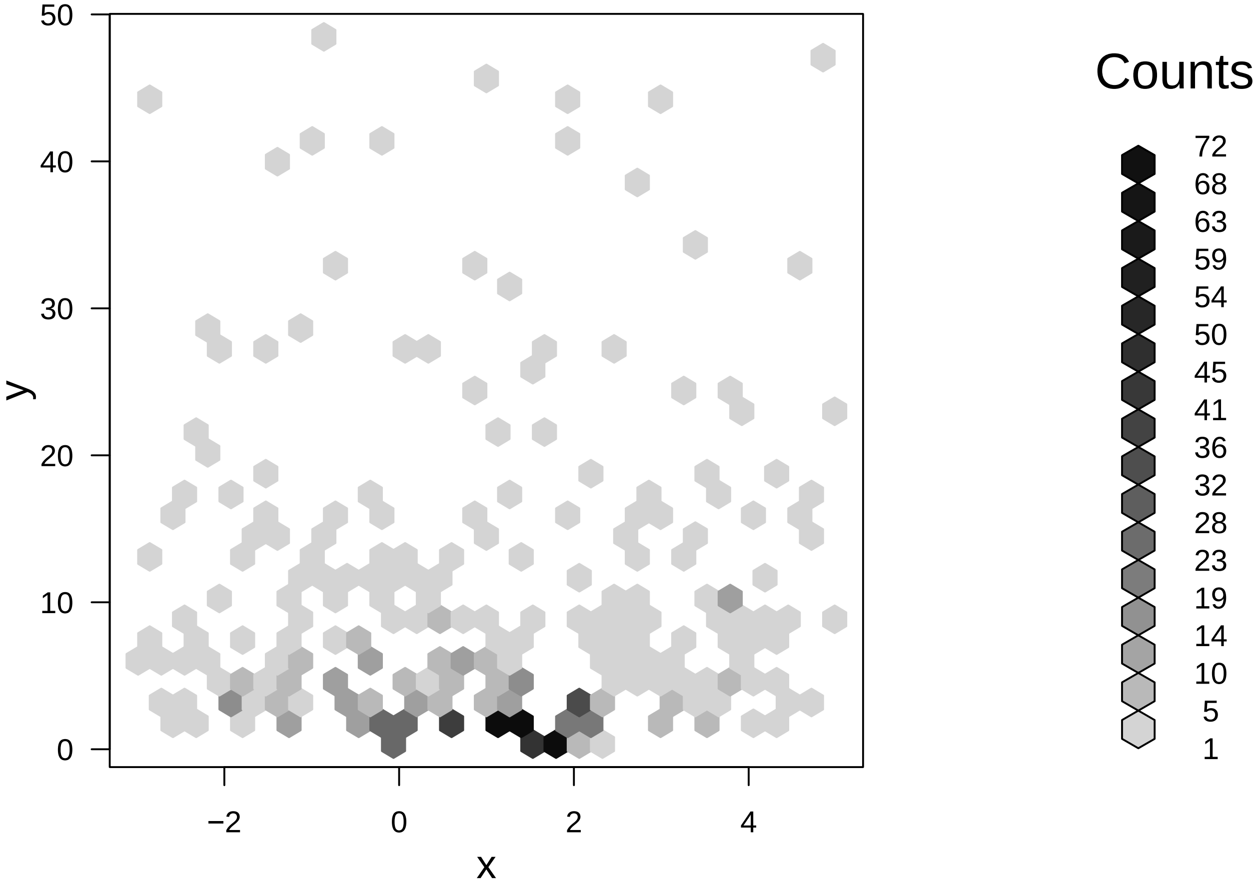

```
persp(xseq, yseq, z, theta=15, phi=0)
```



(b)

Let's start with flat importance function of uniform [0, 4]. (This is not sampling on a grid, but it's fine.)

```
# flat importance function
Nsamples = 1000
samplesunif = cbind(runif(Nsamples, min = -3, max = 5), runif(Nsamples, min = 0.01, max = 50))
w = apply(samplesunif, 1, target)
idx = sample(1:length(w), replace = TRUE, prob = w/sum(w))
samplestarget2d = samplesunif[idx, ]
rf <- colorRampPalette(rev(brewer.pal(11,'Spectral')))
h <- hexbin(samplestarget2d)
plot(h, xlab = 'x', ylab = 'y')
```




```
print(Nsamples / (1 + var(w)/(mean(w))^2))
```

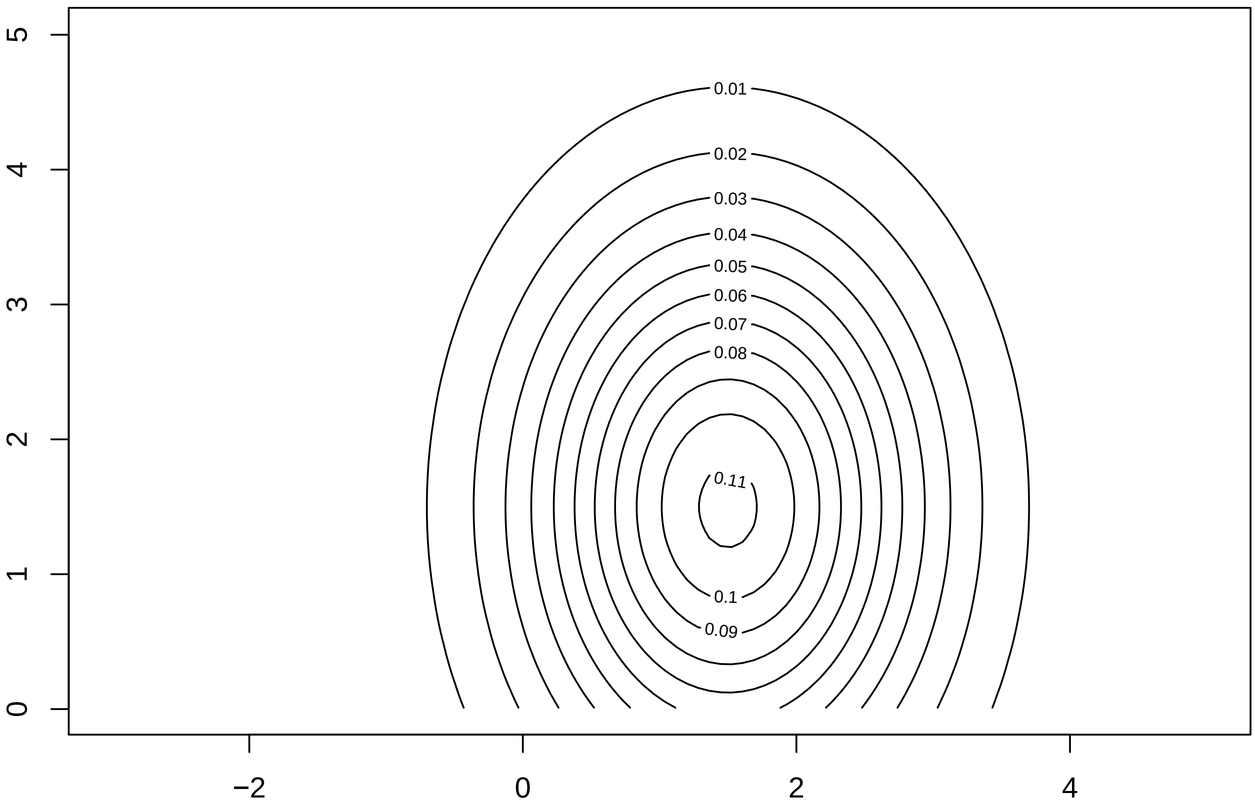
```
## [1] 63.59397
```

We can see that the ESS now is around 64.

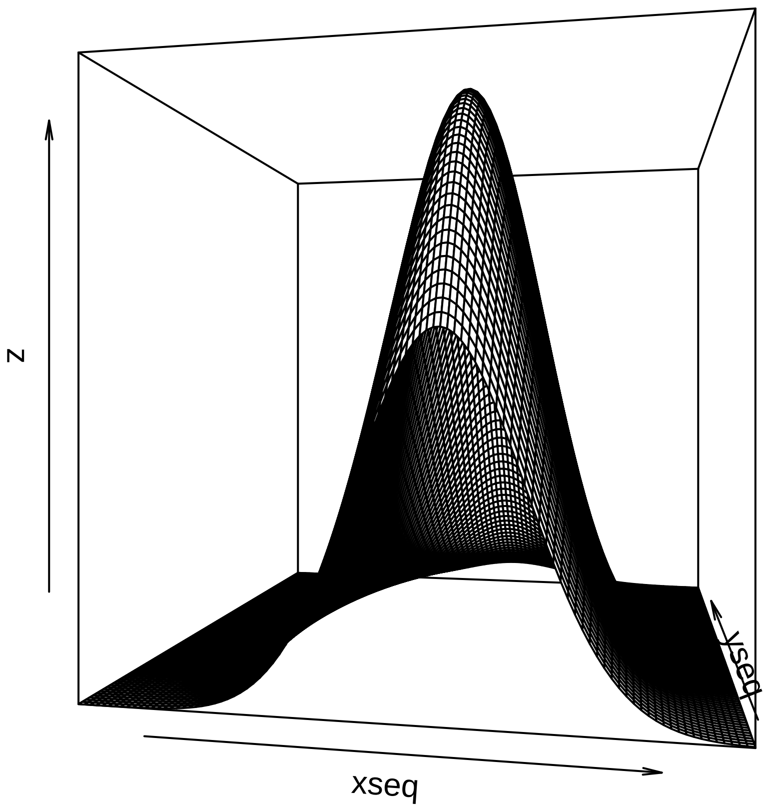
Next we use a Gaussian distribution as propoal (Note in general it's NOT OK to use a proposal distribution with lighter tails! We allow such a choice only because the target distribution is truncated.)

```
# importance function
g <- function(para){
    mu1 = c(1.5,1.5)
    Sigma1 = sqrt(c(1,2))
    x1 = sum(dnorm(para, mu1, Sigma1, log = TRUE))
    return (exp(x1))
```

```
}
xseq = seq(-3, 5, length.out = 100)
yseq = seq(0.01, 5, length.out = 100)

z = matrix(0, length(xseq), length(yseq))
for(i in 1:length(xseq)){
    for(j in 1:length(yseq)){
        z[i, j] = g(c(xseq[i], yseq[j]))
    }
}
contour(xseq, yseq, z)
```




```
persp(xseq, yseq, z, theta=10, phi=0)
```




```r
# new importance function
for( m in c(50, 100, 200, 500, 1000)){
    samplesG = cbind(rnorm(Nsamples, 1, sqrt(1)), rnorm(Nsamples, 1, sqrt(1)))

    w = apply(samplesG, 1, target) / apply(samplesG, 1, g)
    w = na.omit(w)
    idx = sample(1:length(w), replace = TRUE, prob = w/sum(w))
    samplestarget = samplesG[idx, ]
    ess = m / (1 + var(w)/(mean(w))^2)
    print(paste0('ESS(', m, ') = ', ess))
}
```

```
## [1] "ESS(50) = 18.7911885614051"
## [1] "ESS(100) = 24.8506648473961"
## [1] "ESS(200) = 45.1006844268438"
## [1] "ESS(500) = 64.3423438854661"
## [1] "ESS(1000) = 523.609275588432"
```

A much better ESS.

Rejection Control Algorithm

Suppose $\pi(x) \propto$ Gaussian density with mean 0 and standard deviation 0.3, $g(x) \propto$ student t distribution with 2 degrees of freedom.

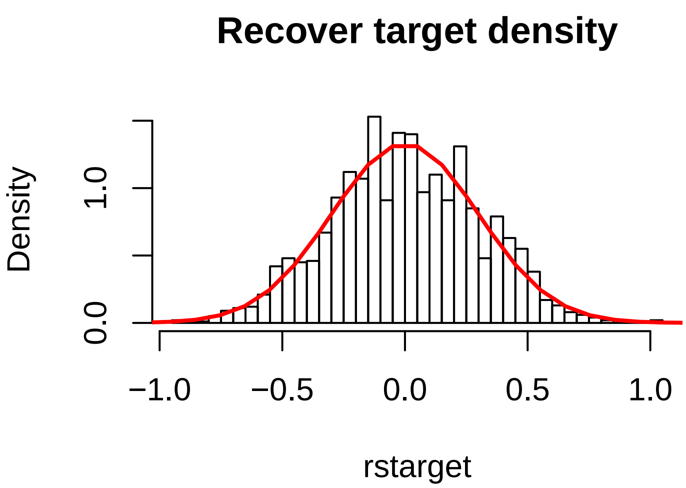
```r
rejectioncontrol <- function(Nsamples, samplefromtrial, logtarget, logtrial,
                             c, dimension = 1){
    samplesX <- samplefromtrial(Nsamples)
    if(dimension == 1){
```

```
    weights <- sapply(samplesX, function(x) logtarget(x) - logtrial(x))
    }
    if(dimension > 1){
    weights <- as.numeric(apply(samplesX, 1, function(x) logtarget(x) - logtrial(x)))
    }
    logc <- -Inf
    if(c > 0){
    logc <- log(c)
    }
    acceptrates <- rep(1, Nsamples)
    indicator <- which(weights < logc)
    if(length(indicator) > 0){
    acceptrates[indicator] <- exp(weights[indicator] - logc)
    }
    acceptindicator <- which(runif(Nsamples) < acceptrates)
    if(dimension == 1){
    acceptedsamples <- samplesX[acceptindicator]
    }
    if(dimension > 1){
    acceptedsamples <- samplesX[acceptindicator, ]
    }
    weightsnew <- weights[acceptindicator] - log(acceptrates[acceptindicator])
    weightsnew <- exp(weightsnew - max(weightsnew))
    weightsnew <- weightsnew / sum(weightsnew)
    return(list(samples = acceptedsamples, weights = weightsnew))
}
```

1. Implement the importance sampling with and without rejection control.

Implementation without rejection control.

```
m = 2000
set.seed(700)
a = 10
target = function(x) {dnorm(x, mean = 0, sd = 0.3)}
n = m
# importance function: student t with 2 df
rs = rt(n, df = 2)
g = dt(rs, df = 2)
q = target(rs)
w = q / g
rstarget = sample(rs, size = n, prob = w, replace = TRUE)
hist(rstarget, 30, freq = FALSE, main = 'Recover target density')
x = seq(-a, a, length.out = 200)
c = sum(target(x) * (x[2] - x[1]))
lines(x, target(x)/c, col = 'red', lwd = 2)
```




```
ess = m / (1 + var(w)/(mean(w))^2)
print(paste0('ESS(', m, ') = ', ess))
```

```
## [1] "ESS(2000) = 722.131849768128"
```

Implementation with rejection control.

```
samplefromtrial <- function(n){
    rt(n, df = 2)
}
logtrial <- function(x){
    dt(x, df = 2, log = TRUE)
}
logtarget <- function(x){
    dnorm(x, mean = 0, sd = 0.3, log = TRUE)
}
Nsamples <- 2000
c = 5
RCresult <- rejectioncontrol (Nsamples, samplefromtrial, logtarget,
                              logtrial, c, dimension = 1)
w = RCresult$weights
m = length(w)
ess = m / (1 + var(w)/(mean(w))^2)
print(paste0('ESS(', m, ') = ', ess))
```

```
## [1] "ESS(411) = 411"
```

Rejection control provides with LESS effective samples! However this saves computation, as we have to evaluate h on much less samples.

2. Do you see improvement by using rejection control with different C?

```
for(c in c(0.01, 0.1, 1, 2, 3, 5, 10)){
    RCresult <- rejectioncontrol (Nsamples, samplefromtrial, logtarget,
                              logtrial, c, dimension = 1)
    w = RCresult$weights
    m = length(w)
    ess = m / (1 + var(w)/(mean(w))^2)
    print(paste0('c = ',c, '; ESS(', m, ') = ', ess))
}
```

```
## [1] "c = 0.01; ESS(1265) = 709.58223838829"
## [1] "c = 0.1; ESS(1143) = 745.71178400594"
## [1] "c = 1; ESS(905) = 754.685422757226"
## [1] "c = 2; ESS(736) = 692.846226062537"
## [1] "c = 3; ESS(646) = 640.439335366042"
## [1] "c = 5; ESS(385) = 385"
## [1] "c = 10; ESS(194) = 194"
```

No.

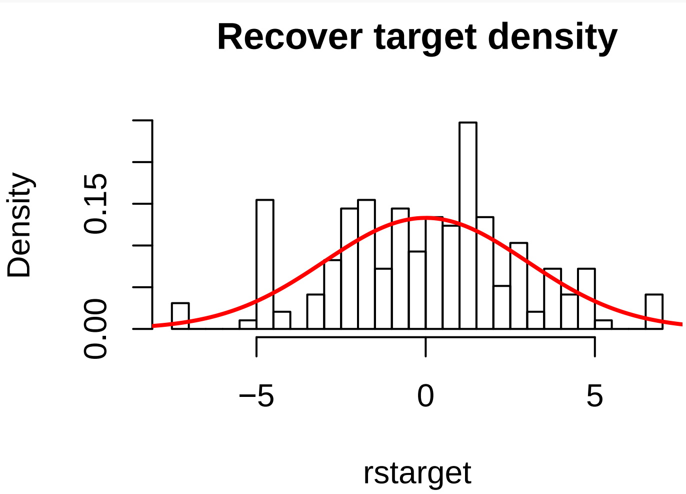
3. What value of c do you choose to use finally and why?

In this implementation, the smaller the better. Since we always discard samples. When $c \to 0$, we are back in the Importance Sampling case.

4. Repeat the above procedures when $\pi(x) \propto$ Gaussian density with mean 0 and standard deviation 3.

Without rejection control.

```r
set.seed(700)
a = 10
target = function(x) {dnorm(x, mean = 0, sd = 3)}
n = m
# importance function: student t with 2 df
rs = rt(n, df = 2)
g = dt(rs, df = 2)
q = target(rs)
w = q / g
rstarget = sample(rs, size = n, prob = w, replace = TRUE)
hist(rstarget, 30, freq = FALSE, main = 'Recover target density')
x = seq(-a, a, length.out = 200)
c = sum(target(x) * (x[2] - x[1]))
lines(x, target(x)/c, col = 'red', lwd = 2)
```




```r
ess = m / (1 + var(w)/(mean(w))^2)
print(paste0('ESS(', m, ') = ', ess))
```

```
## [1] "ESS(194) = 95.4471120224163"
```

With rejection control and different C.

```
samplefromtrial <- function(n){
    rt(n, df = 2)
}
logtrial <- function(x){
    dt(x, df = 2, log = TRUE)
}
logtarget <- function(x){
    dnorm(x, mean = 0, sd = 3, log = TRUE)
}
Nsamples <- 2000
for(c in c(0.01, 0.1, 1, 2, 3, 5, 10)){
    RCresult <- rejectioncontrol (Nsamples, samplefromtrial, logtarget,
                                  logtrial, c, dimension = 1)
    w = RCresult$weights
    m = length(w)
    ess = m / (1 + var(w)/(mean(w))^2)
    print(paste0('c = ',c, '; ESS(', m, ') = ', ess))
}
```

```
## [1] "c = 0.01; ESS(1996) = 982.844581618895"
## [1] "c = 0.1; ESS(1985) = 997.496787312193"
## [1] "c = 1; ESS(1293) = 920.278698503296"
## [1] "c = 2; ESS(827) = 751.096901982948"
## [1] "c = 3; ESS(600) = 587.822124571759"
## [1] "c = 5; ESS(412) = 412"
## [1] "c = 10; ESS(199) = 199"
```