Maggie O'Shea
Bayesian Statistical Modeling and Computation
Professor Klaus Keller
Problem Set 4
February 14, 2025

1. **Revisit the estimation problem about the fuel level (just this part) from the previous problem set using an implementation of the Metropolis Hasting algorithm.**

I implemented the Metropolis Hastings algorithm to estimate the tank in the fuel. Here I used a log-likelihood function for a normal distribution to evaluate candidate values in the MCMC analysis, and my initial value was the observed 34 liters of fuel. The resulting mean estimate of fuel from the MCMC results was 33.72 (Figure 1).

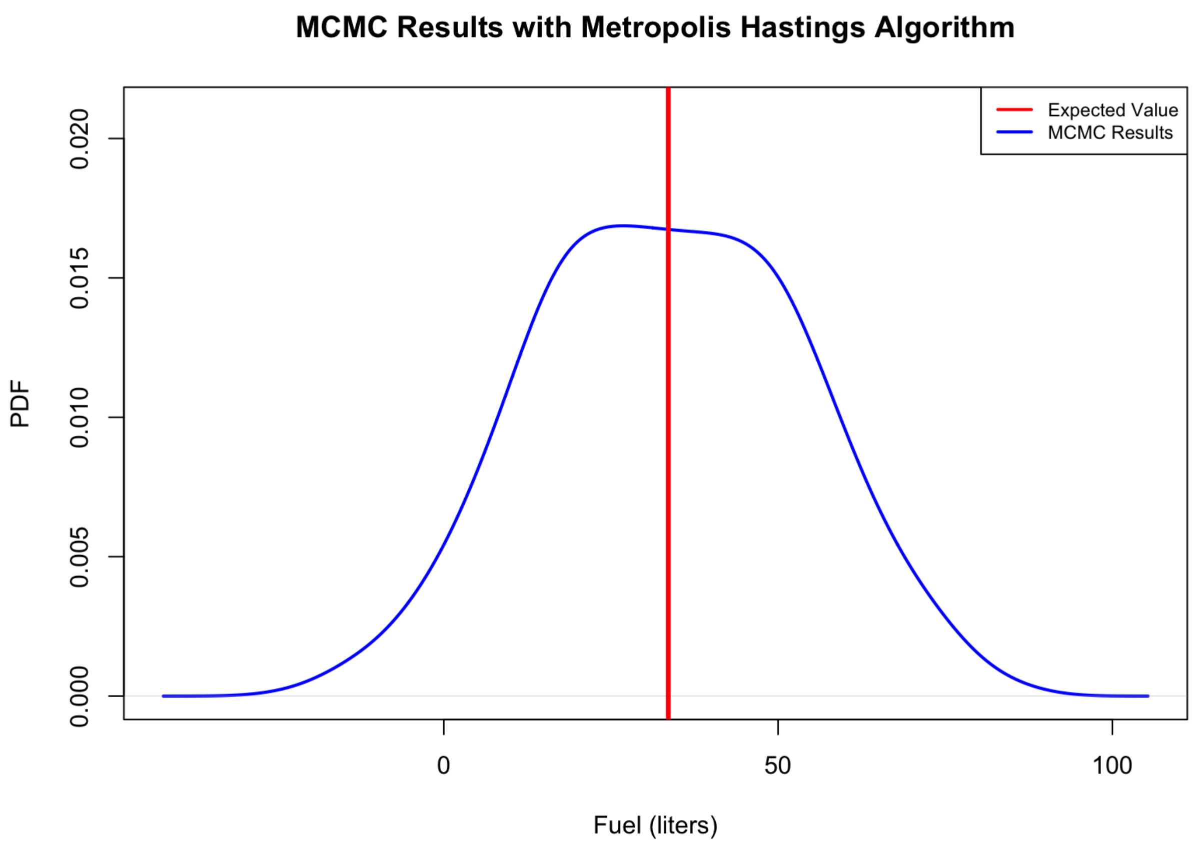

Figure 1: Results from Metropolis Hastings Markov Chain Monte Carlo simulation. Resulting expected value in red and PDF in blue.

Some choices that I made here include the seed, as well as the initial value, the step size, and the number of iterations. First, the seed introduces uncertainty – using multiple seeds would help evaluate the uncertainty and show the uncertainty bounds around the expected value of fuel, for example, brought on by seed uncertainty. Additionally, I selected the initial value - not shown here are tests in which I ran the MCMC with different initial values. I found that with these changes the expected value for fuel did not change dramatically, however this was an informal test not shown in a plot. Still, this does introduce some uncertainty and showing results from multiple starting points may result in some variability around the results. The more impactful choice was the step size, which I did make smaller to support convergence. While this didn’t necessarily change the expected value of fuel in a significant way, it did, again, help facilitate convergence. Relatedly, I chose the number of samples also based on convergence – I started with a lower number of samples (10,000) and increased the number of samples until the credible interval width (0.05) test for convergence was passed. Each of these choices could introduce uncertainty in the result and exploring them more deeply and quantitatively would likely result in a better understanding of the range of results associated with different analytical choices. This code was made reproducible by commenting the code explicitly and setting seeds for reproducing the randomness used here.

A. assess whether your numerical inference about the mode converges (5 points),
Convergence was tested here using a test in which I estimate the credible interval and estimate if it is sufficiently small to consider the estimate precise, or converging around a particular value. In this case the threshold was 0.05. This value was selected based on previous work in the course using this threshold (See Workflow_example.R), however it’s quite narrow. A higher threshold might find convergence with fewer samples and require less computational power. This may introduce a type of uncertainty in which the threshold for convergence varies the result. Still, because having enough fuel for a flight is a safety concern, perhaps this narrow of a threshold is defensible. Beyond this, the uncertainty here is primarily related to the uncertainty discussed above - that is, the uncertainty accumulated when generating the results from the MCMC analysis. This was again made reproducible by commenting the code used to test for convergence.

B. define and use a positive control to assess the accuracy of your implemented method for the inference in part a (5 points), and

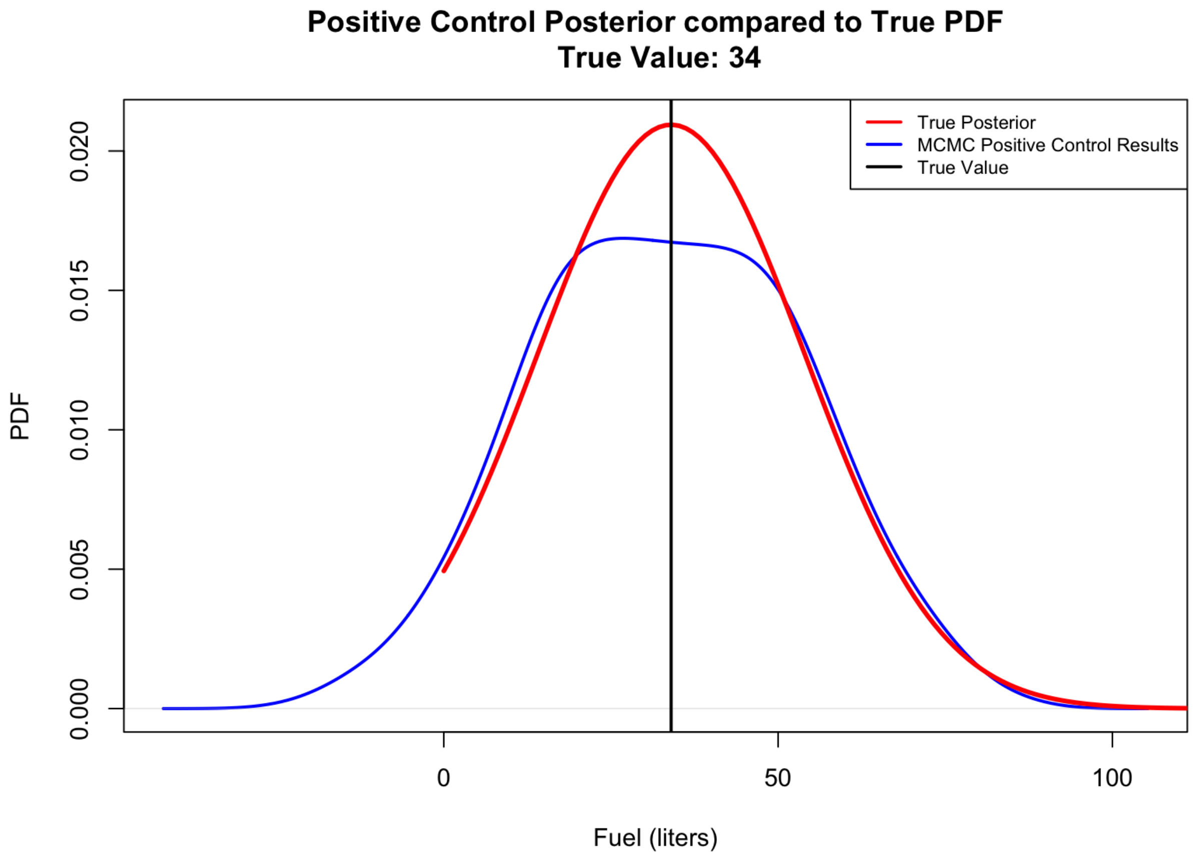

Figure 2: Positive Control comparing Metropolis Hastings Markov Chain Monte Carlo results with true PDF based on value 34 liters. True value (34) in black.

Here I used a positive control in which I set the true value to 34 liters. I then estimated a PDF using a prior uniform distribution of 0 to 182 and a likelihood function with the mean 34 and SD of 20. After running the MCMC and finding convergence the expected value from this analysis was 33.581 (Figure 2). It was very close to the true value selected (34). However, the two PDFs do differ (Figure 2). In particular, the uniform distribution between 0 and 182 that sets limits on the true PDF was not observed in the MCMC result, but, rather, the MCMC PDF allows for values beyond 0 which is impossible (to have negative fuel). In this way the expected value result from the MCMC is close to the truth, however, the resulting PDF includes values that are physically impossible.

C. **Compare the required number of runs needed for converged inferences in part b from the Metropolis Hasting algorithm with the Bayes Monte Carlo algorithm (5 points).**

Based on the convergence test selected, which was the 0.05 credible interval test, the two algorithms required the same number of runs. Lower runs were tested for each algorithm, increasing from 10,000 to 100,000 and then 2,000,000 and 3,000,000. Due to such a low threshold (0.05), the number of runs for each was quite high. When lowering the threshold such that the estimate of fuel could be within 0.25 liters, only 200,000 samples were needed.

Appendix I:

Code written in software R (Version 2024.12.0+467).

- Downloaded from: https://www.r-project.org/

Saved to Github Repository:
https://github.com/maggieoshea/BayesianStatisticalModelingandComputation/tree/main/Problem Set4

Full R Script below.

```
############################################
##  file: margaret.oshea.gr@dartmouth.edu#4.R
## Written on R Version 2024.12.0+467
######################################################
##  Maggie O'Shea
##  copyright by the author
##  distributed under the GNU general public license
##  https://www.gnu.org/licenses/gpl.html
##  no warranty (see license details at the link above)
######################################################
##   Course: Bayesian Statistical Modeling & Computation
##   Professor Klaus Keller
##   February 14, 2025
##   Problem Set #4
##################################################
# contact: margaret.oshea.gr@dartmouth.edu
##################################################
# sources:
# Coding Metropolis Hastings in R: https://rpubs.com/
ROARMarketingConcepts/1063733
# Bayesian Modeling Overview site: https://francescacapel.com/
BayesianWorkflow/notebooks/model_building.html
# Bayesian Statistics and Computation Course code: Workflow_Example.R
# Plotting MCMC results: https://www.bayesrulesbook.com/chapter-7
# Plotting histogram in R: https://stackoverflow.com/questions/
14825187/making-ggplot2-plot-density-histograms-as-lines
# Log-likelihood vs Likelihood: https://stats.stackexchange.com/
questions/289190/theoretical-motivation-for-using-log-likelihood-vs-
likelihood
# coding Log-likelihood function: https://rpubs.com/Koba/MLE-Normal
# Quantile function: https://www.rdocumentation.org/packages/stats/
versions/3.6.2/topics/quantile
# Conversation with Alexis Hudes
####################################################
# Clear any existing variables and plots.
rm(list = ls())
graphics.off()

## Packages ##
# If packages ggplot2 and dplyr not already downloaded, un-tag (remove
the #) to download packages before running the rest of the script.
#install.packages("ggplot2")
#install.packages("dplyr")
library(ggplot2)
library(dplyr)


# Q1: Revisit the estimation problem about the fuel level
# (just this part) from the previous problem set using an
#implementation of the Metropolis Hasting algorithm
```

```
# Log-likelihood function for a normal distribution to use in the
Metropolis-Hastings alg.
# (fn short for function!)
log_likelihood_fn <- function(mu_now, observed, sd) {
  resid = observed - mu_now
  log_likelihood = -0.5 * log(2 * pi) - 0.5 *log(sd) - (resid^2) / (2
* sd^2)
  return(log_likelihood)
}

## Metropolis-Hastings algorithm (mh)
# Inputs are:
# n_iter: number of iterations:
# mu_initial: starting point mean (mu)
# candidate_sd: similar to step_size it is the standard deviation for
the selection of the candidate mean to be used in the MH and compared
to likelihood of original mean
mh <- function(n_iter, mu_initial, candidate_sd) {

  ## Initialize
  mu_out = numeric(n_iter)
  accept = 0
  mu_now = mu_initial
  original_likelihood = log_likelihood_fn(mu_now, observed, tanksd)

  ## Iterate through candidates
  for (i in 1:n_iter) {
    ## Randomly sample a candidate
    mu_cand = rnorm(n=1, mean=mu_now, sd=candidate_sd)

    ## Likelihood Function with Candidate
    lk_cand = log_likelihood_fn(mu_cand, observed, tanksd)

    ## Acceptance Ratio
    alpha = min(1, exp(lk_cand - original_likelihood))
    ## Draw variable between 0 and 1 to compare alpha to for
acceptance
    # so if the acceptance ratio is greater than the randomly sampled
value from 0 to 1 then its accepted
    u = runif(1)
    if (u < alpha) { # if random variable < alpha, then accept
      mu_now = mu_cand
      accept = accept + 1 # count acceptances
      lk_now = lk_cand
    }

    ## Save iteration's mu
    mu_out[i] = mu_now
  }
  ## return list of results
  list(mu=mu_out, accept=accept/n_iter)
}

# Function defined.. now run MCMC!
set.seed(930) # set the random seed for reproducibility
mu_initial = 50
observed = 34
tanksd = 20
n_iter = 30*10^5
candidate_sd=1
mcmcresult = mh(n_iter, mu_initial, candidate_sd)

plot(mcmcresult$mu, type = "l", col = "blue",
     xlab = "Iteration", ylab = "Fuel (liters)",
     main = "Values of Fuel in MCMC")
abline(h = mean(mcmcresult$mu), col = "black", lty = 2, lwd = 2)
print(mean(mcmcresult$mu))

mu_samples <- mcmcresult$mu
mc_df <- data.frame(mu = mcmcresult$mu)
plot(density(mcmcresult$mu, adjust=5), col = "blue",
     lwd=2,
     ylab= "PDF",
     xlab='Fuel (liters)',
     main="MCMC Results with Metropolis Hastings Algorithm",
     ylim=c(0, 0.021))
abline(v=mean(mc_df$mu), col = "red", lwd = 3)
legend("topright", legend = c("Expected Value", "MCMC Results"),
       col = c("red", "blue"),
       lwd = 2,
       lty = 1,
       cex = 0.75)



ggplot(mc_df, aes(x = mu), title="MCMC PDF with Expected Value mu") +
  geom_density(adjust=10)+
  #geom_histogram(aes(y = ..density..), color = "white", bins = 15)+
  geom_vline(xintercept=mean(mc_df$mu), color='red')


## Q1 A. assess whether your numerical inference about the mode
converges
conv_test <- 2 * qt(0.975, length(mu_samples) - 1) * sd(mu_samples) /
sqrt(length(mu_samples))

# Now check if the CI is sufficiently small (<= 0.05)
if (conv_test <= 0.05) {
  print("CI is sufficiently small")
} else {
  print("CI is too large")
}


## Q1 B. define and use a positive control to assess the accuracy of
your implemented
#method for the inference in part a
# Positive Control

# run the grid based method to get the posterior with a combined
gaussian and uniform dist.
set.seed(930)
max=182
min=0
observed = 34
tanksd = 20
# grid-based method
gridvalues = seq(min, max, length.out=182)
# prior is uniform distribution from 0 to 182
uniform_prior <- dunif(gridvalues, min=min, max=max)
likelihood_values <- dnorm(gridvalues, mean=observed, sd=tanksd)
posterior <- likelihood_values*uniform_prior
posterior_normed <- posterior / sum(posterior)
# Plotting all together
plot(gridvalues, posterior_normed, col = "red",
     lwd=2,
     ylab= "PDF",
     xlab='Fuel (liters)',
     main="Positive Control Posterior")

# Add uniform distribution
lines(gridvalues, uniform_prior, col = "brown", lwd = 3)
segments(0, 0, 0, max(uniform_prior), col = "brown", lwd = 3)
segments(182, 0, 182, max(uniform_prior), col = "brown", lwd = 3)
legend("topright", legend = c("True Posterior", "Uniform Prior"),
       col = c("red", "brown"),
       lwd = 2,
       lty = 1,
       cex = 0.75)


# MCMC run for positive control
mu_initial = 50
tanksd = 20
n_iter = 30*10^5
candidate_sd=1
observed = 34
positive_control = mh(n_iter, mu_initial, candidate_sd)
```

```
PC_samples = positive_control$mu

expectedvalue_PC = mean(PC_samples)
print(expectedvalue_PC)

positive_control_df <- data.frame(mu = positive_control$mu)
plot(density(positive_control_df$mu, adjust=5), col = "blue",
     lwd=2,
     ylab= "PDF",
     xlab='Fuel (liters)',
     main="Positive Control Posterior compared to True PDF\n True
Value: 34",
     ylim=c(0, 0.021))
lines(gridvalues, posterior_normed, col = "red", lwd = 3)
abline(v=34, color="black", lwd=2)
legend("topright", legend = c("True Posterior", "MCMC Positive Control
Results", 'True Value'),
       col = c("red", "blue", "black"),
       lwd = 2,
       lty = 1,
       cex = 0.75)



# test convergence of positive control
conv_PC <- 2 * qt(0.975, length(PC_samples) - 1) * sd(PC_samples) /
sqrt(length(PC_samples))

# Now check if the CI is sufficiently small (<= 0.05)
if (conv_PC <= 0.05) {
  print("Confidence Interval is sufficiently small")
} else {
  print("Confidence Interval is too large")
}



## Testing a higher threshold for convergence ##
# MCMC run
mu_initial = 50
tanksd = 20
n_iter = 200000
candidate_sd=1
conv_test = mh(n_iter, mu_initial, candidate_sd)
convtest_samples = conv_test$mu

# test convergence of positive control
convtest_CI <- 2 * qt(0.975, length(convtest_samples) - 1) *
sd(convtest_samples) / sqrt(length(convtest_samples))
```

```
# Now check if the CI is sufficiently small (<= 0.05)
if (convtest_CI <= 0.25) {
  print("Confidence Interval is sufficiently small")
} else {
  print("Confidence Interval is too large")
}
```