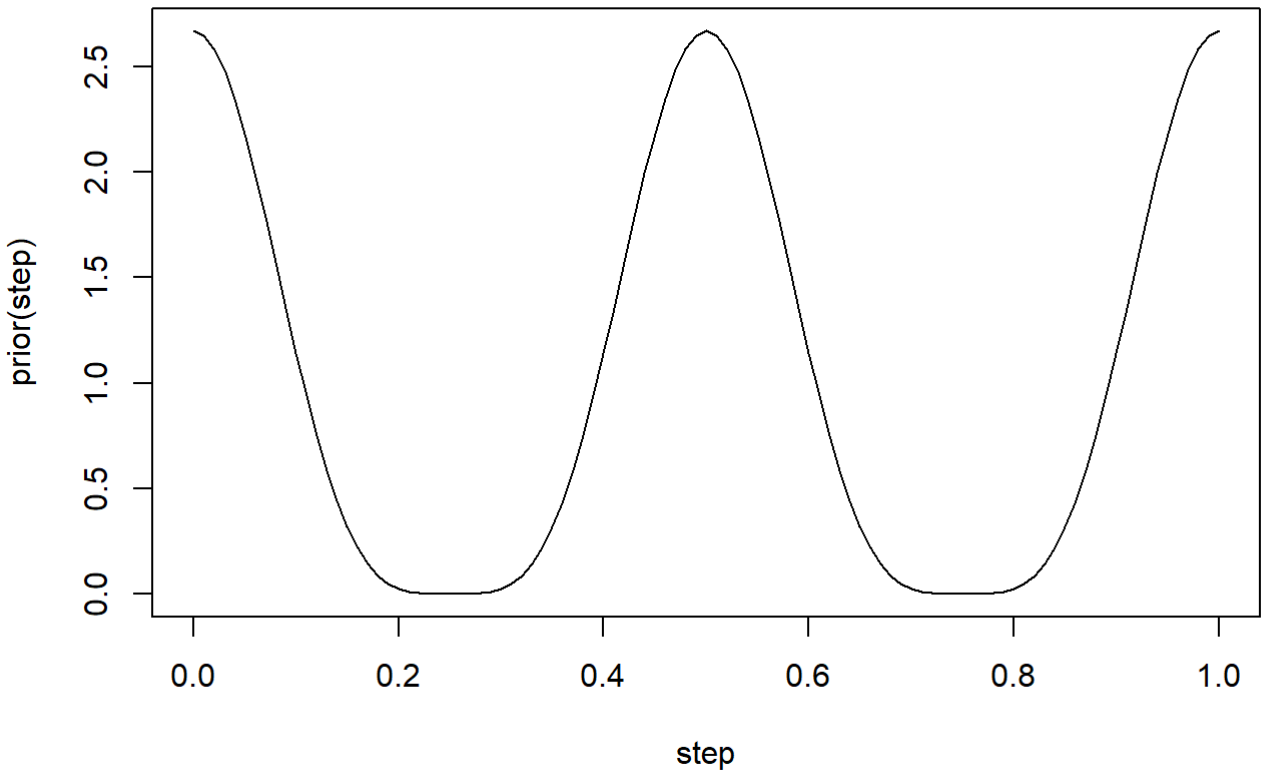
STATS 451 Homework 4

Yuzhou Peng

2024-02-20

Problem 1

```
prior <- function(x) {
  return (((cos(4*pi*x)+1)^2)/1.5)
}

step <- seq(0,1, by = 0.01)


plot(step, prior(step),
     type = "l")
```

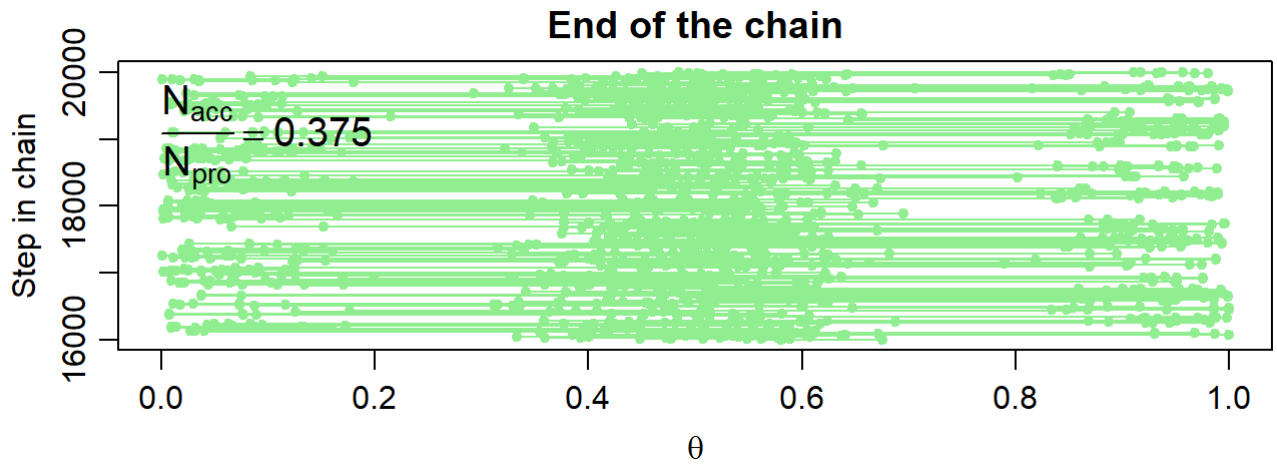


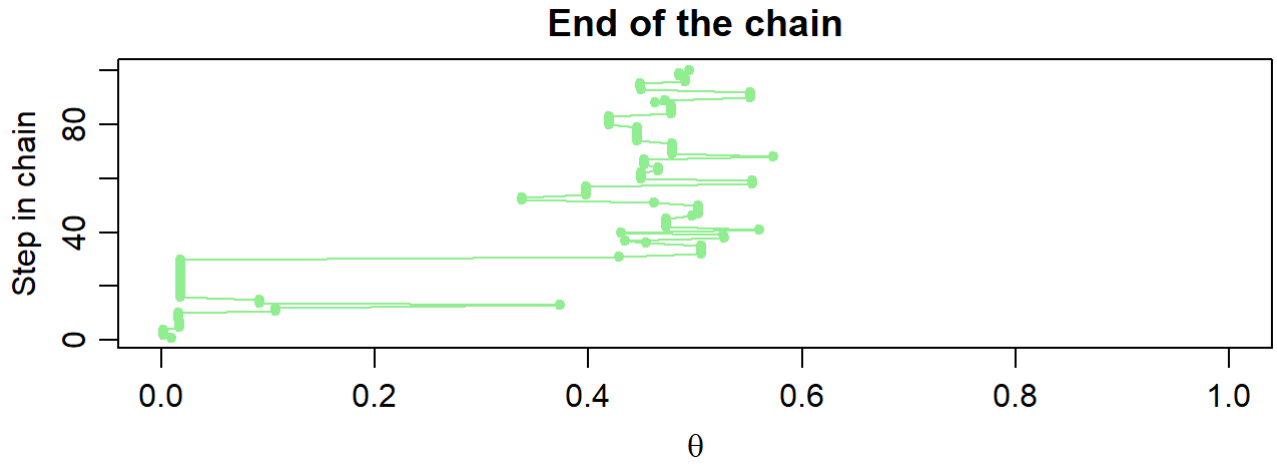
Problem 2

```
my_data = c()

# Define the likelihood function
likelihood <- function(theta, data) {
  z = sum(data)
  N = length(data)
  p_data_given_theta = theta^z * (1-theta)^(N-z)
  p_data_given_theta[ theta > 1 | theta < 0 ] = 0
  return (p_data_given_theta)
}

# Prior is defined in Problem 1

# Define the relative probability of the target distribution,
# as a function of vector theta. For our application, this
# target distribution is the unnormalized posterior distribution.
target_rel_prob <- function(theta, data){
  rel_posterior <- likelihood(theta, data) * prior(theta)
  return (rel_posterior)
}

# Number of iterations
traj_length <- 20000
# Create trajectory vector
trajectory <- rep(0, traj_length)



# Initializaiton
trajectory[1] <- 0.01

#Burn-in index
burn_in = ceiling( 0.2 * traj_length)



n_accepted <- 0
n_rejected <- 0

#Genertate random walks
set.seed(47405)
proposal_sd <- 0.2 #standard deviation of random walks

#For the for loop, input the target density function and iteration times
#and output the trajectory
for (t in 1:(traj_length -1)){
  current_position <- trajectory[t]
  proposed_jump <- rnorm(1, mean = 0, sd = proposal_sd)
  #Compute the acceptance probability
  prob_accept <- min(1,
                     target_rel_prob(current_position + proposed_jump, my_data)/target_rel_prob
(current_position, my_data))
  # Generate a random uniform value from the interval [0,1] to
  # decide whether or not to accept the proposed jump.
  if (runif(1) < prob_accept){
```

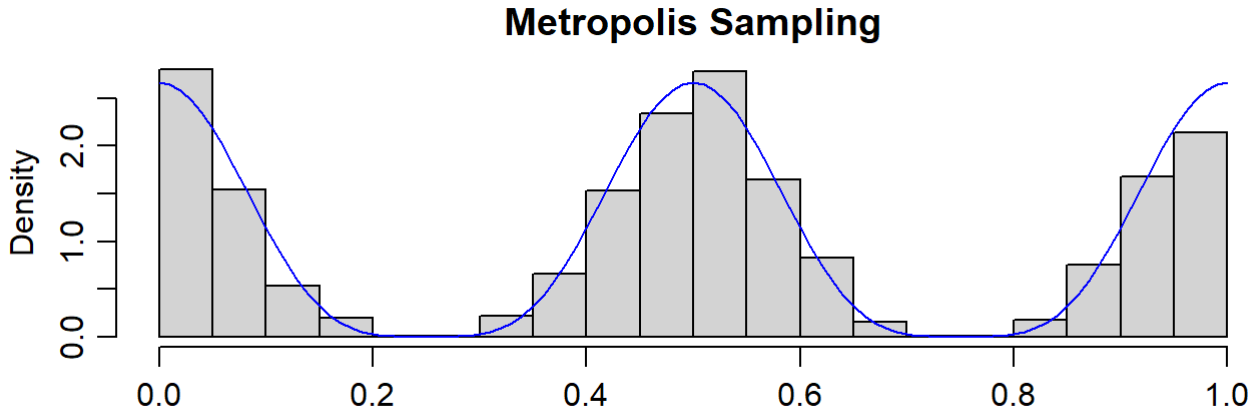
```
    #accept the proposed jump
    trajectory[t+1] <- current_position + proposed_jump
    if (t > burn_in){n_accepted = n_accepted + 1}
  } else {
    #reject the proposed jump and stay at the same point
    trajectory[t+1] <- current_position
    if (t > burn_in) {n_rejected = n_rejected + 1  }
  }
}

#Extract the post-burn-in portion of the trajectory
accepted_traj = trajectory[ (burn_in+1) : length(trajectory) ]
```

```
# Visualization

# Display the chain
layout( matrix(1:2,nrow=2) )
par(mar=c(3,4,2,1),mgp=c(2,0.7,0))

#Trace plot, end of the trajectory
idx_to_plot <- (traj_length - burn_in):traj_length

plot(trajectory[idx_to_plot], idx_to_plot,
     main = "End of the chain",
     xlab = bquote(theta), xlim = c(0,1),
     ylab = "Step in chain",
     type = "o", pch = 20, col = "lightgreen", cex.lab = 1)

# Display proposal SD and acceptance ratio
text( 0.0 , traj_length , adj=c(0.0,1.1) , cex=1.25 ,
      labels = bquote( frac(N[acc],N[pro]) ==
                       .(signif( n_accepted/(n_accepted+n_rejected) , 3 ))))

#Trace plot, start of the trajectory
idx_to_plot <- 1:100
plot(trajectory[idx_to_plot], idx_to_plot,
     main = "End of the chain",
     xlab = bquote(theta), xlim = c(0,1),
     ylab = "Step in chain",
     type = "o", pch = 20, col = "lightgreen", cex.lab = 1)

# Indicate burn in limit:
if ( burn_in > 0 )  {
  abline(h=burn_in,lty="dotted")
  text( 0.5 , burn_in+1 , "Burn In" , adj=c(0.5,1.1) )
}
```






```
# Posterior histogram
layout( matrix(1:2,nrow=2) )
par(mar=c(3,4,2,1),mgp=c(2,0.7,0))

idx_to_plot <- (traj_length - burn_in):traj_length

hist(trajectory[idx_to_plot],
     freq = FALSE, breaks = 15,
     xlab = '', main = 'Metropolis Sampling', xlim = c(0, 1))

lines(step, prior(step), col = "blue")
```

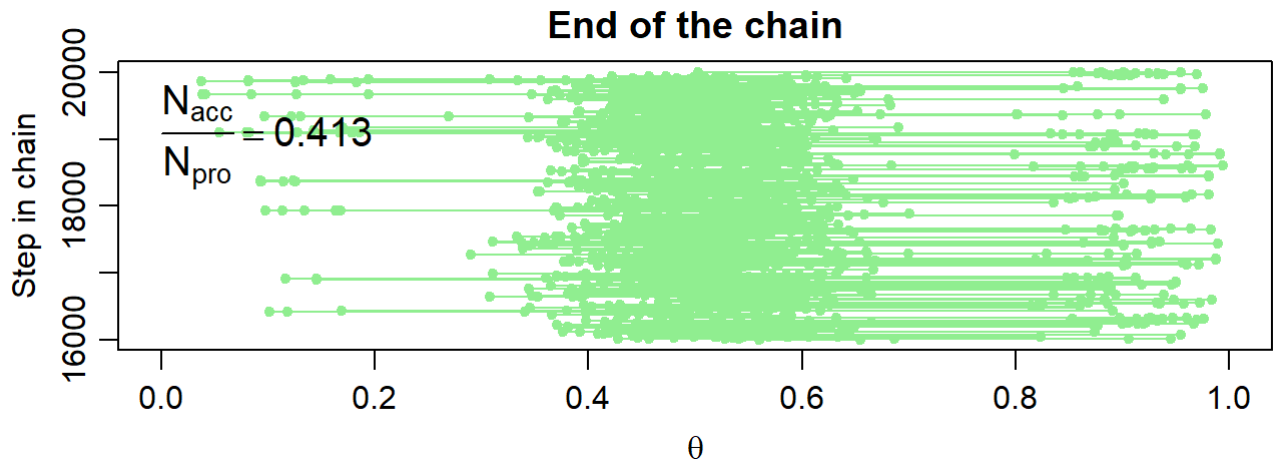


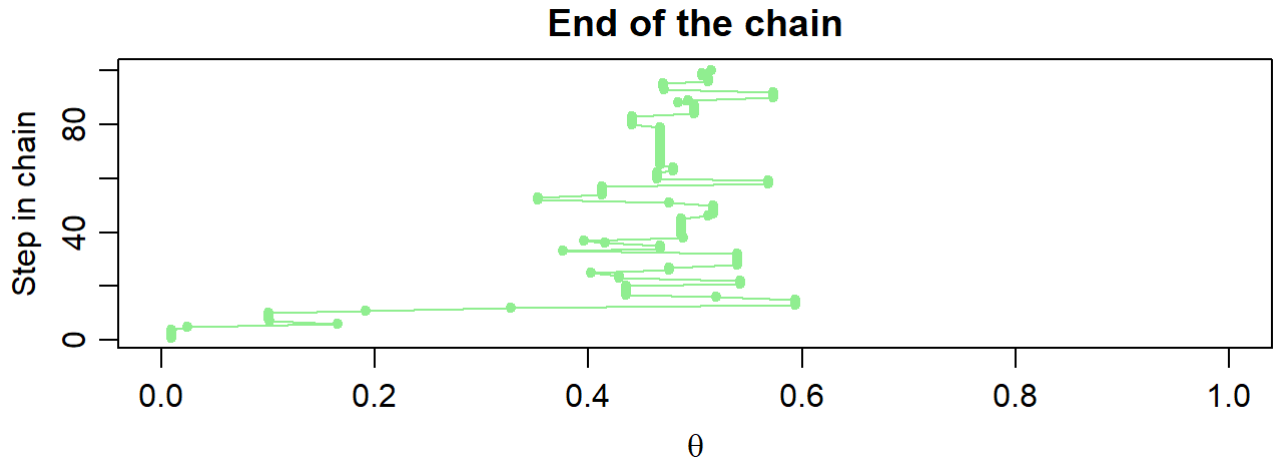
Yes, the histogram of trajectory looks similar to the graph of previous part.

Problem 3

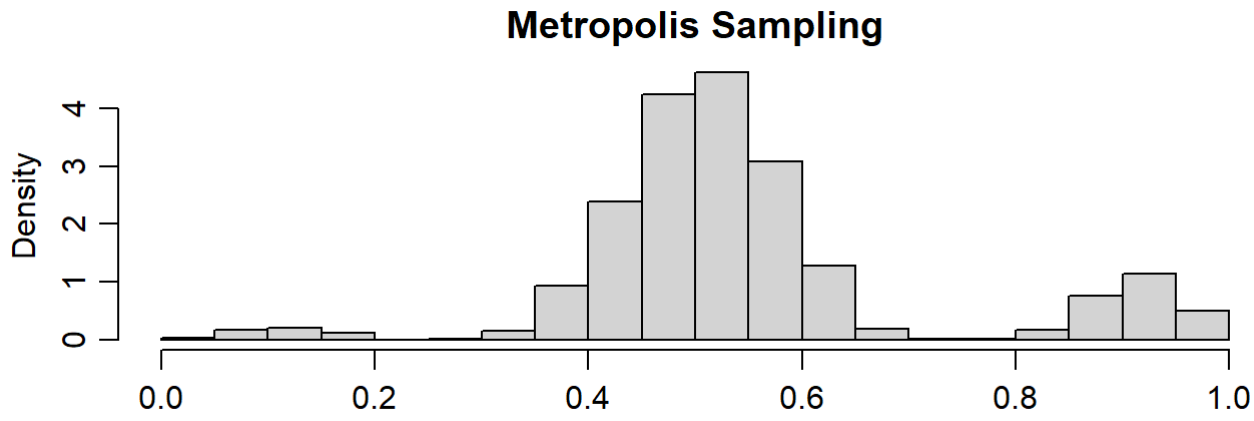
```
my_data = c(0,1,1)

# Define the likelihood function
likelihood <- function(theta, data) {
  z = sum(data)
  N = length(data)
  p_data_given_theta = theta^z * (1-theta)^(N-z)
  p_data_given_theta[ theta > 1 | theta < 0 ] = 0
  return (p_data_given_theta)
}

# Prior is defined in Problem 1

# Define the relative probability of the target distribution,
# as a function of vector theta. For our application, this
# target distribution is the unnormalized posterior distribution.
target_rel_prob <- function(theta, data){
  rel_posterior <- likelihood(theta, data) * prior(theta)
  return (rel_posterior)
}

# Number of iterations
traj_length <- 20000
# Create trajectory vector
trajectory <- rep(0, traj_length)



# Initializaiton
trajectory[1] <- 0.01

#Burn-in index
burn_in = ceiling( 0.2 * traj_length)



n_accepted <- 0
n_rejected <- 0

#Genertate random walks
set.seed(47405)
proposal_sd <- 0.2 #standard deviation of random walks

#For the for loop, input the target density function and iteration times
#and output the trajectory
for (t in 1:(traj_length -1)){
  current_position <- trajectory[t]
  proposed_jump <- rnorm(1, mean = 0, sd = proposal_sd)
  #Compute the acceptance probability
  prob_accept <- min(1,
                     target_rel_prob(current_position + proposed_jump, my_data)/target_rel_prob
(current_position, my_data))
  # Generate a random uniform value from the interval [0,1] to
  # decide whether or not to accept the proposed jump.
  if (runif(1) < prob_accept){
```

```
    #accept the proposed jump
    trajectory[t+1] <- current_position + proposed_jump
    if (t > burn_in){n_accepted = n_accepted + 1}
  } else {
    #reject the proposed jump and stay at the same point
    trajectory[t+1] <- current_position
    if (t > burn_in) {n_rejected = n_rejected + 1  }
  }
}

#Extract the post-burn-in portion of the trajectory
accepted_traj = trajectory[ (burn_in+1) : length(trajectory) ]

# Visualization

# Display the chain
layout( matrix(1:2,nrow=2) )
par(mar=c(3,4,2,1),mgp=c(2,0.7,0))

#Trace plot, end of the trajectory
idx_to_plot <- (traj_length - burn_in):traj_length

plot(trajectory[idx_to_plot], idx_to_plot,
     main = "End of the chain",
     xlab = bquote(theta), xlim = c(0,1),
     ylab = "Step in chain",
     type = "o", pch = 20, col = "lightgreen", cex.lab = 1)

# Display proposal SD and acceptance ratio
text( 0.0 , traj_length , adj=c(0.0,1.1) , cex=1.25 ,
      labels = bquote( frac(N[acc],N[pro]) ==
                       .(signif( n_accepted/(n_accepted+n_rejected) , 3 ))))

#Trace plot, start of the trajectory
idx_to_plot <- 1:100
plot(trajectory[idx_to_plot], idx_to_plot,
     main = "End of the chain",
     xlab = bquote(theta), xlim = c(0,1),
     ylab = "Step in chain",
     type = "o", pch = 20, col = "lightgreen", cex.lab = 1)

# Indicate burn in limit:
if ( burn_in > 0 )  {
  abline(h=burn_in,lty="dotted")
  text( 0.5 , burn_in+1 , "Burn In" , adj=c(0.5,1.1) )
}
```






```
# Posterior histogram
layout( matrix(1:2,nrow=2) )
par(mar=c(3,4,2,1),mgp=c(2,0.7,0))

idx_to_plot <- (traj_length - burn_in):traj_length

hist(trajectory[idx_to_plot],
     freq = FALSE, breaks = 15,
     xlab = '', main = 'Metropolis Sampling', xlim = c(0, 1))
```

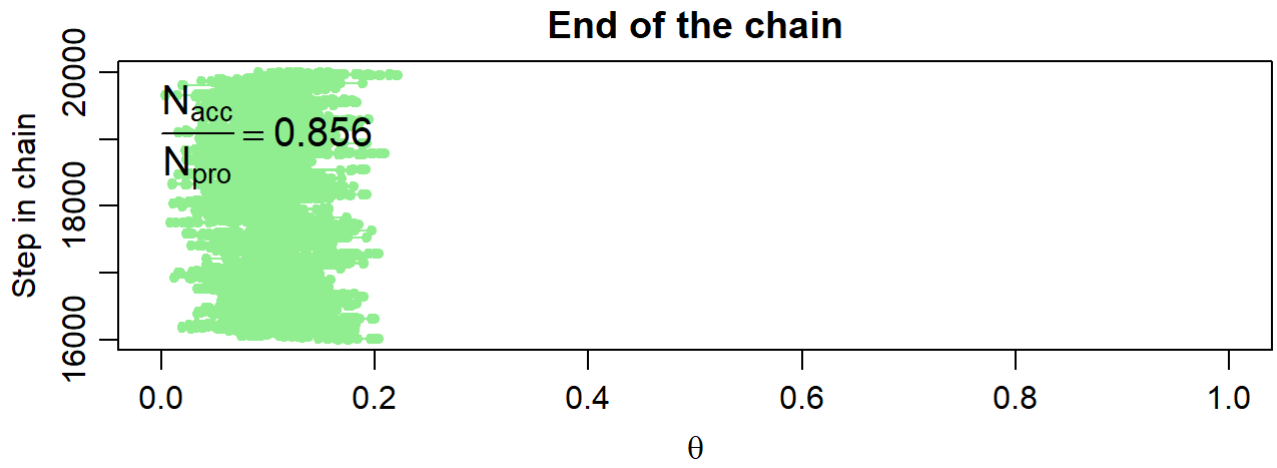


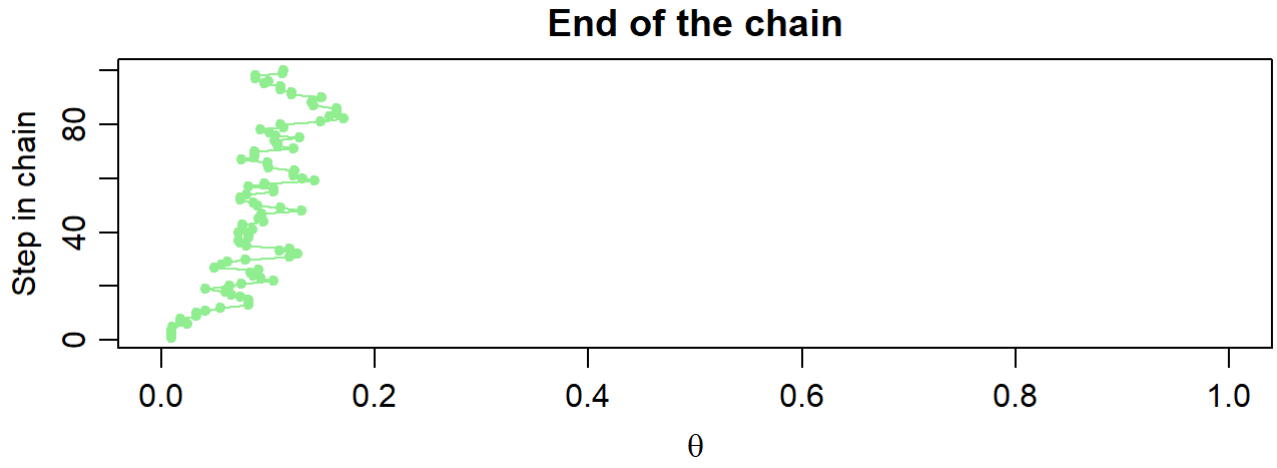
The posterior distribution makes sense because:
(1) Looking at the end of train, the trace plot display an distribution that is invariant of time (steps).
(2) The metropolis samples cover full range of support, (0,1) in this particular case.

Problem 4

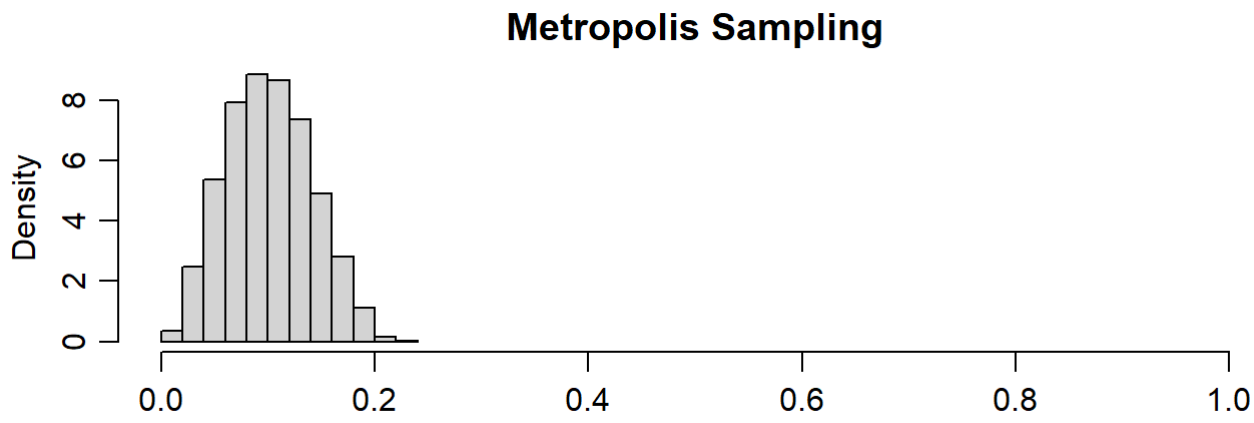
```
my_data = c(0,1,1)

# Define the likelihood function
likelihood <- function(theta, data) {
  z = sum(data)
  N = length(data)
  p_data_given_theta = theta^z * (1-theta)^(N-z)
  p_data_given_theta[ theta > 1 | theta < 0 ] = 0
  return (p_data_given_theta)
}

# Prior is defined in Problem 1

# Define the relative probability of the target distribution,
# as a function of vector theta. For our application, this
# target distribution is the unnormalized posterior distribution.
target_rel_prob <- function(theta, data){
  rel_posterior <- likelihood(theta, data) * prior(theta)
  return (rel_posterior)
}

# Number of iterations
traj_length <- 20000
# Create trajectory vector
trajectory <- rep(0, traj_length)



# Initializaiton
trajectory[1] <- 0.01

#Burn-in index
burn_in = ceiling( 0.2 * traj_length)



n_accepted <- 0
n_rejected <- 0

#Genertate random walks
set.seed(47405)
proposal_sd <- 0.02 #standard deviation of random walks

#For the for loop, input the target density function and iteration times
#and output the trajectory
for (t in 1:(traj_length -1)){
  current_position <- trajectory[t]
  proposed_jump <- rnorm(1, mean = 0, sd = proposal_sd)
  #Compute the acceptance probability
  prob_accept <- min(1,
                     target_rel_prob(current_position + proposed_jump, my_data)/target_rel_prob
(current_position, my_data))
  # Generate a random uniform value from the interval [0,1] to
  # decide whether or not to accept the proposed jump.
  if (runif(1) < prob_accept){
```

```
    #accept the proposed jump
    trajectory[t+1] <- current_position + proposed_jump
    if (t > burn_in){n_accepted = n_accepted + 1}
  } else {
    #reject the proposed jump and stay at the same point
    trajectory[t+1] <- current_position
    if (t > burn_in) {n_rejected = n_rejected + 1  }
  }
}

#Extract the post-burn-in portion of the trajectory
accepted_traj = trajectory[ (burn_in+1) : length(trajectory) ]


# Visualization

# Display the chain
layout( matrix(1:2,nrow=2) )
par(mar=c(3,4,2,1),mgp=c(2,0.7,0))

#Trace plot, end of the trajectory
idx_to_plot <- (traj_length - burn_in):traj_length

plot(trajectory[idx_to_plot], idx_to_plot,
     main = "End of the chain",
     xlab = bquote(theta), xlim = c(0,1),
     ylab = "Step in chain",
     type = "o", pch = 20, col = "lightgreen", cex.lab = 1)

# Display proposal SD and acceptance ratio
text( 0.0 , traj_length , adj=c(0.0,1.1) , cex=1.25 ,
      labels = bquote( frac(N[acc],N[pro]) ==
                         .(signif( n_accepted/(n_accepted+n_rejected) , 3 ))))

#Trace plot, start of the trajectory
idx_to_plot <- 1:100
plot(trajectory[idx_to_plot], idx_to_plot,
     main = "End of the chain",
     xlab = bquote(theta), xlim = c(0,1),
     ylab = "Step in chain",
     type = "o", pch = 20, col = "lightgreen", cex.lab = 1)

# Indicate burn in limit:
if ( burn_in > 0 )  {
  abline(h=burn_in,lty="dotted")
  text( 0.5 , burn_in+1 , "Burn In" , adj=c(0.5,1.1) )
}
```






```
# Posterior histogram
layout( matrix(1:2,nrow=2) )
par(mar=c(3,4,2,1),mgp=c(2,0.7,0))

idx_to_plot <- (traj_length - burn_in):traj_length

hist(trajectory[idx_to_plot],
     freq = FALSE, breaks = 15,
     xlab = '', main = 'Metropolis Sampling', xlim = c(0, 1))
```

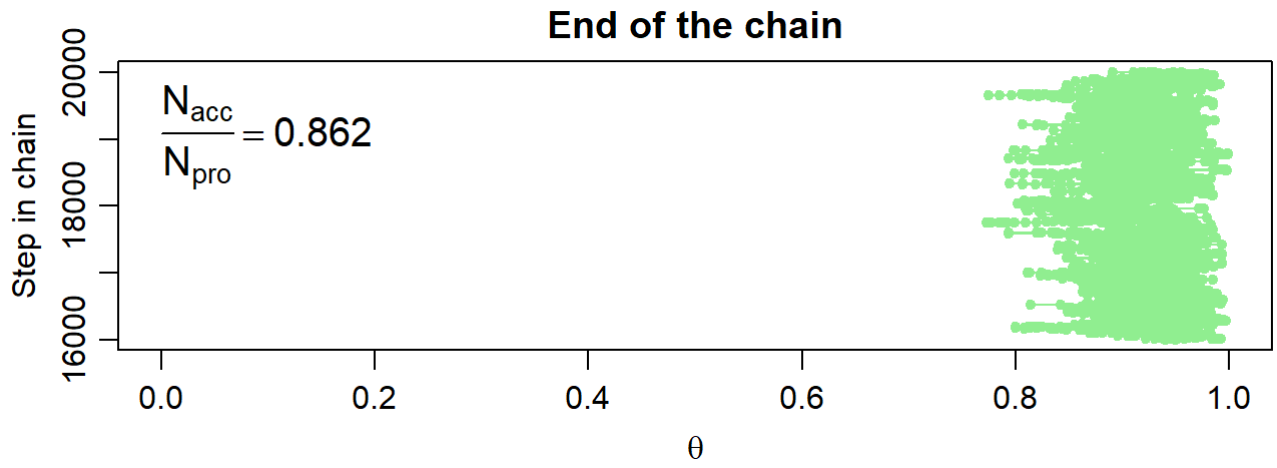


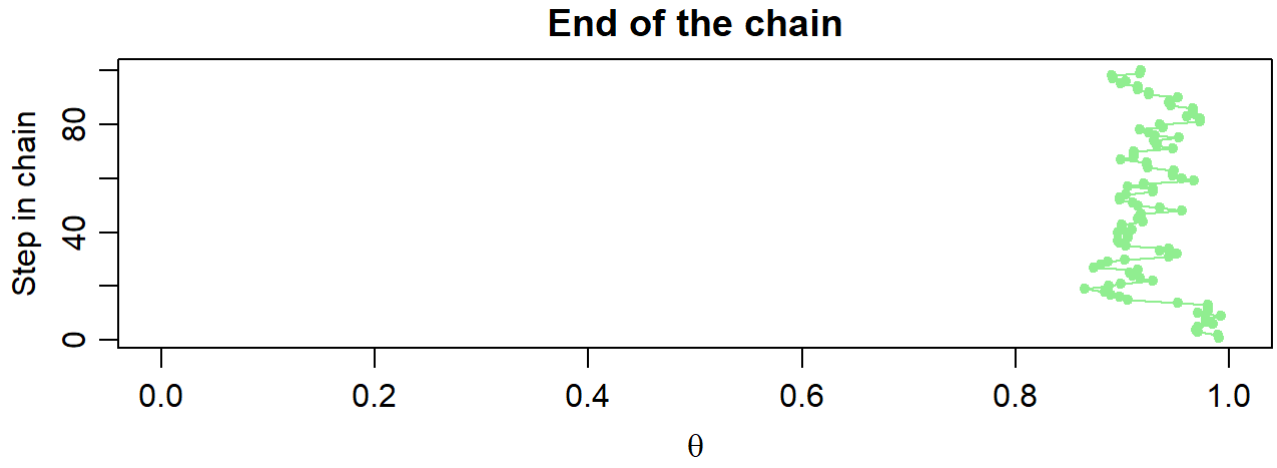
The posterior sample do not make sense because the samples do not mix well.
The range of support, (0,1), is not fully covered by the metropolis sample.
The reason of this is the variance of random walk is so small that our algorithm have very low probability to walk across the area where density is 0 and explore the area outside
of (0,0.2).

Problem 5

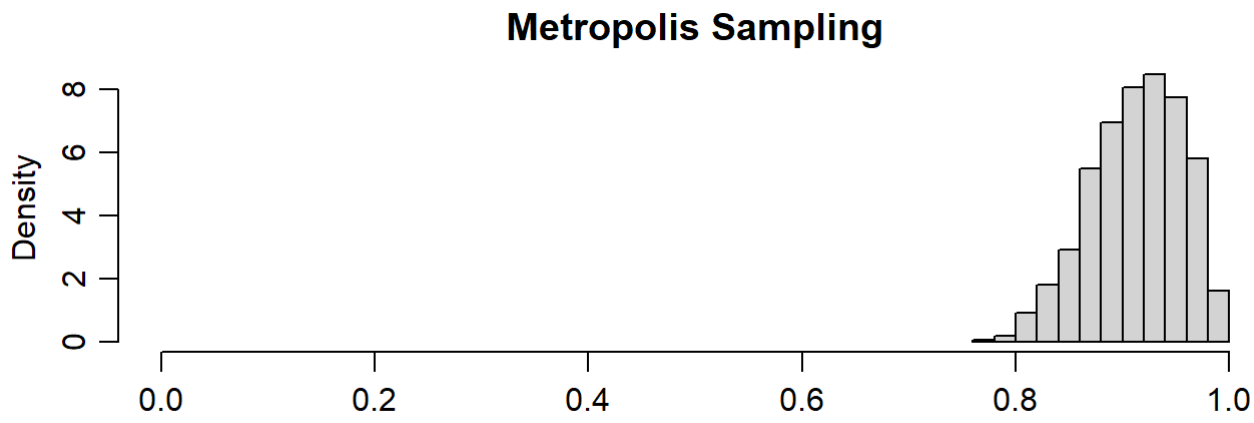
```
my_data = c(0,1,1)

# Define the likelihood function
likelihood <- function(theta, data) {
  z = sum(data)
  N = length(data)
  p_data_given_theta = theta^z * (1-theta)^(N-z)
  p_data_given_theta[ theta > 1 | theta < 0 ] = 0
  return (p_data_given_theta)
}

# Prior is defined in Problem 1

# Define the relative probability of the target distribution,
# as a function of vector theta. For our application, this
# target distribution is the unnormalized posterior distribution.
target_rel_prob <- function(theta, data){
  rel_posterior <- likelihood(theta, data) * prior(theta)
  return (rel_posterior)
}

# Number of iterations
traj_length <- 20000
# Create trajectory vector
trajectory <- rep(0, traj_length)



# Initializaiton
trajectory[1] <- 0.99

#Burn-in index
burn_in = ceiling( 0.2 * traj_length)



n_accepted <- 0
n_rejected <- 0

#Genertate random walks
set.seed(47405)
proposal_sd <- 0.02 #standard deviation of random walks

#For the for loop, input the target density function and iteration times
#and output the trajectory
for (t in 1:(traj_length -1)){
  current_position <- trajectory[t]
  proposed_jump <- rnorm(1, mean = 0, sd = proposal_sd)
  #Compute the acceptance probability
  prob_accept <- min(1,
                     target_rel_prob(current_position + proposed_jump, my_data)/target_rel_prob
(current_position, my_data))
  # Generate a random uniform value from the interval [0,1] to
  # decide whether or not to accept the proposed jump.
  if (runif(1) < prob_accept){
```

```
    #accept the proposed jump
    trajectory[t+1] <- current_position + proposed_jump
    if (t > burn_in){n_accepted = n_accepted + 1}
  } else {
    #reject the proposed jump and stay at the same point
    trajectory[t+1] <- current_position
    if (t > burn_in) {n_rejected = n_rejected + 1  }
  }
}

#Extract the post-burn-in portion of the trajectory
accepted_traj = trajectory[ (burn_in+1) : length(trajectory) ]


# Visualization

# Display the chain
layout( matrix(1:2,nrow=2) )
par(mar=c(3,4,2,1),mgp=c(2,0.7,0))

#Trace plot, end of the trajectory
idx_to_plot <- (traj_length - burn_in):traj_length

plot(trajectory[idx_to_plot], idx_to_plot,
     main = "End of the chain",
     xlab = bquote(theta), xlim = c(0,1),
     ylab = "Step in chain",
     type = "o", pch = 20, col = "lightgreen", cex.lab = 1)

# Display proposal SD and acceptance ratio
text( 0.0 , traj_length , adj=c(0.0,1.1) , cex=1.25 ,
      labels = bquote( frac(N[acc],N[pro]) ==
                         .(signif( n_accepted/(n_accepted+n_rejected) , 3 ))))

#Trace plot, start of the trajectory
idx_to_plot <- 1:100
plot(trajectory[idx_to_plot], idx_to_plot,
     main = "End of the chain",
     xlab = bquote(theta), xlim = c(0,1),
     ylab = "Step in chain",
     type = "o", pch = 20, col = "lightgreen", cex.lab = 1)

# Indicate burn in limit:
if ( burn_in > 0 )  {
  abline(h=burn_in,lty="dotted")
  text( 0.5 , burn_in+1 , "Burn In" , adj=c(0.5,1.1) )
}
```






```
# Posterior histogram
layout( matrix(1:2,nrow=2) )
par(mar=c(3,4,2,1),mgp=c(2,0.7,0))

idx_to_plot <- (traj_length - burn_in):traj_length

hist(trajectory[idx_to_plot],
     freq = FALSE, breaks = 15,
     xlab = '', main = 'Metropolis Sampling', xlim = c(0, 1))
```



The posterior sample still do not make sense for the same reason mentioned in Problem 4.
The reason of this is the variance of random walk is so small that our algorithm have very low probability to walk across the area where density is 0 and explore the area outside
of (0.7, 1).

In conjunction with previous part, we know:
(1) The design of random walk may affect effectiveness of the metropolis algorithm. Only appropriate random walk can output reasonable samples.
(2) To check if the metropolis samples make sense, we can run the algorithm using different starting points. If the results are sensitive to the selection of initial
points, we may also conclude the samples do not make sense.