Assignment 4

Rory Sarten 301005654

14 October, 2020

Question 1

a)

```
food_prices <- readr::read_delim("food_prices_kg2019.csv", delim = ",", col_types = readr::cols())
theta_est <- IQR(food_prices$Data_value) %>% round(3)
theta_est
```

```
## [1] 6.675
```

b)

```
set.seed(1)
N <- 1e4
boot_IQR <- 1:N %>%
  lapply(function(i) sample(food_prices$Data_value, replace = TRUE)) %>%
  sapply(IQR) %>%
  round(3)

## standard error of estimator
sd(boot_IQR) %>% round(3)
```

```
## [1] 1.197
```

```
## standard 95% bootstrap confidence interval
(theta_est + 1.96*c(-1, 1)*sd(boot_IQR)) %>% round(3)
```

```
## [1] 4.328 9.022
```

c)

```
## Efron's interval
quantile(boot_IQR, probs = c(0.025, 0.975)) %>% round(3)
```

```
##   2.5%  97.5%
##  5.290 10.105
```

d)

```
## Hall's interval
hall <- (2 * theta_est - quantile(boot_IQR, probs = c(0.975, 0.025))) %>% round(3)
names(hall) <- c("2.5%", "97.5%")
hall
```

```
##  2.5% 97.5%
## 3.245 8.060
```

e)

```
## bias
bias <- (mean(boot_IQR) - theta_est) %>% round(3)
bias
```

```
## [1] 0.522
```

```
## size of bias in relation to the std error
bias_size <- (bias/sd(boot_IQR)) %>% round(3)
bias_size
```

```
## [1] 0.436
```

The bias is approximately 44% of the $s.e.(\hat{\theta})$. The size of this bias is considerable.

f)

```
## bias corrected Efron interval
(quantile(boot_IQR, probs = c(0.025, 0.975)) - bias) %>% round(3)
```

```
##  2.5% 97.5%
## 4.768 9.583
```

The lower bound of the confidence interval is above $4. We reject the hypothesis that the test IQR could be below 4NZD at the 5% confidenc interval.

Question 2

a)

1. Calculate the observed $\hat{\beta}$ and $\hat{\sigma}^2$ from the observed data
2. Draw a sample of the observations with replacement and calculate a new estimate $\hat{\beta}_b^*$ from the sample
3. Repeat step 2 N times
4. Calculate $s.e.(\hat{\beta}^*)$ as the standard error over the results of the bootstrapped samples
5. Calculate $\hat{\beta} \pm 1.96 \times s.e.(\hat{\beta}^*)$ (for 95% confidence interval)

b)

```
galaxy <- readr::read_delim("galaxies.csv", delim = ",", col_types = readr::cols())
velocity <- galaxy$v %>% as.numeric()
galaxy$d <- as.numeric(galaxy$d)
distance <- galaxy$d

##########################################
## Calculations
calc_beta <- function(v, d) sum(v)/sum(d)

calc_sigma <- function(v, d) {
  beta_est <- calc_beta(v, d)
  mean(1/d*(v - beta_est*d)^2)
}
##########################################

n <- length(distance)
beta_est <- calc_beta(velocity, distance)
sigma_est <- calc_sigma(velocity, distance)

N <- 1e4
```

```r
set.seed(1)
bootstrap_velocity <- 1:N %>%
  lapply(function(i) {
    beta_est*distance + rnorm(n, sd = sqrt(sigma_est * distance))})

bootstrap_beta <- bootstrap_velocity %>%
  sapply(calc_beta, distance)

bootstrap_sigma2 <- bootstrap_velocity %>%
  sapply(calc_sigma, distance)

estimate <- mean(bootstrap_beta)
ci <- beta_est + 1.96 * c(-1, 1) * sd(bootstrap_beta)
results <- c(estimate, ci) %>% round(3)
names(results) <- c("Estimate", "Lower", "Upper")
results ## boostrap results for beta
```

```
## Estimate    Lower    Upper
##   76.036   59.630   92.351
```

```r
bootstrap_sigma2_est <- mean(bootstrap_sigma2)
ci <- sigma_est + 1.96 * c(-1, 1) * sd(bootstrap_sigma2)
sigma2_results <- c(bootstrap_sigma2_est, ci)
names(sigma2_results) <- c("Estimate", "Lower", "Upper")
sigma2_results ## bootstrap results for sigma squared
```

```
##  Estimate     Lower     Upper
##  8834.016  2164.639 16987.697
```

```r
## Reserve results for part d)
bootstrap_results <- rbind(results, sigma2_results)
rownames(bootstrap_results) <- c("Beta", "Sigma2")
```

Using the boot package

```r
boot_beta <- function(dataset, beta_est, sigma_est) {
  v <- beta_est*dataset$d + rnorm(n, sd = sqrt(sigma_est * dataset$d))
  calc_beta(v, dataset$d)
}

library(boot)
set.seed(1)
boot_stats <- boot(galaxy,
                   sim = "parametric",
                   statistic = boot_beta,
                   R = N,
                   beta_est = beta_est,
                   sigma_est = sigma_est)

boot_stats_se <- boot_stats$t %>% sd() %>% round(3)

estimate <- mean(boot_stats$t)
ci <- beta_est + 1.96 * c(-1, 1) * boot_stats_se

boot_results <- c(estimate, ci) %>% round(3)
```

```
names(boot_results) <- c("Estimate", "Lower", "Upper")
boot_results
```

```
## Estimate    Lower    Upper
##   76.035   59.630   92.350
```

c)

$y_i \sim N(\beta x_i, \sigma^2 x_i)$

$$L(\beta, \sigma | \boldsymbol{y}) = \prod_{i=1}^{n} \left[(2\pi\sigma^2 x_i)^{1/2} \exp(-\frac{(y_i - \beta x_i)^2}{2\sigma^2 x_i}) \right]$$

$$\ell = -\frac{1}{n}\sum_{i=1}^{n} \log(2\pi\sigma^2) - \frac{1}{n}\sum_{i=1}^{n} \log(x_i) - \frac{1}{2\sigma^2}\sum_{i=1}^{n} x_i^{-1}(y_i - \beta x_i)^2$$

Solving for $\hat{\beta}$ we only need the last term.

$$\frac{\partial \ell}{\partial \beta} = \frac{1}{\sigma^2}\sum_{i=1}^{n}(y_i - \beta x_i) \quad \text{setting equal to 0}$$

$$\frac{1}{\sigma^2}\sum_{i=1}^{n} y_i = \beta \frac{1}{\sigma^2}\sum_{i=1}^{n} x_i$$

$$\beta = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i}$$

$$\hat{\beta} = \frac{\bar{y}}{\bar{x}}$$

Solving for $\hat{\sigma}^2$

$$\frac{\partial \ell}{\partial \sigma^2} = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_{i=1}^{n} x_i^{-1}(y_i - \beta x_i)^2 \quad \text{setting equal to 0}$$

$$\frac{n}{2\sigma^2} = \frac{1}{2\sigma^4}\sum_{i=1}^{n} x_i^{-1}(y_i - \beta x_i)^2$$

$$\frac{2\sigma^4}{2\sigma^2} = \frac{1}{n}\sum_{i=1}^{n} x_i^{-1}(y_i - \beta x_i)^2$$

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{x_i}(y_i - \beta x_i)^2$$

d)

```
loglikelihood <- function(params, x, y) {
  n <- length(x)
  beta_ <- params[1]
  sigma2_ <- params[2]
  -(n/2)*log(2*pi*sigma2_) - 0.5 * sum(log(x)) - (1/(2*sigma2_)) * sum((y - beta_*x)^2 / x)
}

result <- optim(par = c(40, 250),
                fn = loglikelihood,
                x = distance, y = velocity,
                lower = c(0, 250), upper = c(Inf, Inf),
                method = "L-BFGS-B",
                control = list(fnscale = -1),
```

```
                hessian = TRUE)

opt_mle_se <- solve(-result$hessian) %>% diag() %>% sqrt() %>% round(3)
ci <- result$par[1] + c(-1, 1) * 1.96 * opt_mle_se[1]
likelihood_beta_results <- c(result$par[1], ci) %>% round(3)
names(likelihood_beta_results) <- c("Estimate", "Lower", "Upper")
#likelihood_beta_results

ci <- result$par[2] + c(-1, 1) * 1.96 * opt_mle_se[2]
likelihood_sigma_results <- c(result$par[2], ci) %>% round(3)
names(likelihood_sigma_results) <- c("Estimate", "Lower", "Upper")
#likelihood_sigma_results

output <- rbind(likelihood_beta_results, likelihood_sigma_results)
rownames(output) <- c("Beta", "Sigma2")
#output
```

```
knitr::kable(bootstrap_results, caption = "Bootstrap Estimates")
```

Table 1: Bootstrap Estimates

	Estimate	Lower	Upper
Beta	76.036	59.630	92.351
Sigma2	8834.016	2164.639	16987.697

```
knitr::kable(output, caption = "optim Estimates")
```

Table 2: optim Estimates

	Estimate	Lower	Upper
Beta	75.990	59.548	92.433
Sigma2	9576.071	2223.131	16929.011

The estimates for $\hat{\beta}$ are very similar. The 95% confidence interval widths are 32.721 and 32.885

The estimates for $\hat{\sigma}^2$ are more divergent. The 95% confidence interval widths are 14823.1 and 14705.9. This is not surprising as the log-likelihood function is not symmetric around the test statistic.

Question 3

a)

irreducible: the chain must be irreducible because it must be able to fully explore the parameter space. It must be able to reach any point in the space given enough steps.

aperiodic: the chain must be aperiodic to avoid cycling between sets of values at regular intervals, this would prevent adequate mixing.

b)

Function setup code

```r
dirichletMH <- function(n, start, alpha, prop_width) {

  x_values <- vector("list", n)
  proposals <- vector("list", n)
  accepted <- rep(0, n)

  ## initialise chain
  x_values[[1]] <- start
  accepted[1] <- 1

  proposal_value <- function(prev_x, pos, prop_width)
    runif(1, min = prev_x[pos] - prop_width, max = prev_x[pos] + prop_width)

  for (i in 2:n) {
    prev_x <- x_values[[i-1]]
    x1 <- proposal_value(prev_x, 1, prop_width)
    x2 <- proposal_value(prev_x, 2, prop_width)
    x3 <- 1 - x1 - x2
    y <- c(x1, x2, x3)
    proposals[[i]] <- c(x1, x2, x3)

    r <-
      mixtools::ddirichlet(x = y, alpha)/
      mixtools::ddirichlet(x = prev_x, alpha)

    aprob <- min(1, r)
    if (runif(1) < aprob) {
      x_values[[i]] <- y
      accepted[i] <- 1
    } else {
      x_values[[i]] <- prev_x
      accepted[i] <- 0
    }
  }
  list(x_values = x_values, proposals = proposals, accepted = accepted)
}

## initialise common variables
burnin <- 1000
nmcmc <- 4000
lastk <- (nmcmc+1):(burnin+nmcmc) ## last 1000 values
start <- c(0.1, 0.1, 0.8)
alpha <- c(1, 1, 1)
```

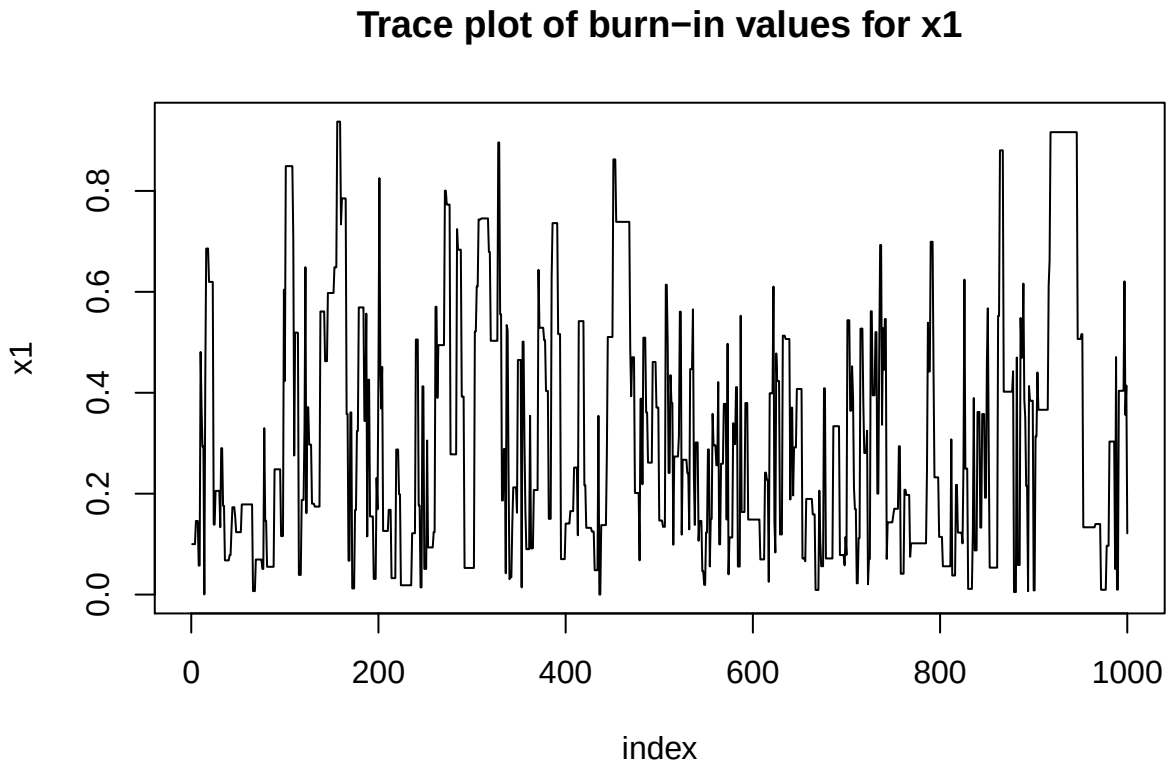
Proposal Distribution: Uniform up to 0.5 either side of the current point

i.

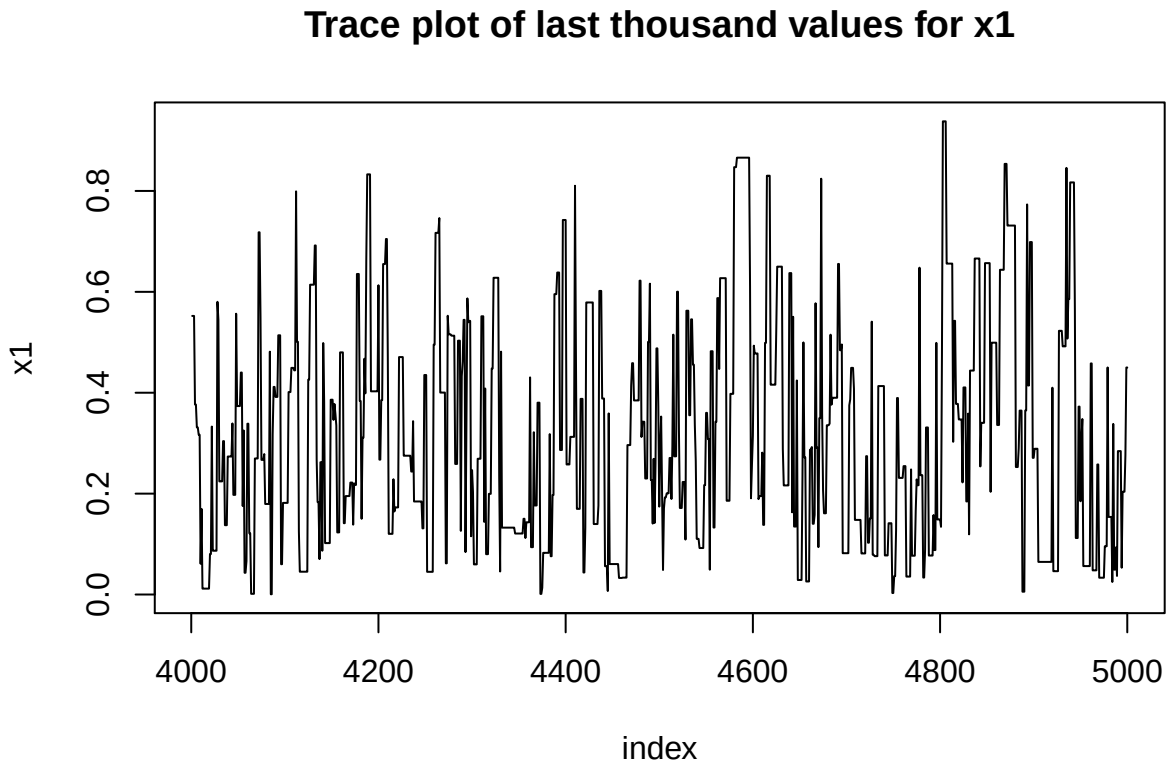
```r
set.seed(101)
results <- dirichletMH(burnin + nmcmc, start, alpha, prop_width = 0.5)
x_prop1 <- do.call(rbind, results$x_values) %>% as.data.frame()
names(x_prop1) <- paste0("x", 1:3)
```

ii.

```
plot(1:burnin, x_prop1$x1[1:burnin], type = "l",
     main = "Trace plot of burn-in values for x1", xlab = "index", ylab = "x1")
```




```
plot(lastk, x_prop1$x1[lastk], type = "l",
     main = "Trace plot of last thousand values for x1", xlab = "index", ylab = "x1")
```



iii. Acceptance ratio for last 1000 points in the chain

```
mean(x_prop1$x1[lastk])
```

```
## [1] 0.3160568
```

iv.

The burn-in is probably long enough as the trace plot shows it generally approaching as much of a convergence as is achieved in later stages.

It is hard to say how well it is mixing; there are a lot of points where x1 gets very high or low where it remains for too long, but it does appear to be exploring the space appropriately around a convergence.

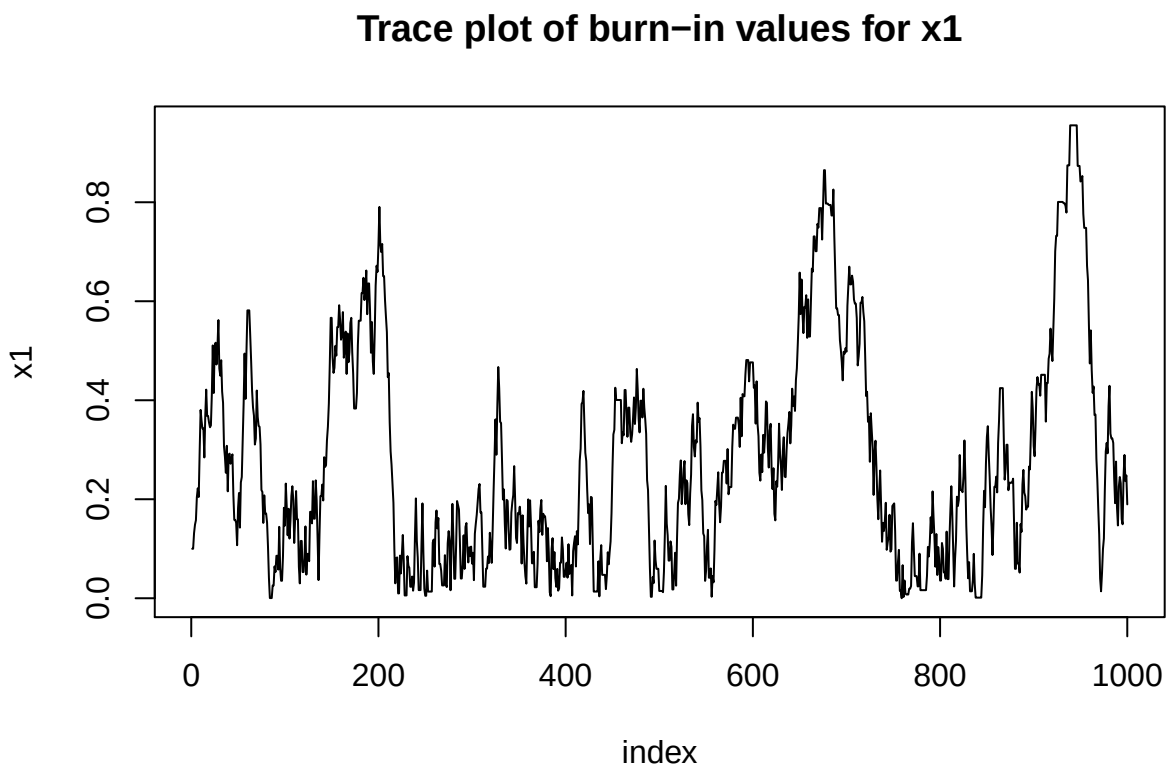
Proposal Distribution: Uniform up to 0.1 either side of the current point

i.

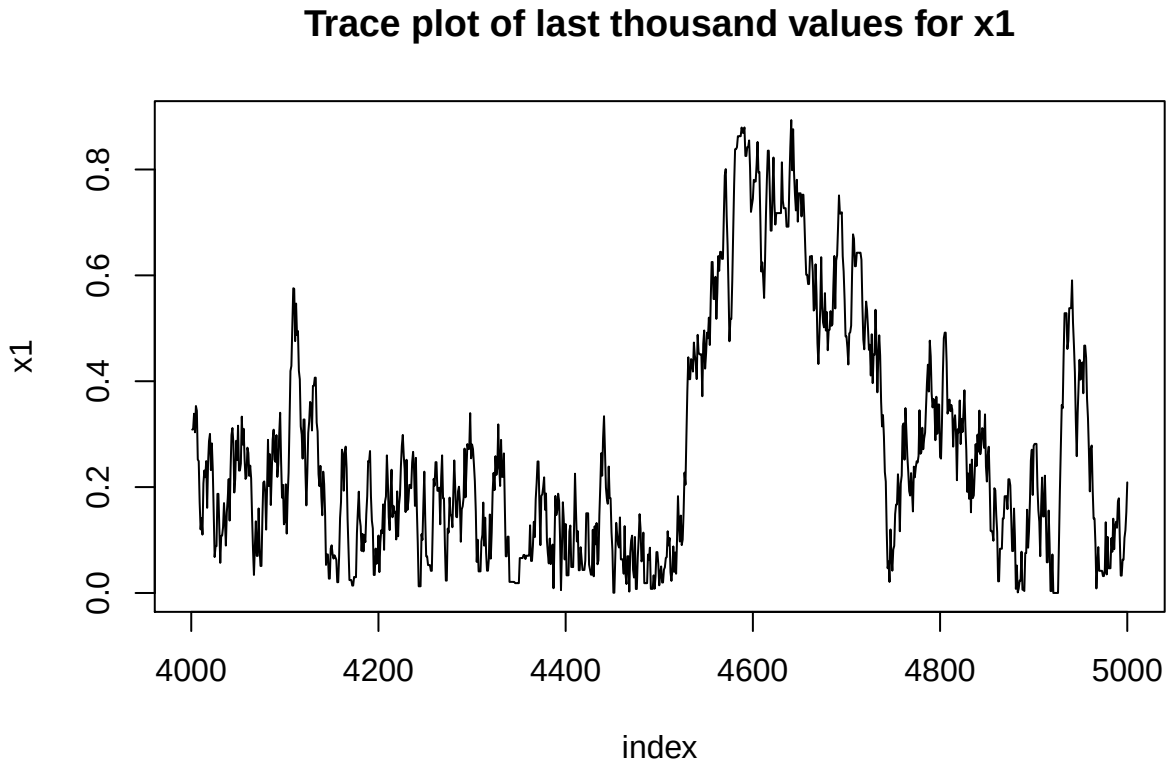
```
set.seed(101)
results <- dirichletMH(burnin + nmcmc, start, alpha, prop_width = 0.1)
x_prop2 <- do.call(rbind, results$x_values) %>% as.data.frame()
names(x_prop2) <- paste0("x", 1:3)
```

ii.

```
plot(1:burnin, x_prop2$x1[1:burnin], type = "l",
     main = "Trace plot of burn-in values for x1", xlab = "index", ylab = "x1")
```




```
plot(lastk, x_prop2$x1[lastk], type = "l",
     main = "Trace plot of last thousand values for x1", xlab = "index", ylab = "x1")
```



iii. Acceptance ratio for last 1000 points in the chain

```r
mean(x_prop2$x1[lastk])
```

```
## [1] 0.2688583
```

iv.

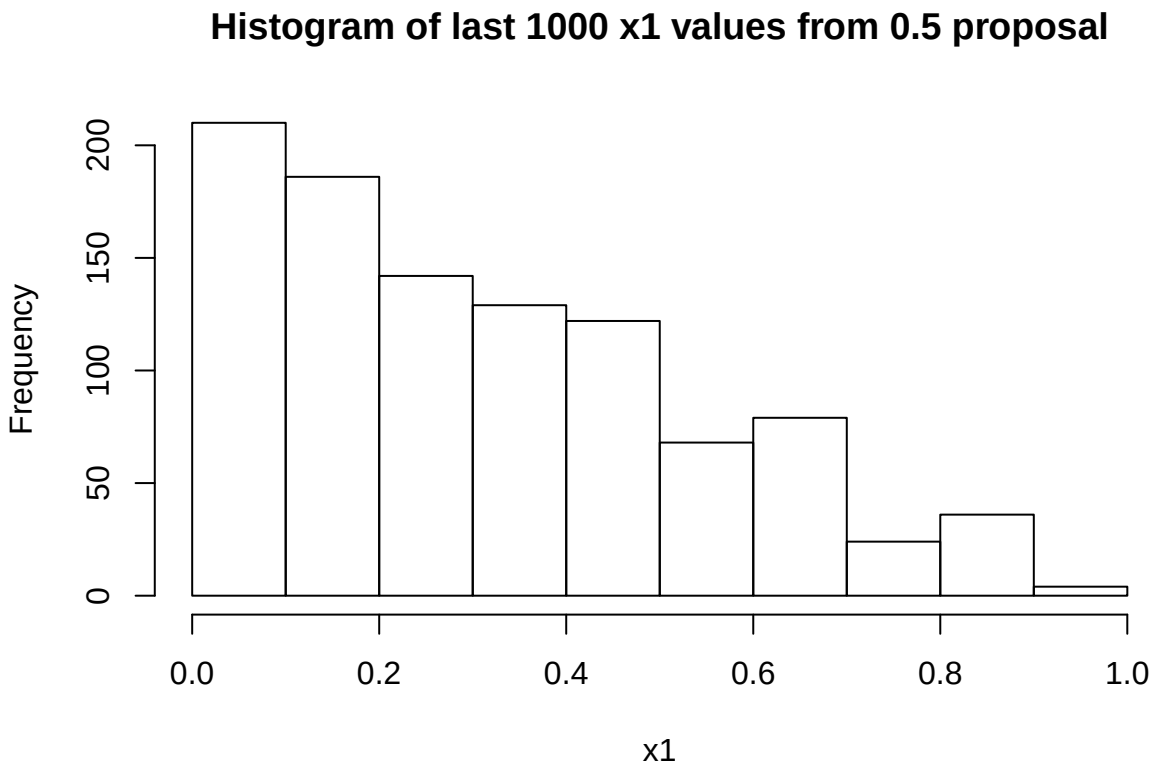
The burn-in is likely never going to be long enough to achieve convergence with this proposal.

It is mixing very poorly; it is not moving freely around the space and values are too correlated with previous values.

c)

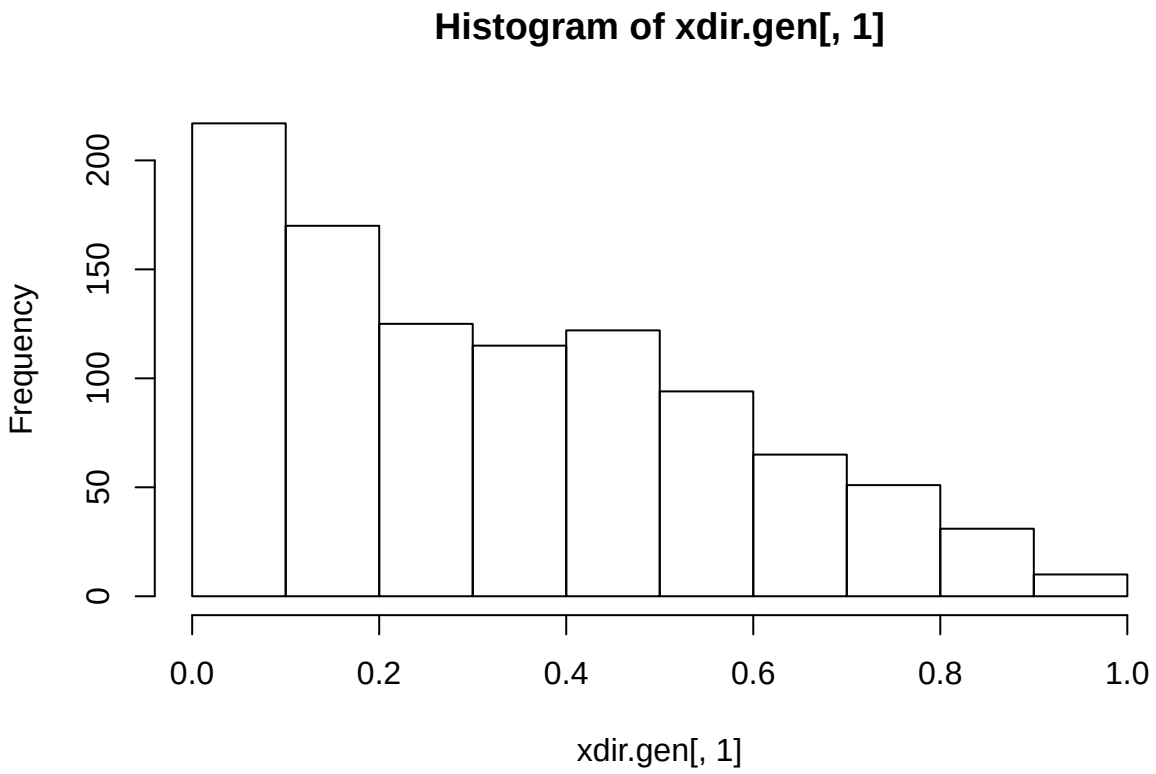
There are still some reservations about the first (0.5) proposal, but it is clearly mixing and converging much better than the alternative.

```r
hist(x_prop1$x1[lastk], main = "Histogram of last 1000 x1 values from 0.5 proposal", xlab = "x1", breaks
```



d)

```r
set.seed(100)
rDirichlet <- function(ndraw, alpha) {
  gamdraw <- rgamma(ndraw, shape = alpha, rate = 1)
  gamdraw/sum(gamdraw)
}
xdir.gen <- do.call(rbind, lapply(1:1000, function(i) rDirichlet(3, 1)))
hist(xdir.gen[,1])
```



The general shape of the distributions looks similar, suggesting the MCMC chain is an appropriate distribution and given more values for both sets of data they would likely converge to the same shape.