✓

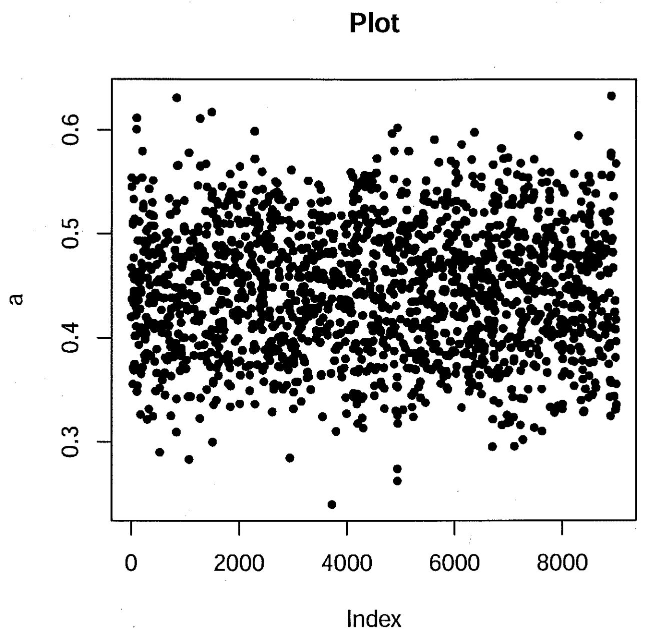
1. Use Metropollis algorithm to simulate a Markov Chain that have the following distribution as its stationary/limiting distribution. The distribution in question is a so called posterior distribution. Its density function is proportional to the product of
(a). a prior distribution that is a beta density, in particular beta(1, 2) say f(p).
(b). binomial likelihood, here viewed as a function of p: (n choose k) p^k (1-p)^(n-k)
We take n=80, k=36.

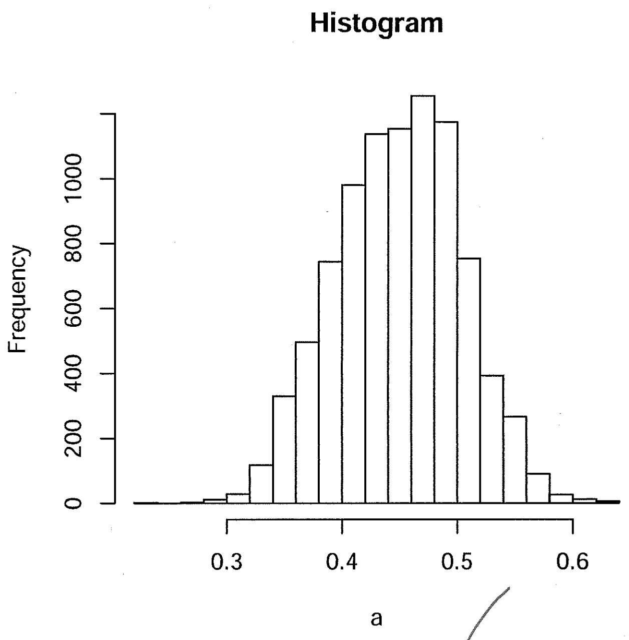
R code to simulate a Markov Chain with Beta Prior and Binomial Likelihood is below:

```
fprior = function(mu) {
    dbeta(mu, shape1 = 1, shape2 = 2)
}
lik = function(mu) {
    prod(dbinom(prob = mu, size = 80, x = 36))
}
fpost <- function(mu) {
    fprior(mu) * lik(mu)
}

MetroBeta <- function(n, a) {
    vec <- vector("numeric", n)
    x <- runif(1)
    vec[1] <- x
    for (i in 2:n) {
        can <- x + runif(1, min = -a, max = a)
        while (can <= 0 | can >= 1) {
            can <- x + runif(1, min = -a, max = a)
        }

        aprob <- fpost(can)/fpost(x)
        u <- runif(1)
        if (u < aprob) {
            x <- can
        }
        vec[i] <- x
    }
    return(vec)
}
```

Now looking at a plot and histogram with $n = 9,000$ and $a = .5$ we get:

```
a = MetroBeta(9000, 0.5)
plot(a, pch = 20, main = "Plot")
hist(a, main = "Histogram")
```





✓

2. **This question at www.ms.uky.edu/~mai/sta624/2013HM05.pdf.**

Considering a question like this my first intuition would be to write a bit of code to solve the problem. Simulate many many times until the expected number of visits is obtained. My code to do so is below.

```
mat = matrix(c(0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0.5, 0, 0.5, 0, 0, 0, 1/3, 1/3,
    0, 0, 0, 1/3, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1/4, 0, 1/4, 0, 1/4, 1/4,
    0, 0, 0, 1/3, 0, 1/3, 0, 0, 1/3, 0, 0, 0, 0, 1/2, 0, 0, 1/2, 0, 0, 0, 0,
    0, 1/2, 1/2, 0), nrow = 8)
mat = t(mat)
place = 1

ratrun = function(t = 1000, start = 1, cheesestate = 4, runtimes = 100) {
    times = rep(0, runtimes)
    # plot(0,start,pch=20,xlim=c(0,1000),ylim=c(0,10))
    for (j in 1:runtimes) {
        place = start
        # plot(0,start,pch=20,xlim=c(0,t),ylim=c(0,10))
        for (i in 1:t) {
            if (place == cheesestate) {
                times[j] = i - 1
            }
            if (place == cheesestate) {
                break
            }
            place = sample(c(1, 2, 3, 4, 5, 6, 7, 8), size = 1, prob = mat[place,
                ])
            # points(i,place,pch=20)
        }
    }
    return(list(times = times, avg_time = c(mean(times), median(times)), sd_time = sd(times)))
}
```

Running this code will yield the following results for runtimes=1,000,000.

```
results = ratrun(t = 1000, start = 1, cheesestate = 4, runtimes = 1e+06)
results$avg_time[1]

## [1] 24.65
```

✓ O.K.

We would also like to know if our result of the mean time is accurate. We can checking our results with the follow theoretical formula: $t = N\mathbf{1}$ where $N = (I_t - Q)^{-1}$ will yield the correct expected number of steps before being absorbed into the cheesestate. Doing this derivation gives $t = 24\frac{2}{3} \approx 24.66$. So we can see the simulation results gave similar results. ✓