# Importance Sampling Examples

Patrick Graham

19 September 2021

## Using a t as the importance sampler for a normal

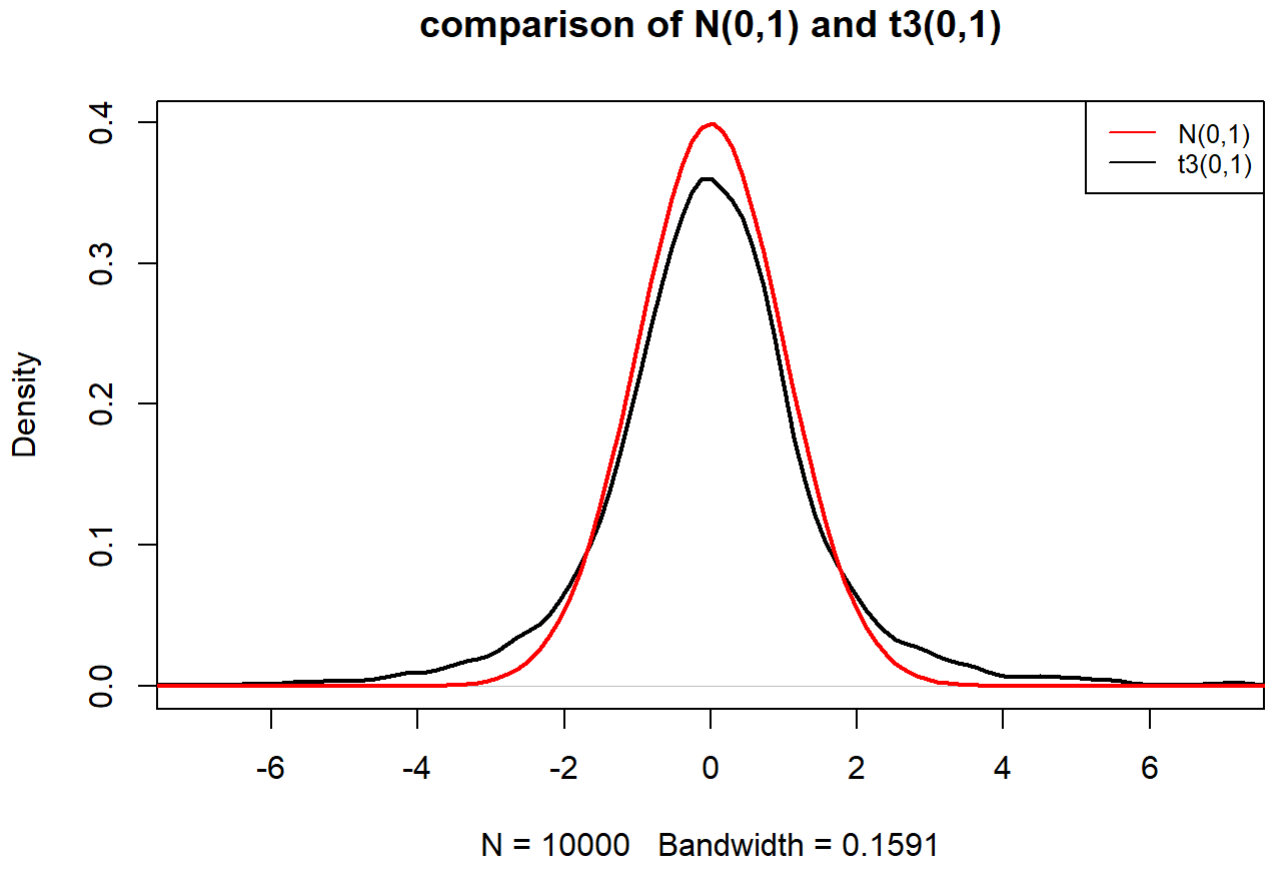
Draw a sample of values from a $t_3(0, 1)$

```
t <- rt(10000,df=3)
summary(t)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -16.40076  -0.75189  -0.00535   0.01449   0.74280  54.02090
```

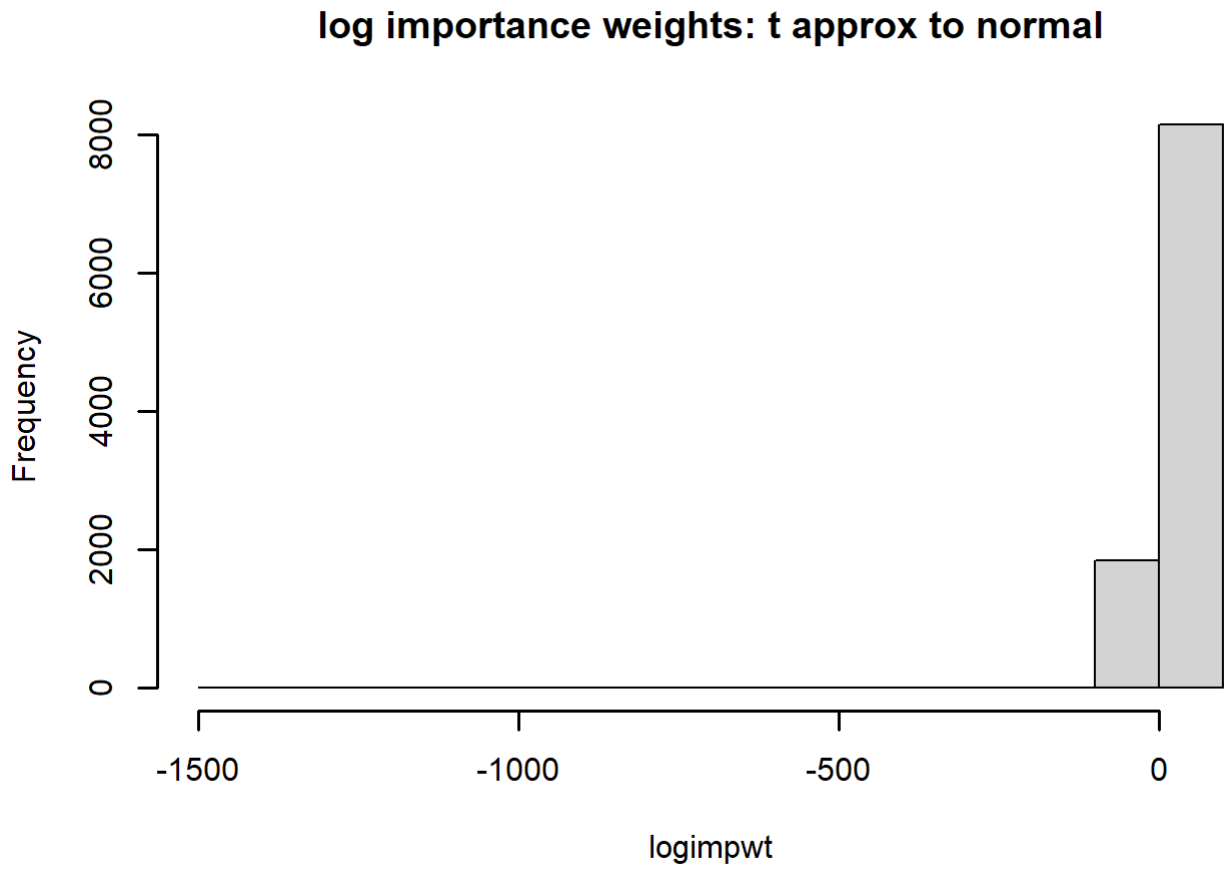
for comparison evaluate normal(0,1) density at same points that tdensity is evaluated

```
tdensity <- density(t)
ndensity <- dnorm(tdensity$x)
plot(density(t),col="black",main="comparison of N(0,1) and t3(0,1)",ylim=c(0,max(nden
sity)),xlim=c(-7,7),lwd=2.0)
lines(tdensity$x,ndensity,col="red",lwd=2.0)
legend("topright",legend=c("N(0,1)","t3(0,1)"),lwd=c(1.2,1.2),col=c("red","black"),ce
x=0.8)
```

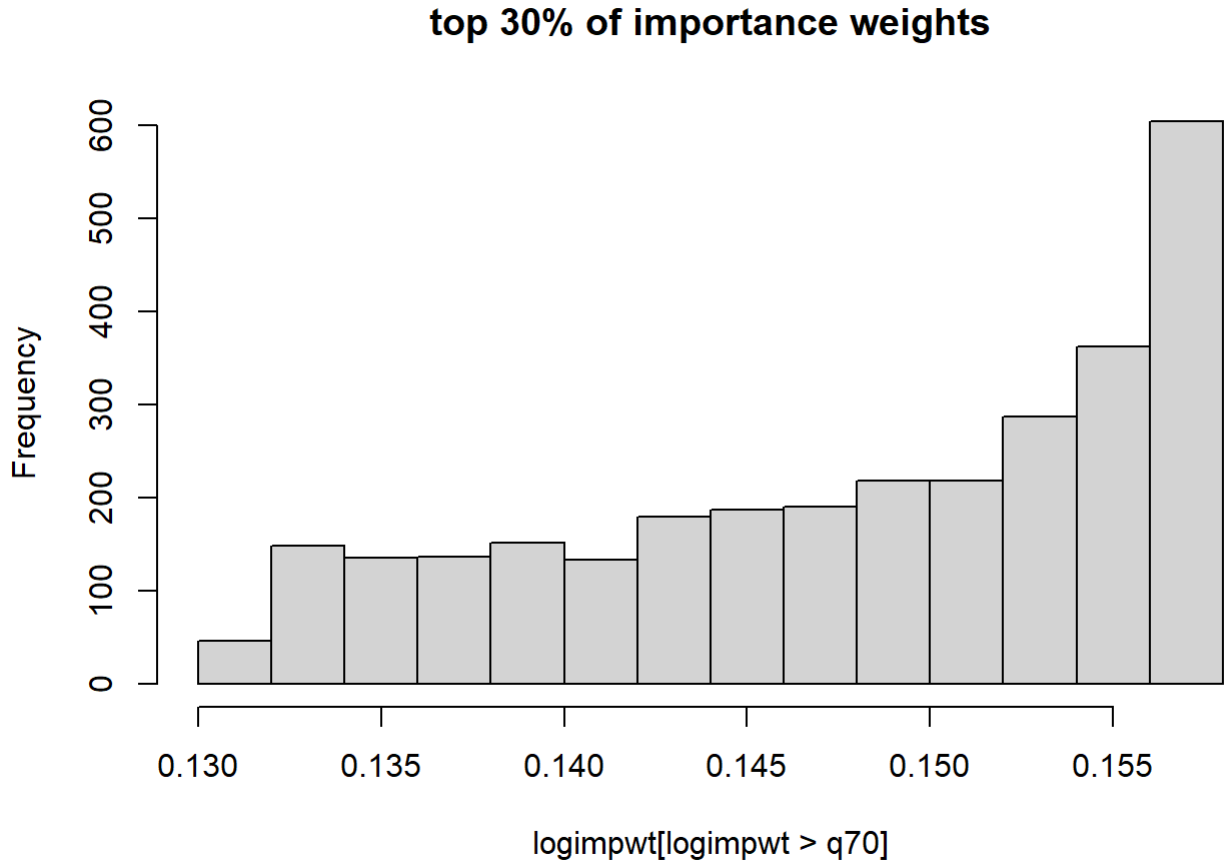


Now we will implement the importance sampling algorithm by computing the importance sampling weights: true density / approximating density.

```
##get log-importance ratios
logimpwt <-  dnorm(t,mean=0,sd=1,log=TRUE) - dt(t,df=3,log=TRUE)
##and plot a histogram
hist(logimpwt,main="log importance weights: t approx to normal",lwd=1.5,plot=TRUE)
```




```
##plot largest 30% of weights
q70 <- quantile(logimpwt,probs=0.7)
hist(logimpwt[logimpwt > q70],main="top 30% of importance weights")
```




The distribution of importance weights does not look too bad, and importantly (no pun intended!) there are no extreme outliers. The numerical values are such that we could just directly exponentiate the log importance weights to obtain the importance weights - which we do need ultimately to get the normalised importance importance weights.

However, in order to illustrate the sort of manoeuvre that is sometimes necessary and useful to obtain the importance weights from the log importance weights, we will re-center the log-importance weights by subtracting the mean. This can avoid exponentiating anything too big or small. Note if $l_i = \log(r(\theta_i))$ and $\bar{l} = 1/M \sum_{i=1}^{M} l_i$

$$\begin{aligned}\frac{\exp(l_i - \bar{l})}{\sum_i \exp(l_i - \bar{l})} &= \frac{\exp(-\bar{l})\exp(l_i)}{\sum_i \exp(-\bar{l})\exp(l_i)} \\ &= \frac{\exp(-\bar{l})\exp(l_i))}{\exp(-\bar{l})(\sum_i \exp(l_i)} \\ &= \frac{\exp(l_i)}{\sum_i \exp(l_i)}\end{aligned}$$

```
##compute importance sampling weights
summary(logimpwt)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -1445.2845     0.0822     0.1026    -0.7004     0.1384     0.1573
```

```
summary(exp(logimpwt))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.086   1.108   1.004   1.148   1.170
```

```
impwt <- exp((logimpwt - mean(logimpwt)))
summary(impwt)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.187   2.232   2.023   2.314   2.358
```

```
relwt <- impwt / sum(impwt) ##normalised importance weight

sum(relwt)
```

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

```
Neff <- 1/sum(relwt^2)
Neff
```

```
## [1] 9210.906
```

Note the term $1/\text{mean}(\text{logimpwt})$ cancels from the numerator and denominator in the calculation of the normalised importance weights `relwt` . The effective Monte Carlo size of 9210.91 is 92.11% of the nominal Monte carlo sample size. This is about as good as it gets for importance sampling

We now look at approximations to the posterior mean, standard deviation and tail quantiles. These values can be compared to the well-known true values from the standard normal distribution. Since we have an estimate the effective Monte carlo sample size we can use this to get an approximation to the Monte Carlo error in the estimate of the posterior mean.

```
###posterior mean
posmean <- sum(relwt*t)  ##since sum(relwt) =1
posmean  ###close to zero)
```

```
## [1] -0.01044992
```

```
##variance
posvar <- sum(relwt*(t-posmean)^2)
possd <- sqrt(posvar)
possd  #close to true value of 1
```

```
## [1] 0.9808384
```

```
MCerror <- possd / sqrt(Neff)
MCerror
```

```
## [1] 0.01021989
```

```
##get approximation to probability > 1.96

gt1p96 <- (t > 1.96)  #just  sets up indicator  for  > 1.96

pr1p96 <- sum(relwt*gt1p96)/sum(relwt)  #sum(relwt)=1
pr1p96
```

```
## [1] 0.02239621
```

```
##and compare with unweighted version
mean(gt1p96)
```

```
## [1] 0.0692
```

```
##get approximation to probability > 1.645

gt1p645 <- (t > 1.645)

pr1p645 <- sum(relwt*gt1p645)
pr1p645
```

```
## [1] 0.04638245
```

```
##Compare with  unweighted, i.e.  the sample from  t3
mean(gt1p645)
```

```
## [1] 0.095
```

It certainly looks like the importance sample works pretty well here. Finally, we will resample the original set of values with probability proportional to the normalised important weights, to make it easier to get an approximation to the posterior density.

```
##resample easier to get density plots, and quantiles

newt <- sample(t,size=length(t),replace=TRUE,prob=relwt)
mean(newt)
```

```
## [1] -0.00783543
```

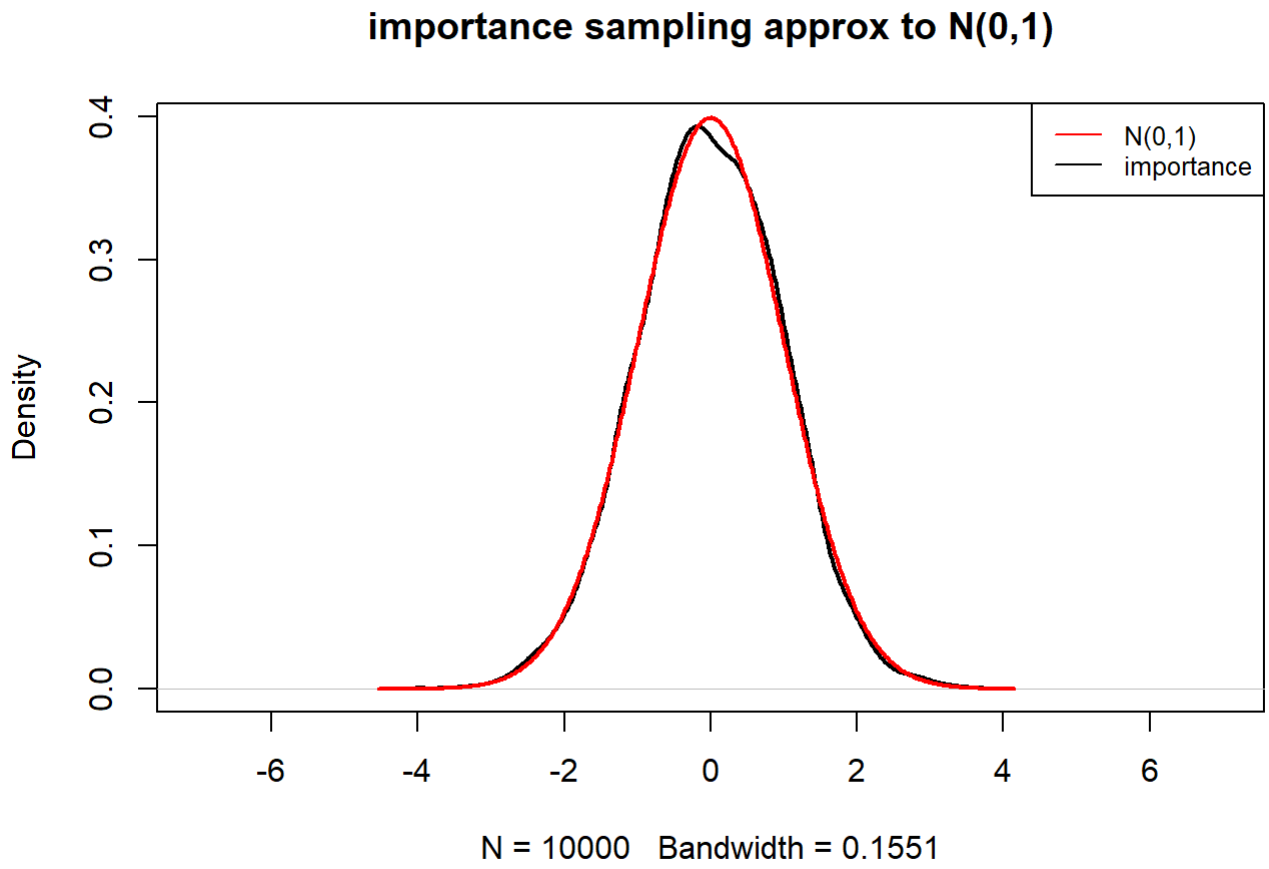
```
sd(newt)
```

```
## [1] 0.9882193
```

```
##get density and obtain normal density values for  comparison
newdensityt <- density(newt,adjust=1.1)

newnorm <- dnorm(newdensityt$x)


plot(newdensityt,main="importance sampling approx to N(0,1)",lwd=2,xlim=c(-7,7))
lines(newdensityt$x,newnorm,lwd=2,col="red")
legend("topright",legend=c("N(0,1)","importance"),lwd=c(1.2,1.2),col=c("red","black"
),cex=0.8)
```




This looks pretty good.

# Importance sampling for the unknown N, known p problem.

We consider the problem of estimating the size of a population, $N$ given an observed count of $X$ individuals obtained from the population through some process with known probability of observation (or capture probability), $\theta$. We can regard the count of $X$ individuals as having been obtained from a Binomial experiment with size parameter $N$ and success probability $\theta$, i.e

$$X \sim \text{Binomial}(N, \theta).$$

In contrast to most Binomial problems, it the size parameter that is unknown. In realistic applications to population estimation problems the success probability will also have to be estimated, and this generally requires some additional data. This takes us into the realm of capture-recapture or multiple systems estimation. However, for now we will just assume the capture probability, $\theta$, is known

We will assume a uniform prior on the the unknown population size parameter and we will use the prior as our approximating density. This is not usually good practice computationally,but is useful for illustrating the role of the likelihood in Bayesian inference.

# set-up for the problem- specify the prior, capture probability and the data.

Since we are using a uniform prior for the population size $N$ we need only specify lower and upper bounds.

```
lower_prior <- 4000
upper_prior <- 6000
```

Specify the capture probability; assumed known here but likely to be estimated in practice.

```
theta <- 0.87 #assumed capture probability
```

and give the data - here it is just the single observed count $X$

```
X <- 4350    #observed number of captures
```

# Set-up the log-likelihood function

(same as in the rejection sampling example) Our model is $[X|N,\theta] \sim \text{Binomial}(\theta, N)$ with $N$ regarded as the unkown parameter that we are trying to estimate.

```
##log pmf for N - log-likelihood
#N is the unknown population size
# n is the observed count
# p0 =  (1-\theta) is the probability of not being captured in the data
logpN <- function(N,n,p0) {
     logdens <- lchoose(N,n) +  (N-n)*log(p0)
  return(logdens)
}
```

Test this log-likelihood function out by trying it at few values

# test the logpN function

```
logpN(N=5000,n=X,p0=(1-theta))
```

```
## [1] 601.7021
```

```
logpN(N=4000,n=X,p0=(1-theta))
```

```
## [1] -Inf
```

Note that $N = 4000$ is less than the observed count and the log-likelihood is therefore $\exp(\text{logpN}) = 0$ - no support for values of the population count less than the observed count - makes sense.

# Run the importance sampling approximation

Specify the desired posterior sample size.

```
npos <- 2000 ##desired posterior sample size
```

Draw from the approximating density. If we use the prior as the approximator this just means draw npos values from a uniform distribution over the prior bounds (since the prior is uniform in this example)

```
testseq <- seq(from=lower_prior,to=upper_prior,by=1)
Napprox <- sample(testseq,size=npos,replace=TRUE) #equivalent
                                                  #to sampling from
                                                  #the uniform prior
summary(Napprox)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    4001    4502    4991    4988    5466    5999
```

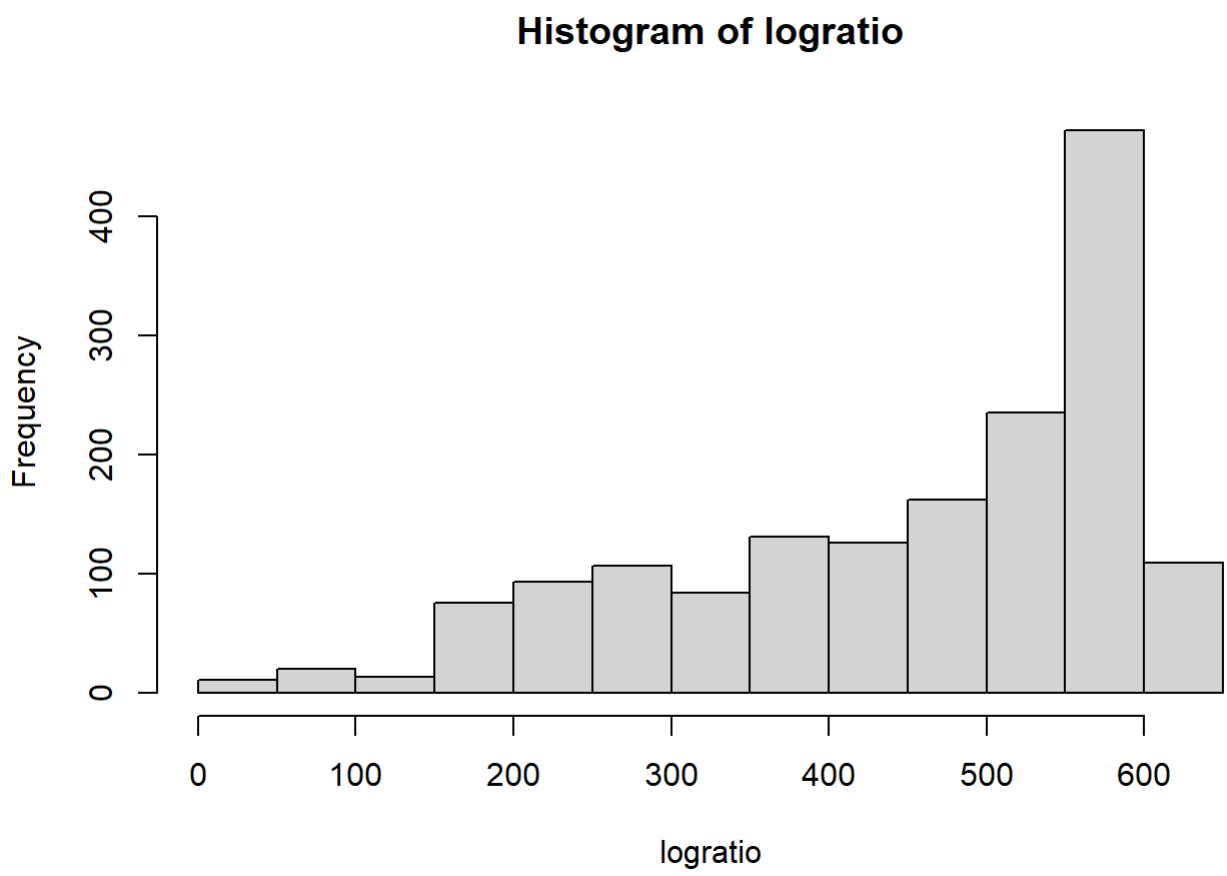
If we use the prior as the approximating density the importance ratio is just the likelihood; hence the log of the importance ratio is the log-likelihood. Hence we can use our log-likelihood function to give the value of the importance ratio at for each of the values generated from the approximating density.

```
logratio <- logpN(N=Napprox,n=X,p0=(1-theta))
summary(logratio)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    -Inf   212.2   441.5    -Inf   564.8   601.7
```

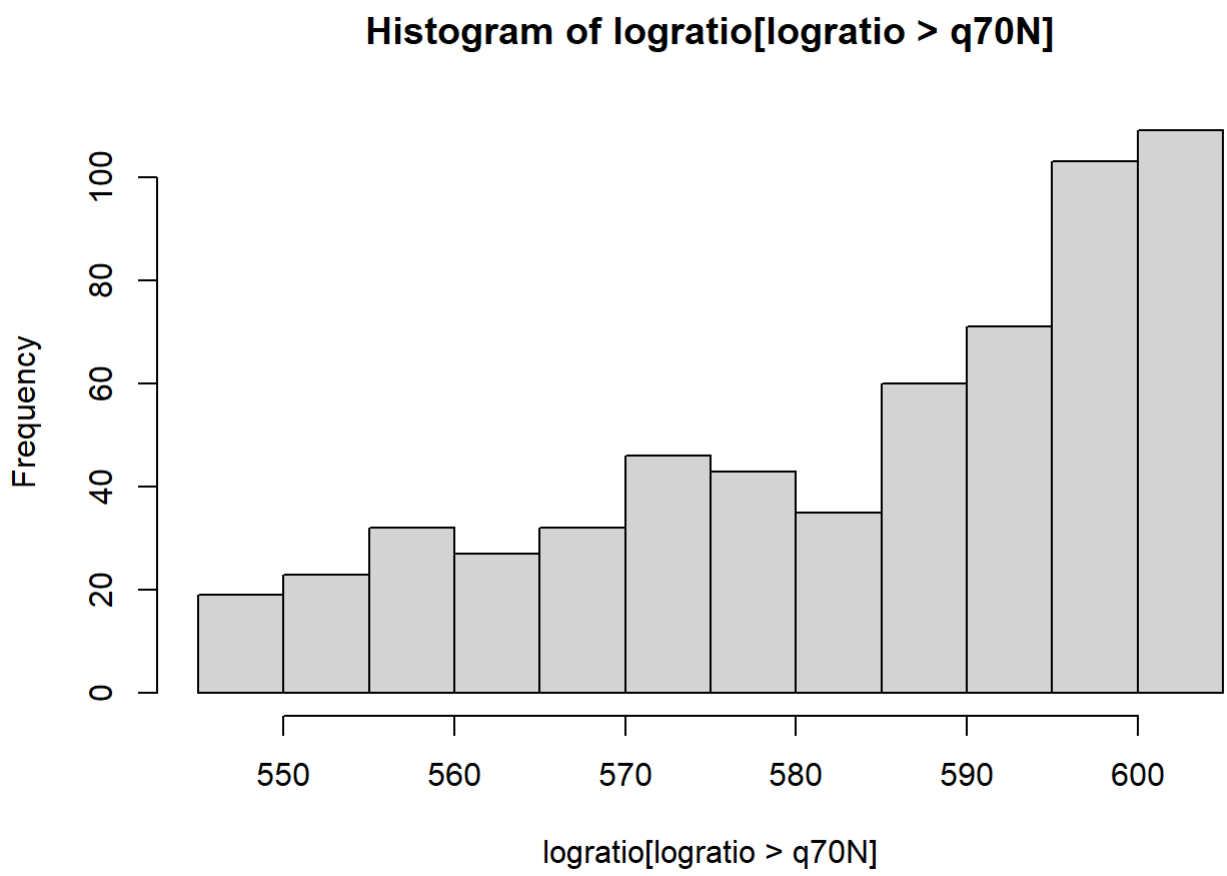
Next we check the distribution of the log importance sampling ratios to make sure the distribution is not dominated by a few large values

```
hist(logratio)
```




Better to look at just the largest values

```
q70N <- quantile(logratio,prob=0.7)
hist(logratio[logratio > q70N])
```




To obtain the posterior histogram we normalise the importance weights. This, effectively converts them to proportions corresponding to the probability mass attached to each of possible values of $N$, sampled from the approximating distribution.

To avoid any numerical difficulties due the presence of some -Inf values among the log importance ratios, we drop these log importance ratios and the corresponding $N$ values from the sample from the approximating density. We normalise the remaining set of importance weights. Note the $\N$ values with log importance weights of -Inf get no weight in the posterior sample, since (exp(-Inf)=0). These $N$ values are less than the observed count, so are impossible, once we have observed the count, $X$.

```
Napprox_trim <- Napprox[Napprox >= X]
length(Napprox_trim)
```

```
## [1] 1640
```

```
logratio_trim <- logratio[Napprox >= X]
summary(logratio_trim)
```

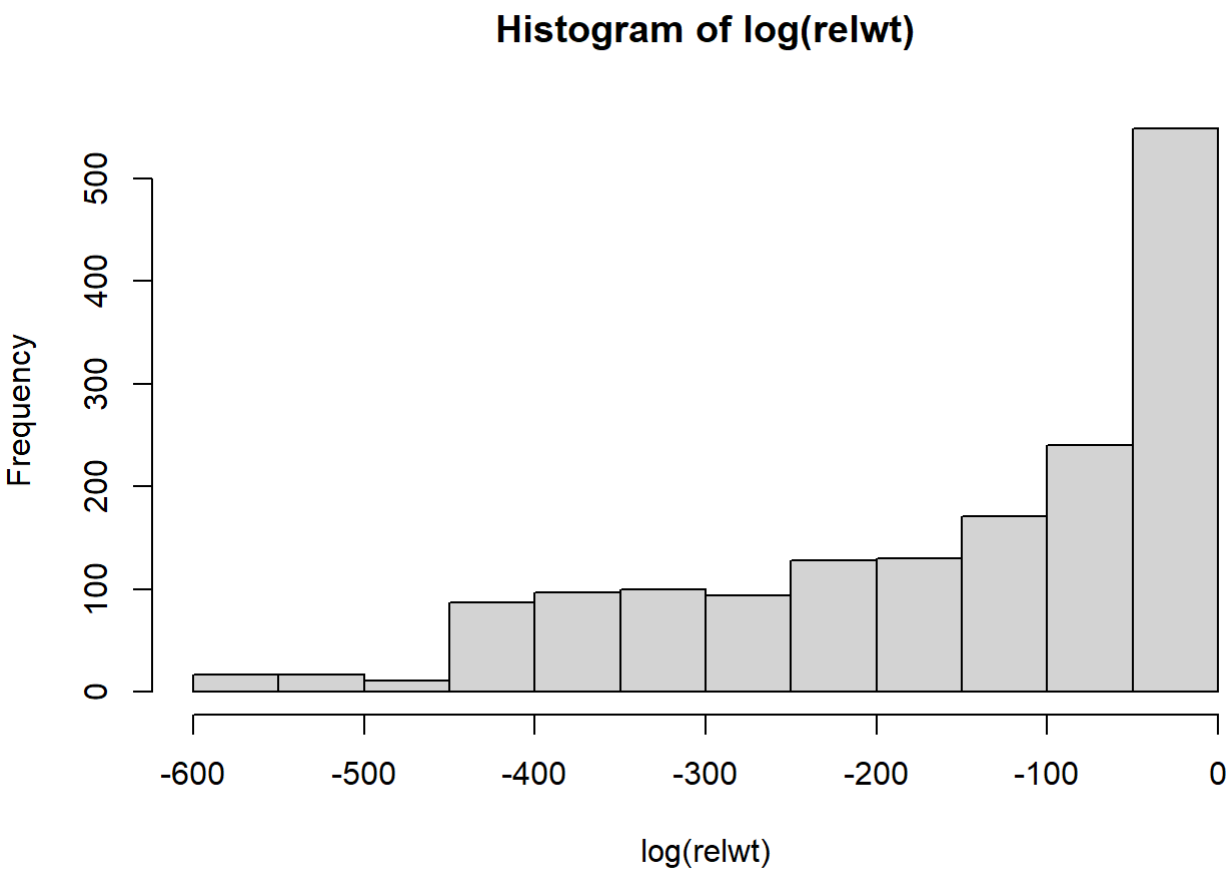
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   6.338 350.913 498.970 449.984 576.968 601.702
```

```
mean_trim <- mean(logratio_trim)

relwt <- ( exp(logratio_trim - mean_trim) /
             sum(exp(logratio_trim - mean_trim))
)
summary(relwt)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000000 0.0000000 0.0000000 0.0006098 0.0000000 0.0139222
```

```
hist(log(relwt))
```




```
head(relwt)
```

```
## [1]  3.741648e-52  5.985697e-59  7.781890e-17 3.235981e-241  3.146451e-88
## [6]  1.018332e-13
```

```
Neff <- 1/sum(relwt^2)
Neff
```

```
## [1] 104.2613
```

The distribution of importance weights is not really ideal. There are many small values most of the information for the posterior will come from a relatively proportion of the originally sample values for $N$. This leads to a small effective Monte Carlo sample size of 104.261333 even though the size of the original sample from the

approximating pmf was 2000. The prior is not a great approximation to the posterior.

Nevertheless, we will carry on and obtain posterior expectation and other summaries for N: This is just a weighted average of the points drawn form the approximator

```
Nmean_imp <- sum(Napprox_trim*relwt)
#Note we have normalised the weights to sum  to one
Nmean_imp
```

```
## [1] 5000.663
```

```
#variance
Nvar_imp <- sum(relwt*(Napprox_trim - Nmean_imp)^2)
Nsd_imp <- sqrt(Nvar_imp)
Nsd_imp
```

```
## [1] 28.1587
```

```
MCerror <- Nsd_imp / sqrt(Neff)
MCerror
```

```
## [1] 2.757725
```

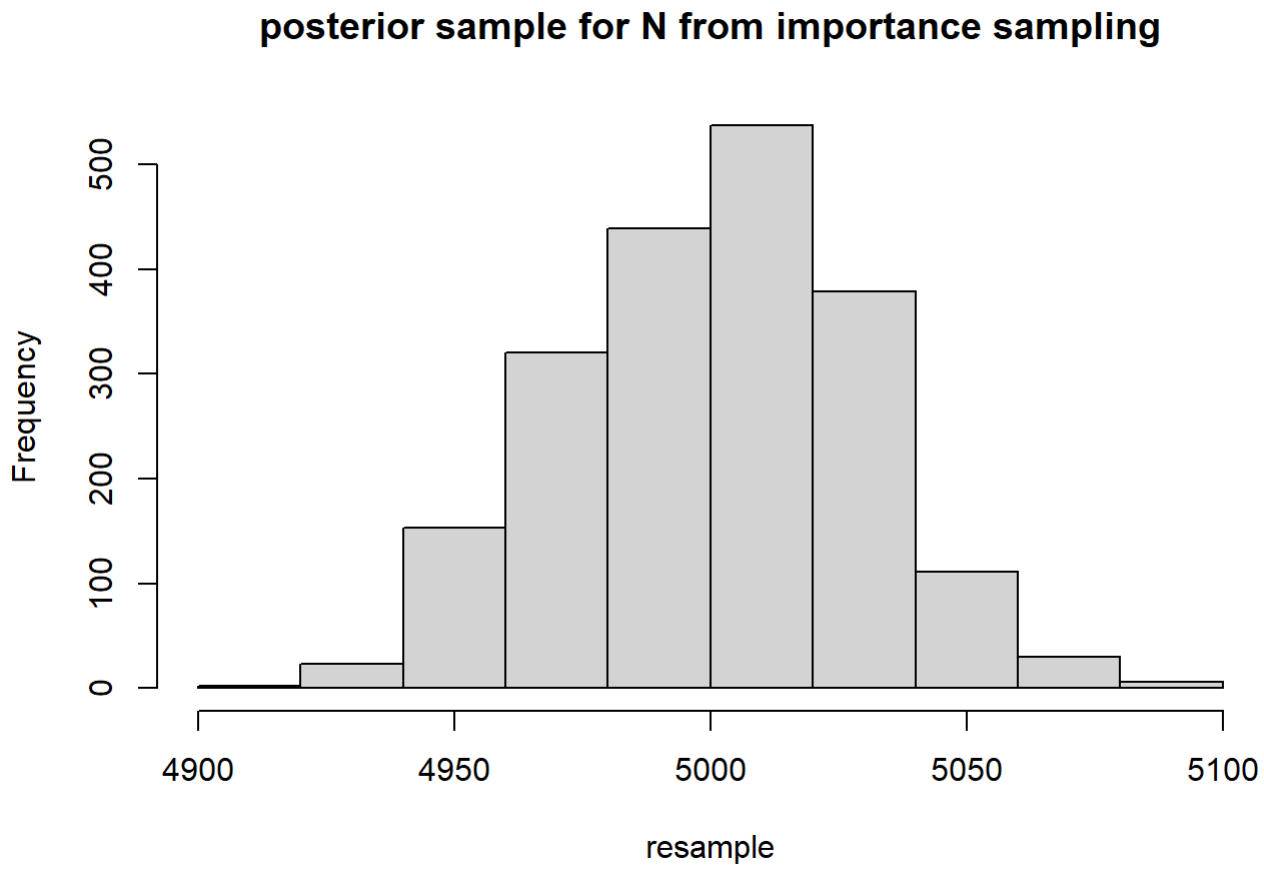
Although the effective Monte Carlo sample size is low, relative to the nominal Monte Carlo sample size the Monte Carlo error is not too terrible given that the posterior mean is 5000.66. This reflects the fact the posterior standard deviation is modest relative to the posteruor mean.

To get other posterior summaries we will make life easier for ourselves by resampling the original approximation with probabilities proportional to the relative importance sampling weights and treat the resulting sample as an approximation to the posterior in subsequent calculations.This in introduces an extra element of random-ness into the posterior sample but we will just live with that for now. Note that the re-sampling here differs from sampling-importance resampling where the aim is to sub-sample a relatively small fraction of the original sample from the approximator. Here we are just trying to get a representation of the posterior distribution implied by the importance sampling approximation.

```
resample <- sample(Napprox_trim,size=npos,prob=relwt,replace=TRUE)
summary(resample)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    4915    4981    5001    5001    5021    5098
```

```
hist(resample,main="posterior sample for N from importance sampling")
```




Compare the posterior implied by importance sampling with the prior (also the approximating density in this example ).

```
summary(Napprox)  #prior
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    4001    4502    4991    4988    5466    5999
```

```
summary(resample) #importance sampling
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    4915    4981    5001    5001    5021    5098
```

```
# get some other quantiles
quant_prior <- quantile(Napprox,probs=c(0.025,0.05,0.5,0.95,0.975))
quant_importance <- quantile(resample,probs=c(0.025,0.05,0.5,0.95,0.975))
```

It certainly looks like the importance sampler has done something. The true population size in this example is 5000, and the posterior mean approximation obtained by importance sampling is close to that.

In this example we can compare with quantiles of the true posterior which can be obtained directly, in this case, since the normalising constant from Bayes theorem can be obtained by summation over the specified prior range for $N$.

Since the likelihood is zero for value of $N$ less than the observed count, $X$ we need only consider the range of $N$ values between the maximum of ($X$ and the prior lower-bound) and the prior upper-bound.

```
range = seq(from=max(X,lower_prior),to=upper_prior,by=1)
logprange <- logpN(range,n=4350,p0=1-theta)
probs <- exp(logprange)/sum(exp(logprange)) ##Normalising the
                                            ##likelihood: the sum
##in n the
                                            ##denominator is the
##normalising
                                            ##constant
cdf <- cumsum(probs)

q025 <- sum(cdf <= 0.025)  ##Finding the position of the 2.5% percentile
q05 <- sum(cdf <= 0.05)
q50 <- sum(cdf <= 0.5)
q95 <- sum(cdf <= 0.95)
q975 <- sum(cdf <= 0.975)

##Find the corresponding  quantiles in range
true_quantiles <- range[(1+c(q025,q05,q50,q95,q975))]
true_quantiles
```

```
## [1] 4947 4956 5000 5045 5054
```

Next, we compare quantiles of the prior, importance sampling approximation and true posterior distributions.

```
quant_prior
```

```
##     2.5%       5%      50%      95%    97.5%
## 4045.000 4094.850 4991.000 5907.050 5957.025
```

```
quant_importance
```

```
##  2.5%    5%   50%   95% 97.5%
##  4948  4957  5001  5048  5058
```

```
true_quantiles
```

```
## [1] 4947 4956 5000 5045 5054
```