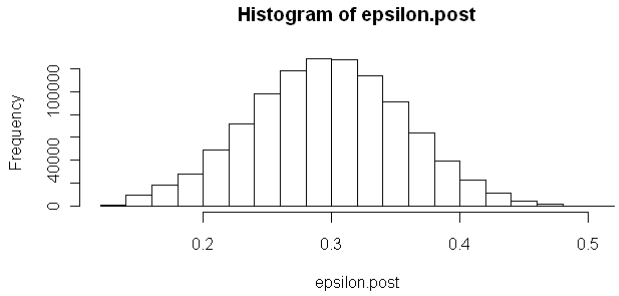
Extra Notes for SIR

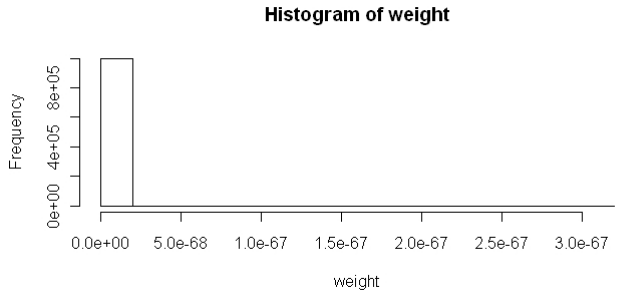
We can further improve the program by using a better proposal for θ_1:

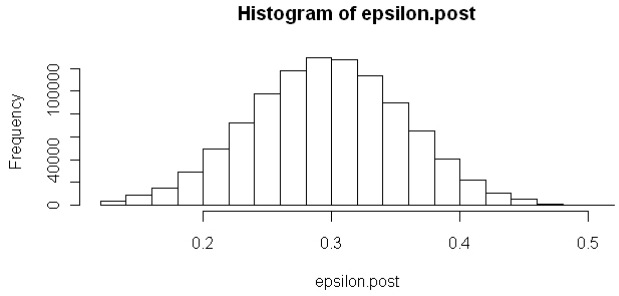
```
SIR.for.epsilon.4 <- function(a1, b1, a2, b2, x1, n1, x2, n2, size=10000)
{
theta1 <- rbeta(size, a1+x1, b1+n1-x1)
epsilon <- rbeta(size, 30,70)
weight <- theta1^(a1+x1-1)*(1-theta1)^(b1+n1-x1-1)*
    ( theta1 - epsilon)^(a2+x2-1)*
     (1- (theta1 - epsilon))^(b2+n2-x2-1)/(dbeta(epsilon, 30, 70)
                   *dbeta(theta1, a1+x1, b1+n1-x1))
epsilon.post<-sample(epsilon, length(epsilon), replace=T, prob=weight)
layout(matrix(c(1,2), byrow=T, nrow=2))
hist(epsilon.post)
hist(weight)
return(quantile(epsilon.post, prob=c(0.025, 0.25, 0.5, 0.75, 0.975)))
}
```

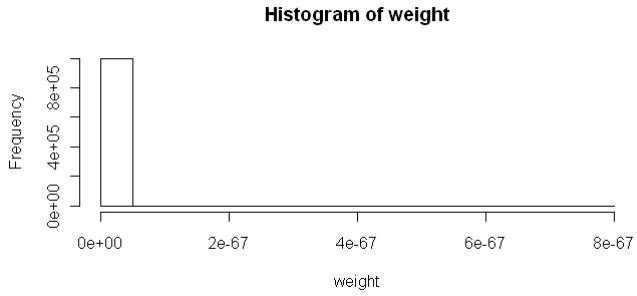
```
> SIR.for.epsilon.4(1,1,1,1,78, 118, 44, 122, 1000000)
     2.5%       25%       50%       75%     97.5%
0.1776777 0.2551830 0.2965333 0.3372757 0.4118818
> SIR.for.epsilon.4(1,1,1,1,78, 118, 44, 122, 1000000)
     2.5%       25%       50%       75%     97.5%
0.1714509 0.2544632 0.2961957 0.3370861 0.4127374
> SIR.for.epsilon.4(1,1,1,1,78, 118, 44, 122, 1000000)
     2.5%       25%       50%       75%     97.5%
0.1777780 0.2553157 0.2965252 0.3372748 0.4117839
> SIR.for.epsilon.4(1,1,1,1,78, 118, 44, 122, 10000000)
     2.5%       25%       50%       75%     97.5%
0.1728371 0.2545237 0.2960805 0.3369084 0.4123999
> SIR.for.epsilon.4(1,1,1,1,78, 118, 44, 122, 10000000)
     2.5%       25%       50%       75%     97.5%
0.1732555 0.2546648 0.2961969 0.3369895 0.4124063
```

Here are the graphs (from SIR3 and SIR 4 programs, respectively):









The weights are still unbalanced, presumably since θ_1 and ϵ are highly correlated, which is ignored in the proposal, so most weights still near zero. However, three decimal accuracy is attained.