Homework 4

Patrick Addona

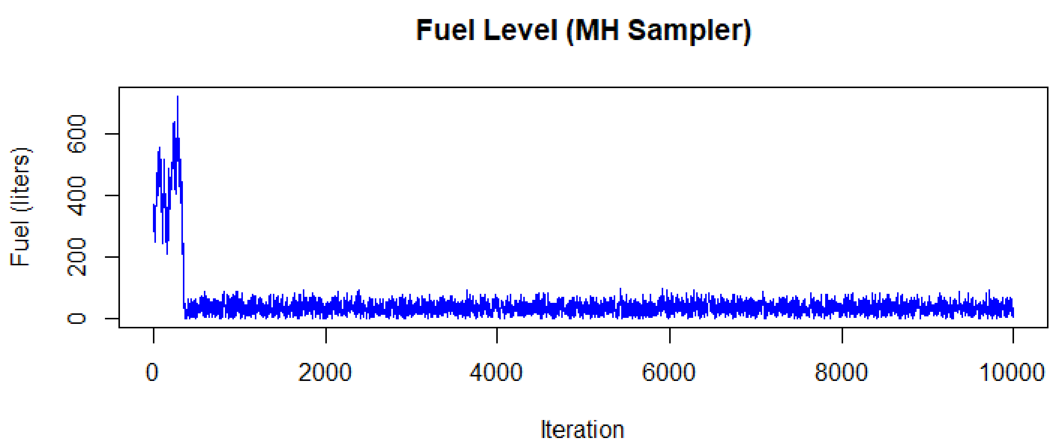
February 15, 2025

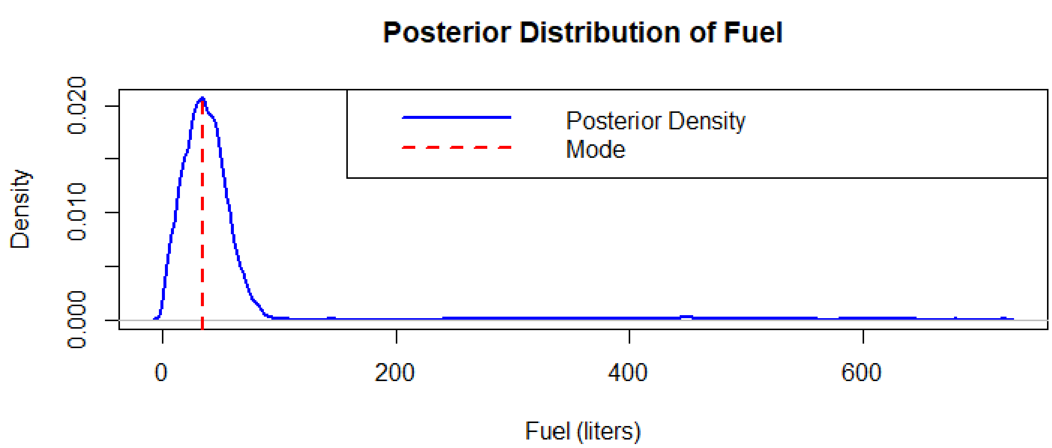
Question 1

Revisit the estimation problem about the fuel level (just this part) from the previous problem set using implementation of the Metropolis Hastings algorithm.
To this end, please: assess whether your numerical inference about the mode converges.

To implement the Metropolis Hastings algorithm in R, I found an introduction to the algorithm published by Ken Wood to be very helpful with the mathematical formulation. I chose to run the chain for 10,000 iterations (which we will see is more than enough), with a standard deviation of 30 to properly probe the posterior space. If I were to choose more iterations, the results would be redundant, and too few iterations may not guarantee convergence. Likewise, a standard deviation that is too high may never accept proposals, and one that is too low may never reject and take much longer to converge.

The results of the algorithm are shown in the following graphs:





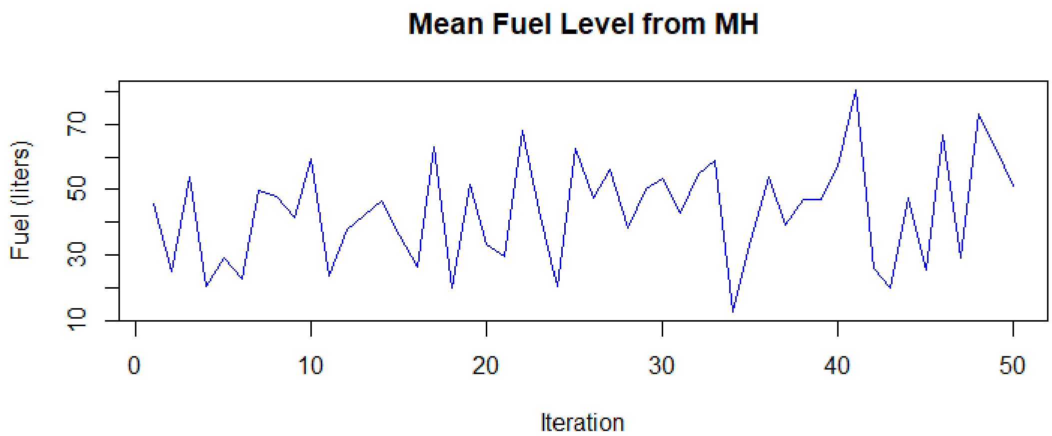
We see that after an initial burn-in period, the MH algorithm converges to the expected value of approximately 34, which is our observed fuel level. Likewise, the posterior distribution created has its peak at the mode of around 34, with the long right tail corresponding to the burn-in period. Were we to truncate this, it would appear cleaner. This analysis is reproducible through the use of a chosen seed.

Ken Wood's Metropolis-Hastings Algorithm page: https://rpubs.com/ROARMarketingConcepts/1063733

Question 2

Please define and use a positive control to assess the accuracy of your implemented method for the inference in Question 1.

To define a positive control, I chose to assume that we know the true fuel level is 40 liters. After generating a noisy observation "y_sim", I ran the MH algorithm 50 times and averaged the posterior means. The means themselves may be quite different, but the mean mean approaches the positive control value. This indicates that our method does indeed converge as expected, though noise is certainly present. Below the posterior means are shown:



As before, this analysis is reproducible with the use of designated seeds.

Question 3

Please compare the required number of runs needed for converged inferences in part b from the Metropolis Hastings algorithm with the Bayes Monte Carlo algorithm.

Because the Bayes Monte Carlo algorithm samples directly from the prior, we do not have to consider burn-in. We do not have to worry about a Markov Chain walking in the correct direction, which means the method is effective for low-dimensional problems. However, the sample size must be large enough so that an appropriate portion of them end up in the more likely regions. In my implementation from last assignment, I utilized 1,000,000 samples, though many less may have done the job.

In comparison, the MH algorithm needs to run for long enough to even accurately sample the posterior. I used 10,000 iterations, which ended up being plenty to account for burn-in. In general, MH would be more suited for a high-dimensional problem, as the samples required would not scale geometrically with dimension like in BMC.

Appendix

Problem 1:

```r
##  author: Patrick Addona
##  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)

set.seed(1)  # for reproducibility

# initial settings
n_iter <- 10000        # total iterations
sigma    <- 30            # proposal standard deviation
y      <- 34           # observed sensor reading
n_accept <- 0          # initialize to later calcuate acceptance prob

# prior: Uniform(0, 182)
prior <- function(x) {
  if (x >= 0 && x <= 182) {
    return(1/182)
  } else {
    return(0)
  }
}

# likelihood: N(y=34 | X, sd=20)
likelihood <- function(x) {
  dnorm(y, mean = x, sd = 20, log = FALSE)
}

# initialize the chain
X <- numeric(n_iter)
X[1] <- 300  # choose value outside prior range to demonstrate convergence

# begin Metropolis-Hastings
for (t in 1:(n_iter-1)) {
  # current state
  x_curr <- X[t]

  # propose x_star ~ N(x_curr, sigma^2)
  x_star <- rnorm(1, mean = x_curr, sd = sigma)

  # compute acceptance ratio
  p_star <- prior(x_star) * likelihood(x_star)
  p_curr <- prior(x_curr) * likelihood(x_curr)

  # ratio, set alpha to 1 if p_curr = 0 to avoid division by 0 and encourage movement towards high-density regions
  # alpha simplifies to p_star / p_curr because the likelihood is Gaussian
  alpha <- min(1, p_star / p_curr)
  if (p_curr == 0) {
    alpha = 1
  }

  # accept/reject
  if (runif(1) < alpha) {
    X[t+1] <- x_star
    n_accept <- n_accept + 1
  } else {
    X[t+1] <- x_curr
  }
}

# remove burn-in
burn_in <- 0 #set to 0 to show convergence
X_post  <- X[(burn_in+1):n_iter]

accept_rate <- n_accept / (n_iter - 1)
cat("Acceptance rate:", accept_rate, "\n")
```

```
# check the mode convergence
dens <- density(X_post)
mode_est <- dens$x[which.max(dens$y)]
cat("Posterior Mode estimate:", mode_est, "\n")

# plot the chain and the posterior density
par(mfrow=c(2,1))

# trace plot of chain values
plot(X_post, type = "l", col="blue",
     main="Fuel Level (MH Sampler)",
     xlab="Iteration", ylab="Fuel (liters)")

# posterior density estimate (after burn-in)
plot(dens, main="Posterior Distribution of Fuel",
     xlab="Fuel (liters)", col="blue", lwd=2)
abline(v=mode_est, col="red", lty=2, lwd=2)

legend("topright", legend=c("Posterior Density","Mode"),
       col=c("blue","red"), lty=c(1,2), lwd=2)
```

Problem 2:

```
##  author: Patrick Addona
##  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)

set.seed(1)  # for reproducibility

true_fuel <- 40
n_reps <- 50
# generate a single noisy sensor reading from N(50, 20^2)
y_obs <- rnorm(1, mean = true_fuel, sd = 20)
n_iter <- 10000
sigma <- 5
n_accept <- 0          # initialize to later calcuate acceptance prob
posterior_means <- numeric(n_reps)

# prior: Uniform(0, 182)
prior <- function(x) {
  if (x >= 0 && x <= 182) {
    return(1/182)
  } else {
    return(0)
  }
}

# likelihood: N(y=34 | X, sd=20)
likelihood <- function(x) {
  dnorm(y_obs, mean = x, sd = 20, log = FALSE)
}

#repeat MH algorithm to show positive control is accurately reached, on average
for (rep in 1:n_reps){
  y_obs <- rnorm(1, mean = true_fuel, sd = 20)
  X <- numeric(n_iter)
  X[1] <- 30

  # begin Metropolis-Hastings
  for (t in 1:(n_iter-1)) {
    # current state
    x_curr <- X[t]

    # propose x_star ~ N(x_curr, sigma^2)
    x_star <- rnorm(1, mean = x_curr, sd = sigma)

    # compute acceptance ratio
    p_star <- prior(x_star) * likelihood(x_star)
    p_curr <- prior(x_curr) * likelihood(x_curr)

```

```r
    # ratio, set alpha to 1 if p_curr = 0 to avoid division by 0 and encourage movement towards high-density regions
    # alpha simplifies to p_star / p_curr because the likelihood is Gaussian
    alpha <- min(1, p_star / p_curr)
    if (p_curr == 0) {
      alpha = 1
    }

    # accept/reject
    if (runif(1) < alpha) {
      X[t+1] <- x_star
      n_accept <- n_accept + 1
    } else {
      X[t+1] <- x_curr
    }
  }

  # remove burn-in
  burn_in <- 0 #set to 0 to show convergence
  X_post  <- X[(burn_in+1):n_iter]

  accept_rate <- n_accept / (n_iter - 1)

  posterior_means[rep] <- mean(X_post)
}

cat("Mean of posterior means =", mean(posterior_means), "\n")

# trace plot of chain values
plot(posterior_means, type = "l", col="blue",
     main="Mean Fuel Level from MH",
     xlab="Iteration", ylab="Fuel (liters)")
```