**ENGS 107: Problem Set #4: Markov Chain Monte Carlo**

**PROBLEM SET OVERVIEW:**

***1. Analyses***

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

***To this end, please:***

> ***A. Assess whether your numerical inference about the mode converges***
>
> ***B. Define and use a positive control to assess the accuracy of your implemented method for the inference in part A***
>
> ***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***

---

**TASK:**

Extending upon the tasks outlined in problem set #3, the **initial task** of this problem set was to revisit the estimation problem about the fuel level in the airplane tank, (and only this part), from problem set #3 using an implementation of the Metropolis Hasting algorithm. More specifically, this task asked the student to determine the Bayesian update from the likelihood function and their previously defined proper prior using the Metropolis Hasting algorithm (which is a Markov Chain Monte Carlo method) with a defined convergence and stopping criteria across multiple fixed seeds. The student should have calculated the posterior estimates of the 1) expected value of available fuel (the mean), 2) the most likely value of available fuel (the mode), and 3) the probability of negative fuel in the tank from the Metropolis-Hasting algorithm. To visualize the results, the student should have plotted their posterior distributions from the Metropolis-Hasting algorithm for each of their seeds with the likelihood function and the proper prior.

Following this initial task, **Task A** of this problem set required the student to assess whether their numerical inference about the mode converges. To do this, the student should have 1) calculated the variance of the mode over the course of the Markov Chain Monte Carlo method across all of their seeds and 2) plotted the variance of the mode across the seeds (as a single curve) over the number of iterations (to observe how the variance of the mode declines as the number of iterations increase) and 3) concluded if their inference about the mode converged for each seed.

Building upon these tasks, **Task B** of this problem set required the student to define and use a positive control to assess the accuracy of their implemented method for convergence in Task A. For this task, the student should have used the fuel sensor reading of 34 liters as their positive control, as this was a known value. They also should have performed an accuracy assessment to

determine if their estimated mode for each of their seeds from the Metropolis-Hasting algorithm was within their defined accuracy threshold and plotted the results. In addition, the student should have also plotted their positive control (with a gaussian distribution and a specified standard deviation range) with their posteriors from the Metropolis-Hasting algorithm for each of their seeds to 1) visualize the relationship between the positive control and their posteriors and 2) as an additional approach for assessing the accuracy of their implemented inference method.

As the final task of this problem set, **Task C** required the student to compare the required number of runs needed for converged inferences from the Metropolis-Hasting algorithm with the Bayes Monte Carlo method (from problem set #3). For this task, the student should have determined the Bayesian update from the likelihood function and their proper prior using a Bayes Monte Carlo method with the same convergence and stopping criteria as the Metropolis-Hasting algorithm. They also should have plotted the posteriors from the Bayes Monte Carlo method with the likelihood and the proper prior for each of their seeds and printed the estimates for the 1) expected value of available fuel (the mean), 2) the most likely value of available fuel (the mode), and 3) the probability of negative fuel in the tank along with the convergence metrics for each of the seeds. As a final visualization, the student should have also produced a plot of the posteriors from the Bayes Monte Carlo method overlaid on the posteriors from the Metropolis-Hasting algorithm for all of their seeds along with the likelihood function and the proper prior. To complete the analysis, the student also should have produced a table comparing the number of iterations required for convergence between the Metropolis-Hasting algorithm and Bayes Monte Carlo method for each of their seeds and provided an explanation for the observed differences.

---

## APPROACH:

### *Overarching Approach*:

To accomplish the tasks described above, my **overarching approach** was to produce a cohesive code script in R that systematically addressed each of the tasks. I decided to visually divide my code script into distinct sections for each task and I divided each task into a series of sub-tasks that are delineated by smaller horizontal boundaries in the code script. I also included detailed notes within the code script to articulate the purpose of each element of the code. Detailed below is an overview of my approach for accomplishing each of the tasks in this problem set.

### *Approach for Initial Task*:

***The initial task was to revisit the estimation problem about the fuel level (just this part) from the previous problem set using an implementation of the Metropolis-Hasting algorithm.***

The **first phase of my approach** was focused on establishing the necessary environment for using a Metropolis-Hasting algorithm to determine the Bayesian update from the likelihood function and prior. To do this, I: 1) restated the problem statement in the header of the code notes, 2) specified the inputs for the total fuel tank capacity, the fuel sensor reading, and the fuel sensor standard deviation, 3) set three random, fixed seeds (i.e., 3, 5, and 9) to facilitate

controlled randomness and reproducibility of the results, and 4) created, named and opened a PDF file to plot and print all of the necessary results for each of the tasks.

The **second phase of my approach** was to produce a plot of the likelihood function for each of my three seeds (i.e., 3, 5, and 9). To do this, I: 1) defined the colors and line widths for each of the three seeds for the subsequent plot of the likelihood function, 2) initialized separate lists to store the estimated values for each seed and the calculated values for the plot of the likelihood function, 3), set up a for loop to iterate over each of the three seeds to calculate the probability density function (PDF) for usable fuel in the tank (i.e., the likelihood function) using the dnorm function. Within this for loop, I first specified the fuel level range from -60 to 182 liters (to ensure that the left tail of the distribution was not truncated in the same manner as problem set #3) and subsequently calculated the expected value of available fuel (the mean), the most likely value of available fuel (the mode), and the probability of negative fuel in the tank for each seed. To visualize these results, I produced a plot of the PDF for the usable fuel in the tank (the likelihood function) for each of the three seeds and used vertical lines to denote the mean, the mode, and ± 1 standard deviation from the fuel sensor reading for reference. I also printed the calculated estimates from the likelihood function for each of the seeds in the PDF file.

The **third phase of my approach** was to define my proper prior (in the same manner as for problem set #3). To define my prior, I first set the lower bound of the airplane fuel tank capacity to 0 liters to reflect my prior knowledge that fuel levels cannot be negative. Since probability density functions must integrate to 1, I next determined the prior height over my specified range of fuel levels from 0 liters to 182 liters. This ensured that the total probability summed to 1, making it a valid probability distribution. I next used an ifelse statement to define the prior probability distribution, such that fuel levels inside of the range of 0 liters to 182 liters had a constant probability density and that fuel levels outside of the range (i.e., negative fuel levels) were a assigned a probability value of zero. I next used the seq function to create a sequence of fuel values for plotting the prior and subsequently used the rep function to calculate the prior probability values corresponding to each fuel level. To visualize my proper prior, I plotted my proper prior along with the likelihood function for each of the three seeds to highlight the realistic bound (i.e., lower and upper limits) for the airplane fuel tank capacity introduced through the prior. To this plot I also added vertical lines to denote the mean, the mode, and ± 1 standard deviation from the fuel sensor reading for reference.

With this initial groundwork laid in my code script, the **fourth phase of my approach** was to perform the likelihood estimation, define the prior probability, and calculate the posterior probability before implementing the Metropolis-Hasting algorithm. To do this, I first used the dnorm function to calculate the likelihood of observing a certain fuel level (using a normal distribution centered at the fuel sensor reading). If the fuel level was negative, it was assigned a probability of zero. Second, I specified the prior probability assuming a uniform distribution such that all of the fuel levels within the realistic range of 0 liters to 182 liters were equally likely. If the fuel level was outside of this range, it was assigned a probability of zero. Third, I calculated the posterior probability (unnormalized) by multiplying the likelihood function with the prior. If the fuel level was negative, it was assigned a probability of zero. Fourth, I created a compute_mode variable to find the most frequently occurring fuel level from the sampled data. I

rounded all samples to 2 decimal places to ensure that minor variations didn't affect the mode calculation and returned the most common fuel level (mode).

The **fifth phase of my approach** was to define the Metropolis-Hasting algorithm (a Markov Chain Monte Carlo method) to generate random samples from the posterior distribution. First, I established my convergence and stopping criteria. To do this, I specified the mode tolerance as 0.5 (0.5 L), the check interval as 1000 to check for convergence of the mode every 1000 iterations, and specified the stable checks as 30 to require 30 stable checks to be passed in which the mode was below the tolerance threshold to deem convergence. Second, I specified the burn in period as 30% to discard the first 30% of the samples to reduce the effect of the initial samples on the inference. Third, I set the random seed (from my list of three fixed seeds) to enable reproducibility of the results. Fourth, I initialized a numeric vector to store the samples up to the maximum number of iterations and ensured that the initial value was non-negative. Fifth, I created a mode_history variable to store the mode values to track stability of the mode. Sixth, I initialized a counter to track the number of consecutive stable checks to assess convergence.

The **sixth phase of my approach** was to run a for loop to perform the Metropolis-Hasting algorithm (MCMC method) from iteration 2 up to the maximum number of iterations (1 million) to 1) propose new candidate samples, 2) calculate the acceptance ratio, and 3) accept or reject the proposed sample. To do this, I first used the rnorm function to propose a new candidate sample using a normal distribution centered at the previous sample. Second, I used an if else statement to ensure that the proposed sample was not negative. If the sample was negative, it was immediately rejected. If the proposed sample was not rejected, the acceptance ratio (the Metropolis-Hasting acceptance criterion) was calculated in which the posterior probability of the proposed fuel level was compared to the posterior probability of the current fuel level. Third, the runif() function was used to draw a random number from the uniform distribution. If the random number was smaller than the value from the calculated acceptance ratio, then the proposed (new) fuel level was accepted, otherwise the previous fuel level value was retained.

The **seventh phase of my approach** was to perform convergence checking after the initial burn in period (the first 30% of samples) to assess if the mode had converged. To do this, I first calculated the mode of the sampled distribution. Second, I updated the mode history by removing the oldest value and adding the newest value. Third, I used an if else statement to check if the mode had remained stable over the most recent set of 1000 iterations. If the mode remained constant or varied within the specified mode tolerance of 0.5 liters over the last 1000 iterations, than the stable checks value was increased by 1, however, if the mode fluctuated outside of the mode tolerance, than the stable checks count was returned to a value of 0. Fourth, I used an if statement to stop or terminate the number of iterations early if the mode had stabilized within the tolerance threshold of 0.5 liters for 30 consecutive stable checks. Fifth, if convergence occurred, I used the return function to return the sampled fuel levels and the iteration number at which convergence was detected. If no early stopping converged prior to reaching the maximum number of iterations (1 million), then the full sample set was returned (excluding burn in period).

The **eighth phase of my approach** was to run the Metropolis Hasting algorithm for each of my three fixed seeds. To do this, I first specified the maximum number of iterations for the Metropolis Hasting algorithm as 1 million. Second, I defined an initial value based on the fuel

sensor reading of 34 liters and ensured that this value was not negative. Third, I defined the standard deviation of the proposal distribution as 10 to control the step size when proposing new values within the Metropolis-Hastings algorithm. Fourth, I initialized two separate lists to store the generated samples from the Metropolis-Hasting algorithm for each seed and to store the number of iterations at which convergence was achieved for each seed. Fifth, I used a for loop to perform the Metropolis Hasting algorithm for each of the three seeds. Sixth, I calculated the posterior distribution using the density function from the sampled fuel levels.

As the **ninth and final phase of my approach for the initial task** of implementing a Metropolis Hasting algorithm to determine the Bayesian update from the likelihood function and posterior for each of my three seeds, I produced a plot of the posteriors from the Metropolis Hasting algorithm with the likelihood function and prior for each of my three seeds. I used separate for loops to add the likelihood functions and posteriors for each of the three seeds to the plot. I also included vertical lines to denote the expected value of available fuel, the most likely value of available fuel, and the standard deviation range for the fuel sensor error for reference. I subsequently printed the posterior estimates for the 1) expected value of available fuel (the mean), 2) the most likely value of available fuel (the mode), and 3) the probability of negative fuel in the tank for each of the three onto a blank page in the PDF file.

***Approach for Task A:***

***Task A was to assess whether the numerical inference about the mode converges***

To assess whether my numerical interference about the mode converges across different Markov Chain Monte Carlo (MCMC) simulations (1 chain for each seed) [following my implementation of the Metropolis-Hasting algorithm to determine the Bayesian update from the likelihood function and prior], the **first phase of my approach** was to perform iterative mode tracking across the MCMC chains. To do this, I first used the lapply function to iterate over the MCMC chains for each of my three fixed seeds (3, 5, 9) to track the evolution of the mode estimate across the iterations. Second, I defined a track interval as 1000 to record the mode after every 1000 iterations of the simulation. Third, I re-defined the burn-in period (the first 30% of samples) to ensure that only samples after the burn-in period were considered. The purpose of this was to ensure that the initial 30% of samples did not improperly bias the mode estimation. Fourth, I used a for loop to extract and record the mode at each increment of 1000 iterations (starting after the burn-in period) using the compute_mode function.

The **second phase of my approach** was to align the mode tracking sequences. To do this, I first used the min function to determine the minimum length of the tracked mode sequences for each of my three fixed seeds. Second, I used the lapply function to trim each mode tracking sequence to match the shortest sequence so that all of the chains were of the same length for subsequent calculations. Third, I combined the mode tracking data from the three different seeds into a matrix using the do.call function for analysis. Fourth, I calculated the variance of the mode across the three different seeds using the apply function.

The **third phase of my approach** was to produce a simple plot of the variance of the mode across the three seeds (as a single curve) over the number of iterations. I labeled the axes and included a legend to label the key element of the plot (the curve).

The **fourth and final phase of my approach** to accomplish Task A was to print the convergence assessment results for each of the three seeds onto a blank page in the PDF file. To do this, first the posterior estimates for each of the three seeds were calculated: 1) the expected value of available fuel (the mean), 2) the most likely value of available fuel (the mode), and 3) the probability of negative fuel in the tank. Second, the final variance of the mode value across the three seeds was calculated. Third, the final variance of the mode value across the three seeds was printed at the top of the page. Fourth, the mean, mode, convergence iteration, and convergence status were printed for each of the three seeds at the bottom of the PDF page.

***Approach for Task B:***

***Task B was to define and use a positive control to assess the accuracy of the implemented method for the inference in part A***

To accomplish Task B, the **first phase of my approach** was to define a positive control and accuracy threshold. To do this, I first defined my positive control as 34 liters of fuel, as this was the known fuel sensor reading. Second, I defined an accuracy threshold of ± 3 liters of fuel to enable the classification of each mode estimate as either accurate or inaccurate. Third, I initialized a list to store the accuracy assessment results for each of the three seeds.

The **second phase of my approach** was to perform the accuracy assessment by retrieving and evaluating the posterior estimates. To do this, I set up a for loop to iterate through each of the three seeds. First, I retrieved the inferred (estimated) mode from the Metropolis-Hasting algorithm. Second, I calculated the absolute deviation of the inferred (estimated) mode from the positive control value of 34 liters. Third, I checked if the deviation was within the acceptable accuracy threshold of ± 3 liters of fuel. Fourth, I stored the accuracy results (including the mode, the absolute deviation, and the accuracy status).

The **third phase of my approach** was to print the accuracy assessment results on a blank page in the PDF file. To do this, I printed the estimated mode from the Metropolis Hasting algorithm, the deviation of the estimated mode from the positive control, and the accuracy status of either accurate or inaccurate for each seed onto the blank page in the PDF file.

The **fourth phase of my approach** was to produce a plot of the accuracy assessment results to visualize the accuracy of each of the estimated modes relative to the positive control. To do this, I produced a bar plot in which each bar is a different seed. The y-axis of the plot was the deviation in liters. I also included a black dashed reference line for the accuracy threshold. Seeds with estimated mode values at or below the accuracy threshold (≤ 3 liters) were colored as green and seeds with estimated mode values above the accuracy threshold (> 3 liters) were colored red. A legend was also added to the plot to label the key elements of the plot.

The **fifth phase of my approach** was to produce a plot of the positive control overlaid with the posteriors from the Metropolis Hasting algorithm for each of the three seeds to visualize the relationship between the positive control and the posterior distributions. To do this, I first used the seq function to create a sequence of fuel values for the positive control ranging from -60 to 200 in increments of 0.1. Second, I used the dnorm function to calculate the probability density function (PDF) for the positive control. Third, I used the max function to scale the positive control PDF to match the scale of the posteriors from the Metropolis-Hasting algorithm. Fourth, I plotted the positive control (centered at 34 liters with a standard deviation of ± 3 liters following a gaussian normal distribution) with the posteriors from each of the three seeds from the Metropolis Hasting algorithm. To this plot, I also added vertical dashed lines for each of the three estimated mode values color-coded with the color of the posterior distribution for each seed. I also added a legend to label all of the key elements present in the plot.

***Approach for Task C:***

***Task C was to compare the required number of runs needed for converged inferences in part B from the Metropolis Hasting algorithm with the Bayes Monte Carlo algorithm.***

To accomplish Task C, the **first phase of my approach** was to write the code script to perform the Bayesian update from the likelihood function and prior using a Bayes Monte Carlo method for each of my three seeds. To do this, I first defined the Bayes Monte Carlo method using function. For the Bayes Monte Carlo method, I set as many of the model parameters pertaining to convergence and stopping criteria as possible to be the same as in the Metropolis Hasting algorithm for consistency and to enable a valid comparison. For instance, I set the maximum number of iterations as 1 million, the mode tolerance as 0.5, the check interval to 1000 to check for convergence of the mode every 1000 iterations of the simulation, and I set the number of required stable check to 30. Second, I set the seed (from my pre-defined list of three fixed seeds) for reproducibility of the results. Third, I used the runif function to generate up to 1 million prior samples from the uniform prior distribution within the fuel tank capacity range set by the prior (0 to 182 liters). Fourth, I used the sapply function to calculate the unnormalized posterior weight for each sample. This step determines how much each sample contributes to the final posterior distribution. Fifth, I normalized the weights to sum to a value of 1 to ensure that the posterior distribution was a valid posterior distribution. Sixth, I used the sample function to perform importance sampling with replacement. For this step, the method resamples from the samples using the computed weights to approximate the posterior, which helps to concentrate the samples in regions that exhibit a high posterior probability. Seventh, I used the mode_history variable to store the mode values to track the stability of the mode and eighth I initialized a counter to track the consecutive number of stable checks to assess convergence of the mode.

The **second phase of my approach** was to use a for loop to monitor the convergence of the mode after every 1000 iterations of the Bayes Monte Carlo simulation as specified by the check interval variable. Within this for loop, I first used the compute_mode function to calculate the mode of the fuel level distribution. Second, I updated the mode history to track the stability of the mode over the last 1000 iterations. Third, I used an if else statement within the for loop to determine if the mode history indicates convergence. For this step, if the mode values were the same or if the change in the mode was below the mode tolerance of 0.5 liters, than the number of

stable checks was increased by 1. However, if the change in the modes exceed the threshold of 0.5 liters, than the counter for the stable checks was returned to a value of 0. Fourth, an if statement was used to stop the iterations early if 30 consecutive stable checks were achieved, indicating convergence of the mode. Fifth, the return function was used to return all of the generated samples and the total number of iterations run if the for loop completes without detecting convergence.

With this initial groundwork laid for executing the Bayes Monte Carlo method, the **third phase of my approach** was to run the Bayes Monte Carlo method for each of the three seeds. To do this, I first initialized two separate lists to store the generated samples for each seed and to store the iteration at which convergence was achieved for each seed. Second, I used a for loop to run the Bayes Monte Carlo simulation for each of the three seeds. Within the for loop, I stored the generated sample values in a list indexed by seed, and I also stored the iteration at which convergence was reached. Third, outside of the for loop, I subsequently calculated the Bayes Monte Carlo posterior densities using the kernel density estimation (KDE). An if else statement was used to filter out any negative samples, as only positive samples were used. The purpose of the KDE was to smooth the distribution. If there were too few valid samples, the else portion of the if else statement was used to return NULL.

To visualize the results from the Bayes Monte Carlo method, the **fourth phase of my approach** was to plot the posterior distributions from the Bayes Monte Carlo method for each of the three seeds with the likelihood function and my proper prior to observe how the posteriors were offset slightly from the likelihood function as a result of the prior. Separate for loops were used to plot the likelihood functions and the posterior distributions from the Bayes Monte Carlo method for each of the three seeds. I also included vertical lines to mark the expected value of available fuel (the mean), the most likely value of available fuel (the mode), and the standard deviation of the fuel sensor error for reference. I also added a legend to label the key elements of the plot.

The **fifth phase of my approach** was to print the calculated posterior estimates and convergence metrics from the Bayes Monte Carlo method for each of the three seeds. To do this, I first opened a new page in the PDF file and created a title at the top of the page. Second, I calculated the posterior estimates for the mean, mode, and probability of negative fuel in the tank for each of the three seeds. Third, I used a for loop to iterate through each of the three seeds to print the calculated estimates (mean, mode, and probability of negative fuel) and the convergence metrics (the required number of iterations for convergence and the convergence status) on the PDF page.

As a final visualization, the **sixth phase of my approach** was to produce a plot of the posterior estimates from the Bayes Monte Carlo method overlaid on the posterior estimates from the Metropolis Hasting algorithm for each of the three seeds along with the likelihood function and proper prior. I used separate for loops to add the likelihood function, the posteriors for the Metropolis Hasting algorithm, and the posteriors for the Bayes Monte Carlo method to the plot. I also included vertical lines to mark the expected value of available fuel (the mean), the most likely value of available fuel (the mode), and the standard deviation of the fuel sensor error for reference. I also added a large legend to the plot to label all of the key elements of the plot.

To facilitate an easy visual comparison of the required number of iterations for convergence of the mode between the Metropolis Hasting algorithm and the Bayes Monte Carlo method, for the **seventh and final phase of my approach**, I wrote a final section of code script to print a table of the required number of iterations for convergence for both methods for each of the three seeds. To do this, I first loaded the gridExtra and grid libraries into my code script to enable the production of a well formatted table. Second, I stated the three seeds and specified the required iteration for convergence for each of the three seeds for both the Metropolis Hasting algorithm and the Bayes Monte Carlo method. Third, I created a data frame with properly formatted column names to produce the table with the required number of iterations for convergence. Fourth, I created and added a title to the top of the PDF page. Fifth, I printed the table onto the PDF page using the grid.table function. Finally, I used dev.off() to close the entire PDF file.

---

**ASSUMPTIONS:**

To accomplish each of the tasks for this problem set, many assumptions were made in the process of writing the code script. A detailed listing of these assumptions is presented below:

In a similar manner to problem set #3, in terms of the provided inputs, **one assumption** made was that the fuel sensor error follows a Gaussian (normal) distribution (with a standard deviation of 20 liters). This assumes that the measurement errors are symmetrically distributed, meaning that under- and overestimations are equally probable (including extreme underestimations or overestimations), which may not be entirely realistic under all possible real-world scenarios.

A **second assumption** made was that setting three random fixed seeds was a sufficiently large number of seeds to introduced controlled randomness into the analysis (as compared to 5 or 10 seeds or more), while also facilitating reproducibility of the results for each of the seeds. Setting different fixed seeds also assumes that different results will be acquired for each of the seeds.

In terms of plotting the likelihood function, (in a similar manner to problem set #3), a **third assumption** made was that setting the lower bound of the x-axis to -60 liters of fuel (a physically impossible value) was sufficient to prevent any truncation of the left tail of the distribution that would interfere with the calculations for the expected value of available fuel and the most likely value of available fuel from the likelihood function.

In terms of establishing the proper prior, a **fourth assumption** made was that using a prior with a uniform distribution assumed that all fuel levels within the specified, realistic range of 0 liters to 182 liters were equally likely. This assumes no prior knowledge pertaining to airplane fuel tank levels aside from the fact that it is physically impossible to have 1) negative fuel and 2) fuel levels greater than the tank capacity. It is worth noting that previously published data describing airplane fuel consumption patterns could have been used to develop a more informative and detailed prior.

Also in terms of establishing the proper prior, a **fifth assumption** made was that both negative fuel levels and fuel levels exceeding the fuel tank capacity are physically impossible (although

this is known to be a true). This assumption enables the assigning of any negative fuel levels or fuel levels above the airplane tank capacity of 182 liters a probability value of zero.

In terms of calculating the mode of the sample distribution within the Metropolis Hasting algorithm, a **sixth assumption** made was that rounding values to two decimal places for the compute_mode function was an appropriate binning approach for determining the most frequent sample value.

In terms of implementing the Metropolis Hasting algorithm and setting the convergence and stopping criteria, several different assumptions were made. A **seventh assumption** was that 1 million iterations was a sufficiently large enough number of iterations to function as a proper maximum upper limit or bound for the Metropolis Hasting algorithm. In other words, it assumed that convergence of the mode would occur prior to reaching 1 million iterations. It also assumes that if convergence has not been achieved before reaching 1 million iterations, the sampling process has failed.

An **eighth assumption** made was that setting the mode tolerance to 0.5 (i.e., 0.5 liters of fuel) was an appropriate value and also a small enough value to set as a threshold to determine convergence over a set number of stable checks.

A **ninth assumption** made was that setting the check interval to 1000 iterations was an appropriate number of iterations to use to periodically check if the change in the mode value from the previous check was below a threshold of 0.5 liters. This assumption was specifically made in the context of setting the maximum number of iterations to 1 million and in the context of setting the burn-in period to 30%.

A **tenth assumption** made was that setting the number of stable checks to 30 was a sufficiently large enough number of consecutive stable checks to deem that convergence of the mode had occurred. This assumed that no change in the mode outside of a threshold of 0.5 liters would occur after 30 consecutive stable checks of the change in the mode being under a value of 0.5.

An **eleventh assumption** made was that setting the burn-in period to 30% was an appropriate percentage of the samples to remove (discard) from the analysis to minimize the impact of the bias from these noisy initial values on the mode estimate calculations and on the calculations for the variance of the mode across the seeds. This also assumes that the first 30% of the samples do not represent the stationary distribution, and thus should be discarded.

Broadly in terms of the convergence and stopping criteria, a **twelfth assumption** made was that setting the mode tolerance, check interval, stable checks, burn-in period, and maximum iterations to their respective values collectively was an appropriate set of convergence and stopping criteria to accurate assess if the modes had converged for each of the seeds.

A **thirteenth assumption** made was that the setting proposal standard deviation (step size) to 10 was an appropriate value for effectively exploring the target distribution.

A **fourteenth assumption** made for this problem set was that examining the mode (the most likely value) was a reliable metric for assessing convergence in the context of this problem.

In terms of assessing whether the numerical inference about the mode converges, a **fifteenth assumption** made was that setting the tracking interval to 1000 was an appropriate and frequent enough value to set for recording the mode value. In other words, this assumes that in the context of the maximum number of iterations and the convergence and stopping criteria, setting the tracking interval to 1000 was a small enough iteration range for recording the mode a sufficient number of times.

Also in terms of assessing whether the numerical inference about the mode converges, a **sixteenth assumption** made was that through the process of trimming the chains to match the shortest chain for the subsequent variance calculation, the shortest chain was still a long enough chain to be representative for the original length of the other two chains.

Also in terms of assessing whether the numerical inference about the mode converges, a **seventeenth assumption** made was that a lower variance of the mode across the seeds was indicative of convergence across the chains. This assumption was made in the context of using the variance of the mode estimates across the different seeds to assess stability of the mode. Following this same line of thinking, an **eighteenth assumption** made was that plotting the variance of the mode across the seeds over the number of iterations was an appropriate visual approach for examining if convergence of the mode had occurred over the iterations.

A **nineteenth assumption** made was that setting the fuel sensor reading of 34 liters as the positive control was an accurate and appropriate selection as a positive control for assessing the performance of the inference method considering that this was a previously known and provided value in the problem statement from problem set #3.

Also in terms of the positive control, a **twentieth assumption** made was that setting the acceptable threshold for the accuracy assessment as ± 3 liters from the positive control value of 34 liters was an appropriate arbitrary threshold to set to determine if each of the estimated modes from each of the three seeds was accurate or not. In other words, this assumes that deviations within this range from the positive control are negligible and that the inference method is valid if the mode estimates from the Metropolis Hasting algorithm fall within this range.

In terms of plotting the accuracy assessment results, a **twenty-first assumption** made was that using a bar plot was an appropriate visual approach for visualizing if the mode estimates for each of the three seeds were above or below the arbitrarily set accuracy threshold of ± 3 liters of fuel from the positive control value of 34 liters of fuel.

In terms of plotting the positive control with the posteriors from the Metropolis Hasting algorithm, a **twenty-second assumption** made was that the positive control followed a normal Gaussian distribution centered at 34 liters of fuel with a standard deviation of ± 3 liters of fuel.

Also in terms of the accuracy assessment, a **twenty-third assumption** made was that assessing accuracy using a simple, binary decision of either accurate or inaccurate was an appropriate

approach for this problem compared a choice of considering confidence intervals to produce a more refined accuracy assessment for the implemented inference method.

In terms of the setup of the Bayes Monte Carlo method to determine the Bayesian update from the likelihood function and prior, several assumptions were made in a similar manner to the assumptions made in the process of establishing the convergence and stopping criteria for the Metropolis Hasting algorithm. For instance, a **twenty-fourth** assumption made was that setting the maximum number of iterations to 1 million was a sufficiently large enough number of iterations to function as a proper maximum upper limit for the Bayes Monte Carlo method. In other words, it assumed that convergence of the mode would occur prior to 1 million iterations.

Also in terms of the setup of the convergence and stopping criteria for the Bayes Monte Carlo method, a **twenty-fifth assumption** made was that setting the mode tolerance to 0.5 (0.5 liters) was an appropriate and small enough value to set as a threshold to determine convergence over a set number of stable checks. A **twenty-sixth assumption** made was that setting the tracking interval to 1000 was an appropriate and frequent enough value to set for recording the mode value. This assumed that in the context of the maximum number of iterations and the convergence and stopping criteria, setting the tracking interval to 1000 was a small enough number of iterations to record the mode a sufficient number of times. A **twenty-seventh assumption** made was that setting the number of stable checks to 30 was large enough number of consecutive stable checks to deem that convergence of the mode had occurred. This assumed that no change in the mode outside of a threshold of 0.5 liters would occur after 30 stable checks.

In terms of performing importance sampling with replacement within the Bayes Monte Carlo method, a **twenty-eighth** assumption arises from the weighting of samples based on their posterior probability and then performing resampling with replacement. This process assumes that importance sampling provides an accurate approximation of the posterior distribution.

In terms of calculating the Bayes Monte Carlo posterior densities, the density function was used with an adjustment factor of 1.5 to smooth the estimate. This made a **twenty-ninth assumption** in that the bandwidth adjustment of 1.5 provided a sufficiently smooth estimate.

A **thirtieth assumption** made was that producing a plot of the posteriors from the Metropolis Hasting algorithm overlaid with the posteriors from the Bayes Monte Carlo method for each of the three seeds was an appropriate visual approach for comparing these alternative approaches for determining the Bayesian update from the likelihood function and proper prior.

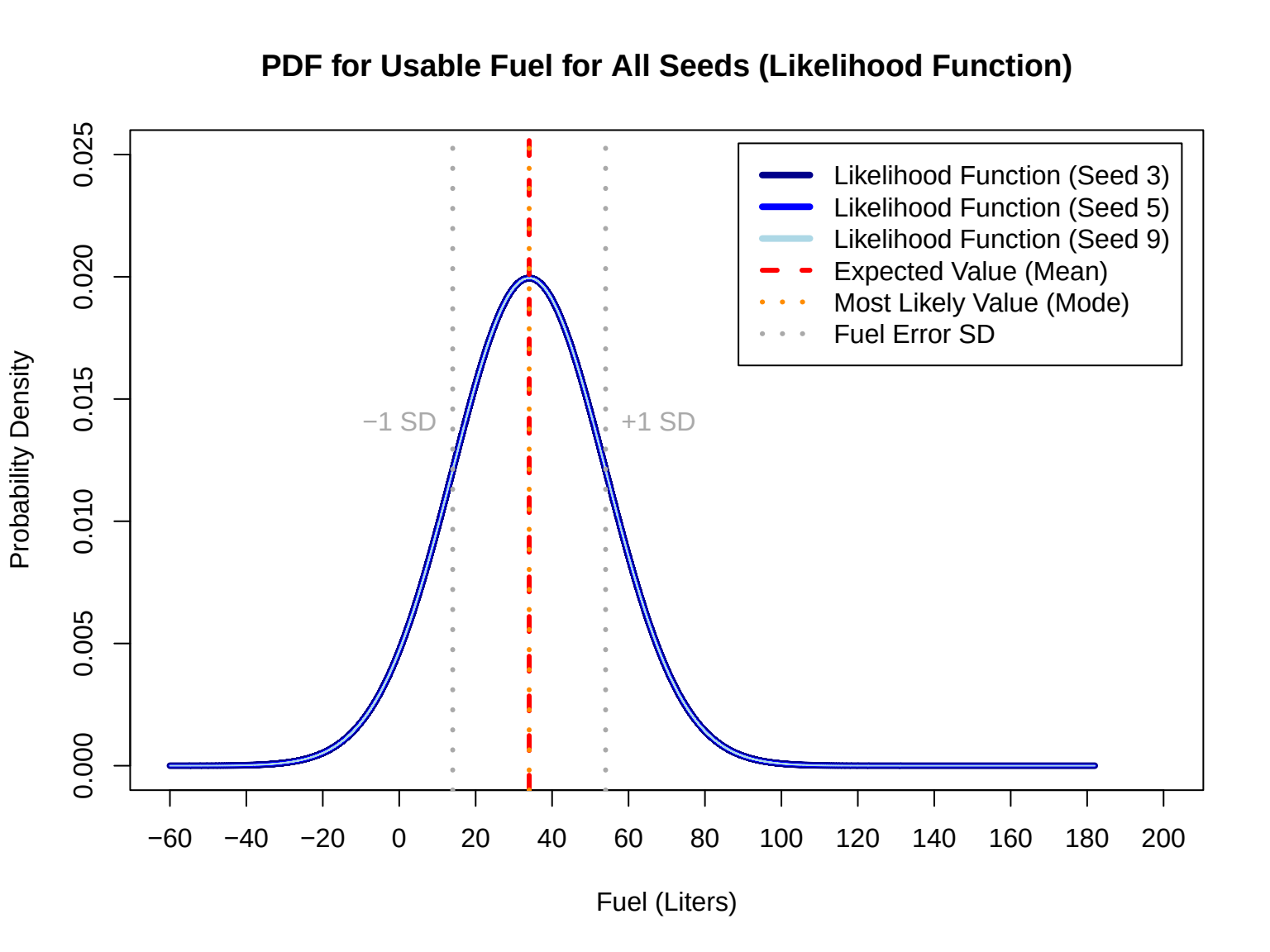
A **thirty-first assumption** made was that using a table was the best approach for comparison and visualizing the differences in the required number of iterations needed for convergence between the Metropolis Hasting algorithm and the Bayes Monte Carlo method.

A **thirty-second and final assumption** made was that the selected colors, fonts, character sizes, labels, legends etc. and formatting of these elements were the best selections and decisions for visualizing and printing the results from this analysis of each of the tasks in this problem set.

**RESULTS & FIGURES:**

# RESULTS AND FIGURES FOR INITIAL TASK

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

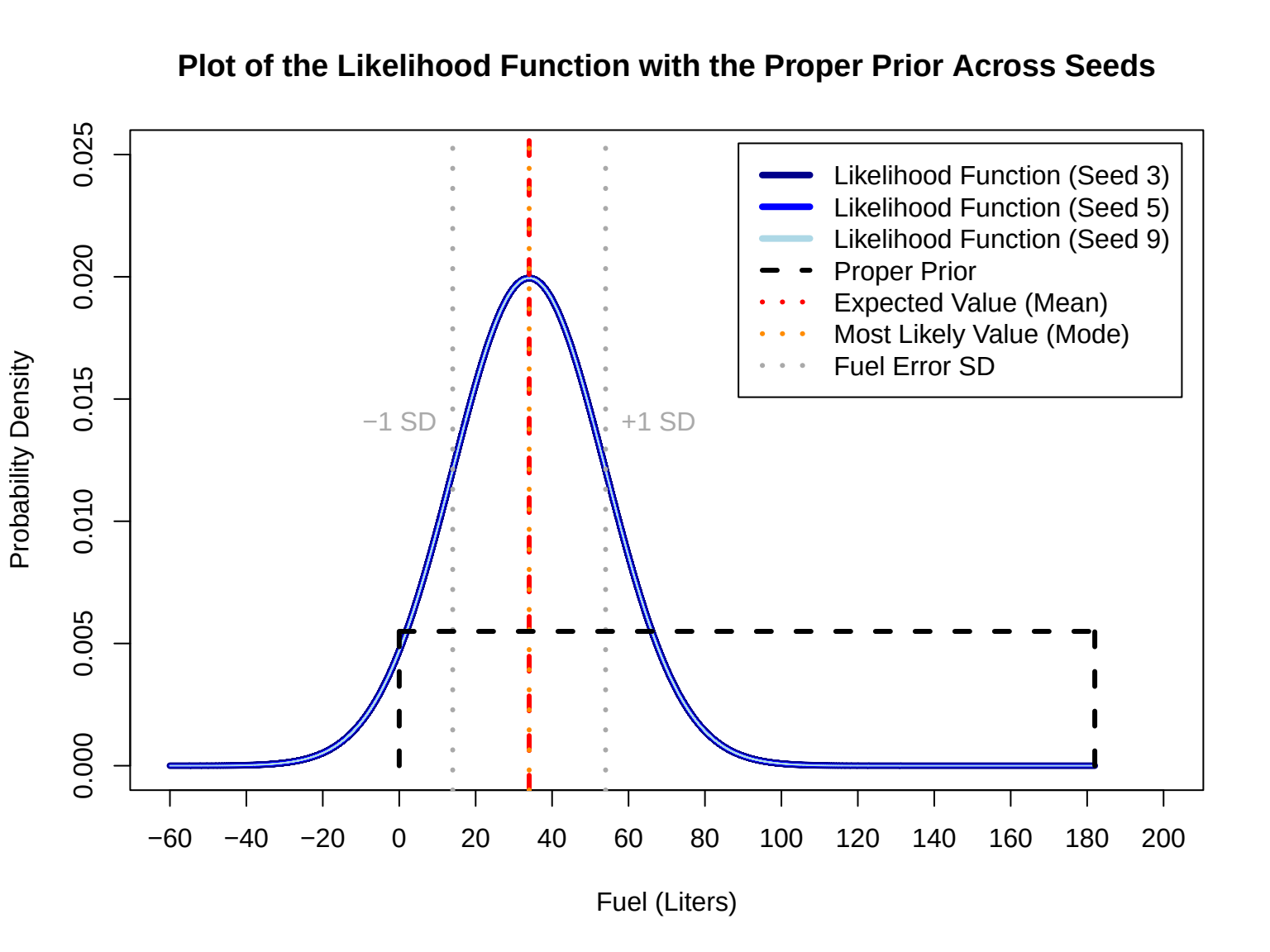

PDF for Usable Fuel for All Seeds (Likelihood Function)
Likelihood Function (Seed 3)
Likelihood Function (Seed 5)
Likelihood Function (Seed 9)
Expected Value (Mean)
Most Likely Value (Mode)
Fuel Error SD
−1 SD
+1 SD
Probability Density
0.000
0.005
0.010
0.015
0.020
0.025
−60
−40
−20
0
20
40
60
80
100
120
140
160
180
200
Fuel (Liters)

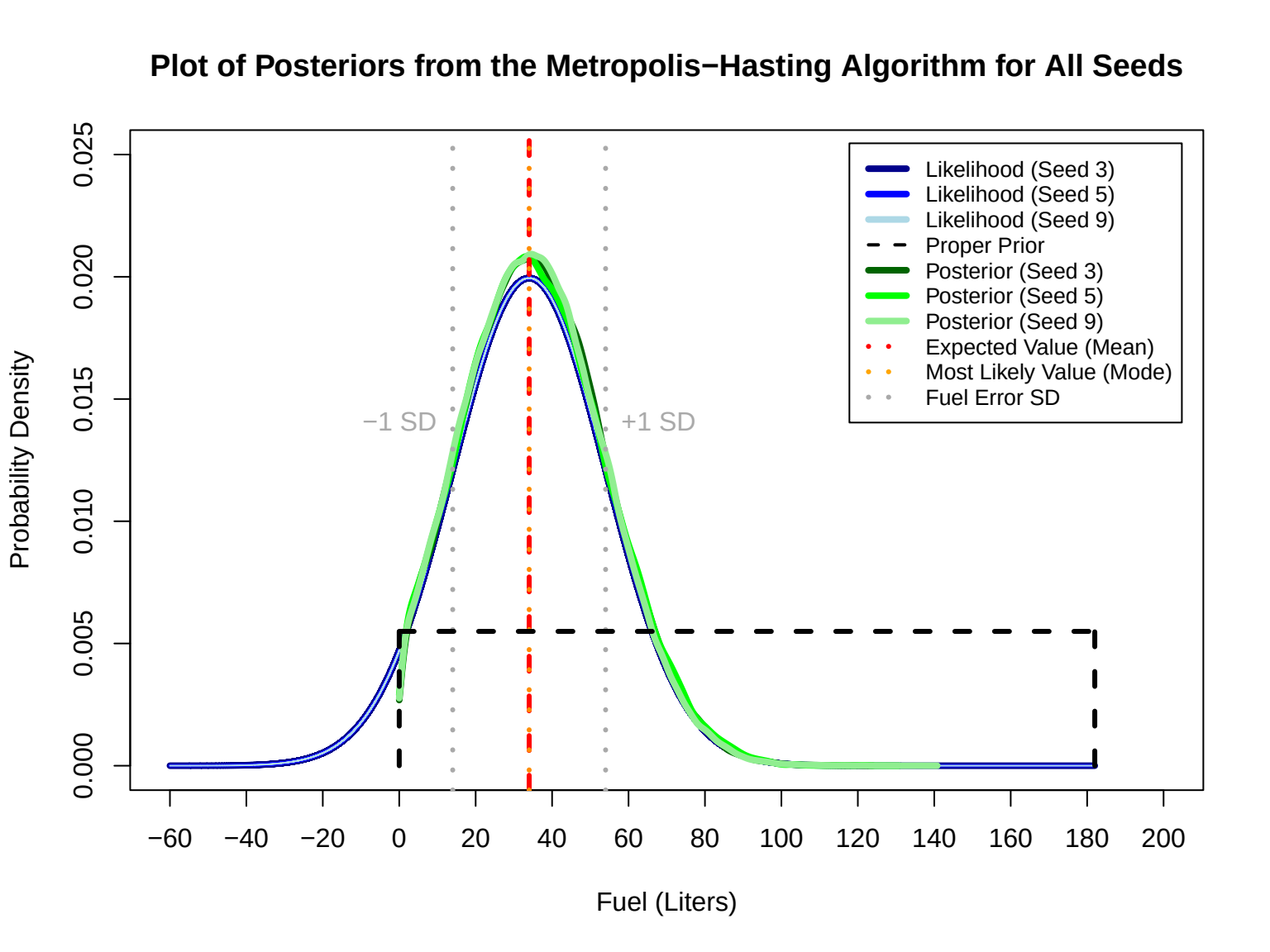
## Estimates from PDF of Usable Fuel for All Seeds (Likelihood Function)

**Seed 3**
Expected Value of Available Fuel (Mean): 34 liters
Most Likely Value of Available Fuel (Mode): 34 liters
Probability of Negative Fuel in the Tank: 4.5 %

**Seed 5**
Expected Value of Available Fuel (Mean): 34 liters
Most Likely Value of Available Fuel (Mode): 34 liters
Probability of Negative Fuel in the Tank: 4.5 %

**Seed 9**
Expected Value of Available Fuel (Mean): 34 liters
Most Likely Value of Available Fuel (Mode): 34 liters
Probability of Negative Fuel in the Tank: 4.5 %


Plot of the Likelihood Function with the Proper Prior Across Seeds
Likelihood Function (Seed 3)
Likelihood Function (Seed 5)
Likelihood Function (Seed 9)
Proper Prior
Expected Value (Mean)
Most Likely Value (Mode)
Fuel Error SD
−1 SD
+1 SD
Probability Density
0.000
0.005
0.010
0.015
0.020
0.025
−60
−40
−20
0
20
40
60
80
100
120
140
160
180
200
Fuel (Liters)


Plot of Posteriors from the Metropolis−Hasting Algorithm for All Seeds
Likelihood (Seed 3)
Likelihood (Seed 5)
Likelihood (Seed 9)
Proper Prior
Posterior (Seed 3)
Posterior (Seed 5)
Posterior (Seed 9)
Expected Value (Mean)
Most Likely Value (Mode)
Fuel Error SD
−1 SD
+1 SD
Probability Density
0.000
0.005
0.010
0.015
0.020
0.025
−60
−40
−20
0
20
40
60
80
100
120
140
160
180
200
Fuel (Liters)

# Posterior Estimates from the Metropolis−Hasting Algorithm for All Seeds

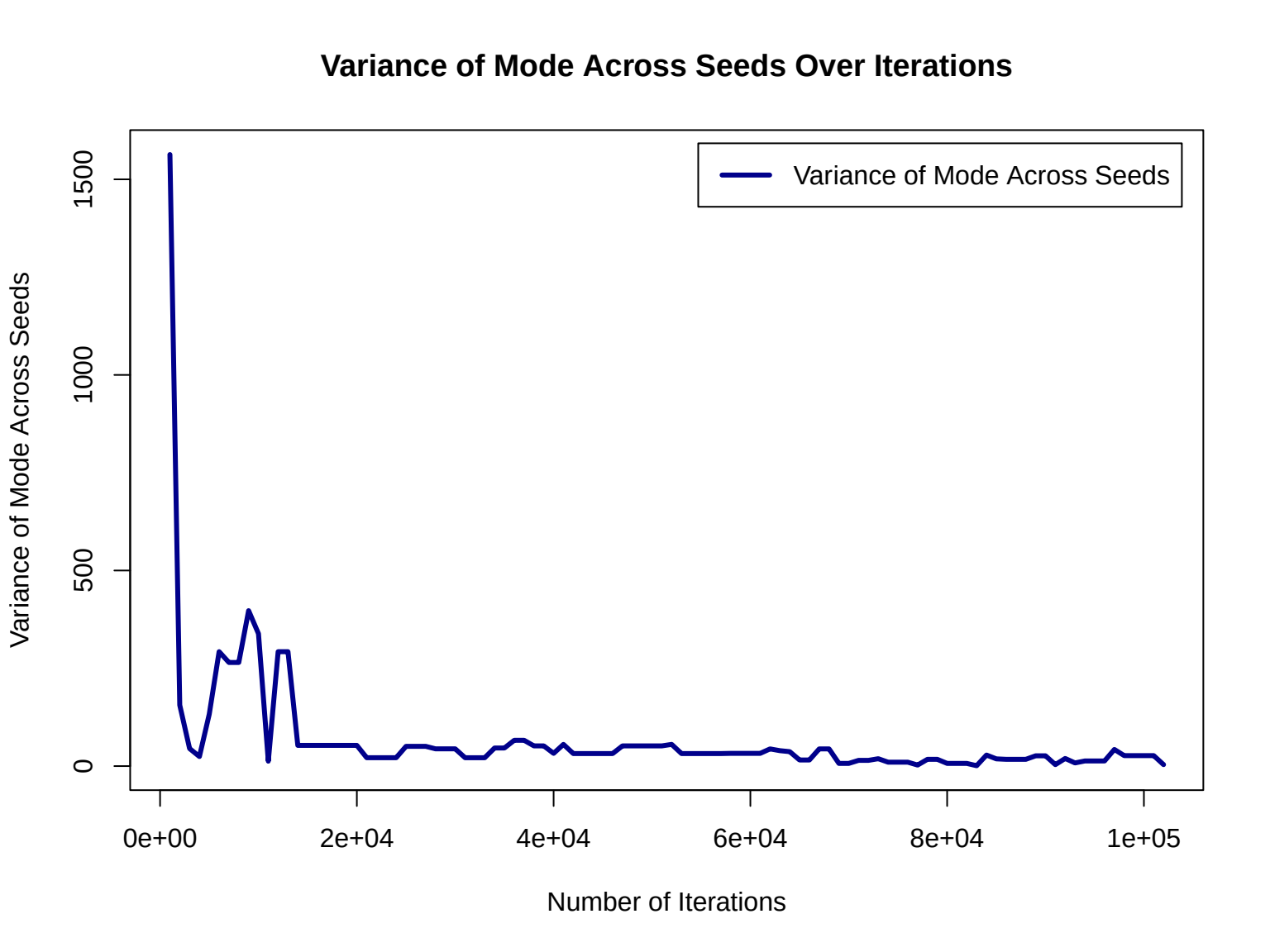
**Seed 3**
Expected Value of Available Fuel (Mean): 36.01 liters
Most Likely Value of Available Fuel (Mode): 30.79 liters
Probability of Negative Fuel in the Tank: 0 %

**Seed 5**
Expected Value of Available Fuel (Mean): 36.06 liters
Most Likely Value of Available Fuel (Mode): 33.41 liters
Probability of Negative Fuel in the Tank: 0 %

**Seed 9**
Expected Value of Available Fuel (Mean): 35.83 liters
Most Likely Value of Available Fuel (Mode): 32.13 liters
Probability of Negative Fuel in the Tank: 0 %

# RESULTS AND FIGURES FOR TASK A

# (Assessing whether the numerical inference about the mode converges)


Variance of Mode Across Seeds Over Iterations
Variance of Mode Across Seeds
Variance of Mode Across Seeds
0
500
1000
1500
0e+00
2e+04
4e+04
6e+04
8e+04
1e+05
Number of Iterations

# Assessment of Mode Convergence Across all Seeds

**Final Variance of Mode Across Seeds: 3.7**

**Seed 3**
Most Likely Value of Available Fuel (Mode): 30.79 liters
Convergence Iteration: 552000
Convergence Status: Converged Successfully

**Seed 5**
Most Likely Value of Available Fuel (Mode): 33.41 liters
Convergence Iteration: 446000
Convergence Status: Converged Successfully

**Seed 9**
Most Likely Value of Available Fuel (Mode): 32.13 liters
Convergence Iteration: 695000
Convergence Status: Converged Successfully

# RESULTS AND FIGURES FOR TASK B

# (Defining and using a positive control to assess the accuracy of the implemented method for the inference in part a)

# Accuracy Assessment of Inference Method Using Positive Control

**Seed 3**

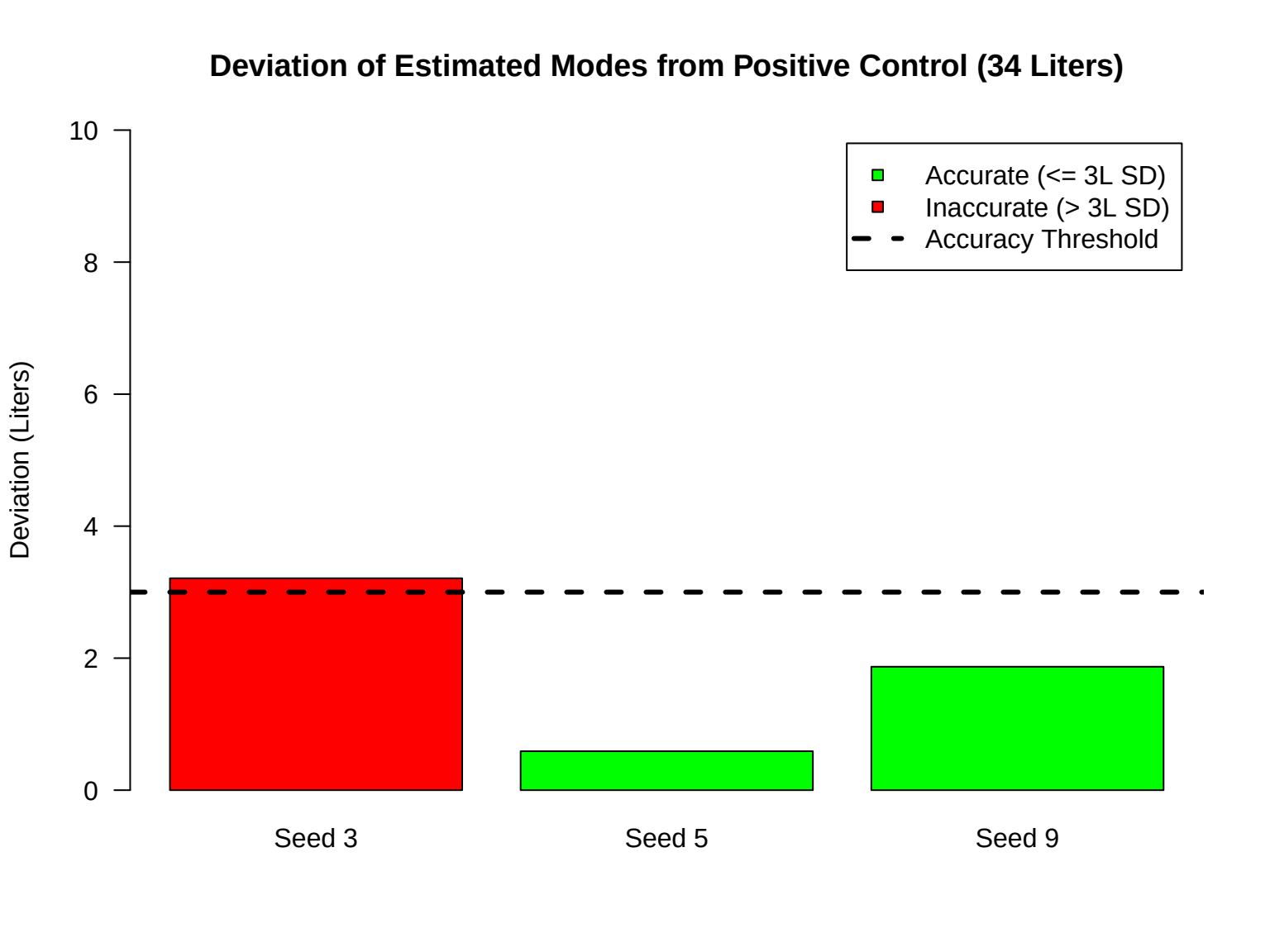
Estimated Mode: 30.79 liters
Deviation from Positive Control (34 L): 3.21 liters
Accuracy Status: Inaccurate

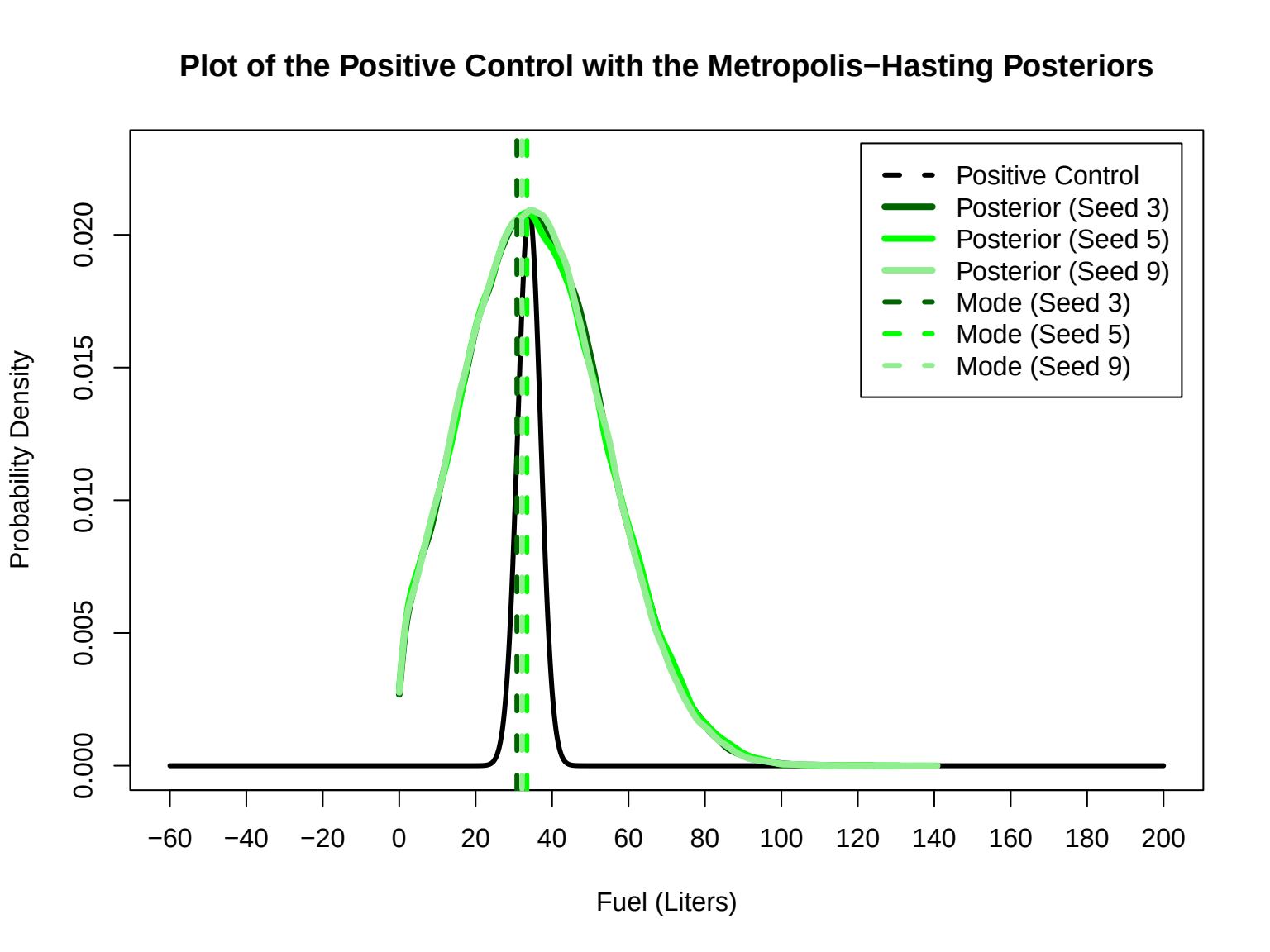
**Seed 5**

Estimated Mode: 33.41 liters
Deviation from Positive Control (34 L): 0.59 liters
Accuracy Status: Accurate

**Seed 9**

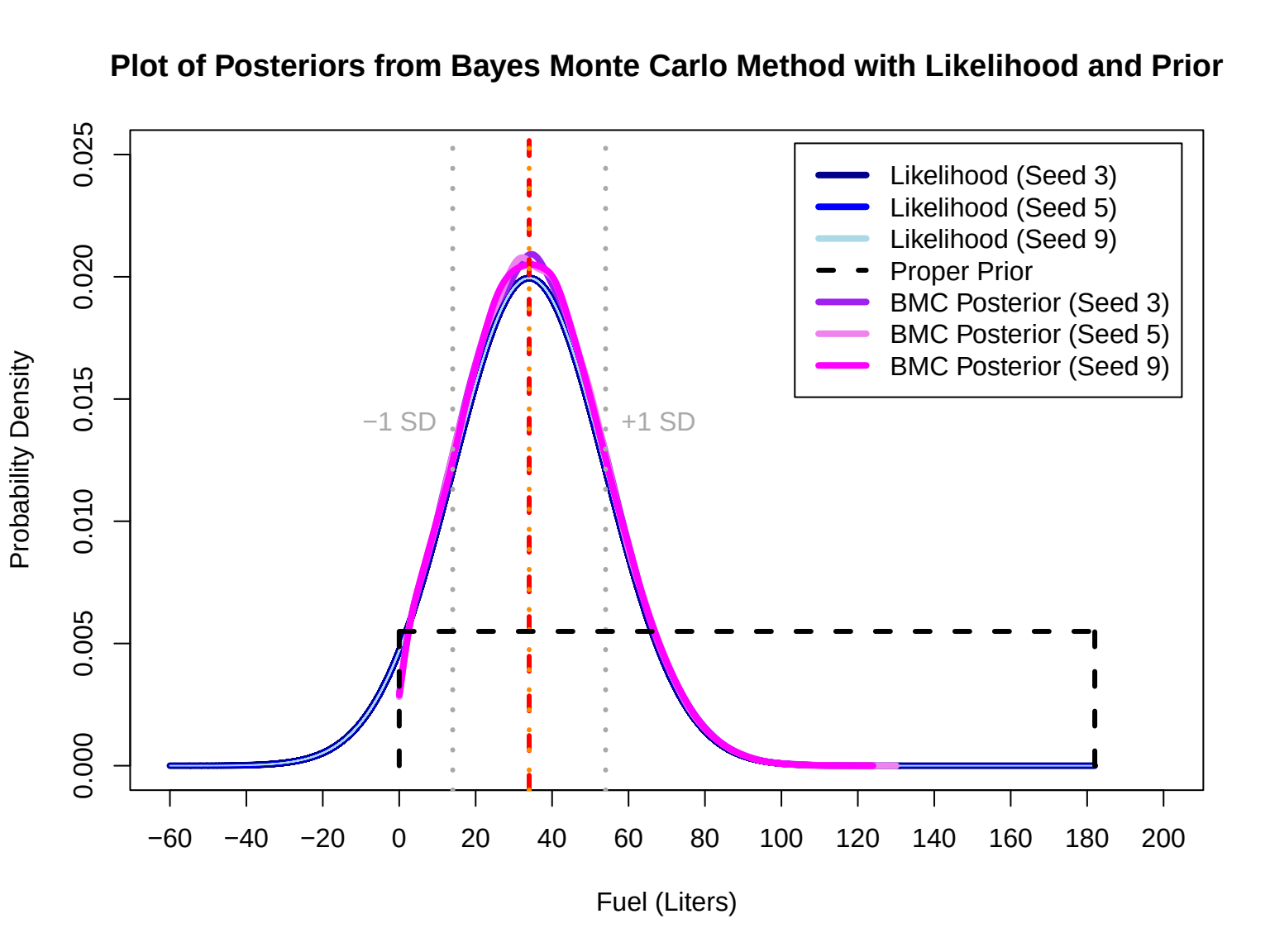
Estimated Mode: 32.13 liters
Deviation from Positive Control (34 L): 1.87 liters
Accuracy Status: Accurate


Deviation of Estimated Modes from Positive Control (34 Liters)
Deviation (Liters)
10
8
6
4
2
0
Accurate (<= 3L SD)
Inaccurate (> 3L SD)
Accuracy Threshold
Seed 3
Seed 5
Seed 9


Plot of the Positive Control with the Metropolis−Hasting Posteriors
Positive Control
Posterior (Seed 3)
Posterior (Seed 5)
Posterior (Seed 9)
Mode (Seed 3)
Mode (Seed 5)
Mode (Seed 9)
Probability Density
0.000
0.005
0.010
0.015
0.020
−60
−40
−20
0
20
40
60
80
100
120
140
160
180
200
Fuel (Liters)

# RESULTS AND FIGURES FOR TASK C

# (Comparing the required number of runs needed for converged inferences in part b from the Metropolis Hasting algorithm with the Bayes Monte Carlo algorithm)


Plot of Posteriors from Bayes Monte Carlo Method with Likelihood and Prior
Likelihood (Seed 3)
Likelihood (Seed 5)
Likelihood (Seed 9)
Proper Prior
BMC Posterior (Seed 3)
BMC Posterior (Seed 5)
BMC Posterior (Seed 9)
−1 SD
+1 SD
Probability Density
0.000
0.005
0.010
0.015
0.020
0.025
−60
−40
−20
0
20
40
60
80
100
120
140
160
180
200
Fuel (Liters)

## Posterior Estimates and Convergence Metrics from the Bayes Monte Carlo Method for All Seeds

**Seed 3**

Expected Value of Available Fuel (Mean): 35.98 liters
Most Likely Value of Available Fuel (Mode): 31.07 liters
Probability of Negative Fuel in the Tank: 0 %
Number of Iterations Required for Convergence: 202000
Convergence Status: Converged Successfully

**Seed 5**

Expected Value of Available Fuel (Mean): 35.97 liters
Most Likely Value of Available Fuel (Mode): 30.18 liters
Probability of Negative Fuel in the Tank: 0 %
Number of Iterations Required for Convergence: 279000
Convergence Status: Converged Successfully

**Seed 9**

Expected Value of Available Fuel (Mean): 35.98 liters
Most Likely Value of Available Fuel (Mode): 24.96 liters
Probability of Negative Fuel in the Tank: 0 %
Number of Iterations Required for Convergence: 148000
Convergence Status: Converged Successfully

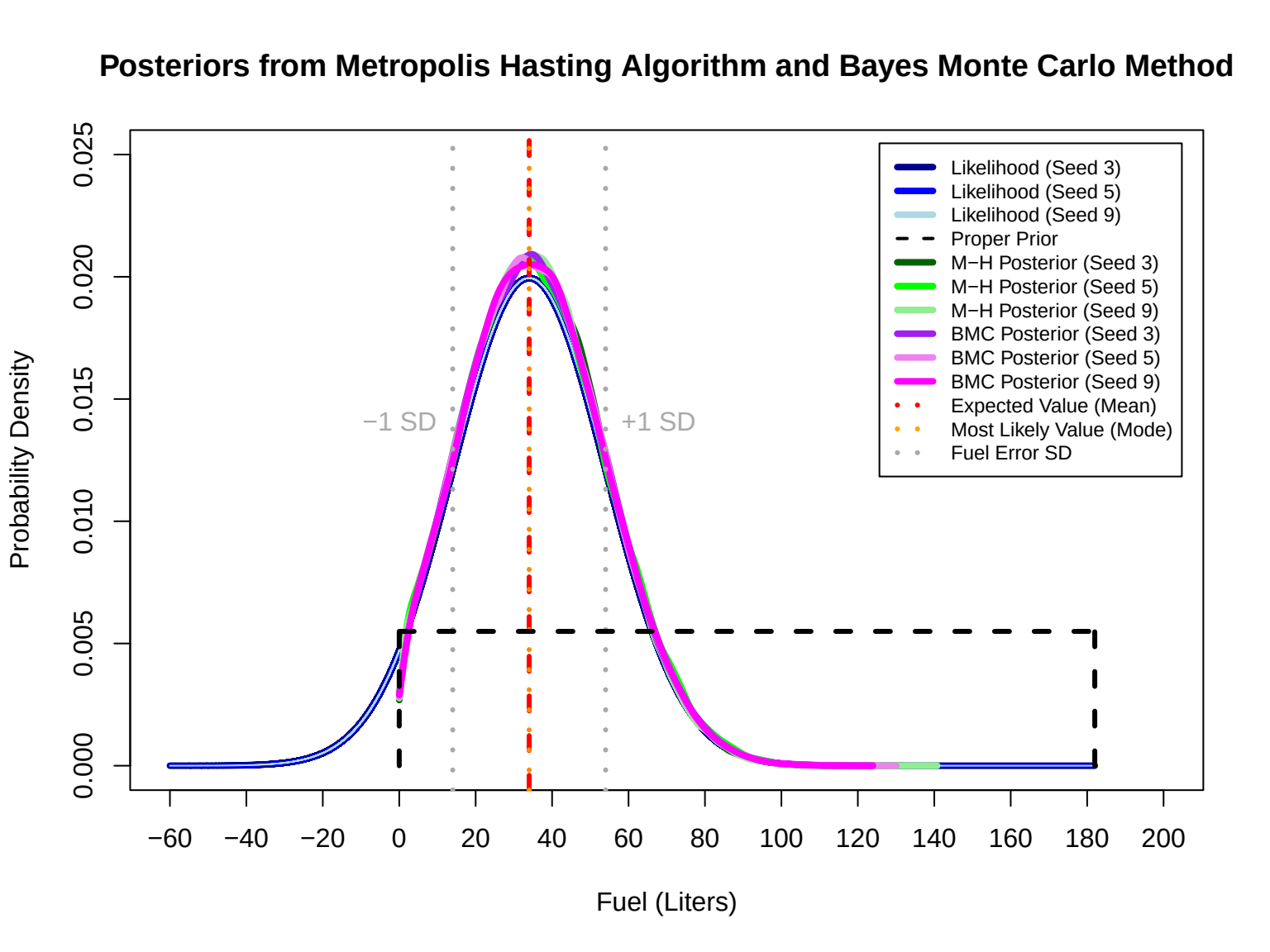

Posteriors from Metropolis Hasting Algorithm and Bayes Monte Carlo Method
Probability Density
0.000
0.005
0.010
0.015
0.020
0.025
−60
−40
−20
0
20
40
60
80
100
120
140
160
180
200
Fuel (Liters)
−1 SD
+1 SD
Likelihood (Seed 3)
Likelihood (Seed 5)
Likelihood (Seed 9)
Proper Prior
M−H Posterior (Seed 3)
M−H Posterior (Seed 5)
M−H Posterior (Seed 9)
BMC Posterior (Seed 3)
BMC Posterior (Seed 5)
BMC Posterior (Seed 9)
Expected Value (Mean)
Most Likely Value (Mode)
Fuel Error SD

**Table of Required Iterations for Convergence for Metropolis−Hastings and Bayes Monte Carlo**

| | Seed | M−H Required Iterations for Convergence | BMC Required Iterations for Convergence |
|---|---|---|---|
| *1* | 3 | 552000 | 202000 |
| *2* | 5 | 446000 | 279000 |
| *3* | 9 | 695000 | 148000 |

**Notes on the Results of Task C:**

Comparison of the required number of runs needed for converged inferences between the Metropolis Hasting algorithm and the Bayes Monte Carlo method for each of the three seeds revealed that convergence happened for each of the three seeds considerably sooner (about 2-4 times faster) in the Bayes Monte Carlo method compared to the Metropolis Hasting algorithm. The primary basis for this observation arises from the differences in sampling efficiency and exploration of the posterior distribution between the two methods.

To examine the results in a bit more detail, the Metropolis Hasting algorithm required between 552,000 – 695,000 iterations for convergence to occur across the three seeds. In contrast, the Bayes Monte Carlo Method only required 148,000 – 279,000 iterations for convergence to occur across the three seeds. This indicates that convergence occurred about 2-4 times faster in the Bayes Monte Carlo method compared to Metropolis Hasting algorithm. The primary basis for this arises from the Bayes Monte Carlo method using direct sampling from the posterior distribution and then performing importance sampling with replacement based on the posterior weights, which efficiently generates a posterior distribution in a fewer number of iterations. In contrast, the Metropolis Hasting algorithm is a Markov Chain Monte Carlo (MCMC) method that uses an iterative process in which each new sample is conditionally dependent on the previous sample. The Metropolis Hasting algorithm rejects many of the proposed samples based upon the calculated acceptance ratio, which can result in the algorithm revisiting regions of the distribution that it has already sampled, which is inefficient and leads to a larger number of required iterations for convergence. Based upon this explanation, it makes sense that fewer iterations were required for convergence to occur across the three fixed seeds for the Bayes Monte Carlo method compared to the Metropolis Hasting algorithm (MCMC method).

**CHOICES:**

Several key choices were made in the process of writing this code script in R to accomplish each of the tasks. **One choice** was to set three fixed seeds (3, 5, and 9) to facilitate controlled randomness and reproducibility of the results. An alternative plausible choice could have been to set a single fixed seed or many fixed, random seeds, (such as 5 or 10 seeds). Setting a single fixed seed would enable reproducibility of the results, but would not enable analysis of the results across multiple seeds. Conversely, setting many fixed seeds (i.e., 5 or 10 seeds) may lead to a more robust analysis (compared to 3 seeds), but would also significantly increase the computational time required to run the code script for many seeds. Thus, I decided to set three fixed seeds (3, 5, and 9) in my code script to introduce controlled randomness and facilitate reproducibility of the results, while not significantly increasing the computation time.

In a similar manner to problem set #3, a **second choice** that I made was to define the range of fuel levels in the airplane tank from -60 liters (an impossible value) up to 182 liters (the maximum tank capacity) to determine the probability density function (PDF) for the usable fuel in the tank (i.e., the likelihood function). An alternative choice could have been to set the lower bound to a value such as -20 liters, however, this decision may have led to a partial truncation of the left tail of the distribution for the likelihood function, and thus may have led to inaccurate estimates for the mean, mode, and probability of negative fuel in the tank from the likelihood

function. Thus, I decided to set the lower bound of the range of fuel levels in the airplane tank to -60 for the likelihood function to ensure that the left tail of the distribution was not truncated to prevent any inaccurate calculated estimates for the mean, mode, and probability of negative fuel in the airplane tank from the likelihood function.

In the process of defining a broad range of fuel levels in the airplane tank, a **third choice** that I made was to set the step (or increment) size to 0.001 (i.e., by = 0.001). A plausible alternative choice could have been to set the step size to a larger value such as 0.01 or a smaller value such as 0.0001, but I opted to set the step size to 0.001 because I thought that this step size would not significantly sacrifice accuracy, while still maintaining good computational efficiency.

In a similar manner to problem set #3, a **fourth choice** that I made was to remove the default x-axis labels, specify my own upper and lower x-axis limits, and add custom x-axis labels to improve the readability of the plot in any instance where I was plotting the likelihood function. An alternative choice could have been to use the default x-axis labels in base R plotting, but this would have undesirably omitted the left tail of the likelihood function from view on the plot. Thus, to ensure that the full likelihood function was visible on the plot, I opted to remove the default x-axis labels, specify my own custom x-axis range of values from -60 to 200, and set custom labels on the x-axis in increments of 20.

In terms of plotting the results in general, a stylistic and **fifth choice** that I made was to include vertical dashed or dotted lines to specify the expected value of available fuel (the mean), the most likely value of available fuel (the mode), and ± 1 standard deviation of the fuel sensor error for reference. An alternative choice could have been to omit all or some of these elements from the plot when plotting the likelihood function, the likelihood function with the prior, or the posteriors from the Metropolis Hasting algorithm or the Bayes Monte Carlo method to visually simplify the plot, but I decided to include these elements as I felt that they were beneficial to reference each time that I was plotting the results.

In the process of defining my proper prior, (in a similar manner to problem set #3), a **six choice** that I made was to constrain the range of possible fuel levels to a physically realistic range of 0 liters (the minimum) to 182 liters (the maximum capacity), because negative fuel levels are not physically possible. An alternative choice could have been to set the range of possible fuel levels to a negative value such as -20 liters, but this would have been a poor decision because negative fuel levels are not physically possible, and thus this would have been a poor prior. Another alternative choice could have been to maintain the lower bound of the range of possible fuel levels at -60 liters (as was the case for the likelihood function). However, the problem with this is that the posteriors would have not been distinct from the likelihood function on the plot, as the prior would not have introduced any new beneficial and realistic prior knowledge or information to achieve more realistic fuel estimates. Thus, I decided to set the range of possible fuel levels for my prior from 0 - 182 liters of fuel, as this introduced a realistic range of possible fuel levels.

Also in terms of setting a proper prior (in a similar manner to problem set #3), a **seventh choice** that I made was to use a truncated uniform prior within the range of 0 liters to 182 liters, which assumed that all values in this plausible range were equally likely before incorporating the fuel sensor reading. One plausible alternative could have been to use a beta distribution prior, which

could have been used to favor likely values and suppress extreme or less likely values (i.e., a completely full or empty tank). Another plausible alternative could have been to use a normal prior centered around a typical refueling level, such as 70% or 75% of the full capacity of the tank. However, I decided to use a truncated uniform prior for simplicity and because I felt that it was realistic to assume that all values within the realistic range were equally probable.

Also in the process of setting my prior, (in a similar manner to problem set #3), an **eighth choice** that I made was to assign a probability of zero to fuel levels outside of the specified range set by my prior. An alternative choice could have been to assign a non-zero probability to negative fuel levels outside of my specified range, however, this may have led to Bayesian updates that may have suggested a non-negligible chance of negative fuel in the tank, which is impossible. Thus, I decided to set the probability to zero for fuel levels outside of my specified range in the prior to ensure that the posterior estimates remained within the realistic bounds of my proper prior.

In general, an **eighth choice** that I made was to normalize the posteriors to sum to a value of 1 to ensure that they formed a valid probability distribution. A less plausible alternative choice could have been to not normalize the posteriors; however, this would have resulted in the posteriors retaining their shape but not being directly interpretable as probability density functions, (as probabilities can only range between 0 and 1). Thus, I decided to normalize the posteriors to sum to 1 to ensure that the posteriors could be interpreted as probability density functions (PDFs) that accurately reflect probability proportions.

To visualize the proper prior, a **ninth choice** that I made was to plot the proper prior using black dashed lines with the likelihood function. An alternative choice could have been to not plot the prior with the likelihood function, but I opted to do so because I felt that it was beneficial to observe the effect of the prior on defining a realistic range of fuel levels from 0 – 182 liters.

In the process of laying the groundwork for the Metropolis Hasting algorithm, a **tenth choice** that I made was to calculate the mode by rounding the samples to two decimal places and finding the most likely (most frequent) value. An alternative choice could have been to use kernel density estimation to provide a smoother estimation of the mode. However, I opted to round the samples to 2 decimal places for the sake of simplicity and because I did not want the KDE to interfere with producing accurate mode estimates.

In terms of implementing the Metropolis Hasting algorithm to determine the Bayesian update from the likelihood function and prior, (and setting the convergence and stopping criteria for the Metropolis Hasting algorithm), many important choices were made. Provided below is a detailed listing of these choices. An **eleventh choice** was to set the maximum number of iterations to 1 million as an upper bound or limit for the Metropolis Hasting algorithm. An alternative choice could have been to set the maximum number of iterations to 500,000 or to a larger value such as 10 million. However, I opted to set the maximum number of iterations to 1 million because I was worried that setting the max_iterations to only 500,000 may not have been enough iterations to enable convergence to occur prior to the maximum number of iterations being reached. I also opted to not set the maximum number of iterations to 10 million after discovering that convergence was occurring between 500,000 – 700,000 iterations and I felt that 10 million

iterations was an unnecessarily large number of iterations to set as an upper limit in light of this. Thus, I opted to set the maximum number of iterations to 1 million.

A **twelfth choice** that I made was to set the mode tolerance to 0.5 (0.5 liters of fuel). Alternative choices could have been to set the mode tolerance to a smaller value such as 0.001 or to a larger value such as 1. I experimented for quite a while with the mode tolerance set at 0.001, however I discovered that it was taking a long time for convergence to occur and after discussions with Dr. Keller, it was determined that 0.001 liters of fuel was an unnecessarily small tolerance to set in the context of considering liters of fuel in an airplane tank. Alternatively, a mode tolerance of 1 may have been a bit too large to enable an accurate and refined assessment of whether or not the mode estimates had truly converged for each seed or not. Thus, I opted to set the mode tolerance to 0.5 liters because I felt that this was a sufficiently small enough value to set as a tolerance threshold to examine whether or not the mode estimates had converged for each of the seeds.

A **thirteenth choice** that I made was to set the check interval value to 1000 to check whether convergence had occurred or not every 1000 iterations of the Metropolis Hasting algorithm. An alternative choice could have been to set the check interval to 500 or to a larger value such as 5000. In the context of setting 1 million iterations as the maximum number of iterations, I felt as if setting the check interval to every 500 iterations was too frequent for assessing convergence. Alternatively, I decided that 5000 iterations was not frequent enough to assess convergence. Thus, I decided to set the check interval value to 1000 because I felt that 1000 was a frequent, yet large enough value in the context of the max_iterations to regularly assess convergence.

A **fourteenth choice** that I made was to set the number of stable checks to 30. In the context of setting the convergence and stopping criteria for the Metropolis Hasting algorithm, setting the stable checks to 30 means that 30 consecutive stable checks must be passed in which the change in the mode is below the threshold of 0.5 liters of fuel in order to deem convergence and stop the number of iterations early. An alternative choice could have been to set the number of stable checks to a value of 10 or to set the number of stable checks to a larger value such as 50. However, I felt that only 10 stable checks was not a sufficiently large enough number of consecutive stable checks to deem convergence, as I was worried that fluctuations in the mode estimates could occur after only 10 stable checks. Conversely, I experimented with setting the number of stable checks to 50 and I discovered that this significantly increased the computation time to run the code script. Although setting the number of stable checks to 50 would increase my confidence in whether convergence truly occurred or not, I thought that it was unnecessary to set the number of stable checks to this large of a value in consideration of the computation time.

A **fifteenth choice** that I made was to set the burn in period for the Metropolis Hasting algorithm to 30% of the samples. This decision means that the first 30% of the generated samples were discarded and not considered in the subsequent analysis. An alternative choice could have been to set the burn-in period to 20% or to 50%, however, I felt as if 20% was not a large enough percentage of samples to eliminate bias from the initial sampling. Conversely, I felt as if 50% was omitting too many of the samples, and thus may have negatively impacted the mode estimates. Thus, I decided to set the burn-in period to 30% as I felt that this was a sufficiently large enough number of initial samples to discard to remove the bias from these samples, while also not negatively impacting the mode estimate calculations.

In terms of assessing convergence for the Metropolis Hasting algorithm, a **sixteenth choice** that I made was to set a counter to keep track of the number of consecutive stable checks. An alternative choice could have been to not set a counter, however, this would have been a poor choice as there would have been no method in the code script to track the number of consecutive stable checks to assess convergence. Thus, I decided to set a stable counter variable as a simple approach for tracking the number of consecutive stable checks for assessing convergence and for enabling early stopping of the iterations if convergence had occurred for a particular seed.

A **seventeenth choice** that I made was to set the standard deviation of the proposal distribution for the Metropolis Hasting algorithm to a value of 10. An alternative choice could have been to set the standard deviation value to a smaller value such as 5 or a larger value such as 20. Setting the proposal standard deviation to a smaller value such as 5 would have resulted in smaller, slower-moving steps, whereas setting the proposal standard deviation to 20 would have resulted in large steps or jumps. Due to the impact of this decision on mixing and convergence efficiency, I opted to set the proposal standard deviation value of 10 to enable mid-size steps to be taken, while not setting the value too small, which would have increased the computational time.

In terms of calculating the acceptance ratio to assist in the process of determining whether to accept or reject proposed samples, an **eighteenth choice** that I made was to use the standard Metropolis Hasting acceptance criterion to ensure correct sampling from the posterior. An alternative choice could have been to use the Metropolis algorithm (symmetric proposal), since this is approach is computationally efficient and simple to use. However, this approach is only valid for symmetric proposals. Thus, I opted to use the standard Metropolis Hasting acceptance criterion for the sake of simplicity in the process of accepting or rejecting proposed samples.

In terms of determining the posterior density estimations for the Metropolis Hasting algorithm, a **nineteenth choice** that I made was to use the kernel density estimation (KDE), as this provides a smooth estimate of the posterior. An alternative choice could have been to use a histogram-based density estimation. Despite being a simpler, approach, the estimates are less smooth and thus I opted to use KDE for my posterior density estimations for the Metropolis Hasting algorithm.

In terms of defining the likelihood function, prior probability, and posterior probability for use in the Metropolis Hasting algorithm, a **twentieth decision** that I made was to assign a probability value of zero to any negative fuel levels that were outside of the realistic range set by the proper prior. An alternative, but poor choice, could have been to assign a non-zero probability to any negative fuel estimates, but this would have contradicted the purpose of establishing the proper prior and it also would have negatively impacted the calculated estimates for the mean, the mode, and the probability of negative fuel in the tank from the posteriors for each of the seeds. Thus, I opted to assign all negative fuel levels a probability value of zero for each of these cases.

In terms of assessing whether the numerical inference about the mode converges, a **twenty-first** decision that I made was to set the track interval to 1000. This choice means that the mode will be recorded every 1000 iterations. An alternative choice could have been to set the track interval to a smaller number such as 500, or to a larger value such as 5000. Setting the track interval to 500 would have resulted in more frequent tracking (i.e., every 500 iterations instead of 1000), which produces high-resolution results, but at the expensive of computational time. Conversely,

setting the tracking interval to 5000 would have reduced computational time, but also would have led to the mode being recorded less frequently, and thus producing less refined results.

Also in terms of assessing whether the numerical inference about the mode converges, a **twenty-second choice** that I made was to use the lapply function to track the mode evolution by using lapply to apply a function to each element in my Metropolis Hasting samples (mh_samples), which contains the MCMC samples for each of the three seeds. An plausible alternative choice could have been to use a for loop, but I opted to use the lapply function because it improves readability and simplifies the code script.

Also in terms of assessing whether the numerical inference about the mode converges, a **twenty-third choice** that I made was to ensure consistency across each of my three seeds by determining the minimum length of my three Markov chains (1 for each seed), and subsequently trimming the other two chains to the length of the shortest train for subsequent analysis. An alternative choice could have been to use padding with NA instead of trimming the chains to preserve the original lengths, but I decided to trim the chains to ensure a consistent length for all of the Markov chains as I felt that this was the most straightforward approach for subsequent analysis.

Also in terms of assessing whether the numerical inference about the mode converges, a **twenty-fourth choice** that I made was to combine the mode tracking data into a matrix using the do.call(cbind, mode_tracking) function. The purpose of this was to bind these vectors column-wise into a matrix, such that each column represented the mode estimates for a specific seed (and each row represented a specific iteration step. An alternative choice could have been to use: do.call(rbind, mode_tracking) instead of cbind, as this would store the different mode estimates for the seeds in rows instead of columns. However, I opted to store them in columns, as I felt that this made it easier to calculate the variance across the seeds at each tracking interval.

Also in terms of assessing whether the numerical inference about the mode converges, a **twenty-fifth choice** that I made was to calculate (and subsequently plot) the variance of the mode estimates across the seeds at each tracking interval using the apply() function. The purpose of this choice was to measure how much the mode estimates vary across different seeds at each iteration step. An alternative choice could have been to calculate the standard deviation of the mode across the seeds, as standard deviation is potentially a bit more interpretable than variance (standard deviation is the square root of the variance). However, I decided to calculate the variance of the mode across the seeds because this captured the dispersion in square units, which I felt was a bit more useful for analyzing whether the numerical inference about the mode converged for each of the seeds.

In terms of defining and using a positive control to assess the accuracy of the implemented inference method, a **twenty-sixth choice** that I made was to set the positive control as the fuel sensor reading of 34 liters. An alternative choice could have been to set the positive control as a different fuel level value such as 30 liters, but my basis for setting the positive control as 34 liters was because this fuel sensor reading was a known value that was provided in problem set #3.

Also in terms of setting the positive control, a **twenty-seventh choice** that I made was to define the acceptable threshold for accuracy assessment as ± 3 liters of fuel from the positive control.

An alternative choice could have been to set a stricter threshold, such as ± 1 liter of fuel, however, I decided that this threshold for accuracy assessment was too strict, as 1 liter is a relatively small quantity of fuel (i.e., 1 liter = 0.264 gallons). An alternative choice could have also been to set a looser threshold for accuracy assessment, such as ± 5 liters of fuel from the positive control. However, I decided that this threshold was too loose and thus would not provide a very robust accuracy assessment, (as 5 liters is equivalent to 1.32 gallons of fuel). Therefore, I decided to set the acceptable threshold for accuracy assessment at ± 3 liters of fuel from the positive control of 34 liters because 3 liters is equivalent to about 0.80 gallons of fuel, which I felt was a reasonable threshold for performing an accuracy assessment of the mode estimates since 3 liters is still considerably less than 1 gallon of fuel in the airplane tank. (Note: I decided to conceptualize this in terms of gallons since I am more familiar with gallons than liters).

In terms of plotting the accuracy assessment results, a **twenty-eighth choice** that I made was to visualize the results using a bar plot. An alternative choice to visualize the results could have been to use a dot plot, but I decided to use a bar plot because I wanted to emphasize whether the deviation of each estimated mode value for each seed either exceeded or was below the accuracy assessment threshold of ± 3 liters of fuel and I felt that a bar plot was the best approach for this. As an additional note, I also decided to plot my positive control as a normal Gaussian distribution with my specified standard deviation range along with the posteriors from the Metropolis Hasting algorithm for each of my three seeds (with the estimated mode values for each seed as a vertical dashed line) as an additional approach for visualizing the relationship between the positive control and my posterior distributions.

To compare the required number of runs needed for converged inferences between the Metropolis Hasting algorithm and the Bayes Monte Carlo algorithm, I decided to re-perform the Bayes Monte Carlo algorithm within the code script. The **primary (and twenty-ninth) choice** that I made here was to use the same convergence and stopping criteria for the Bayes Monte Carlo method as in the Metropolis Hasting algorithm to facilitate a reasonable comparison between the results for the two methods. If different convergence and stopping criteria were used between the two algorithms, proper comparisons and conclusions could not be drawn. For instance, I left the maximum number of iterations at 1 million (for reasons previously described), I set the mode tolerance to 0.5 (for reasons previously described), I set the check interval to 1000 (for reasons previously described), and I set stable checks to 30 (for reasons described earlier).

A **thirtieth choice** that I made in the writing of the code script to perform the Bayes Monte Carlo method was to perform importance sampling with replacement using the normalized posterior weights (to ensure that the probabilities sum to a value of 1 before resampling). An alternative choice could have been to use importance sampling without replacement, however in this approach samples cannot be selected more than once, which can lead to over- or under-representation of certain regions, which can lead to inaccurate representations of the posterior distribution. Thus, I decided to perform importance sampling with replacement because this approach allowed for high-probability samples to be selected more frequently, which resulted in a more accurate representation of the posterior distribution.

A **thirty-first choice** that I made, (in a similar manner to the Metropolis Hasting algorithm) was to use kernel density estimation (KDE) to smooth the posterior distribution (with adjust = 1.5 to

modify bandwidth). An alternative choice could have been to use a simple histogram approximation, however histograms are sensitive to the bin size, which can result in a more granular (less refined) estimation of the posterior distribution.

A **thirty-second choice** that I made was to produce a table compare the required number of iterations needed for converged inference for the Metropolis Hasting algorithm and the Bayes Monte Carlo method for the three seeds. An alternative approach could have been to list the required number of iterations required for convergence for each of the seeds for each algorithm, but I felt that making a table of the results was the easiest approach to visually compare the required number of iterations for convergence to occur for each of the three seeds.

It is also worth mentioning that many small choices were also made in the process of producing the plots to visualize the results and to print the calculated estimates and results. These choices included 1) setting the colors, 2) choosing the font type and font sizes, 3) deciding what elements of the plot to include in the legend and formatting the legend, 4) selecting line widths and line types, 5) determining the axes limits, labels, and increments, 6) setting the alignment for the text for the printed results, 7) importing the gridExtra and grid libraries to produce a table for the convergence results etc. The overarching goal for all of these small stylistic choices for presenting the results was to maximize readability such that the key takeaways from the results could be easily determined by the viewer.

---

**REPRODUCIBILITY OF THE ANALYSES:**

**The analyses performed in the code script in R for this problem set are reproducible as a result of several key elements integrated into the code script**. The most important element facilitating reproducibility of the results in this code script was the setting of three random, fixed seed values (i.e., 3, 5, 9). Setting three different, random, fixed seed values introduces controlled randomness into the analyses, but it also ensures that the results obtained for a particular seed are the same every single time the code script is run. As an example of this, the Metropolis-Hasting algorithm and the Bayes Monte Carlo method both rely on random sampling. However, since each run is initialized with a specific seed, the generated random samples, calculated estimates, and convergence results will be the same each time that the simulations are performed.

A secondary element in the code script that also helps to facilitate reproducibility of the results arises from the use of explicitly defined inputs and model parameters, such as the fuel tank capacity, the fuel sensor reading, and the fuel sensor standard deviation as inputs. It also includes the convergence and stopping criteria (i.e., the mode tolerance, maximum iterations, stable checks, tracking interval etc.), for both of the algorithms. The use of fixed inputs and model parameters ensures that each time these algorithms are run for each of the seeds, the same inputs and model parameters are used every single time. This also ensures that proper comparisons can be drawn between the algorithms as well since the same model parameters are used for both.

---

## REFERENCES (CITATIONS)


1. Sriver, R., & Keller, K. (2024). *coin-example.R.* Retrieved from https://canvas.dartmouth.edu/courses/70487/files/folder/example%20codes?preview=13458702.

2. Short, T. (2004). *R Reference Card.* Retrieved from https://cran.r-project.org/doc/contrib/Short-refcard.pdf.

3. Applegate, P. J., Sriver, R. L., Garner, G. G., Bakker, A., Alley, R. B., & Keller, K. (2016). *Risk Analysis in the Earth Sciences: A Lab Manual with Exercises in R (Version 1.2).* SCRiM. Retrieved from https://leanpub.com/raes.

4. Grolemund, G. (2014). *Hands-On Programming with R: Write Your Own Functions and Simulations.* O'Reilly Media.

5. Venables, W. N., Smith, D. M., & the R Core Team. (2024). *An Introduction to R: Notes on R: A Programming Environment for Data Analysis and Graphics Version 4.4.2.* Retrieved from https://cran.r-project.org/doc/manuals/R-intro.pdf.

6. Rizzo, M. L. (2019). *Statistical Computing with R* (2nd ed.). Chapman & Hall/CRC.

7. Robert, C. P., & Casella, G. (2004). *Monte Carlo Statistical Methods* (2nd ed.). Springer.

8. Wickham, H. (2019). *Advanced R (2nd ed.)*. CRC Press.

9. Ross, S.M. (2006). *Simulation (4th ed)*. Academic Press.

10. Douglas, A., Roos, D., Mancini, F., Couto, A., & Lusseau, D. (2024). *An Introduction to R.* Retrieved from https://intro2r.com/Rbook.pdf.

11. Chang, W. (2013). *R Graphics Cookbook: Practical Recipes for Visualizing Data.* O'Reilly Media.

12. Murrell, P. (2005). *R Graphics* (2nd ed.). Chapman & Hall/CRC.

13. Robert, C.P. (2007). *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation (2nd.)*. Springer.

14. "R Graphics Lectures." (No Date). Retrieved from https://www.stat.auckland.ac.nz/~ihaka/787/lectures-r-graphics.pdf.

15. Gelman, A., Carlin, J.B., Stern, H.S., Dunson, D.B., Vehtari, A. & Rubin, D.B. (2013). *Bayesian Data Analysis (3rd ed.)*. CRC Press.

16. Navarro, D. (2023). The Metropolis-Hastings algorithm. Retrieved from https://blog.djnavarro.net/posts/2023-04-12_metropolis-hastings/

17. Wood, K. (2023). Metropolis-Hastings Algorithm. Retrieved from https://rpubs.com/ROARMarketingConcepts/1063733.

18. Graffelman, J. (No Date). *Nice placement of labels in a plot*. Retrieved from https://search.r-project.org/CRAN/refmans/calibrate/html/textxy.html

19. Brooks, S., Gelman, A., Jones, G.L. & Meng, X.L. (2011). *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC.

20. Signorell, A. (2021). Tables in R – A quick practical overview. Retrieved from https://cran.r-project.org/web/packages/DescTools/vignettes/TablesInR.pdf

21. Roy, V. (2019). Convergence diagnostics for Markov chain Monte Carlo. *Annual Review of Statistics and its Application* **7**, 387-412.

**Attached below is the link to access the files in my ENGS 107 GitHub Repository:**
https://github.com/isaiah-richardson28/Dartmouth-ENGS-107-Winter-2025

**APPENDIX:**

***Note: When copied and pasted into RStudio this code script produces the PDF file as desired***

```
# Course: ENGS 107: Bayesian Statistical Modeling and Computation
# Problem Set 4: Markov Chain Monte Carlo
# Professor: Dr. Klaus Keller PhD
# Due Date: Friday, February 14, 2025 at 11:59 pm
# File name: [isaiah.d.richardson.th@dartmouth.edu].PS#4.R
# Software: R and RStudio
#_____________________________________________________________

__
# Author: Isaiah D. Richardson, M.S. (IDR)
# Copyright: copyright by the author
# License: this code is distributed under the GNU GENERAL PUBLIC LICENSE v3.0
# License: for more information regarding GNU GENERAL PUBLIC LICENSE Version 3 visit:
https://www.gnu.org/licenses/why-not-lgpl.html
# There is no warranty on this R code script for ENGS 107 Question 4A
#_____________________________________________________________

__
# Version 1: last changes made on: February 13, 2025
#_____________________________________________________________

__
# Sources:
# - Personal correspondence with Dr. Klaus Keller during in-person office hours to discuss
expectations for each task, the plot of variance of the mode across iterations for task A, and
confirmation of the correct positive control for task B (02/07/2025; 02/10/2025)
# - Personal correspondence with TA Siddhi Gothivrekar during office hours in-person to
conceptually discuss the assignment expectations and to review the printed results and figures for
each of the tasks (02/11/2025)
# - R help files accessed through R-studio for syntax
# - The coin-example.R from Canvas for general formatting of the code script and layout of the
header notes
# - Short, T. (2004). R Reference Card. For understanding how to use functions and commands
in R including: seq, rep, dnorm, rnorm, pnorm, function, do.call, numeric, as.numeric, list, apply,
lapply, sapply, round, c, table, xaxt, xlim, ylim, segments, abline, bty, lty, lwd, ifelse, max, text,
plot, cex, $, font.main, cex.main
# - Applegate, P.J., Keller, K. et al.(2016). Risk Analysis in the Earth Sciences: A Lab Manual
with Exercises in R. For understanding the RStudio interface and basic functions in R
# - Grolemund, G. Hands-On Programming with R. To gain a better understanding of the
RStudio interface and basic functions in R
# - Venables, W.N., Smith, D.M. & the R Core Team. (2024). An Introduction to R Notes on R:
A Programming Environment for Data Analysis and Graphics Version 4.4.2 for understanding
functions, commands, and the basics of how to plot results
# - Rizzo, M.L. (2019). Statistical Computing with R (2nd edition). For understanding syntax,
functions, and how to perform a Bayes Monte Carlo simulation
```

# - Robert, C. P., & Casella, G. (2004). Monte Carlo Statistical Methods (2nd ed.). Springer. For understanding how to set up a Bayes Monte Carlo simulation and to understand the basics of a Markov Chain Monte Carlo method
# - Wickham, H. (2019). Advanced R (2nd ed.). CRC Press. For understanding basic functions in R and how to perform a Monte Carlo simulation in R
# - Ross, S.M. (2006). Simulation (4th ed). Academic Press. For understanding the basics of convergence and stopping criteria for a Bayes Monte Carlo simulation
# - Douglas, A., Roos, D., Mancini, F., Couto, A. & Lusseau, D. (2024). An Introduction to R. For understanding how to produce plots in R using the base R plot commands
# - Chang, W. (2013). R Graphics Cookbook: Practical Recipes for Visualizing Data. O'Reilly Media. For understanding the syntax, functions, formatting, and basics for producing plots and visualizing data in R
# - Murrell, P. (2005). R Graphics. Chapman & Hall/CRC. For understanding the basics of how to plot results visually and textually in R.
# - Robert, C.P. (2007). The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation (2nd.). Springer. To deepen my conceptual understanding of the likelihood function and prior for determining a Bayesian update
# - "R Graphics Lectures." (No Date). Retrieved from https://www.stat.auckland.ac.nz/~ihaka/787/lectures-r-graphics.pdf. To understand the basics of producing plots in R using base R plotting
# - Gelman, A., Carlin, J.B., Stern, H.S., Dunson, D.B., Vehtari, A. & Rubin, D.B. (2013). Bayesian Data Analysis (3rd ed.). CRC Press. To conceptually understand the basics of the Metropolis Hasting algorithm and a Markov Chain Monte Carlo method
# - Navarro, D. (2023). The Metropolis-Hastings algorithm. Retrieved from https://blog.djnavarro.net/posts/2023-04-12_metropolis-hastings/. To gain a basic conceptual understanding of the purpose of the Metropolis Hasting algorithm
# - Wood, K. (2023). Metropolis-Hastings Algorithm. Retrieved from https://rpubs.com/ROARMarketingConcepts/1063733.To understand the basics of how to implement a Metropolis Hasting algorithm in R
# - Graffelman, J. (No Date). Nice placement of labels in a plot. Retrieved from https://search.r-project.org/CRAN/refmans/calibrate/html/textxy.html. To understand the basics of positioning text and labels within a plot in R
# - Brooks, S., Gelman, A., Jones, G.L. & Meng, X.L. (2011). Handbook of Markov Chain Monte Carlo. Chapman & Hall/CRC. To further my conceptual understanding of Markov Chain Monte Carlo, the Metropolis Hasting algorithm, to understand the purpose of the burn-in period
# - Signorell, A. (2021). Tables in R – A quick practical overview. Retrieved from https://cran.r-project.org/web/packages/DescTools/vignettes/TablesInR.pdf. For understanding how to produce a basic table in R to visualize data
# - Roy, V. (2019). Convergence diagnostics for Markov chain Monte Carlo. Annual Review of Statistics and its Application 7, 387-412. For understanding basic convergence criteria to consider for performing a Markov Chain Monte Carlo, also to better understanding burn-in and convergence checking
#_____
__
# Collaborators:
# - I did not work with any collaborators on problem set #4

```
#_____
__
# To run this code script file:
# 1. Open the ascii file in R
# 2. Use the cursor to highlight the entire code script
# 3. Press either 'Source', 'Ctrl + Enter', or 'run' to run the entire code script
# 4. Open the resulting PDF file to analyze the results and figures
#=================================================================
==========
#### THE PROBLEM ####
# Revisit the estimation problem about the fuel level (just this part) from the previous problem
set
# using an implementation of the Metropolis Hasting algorithm. To this end, please:
# a. assess whether your numerical inference about the mode converges
# b. define and use a positive control to assess the accuracy of your implemented method for the
inference in part a
# 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
#=================================================================
==========
#### IMPLEMENTING THE METROPOLIS-HASTING ALGORITHM TO DETERMINE
THE BAYESIAN UPDATE FROM THE LIKELIHOOD FUNCTION AND PROPER PRIOR
####
#=================================================================
==========
#### THE INPUTS ####

# Specifying the total fuel tank capacity (182 liters)
total_fuel_tank_capacity <- 182

# Specifying the fuel sensor reading (34 liters)
fuel_sensor_reading <- 34

# Specifying the digital fuel sensor standard deviation (20 liters)
fuel_sensor_standard_deviation <- 20
#_____
__

# Setting three fixed seeds to facilitate controlled randomness and reproducibility of the results
seeds <- c(3, 5, 9)
#_____
__

# Creating a PDF file to plot the results and print the calculated estimates
pdf("ENGS_107_Problem_Set_4_Results.pdf", width = 8, height = 6)
```

```
#_____
__

# Defining the colors to plot the likelihood function for each of the three seeds
seed_colors <- c("darkblue", "blue", "lightblue")

# Defining the line widths for the likelihood function for each of the three seeds to enhance
readability
seed_line_widths <- c(4, 2, 1)
#_____
__

# Initializing a list to store the estimates
estimates <- list()

# Initializing a list to store the calculated values for plotting of the likelihood function
plot_data <- list()
#_____
__

# Using a for loop to perform the calculations for each of the three seeds
for (i in seq_along(seeds)) {
  set.seed(seeds[i])

  # Defining a broad range of fuel levels in the airplane tank
  # Setting the step length (or increment size) as 0.001 using by = 0.001
  usable_fuel_in_tank <- seq(-60, total_fuel_tank_capacity, by = 0.001)

  # Using the dnorm function to calculate the probability density function (PDF) for the fuel level
  pdf_usable_fuel <- dnorm(usable_fuel_in_tank, mean = fuel_sensor_reading, sd =
fuel_sensor_standard_deviation)
#_____
__

  # Calculating the expected value of available fuel from the likelihood function
  expected_value_available_fuel <- sum(usable_fuel_in_tank * pdf_usable_fuel) /
sum(pdf_usable_fuel)

  # Calculating the most likely value of available fuel from the likelihood function
  most_likely_value_available_fuel <- usable_fuel_in_tank[which.max(pdf_usable_fuel)]

  # Using the pnorm function to calculate the probability of negative fuel in the airplane tank
  probability_negative_fuel_in_tank <- pnorm(0, mean = fuel_sensor_reading, sd =
fuel_sensor_standard_deviation)
#_____
__
```

```
  # Storing the estimates
  estimates[[i]] <- c(seeds[i], expected_value_available_fuel, most_likely_value_available_fuel,
probability_negative_fuel_in_tank)

  # Storing the plotting data for the likelihood function
  plot_data[[i]] <- list(x = usable_fuel_in_tank, y = pdf_usable_fuel, expected =
expected_value_available_fuel, most_likely = most_likely_value_available_fuel)
}
#_____________________________________________________________________________
__
#### PLOTTING THE PDF FOR USABLE FUEL IN THE TANK (THE LIKELIHOOD
FUNCTION) ####

# Plotting the PDF for usable fuel in the airplane tank (the likelihood function) for each of the
three seeds
# Specifying the ranges for the x-axis and y-axis to ensure that the entire likelihood function is
visible on the plot
plot(NULL, xlim = c(-60, 200), ylim = c(0, 0.025), type = "n",

     # Specifying the title/header for the plot
     main = "PDF for Usable Fuel for All Seeds (Likelihood Function)",

     # Specifying the x-axis label
     xlab = "Fuel (Liters)",

     # Specifying the y-axis label
     ylab = "Probability Density",

     # Removing the default x-axis labels
     xaxt = "n")

# Adding custom labels to the x-axis to enhance the readability of the plot
# Specifying the bounds and labeled increments on the x-axis
axis(1, at = seq(-60, 200, by = 20), labels = seq(-60, 200, by = 20))

# Using a for loop to add the likelihood function for each of the seeds to the plot
for (i in seq_along(seeds)) {
  lines(plot_data[[i]]$x, plot_data[[i]]$y, col = seed_colors[i], lwd = seed_line_widths[i])
}

# Adding a vertical dashed line to denote the expected value of available fuel
abline(v = plot_data[[i]]$expected, col = "red", lwd = 3, lty = 2)

# Adding a vertical dashed line to denote the most likely value of available fuel
abline(v = plot_data[[i]]$most_likely, col = "darkorange", lwd = 3, lty = 3)
```

```
# Adding a set of vertical dashed lines to denote the standard deviation of the digital fuel sensor
abline(v = c(fuel_sensor_reading - 20, fuel_sensor_reading + 20), col = "darkgray", lwd = 3, lty
= 3)

# Adding text to label the +1 and -1 standard deviations of the digital fuel sensor
text(fuel_sensor_reading - 20, max(plot_data[[i]]$y) * 0.7, "-1 SD", pos = 2, col = "darkgray")
text(fuel_sensor_reading + 20, max(plot_data[[i]]$y) * 0.7, "+1 SD", pos = 4, col = "darkgray")

# Adding a legend to label the key elements on the plot of the likelihood function
legend("topright", legend = c("Likelihood Function (Seed 3)", "Likelihood Function (Seed 5)",
"Likelihood Function (Seed 9)", "Expected Value (Mean)", "Most Likely Value (Mode)", "Fuel
Error SD"),
       col = c("darkblue", "blue", "lightblue", "red", "darkorange", "darkgray"), lwd = c(4, 4, 4, 3,
3, 3), lty = c(1, 1, 1, 2, 3, 3), bty = "o", inset = 0.02)

#_____________________________________________________________________________
__
#### PRINTING THE ESTIMATES FROM THE LIKELIHOOD FUNCTION ON A NEW
PAGE IN THE PDF FILE ####

# Opening a new plot to print the calculated estimates from the likelihood function for each of
the three seeds
plot.new()

# Specifying a title for the second page of the PDF file detailing the printed estimates for each of
the three seeds
title(main = "Estimates from PDF of Usable Fuel for All Seeds (Likelihood Function)",
      font.main = 2, cex.main = 1.0)

# Using function to add bold and underlined headers to the page
bold_underline <- function(text) {
  bquote(underline(bold(.(text))))
}

# Specifying the Y-coordinate for text positioning
y_pos <- 0.85

# Specifying the spacing to ensure that all printed text is spaced properly on the page
y_step <- 0.06

# Specifying the list of seed values
seed_numbers <- c(3, 5, 9)

# Using a for loop to print the results for each of three seeds
for (i in seq_along(seed_numbers)) {
```

```
  # Specifying a header for each seed value
  text(0.05, y_pos, bold_underline(paste("Seed", seed_numbers[i])), cex = 1.0, adj = 0)
  y_pos <- y_pos - y_step

  # Printing the expected value of available fuel (the mean) for each seed value
  text(0.05, y_pos, paste("Expected Value of Available Fuel (Mean):", round(estimates[[i]][2], 2),
"liters"), adj = 0)
  y_pos <- y_pos - y_step

  # Printing the most likely value of available fuel (the mode) for each seed value
  text(0.05, y_pos, paste("Most Likely Value of Available Fuel (Mode):", round(estimates[[i]][3],
2), "liters"), adj = 0)
  y_pos <- y_pos - y_step

  # Printing the probability of negative fuel in the airplane tank for each seed value
  text(0.05, y_pos, paste("Probability of Negative Fuel in the Tank:", round(estimates[[i]][4] *
100, 1), "%"), adj = 0)

  # Specifying the spacing between the results for each of the three seeds
  y_pos <- y_pos - y_step - 0.04
}
#=====================================================================
===========
#### DEFINING THE PROPER PRIOR ####

# Creating a variable to specify the minimum realistic fuel capacity in the airplane tank as 0 liters
of fuel
minimum_fuel_tank_capacity <- 0

# Determining the height of the uniform prior distribution that constrains the airplane fuel level
between 0 liters and 182 liters
# Note: The total probability must integrate to 1, thus the height must be set accordingly
# Note: The uniform prior assumes that all fuel levels are equally likely within the realistic range
of 0 - 182 liters
prior_height <- 1 / (total_fuel_tank_capacity - minimum_fuel_tank_capacity)

# Defining the prior probability distribution over the entire range of usable fuel in the tank
# Note: This distribution assigns a constant probability density within the realistic fuel range and
a probability of zero to values outside of the range
prior <- ifelse(usable_fuel_in_tank >= minimum_fuel_tank_capacity & usable_fuel_in_tank <=
total_fuel_tank_capacity, prior_height, 0)

# Creating a sequence of fuel values for plotting the prior
fuel_range <- seq(minimum_fuel_tank_capacity, total_fuel_tank_capacity, length.out = 100)
```

```
# Calculating the prior probability values corresponding to each fuel level
# Note: There is a constant prior probability for all valid fuel levels
prior_values <- rep(prior_height, length(fuel_range))
#_____________________________________________________________________
__
#### PLOTTING THE PROPER PRIOR WITH THE LIKELIHOOD FUNCTION ####

# Plotting the likelihood function for each of the seeds with the prior
# Note: specifying the range of values for the x-axis and y-axis
plot(NULL, xlim = c(-60, 200), ylim = c(0, 0.025), type = "n",

    # Specifying the title/header for the plot
    main = "Plot of the Likelihood Function with the Proper Prior Across Seeds",

    # Specifying the x-axis label
    xlab = "Fuel (Liters)",

    # Specifying the y-axis label
    ylab = "Probability Density",

    # Removing the default x-axis labels
    xaxt = "n")

# Adding custom labels and label increments to the x-axis to enhance readability of the plot
# Note: Specifying the bounds and labeled increments on the x-axis
axis(1, at = seq(-60, 200, by = 20), labels = seq(-60, 200, by = 20))

# Using a for loop to add the likelihood function for each of the seeds to the plot
for (i in seq_along(seeds)) {
  lines(plot_data[[i]]$x, plot_data[[i]]$y, col = seed_colors[i], lwd = seed_line_widths[i])
}

# Adding a vertical dashed line to denote the expected value of available fuel
abline(v = plot_data[[i]]$expected, col = "red", lwd = 3, lty = 2)

# Adding a vertical dashed line to denote the most likely value of available fuel
abline(v = plot_data[[i]]$most_likely, col = "darkorange", lwd = 3, lty = 3)

# Adding a set of vertical dashed lines to denote the standard deviation of the digital fuel sensor
abline(v = c(fuel_sensor_reading - 20, fuel_sensor_reading + 20), col = "darkgray", lwd = 3, lty
= 3)

# Adding text to label the +1 and -1 standard deviations of the digital fuel sensor
text(fuel_sensor_reading - 20, max(plot_data[[i]]$y) * 0.7, "-1 SD", pos = 2, col = "darkgray")
text(fuel_sensor_reading + 20, max(plot_data[[i]]$y) * 0.7, "+1 SD", pos = 4, col = "darkgray")
```

```
# Adding vertical dashed lines to denote the bounds for the proper prior
segments(x0 = minimum_fuel_tank_capacity, y0 = 0, x1 = minimum_fuel_tank_capacity, y1 =
prior_height, col = "black", lwd = 3, lty = 2)
segments(x0 = total_fuel_tank_capacity, y0 = 0, x1 = total_fuel_tank_capacity, y1 =
prior_height, col = "black", lwd = 3, lty = 2)

# Adding a horizontal dashed line for the proper prior between 0 liters and 182 liters to create a
rectangle
segments(x0 = minimum_fuel_tank_capacity, y0 = prior_height, x1 = total_fuel_tank_capacity,
y1 = prior_height, col = "black", lwd = 3, lty = 2)

# Adding a legend to label the key elements on the plot of the likelihood function and the proper
prior
legend("topright", legend = c("Likelihood Function (Seed 3)", "Likelihood Function (Seed 5)",
"Likelihood Function (Seed 9)", "Proper Prior", "Expected Value (Mean)", "Most Likely Value
(Mode)", "Fuel Error SD"),
       col = c("darkblue", "blue", "lightblue", "black", "red", "darkorange", "darkgray"), lwd = c(4,
4, 4, 3, 3, 3, 3), lty = c(1, 1, 1, 2, 3, 3, 3), bty = "o", inset = 0.02)

#=================================================================================
===========
#### IMPLEMENTING THE METROPOLIS HASTING ALGORITHM TO DETERMINE
THE BAYESIAN UPDATE FROM THE LIKELIHOOD FUNCTION AND THE PRIOR ####

# Using function to calculate the likelihood of a given fuel level using a normal distribution
# Note: The likelihood represents the probability of observing the given fuel level
likelihood <- function(fuel_level) {
  # Note: If the fuel level is negative, assign it a probability value of zero
  if (fuel_level < 0) return(0)

  # Using the dnorm function to calculate the likelihood using the normal distribution
  dnorm(fuel_level, mean = fuel_sensor_reading, sd = fuel_sensor_standard_deviation)
}

# Using function to define the prior probability (assumes a uniform distribution)
# Note: The prior represents our initial belief about the fuel level before considering observations
prior_prob <- function(fuel_level) {
  if (fuel_level >= minimum_fuel_tank_capacity & fuel_level <= total_fuel_tank_capacity) {

    # Note: Assigning equal probability to all fuel levels within the capacity of the fuel tank
    return(1 / (total_fuel_tank_capacity - minimum_fuel_tank_capacity))
  } else {
    # Assigning a probability value of zero to fuel levels outside the range specified by the prior (0
liters to 182 liters)
    return(0)
  }
```

```
}

# Using function to calculate the un-normalized posterior probability from the prior and
likelihood
posterior_prob <- function(fuel_level) {
  # Assigning a probability value of zero to fuel levels outside the range specified by the prior
  if (fuel_level < 0) return(0)

  # Calculating the posterior probability
  # Note: posterior probability is proportional to the likelihood * prior (not normalized)
  likelihood(fuel_level) * prior_prob(fuel_level)
}

# Using function to calculate the most likely value (mode) of the sample distribution
compute_mode <- function(samples) {

  # Creating a frequency table (rounding the values to two decimal places for binning)
  freq_table <- table(round(samples, 2))

  # Note: Returning the most frequent value (mode)
  return(as.numeric(names(sort(freq_table, decreasing = TRUE)[1])))
}
#_____________________________________________________________________________
__
#### IMPLEMENTING THE METROPOLIS HASTING ALGORITHM TO DETERMINE
THE BAYESIAN UPDATE FROM THE LIKELIHOOD FUNCTION AND THE PRIOR ####

# Specifying the Metropolis-Hastings algorithm (a Markov Chain Monte Carlo [MCMC]
method)
# Setting the mode tolerance as 0.5 L using the variable tolerance = 0.5. This is the threshold for
allowed variation in the mode value
# Note: max_iterations is the variable for the maximum number of iterations for the algorithm
# Note: init is the initial value for the Markov Chain (starting point)
# Note: proposal_sd is the standard deviation for the proposal distribution. This is setting the step
or increment size
# Note: check_interval = 1000 specifies to check for convergence every 1000 iterations
# Note: stable_checks = 30 specifies that 30 consecutive stable checks (the mode under the
tolerance) for convergence to be determined
# Note: the burn_in_ratio = 0.30 specifies that the initial 30% of samples should be discarded
metropolis_hastings <- function(max_iterations, init, proposal_sd, seed, tolerance = 0.5,
check_interval = 1000, stable_checks = 30, burn_in_ratio = 0.30) {

  # Setting the seed (ensures reproducibility)
  set.seed(seed)

  # Initializing a numeric vector variable for sample storage up to the max number of iterations
```

```
  samples <- numeric(max_iterations)

  # Ensuring that the initial value is not a negative value
  samples[1] <- max(init, 0)

  # Defining the burn-in period (opting to discard the first 30% of samples)
  burn_in <- round(max_iterations * burn_in_ratio)

  # Using the mode_history variable to store mode values to track stability
  mode_history <- numeric(stable_checks)

  # Initializing a counter to track consecutive stable checks for the mode
  stable_count <- 0
#_____________________________________________________________________________
__

  # Using a for loop to run from 2 to maximum iterations (iterating over each step of the Markov
chain)
  for (i in 2:max_iterations) {

    # Using the rnorm function to propose a new candidate sample using a normal distribution
centered at the previous sample
    proposal <- rnorm(1, mean = samples[i - 1], sd = proposal_sd)

    # Using an if else statement to ensure that the proposed sample is not negative. If it is
negative, it is rejected and the previous sample is retained
    if (proposal < 0) {
      samples[i] <- samples[i - 1]
    } else {
      # Calculating the acceptance ratio (Metropolis-Hastings criterion)
      acceptance_ratio <- posterior_prob(proposal) / posterior_prob(samples[i - 1])

      # Using an if else statement to either accept or rejected the proposal
      # Note: A uniform random number runif(1) is drawn from [0,1]. If it is smaller than the
acceptance ratio then the proposal is accepted
      if (runif(1) < acceptance_ratio) {
        samples[i] <- proposal
      } else {
        # Reject and retain the previous value
        samples[i] <- samples[i - 1]
      }
    }
#_____________________________________________________________________________
__

    # Performing a convergence check after the burn-in period (first 30%)
```

```
   # Note: This ensures that convergence checking starts after the burn-in period and is
performed every 1000 iterations (as specified by check_interval)
   if (i > burn_in && i %% check_interval == 0) {

     # Calculating the mode of the sampled distribution
     current_mode <- compute_mode(samples[(burn_in + 1):i])

     # Updating the mode history by removing the oldest value and adding the newest value
     mode_history <- c(mode_history[-1], current_mode)

     # Using an if else statement to check if the mode remains stable over the recent set of
iterations
     # Note: If the mode remains constant or varies within the specified tolerance the stable_count
is incremented by 1
     if (length(unique(mode_history)) == 1 || all(abs(diff(mode_history)) < tolerance)) {
       stable_count <- stable_count + 1
     } else {
       # Resetting the stable_count to 0 if instability is detected
       stable_count <- 0
     }

     # Using an if statement to stop/terminate early if the mode has stabilized for the required
number of stable_checks (i.e., 30 stable checks)
     if (stable_count >= stable_checks) {
       cat("Converged at iteration:", i, "with mode:", round(current_mode, 3), "\n")
       return(list(samples = samples[(burn_in + 1):i], convergence_iter = i))  # Return results
     }
   }
 }

  # Returning the full sample set after burn-in if no early stopping occurred
  return(list(samples = samples[(burn_in + 1):max_iterations], convergence_iter =
max_iterations))
}
#_____________________________________________________________________________
__
# Running the Metropolis-Hastings Algorithm for each seed

# Specifying the maximum number of iterations for the Metropolis-Hasting algorithm (setting
the upper bound to 1 million)
# Note: The iterations will stop early if convergence is detected
max_iterations <- 1000000

# Defining an initial value based on fuel sensor reading of 34 liters
# Note: This ensures that the initial value is not negative
initial_guess <- max(fuel_sensor_reading, 0)
```

```
# Defining the standard deviation of the proposal distribution as 10
# Note: This controls the step size when proposing new values in the Metropolis-Hastings
algorithm
proposal_sd <- 10

# Initializing a list to store the generated samples from the Metropolis-Hasting algorithm for each
seed
mh_samples <- list()

# Initializing a list to store the number of iterations at which convergence was achieved for each
seed
convergence_info <- list()

# Using a for loop to perform the Metropolis-Hasting algorithm for each of the three seeds
for (seed in seeds) {
  result <- metropolis_hastings(max_iterations, initial_guess, proposal_sd, seed)
  # Storing the generated samples (this allows for tracking of the results from each of the seeds)
  mh_samples[[as.character(seed)]] <- result$samples
  # Storing the iteration at which the Markov chain converged
  convergence_info[[as.character(seed)]] <- result$convergence_iter
}

# Calculating the posterior densities using the lapply function
posterior_densities <- lapply(mh_samples, function(samples) density(samples, from = 0))
#_____________________________________________________________________________
__
#### PLOTTING THE POSTERIORS FROM THE METROPOLIS-HASTING ALGORITHM
WITH THE LIKELIHOOD AND PRIOR FOR EACH OF THE SEEDS ####

# Plotting the posterior distribution from the Metropolis-Hasting algorithm for each seed with the
prior and likelihood function
# Note: Specifying the range for the x-axis and the y-axis
plot(NULL, xlim = c(-60, 200), ylim = c(0, 0.025), type = "n",

    # Specifying the title/header for the plot
    main = "Plot of Posteriors from the Metropolis-Hasting Algorithm for All Seeds",

    # Specifying the x-axis label
    xlab = "Fuel (Liters)",

    # Specifying the y-axis label
    ylab = "Probability Density",

    # Removing the default x-axis labels
    xaxt = "n")
```

```
# Adding custom labels to the x-axis to enhance readability of the plot
# Note: Specifying the bounds and labeled increments for the x-axis
axis(1, at = seq(-60, 200, by = 20))

# Using a for loop to plot the likelihood function for each of the three seeds
for (i in seq_along(seeds)) {
  lines(plot_data[[i]]$x, plot_data[[i]]$y, col = seed_colors[i], lwd = seed_line_widths[i])
}

# Using a for loop to plot the posterior for each of the three seeds
posterior_colors <- c("darkgreen", "green", "lightgreen")
for (i in seq_along(seeds)) {
  lines(posterior_densities[[as.character(seeds[i])]]$x,
posterior_densities[[as.character(seeds[i])]]$y,
        col = posterior_colors[i], lwd = 4)
}

# Adding a vertical dashed line to denote the expected value of available fuel
abline(v = plot_data[[i]]$expected, col = "red", lwd = 3, lty = 2)

# Adding a vertical dashed line to denote the most likely value of available fuel
abline(v = plot_data[[i]]$most_likely, col = "darkorange", lwd = 3, lty = 3)

# Adding a set of vertical dashed lines to denote the standard deviation of the digital fuel sensor
abline(v = c(fuel_sensor_reading - 20, fuel_sensor_reading + 20), col = "darkgray", lwd = 3, lty
= 3)

# Adding text to label the +1 and -1 standard deviations of the digital fuel sensor
text(fuel_sensor_reading - 20, max(plot_data[[i]]$y) * 0.7, "-1 SD", pos = 2, col = "darkgray")
text(fuel_sensor_reading + 20, max(plot_data[[i]]$y) * 0.7, "+1 SD", pos = 4, col = "darkgray")

# Adding vertical dashed lines to denote the bounds for the proper prior
segments(x0 = minimum_fuel_tank_capacity, y0 = 0, x1 = minimum_fuel_tank_capacity, y1 =
prior_height, col = "black", lwd = 3, lty = 2)
segments(x0 = total_fuel_tank_capacity, y0 = 0, x1 = total_fuel_tank_capacity, y1 =
prior_height, col = "black", lwd = 3, lty = 2)

# Adding a horizontal dashed line for the proper prior between 0 liters and 182 liters to create a
rectangle
segments(x0 = minimum_fuel_tank_capacity, y0 = prior_height, x1 = total_fuel_tank_capacity,
y1 = prior_height, col = "black", lwd = 3, lty = 2)

# Adding a legend to the plot to label the key elements including the likelihood function for each
seed, the proper prior, the posterior for each seed, the expected value (mean), the most likely
value (mode), and the fuel sensor standard deviation
```

```
legend("topright", legend = c("Likelihood (Seed 3)", "Likelihood (Seed 5)", "Likelihood (Seed 9)", "Proper Prior", "Posterior (Seed 3)", "Posterior (Seed 5)", "Posterior (Seed 9)", "Expected Value (Mean)", "Most Likely Value (Mode)", "Fuel Error SD"),
       col = c("darkblue", "blue", "lightblue", "black", "darkgreen", "green", "lightgreen", "red", "orange", "darkgray"), lwd = c(4, 4, 4, 2, 4, 4, 4, 3, 3, 3), lty = c(1, 1, 1, 2, 1, 1, 1, 3, 3, 3), cex = 0.8, inset = 0.02)
#_____________________________________________________________________________
__
#### PRINTING THE POSTERIOR ESTIMATES FROM THE METROPOLIS-HASTING ALGORITHM FOR EACH OF THE SEEDS ON A NEW PAGE IN THE PDF FILE ####

# Opening a new plot in the PDF file to print the calculated estimates from the posterior distribution
plot.new()

# Specifying a title for this page of the PDF file
title(main = "Posterior Estimates from the Metropolis-Hasting Algorithm for All Seeds",
      font.main = 2, cex.main = 1.2)  # Reduced font size for better fit

# Calculating the posterior estimates for each of the three seeds
# Note: This loops through mh_samples, applying the function to each sample set (one for each seed)
posterior_estimates <- lapply(mh_samples, function(samples) {

  # Calculating the expected value of available fuel (the mean)
  expected_value <- mean(samples)

  # Calculating the most likely value of available fuel (the mode)
  most_likely_value <- compute_mode(samples)

  # Calculating the probability of negative fuel in the tank
  prob_negative_fuel <- mean(samples < 0)

  # Note: Storing the estimates for each seed as a dictionary inside the variable posterior_estimates
  list(mean = expected_value, mode = most_likely_value, prob_negative_fuel = prob_negative_fuel)
})

# Making the headers for each seed in bold font and underlined
bold_underline <- function(text) {
  bquote(underline(bold(.(text))))
}

# Defining the Y-positioning variables for the printed text
# Note: y_pos sets the starting position at the top of the page
```

```
y_pos <- 0.90
# Note: y_step specifies the vertical spacing between the lines of text
y_step <- 0.06

# Using a for loop to loop through each of the three seeds and print the estimates
for (i in seq_along(names(mh_samples))) {

  # Printing the header for each seed
  text(0.05, y_pos, bold_underline(paste("Seed", names(mh_samples)[i])), cex = 1.0, adj = 0)
  y_pos <- y_pos - y_step

  # Printing the expected value of available fuel (the mean)
  text(0.05, y_pos, paste("Expected Value of Available Fuel (Mean):",
round(posterior_estimates[[i]]$mean, 2), "liters"), adj = 0)
  y_pos <- y_pos - y_step

  # Printing the most likely value of available fuel (the mode)
  text(0.05, y_pos, paste("Most Likely Value of Available Fuel (Mode):",
round(posterior_estimates[[i]]$mode, 2), "liters"), adj = 0)
  y_pos <- y_pos - y_step

  # Printing the probability of negative fuel in the tank
  text(0.05, y_pos, paste("Probability of Negative Fuel in the Tank:",
round(posterior_estimates[[i]]$prob_negative_fuel * 100, 1), "%"), adj = 0)
  y_pos <- y_pos - y_step - 0.04
}
#=====================================================================
===========
#### PART A: ASSESSING WHETHER THE NUMERICAL INFERENCE ABOUT THE
MODE CONVERGES ####
#=====================================================================
===========

# Using the lapply function to track the evolution of the mode estimate across the iterations for
each of the three seeds
mode_tracking <- lapply(mh_samples, function(samples) {

  # Defining the track interval as 1000, meaning that the mode will be recorded every 1000
iterations
  track_interval <- 1000
  # Defining the burn-in period (the first 30% of the samples)
  burn_in <- round(length(samples) * 0.30)
  # Initializing a numeric vector to store the mode values over time
  mode_values <- numeric()
```

```
  # Using a for loop to iterate through the samples in increments of 1000 (set by the
track_interval variable) after the burn-in period
  for (i in seq(burn_in + track_interval, length(samples), by = track_interval)) {
    # Calculating the mode at each interval
    mode_values <- c(mode_values, compute_mode(samples[burn_in:i]))
  }
  # Returning the mode values tracked across iterations
  mode_values
})
#_____________________________________________________________________________
__
# Determining the minimum length of the mode tracking results across the seeds to align the data
for analysis
# Note: The purpose of this is to ensure consistency across the seeds because the different seeds
may generate chains of different lengths
min_length <- min(sapply(mode_tracking, length))

# Trimming each mode tracking sequence to match the shortest one for consistency so that each
of the chains is the same length
mode_tracking <- lapply(mode_tracking, function(modes) modes[1:min_length])

# Re-stating the track_interval to ensure that the code script runs properly
track_interval <- 1000

# Creating a sequence of iteration numbers corresponding to the mode tracking points
iterations <- seq(track_interval, track_interval * min_length, by = track_interval)

# Combining the mode tracking data from the three different seeds into a matrix for analysis
mode_matrix <- do.call(cbind, mode_tracking)

# Calculating the variance of the mode across the three different seeds at each tracking point
variance_tracking <- apply(mode_matrix, 1, var, na.rm = TRUE)
#_____________________________________________________________________________
__
### PLOTTING THE VARIANCE OF THE MODE ACROSS THE THREE SEEDS OVER
THE NUMBER OF ITERATIONS ###

# Plotting the variance of the mode across the three seeds over the number of iterations
plot(iterations, variance_tracking, type = "l", col = "darkblue", lwd = 3,

    # Specifying the title/header of the plot
    main = "Variance of Mode Across Seeds Over Iterations",

    # Specifying the x-axis label
    xlab = "Number of Iterations",
```

```
    # Specifying the y-axis label
    ylab = "Variance of Mode Across Seeds")

# Adding a legend to label the key elements of the plot
legend("topright", legend = "Variance of Mode Across Seeds", col = "darkblue", lwd = 3, inset =
0.02)
#_____
__
#### PRINTING THE CONVERGENCE ASSESSMENT RESULTS FOR EACH SEED ON A
NEW PAGE OF THE PDF FILE ####

# Opening a new plot in the PDF file to print the mode, convergence iteration, and convergence
status for each seed
plot.new()

# Specifying a title for this page in the PDF file assessing the mode convergence across all of the
seeds
title(main = "Assessment of Mode Convergence Across all Seeds",
      font.main = 2, cex.main = 1.2)

# Calculating the posterior estimates for each of the three seeds
posterior_estimates <- lapply(mh_samples, function(samples) {

  # Calculating the expected value of available fuel (the mean)
  expected_value <- mean(samples)

  # Calculating the most likely value of available fuel (the mode)
  most_likely_value <- compute_mode(samples)

  # Calculating the probability of negative fuel in the tank
  prob_negative_fuel <- mean(samples < 0)

  # Note: Storing the estimates for each seed as a dictionary inside the variable
posterior_estimates
  list(mean = expected_value, mode = most_likely_value, prob_negative_fuel =
prob_negative_fuel)
})

# Calculating the final variance of the mode across the three seeds
final_variance_mode <- variance_tracking[length(variance_tracking)]

# Displaying the final variance of mode across seeds
text(0.05, 0.95, paste("Final Variance of Mode Across Seeds:", round(final_variance_mode, 2)),
     font = 2, adj = 0)

# Making the headers for each seed in bold font and underlined
```

```
bold_underline <- function(text) {
  bquote(underline(bold(.(text))))
}

# Defining the Y-positioning variables for the printed text
# Note: y_pos sets the starting position at the top of the page
y_pos <- 0.85
# Note: y_step specifies the vertical spacing between the lines of text
y_step <- 0.06

# Using a for loop to loop through each of the three seeds and print the values
for (i in seq_along(names(mh_samples))) {
  # Specifying a header for each seed
  text(0.05, y_pos, bold_underline(paste("Seed", names(mh_samples)[i])), cex = 1.0, adj = 0)
  y_pos <- y_pos - y_step

  # Specifying the most likely value (the mode) for each seed
  text(0.05, y_pos, paste("Most Likely Value of Available Fuel (Mode):",
round(posterior_estimates[[i]]$mode, 2), "liters"), adj = 0)
  y_pos <- y_pos - y_step

  # Specifying the convergence iteration for each seed
  text(0.05, y_pos, paste("Convergence Iteration:", convergence_info[[i]]), adj = 0)
  y_pos <- y_pos - y_step

  # Specifying the convergence status for each seed
  text(0.05, y_pos, paste("Convergence Status:",
                          ifelse(convergence_info[[i]] < max_iterations, "Converged Successfully", "Did
Not Converge")), adj = 0)
  y_pos <- y_pos - y_step - 0.04
}
#=====================================================================
===========
#### PART B: DEFINING AND USING A POSITIVE CONTROL TO ASSESS THE
ACCURACY OF IMPLEMENTED INFERENCE METHOD FROM PART A ####
#=====================================================================
===========
#### ASSESSING ACCURACY OF THE IMPLEMENTED METHOD FOR INFERENCE
USING A POSITIVE CONTROL ####

# Defining the fuel sensor reading as the positive control (34 liters)
positive_control <- 34

# Defining an acceptable threshold for accuracy assessment (within ± 3 liters of fuel)
accuracy_threshold <- 3
```

```
# Initializing a list to store the accuracy assessment results for each of the three seeds
accuracy_results <- list()
#_____________________________________________________________________________
__
# Using a for loop to calculate the accuracy for each of the three seeds
for (seed in seeds) {
  seed_str <- as.character(seed)

  # Retrieving the inferred mode from the Metropolis-Hastings algorithm
  inferred_mode <- posterior_estimates[[seed_str]]$mode

  # Calculating the absolute deviation from the positive control (34 liters)
  deviation <- abs(inferred_mode - positive_control)

  # Checking if the deviation is within the acceptable accuracy threshold of ± 3 liters of fuel
  accurate <- deviation <= accuracy_threshold

  # Storing the accuracy results (the mode, the deviation, and the accuracy status)
  accuracy_results[[seed_str]] <- list(
    inferred_mode = inferred_mode,
    deviation = deviation,
    accurate = accurate
  )
}
#_____________________________________________________________________________
__
#### PRINTING THE ACCURACY ASSESSMENT RESULTS ON A BLANK PAGE IN THE
PDF FILE ####

# Opening a new blank page in the PDF file to print the accuracy assessment results
plot.new()

# Adding a title at the top of the PDF page for the accuracy assessments
title(main = "Accuracy Assessment of Inference Method Using Positive Control",
      font.main = 2, cex.main = 1.2)

# Making the headers for each seed in bold font and underlined
bold_underline <- function(text) {
  bquote(underline(bold(.(text))))
}

# Defining the Y-positioning variables for the printed text
# Note: y_pos sets the starting position at the top of the page
y_pos <- 0.85
# Note: y_step specifies the vertical spacing between the lines of text
y_step <- 0.06
```

```
# Using a for loop to loop through each of the three seeds and print the values
for (seed in seeds) {
  seed_str <- as.character(seed)

  # Specifying a header for each seed
  text(0.05, y_pos, bold_underline(paste("Seed", seed)), cex = 1.0, adj = 0)
  y_pos <- y_pos - y_step

  # Specifying the estimated mode for each seed
  text(0.05, y_pos, paste("Estimated Mode:", round(accuracy_results[[seed_str]]$inferred_mode,
2), "liters"), adj = 0)
  y_pos <- y_pos - y_step

  # Specifying the deviation from the positive control value of 34 liters
  text(0.05, y_pos, paste("Deviation from Positive Control (34 L):",
round(accuracy_results[[seed_str]]$deviation, 2), "liters"), adj = 0)
  y_pos <- y_pos - y_step

  # Specifying the accuracy status
  text(0.05, y_pos, paste("Accuracy Status:",
                          ifelse(accuracy_results[[seed_str]]$accurate, "Accurate", "Inaccurate")), adj = 0)
  y_pos <- y_pos - y_step - 0.04
}

#_____________________________________________________________________________
__
#### PLOTTING THE ACCURACY ASSESSMENT RESULTS ####

# Defining a y-axis range for the accuracy assessment plot
y_ticks <- seq(0, 10, by = 2)

# Using a bar plot to plot the deviations from the positive control (34 liters) for each of the three
seeds
barplot(
  sapply(accuracy_results, function(x) x$deviation),names.arg = paste("Seed", seeds), col =
ifelse(sapply(accuracy_results, function(x) x$accurate), "green", "red"),border = "black",

  # Specifying the title/header for the plot
  main = "Deviation of Estimated Modes from Positive Control (34 Liters)",

  # Specifying the y-axis label
  ylab = "Deviation (Liters)",

  # Specifying the y-axis range
  ylim = c(0, max(y_ticks)),
```

```
  # Specifying that the labels are horizontally oriented
  las = 1
)

# Adding black, dashed reference line at the accuracy threshold of +/- 3 liters of fuel
abline(h = accuracy_threshold, col = "black", lwd = 3, lty = 2)


# Adding a legend to label the key elements of the plot
# Note: pch = 22 indicates a square marker with a border; pt.bg specifies the fill colors; lty
specifies the line type and lwd specifies the line width
legend("topright", legend = c("Accurate (≤ 3L SD)", "Inaccurate (> 3L SD)", "Accuracy
Threshold"), col = c("black", "black", "black"), pch = c(22, 22, NA), pt.bg = c("green", "red",
NA), lty = c(NA, NA, 2), lwd = c(NA, NA, 3), bty = "o", inset = 0.02)
#_____________________________________________________________________________
__
#### PLOTTING THE POSITIVE CONTROL AS GAUSSIAN WITH THE POSTERIORS
FROM THE METROPOLIS-HASTING ALGORITHM ####

# Creating a sequence of fuel values for the positive control (assuming a Gaussian distribution)
# Note: by = 0.1 specifies the step size (or increment size)
positive_control_x <- seq(-60, 200, by = 0.1)

# Calculating the probability density function (PDF) for the positive control
positive_control_pdf <- dnorm(positive_control_x, mean = positive_control, sd = 3)

# Scaling the positive control PDF to match the scale of posteriors
scaling_factor <- max(sapply(posterior_densities, function(d) max(d$y))) /
max(positive_control_pdf)
positive_control_pdf <- positive_control_pdf * scaling_factor
#_____________________________________________________________________________
__

# Plotting the positive control with the posteriors from the Metropolis-Hasting algorithm
plot(NULL, xlim = c(-60, 200), ylim = c(0, max(sapply(posterior_densities, function(d)
max(d$y)) * 1.1)),
    type = "n",

    # Specifying the title/header for the plot
    main = "Plot of the Positive Control with the Metropolis-Hasting Posteriors",

    # Specifying the x-axis label
    xlab = "Fuel (Liters)",

    # Specifying the y-axis label
```

```
    ylab = "Probability Density",

    # Removing the default x-axis labels
    xaxt = "n")

# Adding custom labels to the x-axis and specifying the label increment
axis(1, at = seq(-60, 200, by = 20), labels = seq(-60, 200, by = 20))

# Plotting the positive control (Gaussian distribution)
lines(positive_control_x, positive_control_pdf, col = "black", lwd = 3, lty = 1)

# Using a for loop to plot the posterior distributions for each seed from the Metropolis-Hasting
algorithm
posterior_colors <- c("darkgreen", "green", "lightgreen")
for (i in seq_along(seeds)) {
  lines(posterior_densities[[as.character(seeds[i])]]$x,
posterior_densities[[as.character(seeds[i])]]$y,
       col = posterior_colors[i], lwd = 4)
}

# Adding vertical dashed lines at the estimated modes from the posteriors from the Metropolis-
Hasting algorithm
for (i in seq_along(seeds)) {
  inferred_mode <- posterior_estimates[[as.character(seeds[i])]]$mode
  abline(v = inferred_mode, col = posterior_colors[i], lwd = 3, lty = 2)
}

# Adding a legend to label the key elements of the plot
legend("topright",
       legend = c("Positive Control", "Posterior (Seed 3)", "Posterior (Seed 5)", "Posterior (Seed
9)", "Mode (Seed 3)", "Mode (Seed 5)", "Mode (Seed 9)"),
       col = c("black", "darkgreen", "green", "lightgreen", "darkgreen", "green", "lightgreen"), lwd
= c(3, 4, 4, 4, 3, 3, 3), lty = c(2, 1, 1, 1, 2, 2, 2), bty = "o", inset = 0.02)
#=================================================================
==========
#### PART C: COMPARING THE REQUIRED NUMBER OF RUNS NEEDED FOR
CONVERGED INFERENCES FROM THE METROPOLIS-HASTING ALGORITHM WITH
THE BAYES MONTE CARLO ALGORITHM ####
#=================================================================
==========
#### USING A BAYES MONTE CARLO METHOD TO DETERMINE THE BAYESIAN
UPDATE FROM THE LIKELIHOOD FUNCTION AND PRIOR FOR EACH OF THE THREE
SEEDS ####

# Using function to perform the Bayesian inference using a Monte Carlo method
```

```
# Note: Maximum iterations is set to 1 million (see below) (this is the same value as in the
Metropolis-Hasting algorithm)
# Note: Setting the mode tolerance to 0.5 (the same value as in the Metropolis-Hasting
algorithm)
# Note: Setting the check interval to 1000 (the same value as in the Metropolis-Hasting
algorithm)
# Note: Setting the stable checks to 30 (the same value as in the Metropolis-Hasting algorithm)
bayes_monte_carlo <- function(max_iterations, seed, tolerance = 0.5, check_interval = 1000,
stable_checks = 30) {
   # Setting the seed (using three different fixed seeds)
   set.seed(seed)

  # Generating prior samples
  # Note: This draws up to 1 million random samples from the uniform prior distribution within
the fuel tank capacity range (0 liters to 182 liters)
  samples <- runif(max_iterations, minimum_fuel_tank_capacity, total_fuel_tank_capacity)

  # Calculating the posterior probability for each sample
  weights <- sapply(samples, function(x) posterior_prob(x) * prior_prob(x))

  # Normalizing the weights so they sum to 1
  # Note: This ensures valid probability weights
  weights <- weights / sum(weights)

  # Performing importance sampling with replacement based on the posterior weights
  # Note: replace = TRUE specifies sampling with replacement
  resampled_samples <- sample(samples, size = max_iterations, replace = TRUE, prob =
weights)

  # Using the mode_history variable to store mode values to track stability
  mode_history <- numeric(stable_checks)

  # Initializing a counter to track consecutive stable checks for the mode
  stable_count <- 0
#_____________________________________________________________________________
__

  # Using a for loop to perform convergence monitoring after every 1000 iterations (as specified
by check_interval)
  for (i in seq(check_interval, max_iterations, by = check_interval)) {

    # Calculating the mode of the fuel level distribution
    current_mode <- compute_mode(resampled_samples[1:i])

    # Updating the mode history (this is tracking the stability over the last 1000 iterations)
    mode_history <- c(mode_history[-1], current_mode)
```

```
    # Using an if else statement to determine if the mode history indicates convergence
    if (length(unique(mode_history)) == 1 || all(abs(diff(mode_history)) < tolerance)) {
      # Increasing the stable count by 1 if the changes in the mode are below the tolerance
threshold
      stable_count <- stable_count + 1
    } else {
      # Resetting the stable counter to 0 if the changes in the mode are above the tolerance
threshold
      stable_count <- 0
    }
    # If stability has been observed for 30 stable checks, stop the iterations early due to
convergence
    if (stable_count >= stable_checks) {
      cat("Converged at iteration:", i, "with mode:", round(current_mode, 3), "\n")
      return(list(samples = resampled_samples[1:i], convergence_iter = i))
    }
  }
  # Note: If the for loop completes without detecting early convergence before the maximum
number of iterations are reached, return all of the generated samples
  return(list(samples = resampled_samples, convergence_iter = max_iterations))
}
#_____
__
#### RUNNING THE BAYES MONTE CARLO METHOD FOR EACH OF THE THREE
SEEDS ####

# Initializing a list to store the generated samples for each seed
bmc_samples <- list()

# Initializing a list to store the iteration at which the simulation converged for each seed
bmc_convergence_info <- list()

# Using a for loop to run the Bayes Monte Carlo simulation for each of the three seeds
for (seed in seeds) {

  # Performing the Bayes Monte Carlo simulation for the given seed
  result <- bayes_monte_carlo(max_iterations, seed)

  # Storing the generated sample values in a list indexed by seed
  bmc_samples[[as.character(seed)]] <- result$samples

  # Storing the iteration at which convergence was reached
  bmc_convergence_info[[as.character(seed)]] <- result$convergence_iter
}
```

```
#_____
__
# Calculating the Bayes Monte Carlo posterior densities for subsequent analysis
bmc_posterior_densities <- lapply(bmc_samples, function(samples) {

  # Filtering out negative samples (as only positive values are valid)
  filtered_samples <- samples[samples > 0]

  # Ensuring there are enough samples to calculate a density estimation
  if (length(filtered_samples) > 1) {

    # Calculating the kernel density estimation for the samples (only positive samples)
    # Note: adjust = 1.5 modifies the bandwidth to make the estimate more smooth
    density(filtered_samples, from = min(filtered_samples), to = max(filtered_samples), adjust =
1.5)
  } else {
    # Returning NULL if there are not enough valid samples for density estimation
    NULL
  }
})
#_____
__
#### PLOTTING THE POSTERIORS FROM THE BAYES MONTE CARLO METHOD FOR
EACH OF THE THREE SEEDS WITH THE LIKELIHOOD FUNCTION AND PRIOR ####

# Plotting the posteriors from the Bayes Monte Carlo method with the likelihood function and
prior for each of the three seeds
plot(NULL, xlim = c(-60, 200), ylim = c(0, 0.025), type = "n",

     # Specifying the title/header for the plot
     main = "Plot of Posteriors from Bayes Monte Carlo Method with Likelihood and Prior",

     # Specifying the x-axis label
     xlab = "Fuel (Liters)",

     # Specifying the y-axis label
     ylab = "Probability Density",

     # Removing the default x-axis label
     xaxt = "n")

# Adding custom x-axis labels and a custom increment to increase readability of the plot
axis(1, at = seq(-60, 200, by = 20))

# Using a for loop to plot the likelihood function for each of the three seeds
for (i in seq_along(seeds)) {
```

```
  lines(plot_data[[i]]$x, plot_data[[i]]$y, col = seed_colors[i], lwd = seed_line_widths[i])
}

# Plotting the posterior distributions for each of the three seeds from the Bayes Monte Carlo
method
bmc_posterior_colors <- c("purple", "violet", "magenta")
for (i in seq_along(seeds)) {
  lines(bmc_posterior_densities[[as.character(seeds[i])]]$x,
        bmc_posterior_densities[[as.character(seeds[i])]]$y,
        col = bmc_posterior_colors[i], lwd = 4)
}

# Adding a vertical dashed line to denote the expected value of available fuel
abline(v = plot_data[[i]]$expected, col = "red", lwd = 3, lty = 2)

# Adding a vertical dashed line to denote the most likely value of available fuel
abline(v = plot_data[[i]]$most_likely, col = "darkorange", lwd = 3, lty = 3)

# Adding a set of vertical dashed lines to denote the standard deviation of the digital fuel sensor
abline(v = c(fuel_sensor_reading - 20, fuel_sensor_reading + 20), col = "darkgray", lwd = 3, lty
= 3)

# Adding text to label the +1 and -1 standard deviations of the digital fuel sensor
text(fuel_sensor_reading - 20, max(plot_data[[i]]$y) * 0.7, "-1 SD", pos = 2, col = "darkgray")
text(fuel_sensor_reading + 20, max(plot_data[[i]]$y) * 0.7, "+1 SD", pos = 4, col = "darkgray")

# Adding vertical dashed lines to denote the bounds for the proper prior
segments(x0 = minimum_fuel_tank_capacity, y0 = 0, x1 = minimum_fuel_tank_capacity, y1 =
prior_height, col = "black", lwd = 3, lty = 2)
segments(x0 = total_fuel_tank_capacity, y0 = 0, x1 = total_fuel_tank_capacity, y1 =
prior_height, col = "black", lwd = 3, lty = 2)

# Adding a horizontal dashed line for the proper prior between 0 liters and 182 liters to create a
rectangle
segments(x0 = minimum_fuel_tank_capacity, y0 = prior_height, x1 = total_fuel_tank_capacity,
y1 = prior_height, col = "black", lwd = 3, lty = 2)

# Adding a legend to label the key elements of the plot
legend("topright", legend = c("Likelihood (Seed 3)", "Likelihood (Seed 5)", "Likelihood (Seed
9)", "Proper Prior", "BMC Posterior (Seed 3)", "BMC Posterior (Seed 5)", "BMC Posterior
(Seed 9)"),
       col = c("darkblue", "blue", "lightblue", "black", "purple", "violet", "magenta"), lwd = c(4, 4,
4, 3, 4, 4, 4), lty = c(1, 1, 1, 2, 1, 1, 1), inset = 0.02)
#_____________________________________________________________________________
__
```

```
#### PRINTING THE ESTIMATES FROM THE POSTERIORS FROM THE BAYES MONTE CARLO METHOD AND THE CONVERGENCE METRICS FOR EACH OF THE SEEDS ON A NEW PAGE IN THE PDF FILE ####

# Opening a new plot in the PDF file to print the posterior estimates from the Bayes Monte Carlo method along with the convergence metrics for each of the three seeds
plot.new()

# Specifying a title for this new page in the PDF file
title(main = "Posterior Estimates and Convergence Metrics from the Bayes Monte Carlo Method for All Seeds",
      font.main = 2, cex.main = 0.8)

# Calculating the posterior estimates for each of the three seeds
posterior_estimates <- lapply(bmc_samples, function(samples) {

  # Calculating the expected value of available fuel (the mean)
  expected_value <- mean(samples)

  # Calculating the most likely value of available fuel (the mode)
  most_likely_value <- compute_mode(samples)

  # Calculating the probability of negative fuel in the tank
  prob_negative_fuel <- mean(samples < 0)

  # Note: Storing the estimates for each seed as a dictionary inside the variable posterior_estimates
  list(mean = expected_value, mode = most_likely_value, prob_negative_fuel = prob_negative_fuel)
})

# Making the headers for each seed in bold font and underlined
bold_underline <- function(text) {
  bquote(underline(bold(.(text))))
}
# Defining the Y-positioning variables for the printed text
# Note: y_pos sets the starting position for the text at the top of the page
y_pos <- 0.95
# Note: y_step specifies the vertical spacing between the lines of text
y_step <- 0.045

# Using a for loop to iterate through each of the three seeds and print the calculated estimates and convergence metrics
for (i in seq_along(names(bmc_samples))) {
  seed_name <- names(bmc_samples)[i]
  convergence_iter <- bmc_convergence_info[[seed_name]]
```

```
  # Using an if else statement to determine the convergence status
  if (convergence_iter < max_iterations) {
    convergence_status <- "Converged Successfully"
  } else {
    convergence_status <- "Did Not Converge"
  }

  # Printing the header for each seed
  text(0.05, y_pos, bold_underline(paste("Seed", seed_name)), cex = 0.75, adj = 0)
  y_pos <- y_pos - y_step

  # Printing the expected value of available fuel (the mean) for each seed
  text(0.05, y_pos, paste("Expected Value of Available Fuel (Mean):",
                          round(posterior_estimates[[i]]$mean, 2), "liters"), cex = 0.65, adj = 0)
  y_pos <- y_pos - y_step

  # Printing the most likely value of available fuel (the mode) for each seed
  text(0.05, y_pos, paste("Most Likely Value of Available Fuel (Mode):",
                          round(posterior_estimates[[i]]$mode, 2), "liters"), cex = 0.65, adj = 0)
  y_pos <- y_pos - y_step

  # Printing the probability of negative fuel in the tank for each seed
  text(0.05, y_pos, paste("Probability of Negative Fuel in the Tank:",
                          round(posterior_estimates[[i]]$prob_negative_fuel * 100, 1), "%"),
       cex = 0.65, adj = 0)
  y_pos <- y_pos - y_step

  # Printing the required number of iterations for convergence for each seed
  text(0.05, y_pos, paste("Number of Iterations Required for Convergence:", convergence_iter),
       cex = 0.65, adj = 0)
  y_pos <- y_pos - y_step

  # Printing the convergence status for each seed
  text(0.05, y_pos, paste("Convergence Status:", convergence_status),
       cex = 0.65, adj = 0)
  y_pos <- y_pos - y_step - 0.03
}
#_____________________________________________________________________________
__
#### PLOTTING THE POSTERIORS FROM THE METROPOLIS-HASTING ALGORITHM
AND THE BAYES MONTE CARLO METHOD WITH THE LIKELIHOOD FUNCTION AND
PRIOR ####

# Plotting the posterior distribution for each seed from the Metropolis-Hasting algorithm and the
Bayes Monte Carlo method with the likelihood function and prior
```

```
plot(NULL, xlim = c(-60, 200), ylim = c(0, 0.025), type = "n",

    # Specifying the title/header for the plot
    main = "Posteriors from Metropolis Hasting Algorithm and Bayes Monte Carlo Method",

    # Specifying the x-axis label
    xlab = "Fuel (Liters)",

    # Specifying the y-axis label
    ylab = "Probability Density",

    # Removing the default x-axis labels
    xaxt = "n")

# Adding custom labels to the x-axis to enhance the readability of the plot
# Note: Specifying the bounds and labeled increments (by 20) on the x-axis
axis(1, at = seq(-60, 200, by = 20))

# Using a for loop to plot the likelihood function for each of the three seeds
for (i in seq_along(seeds)) {
  lines(plot_data[[i]]$x, plot_data[[i]]$y, col = seed_colors[i], lwd = seed_line_widths[i])
}

# Using a for loop to plot the posterior for each of the three seeds from the Metropolis Hasting
Algorithm
posterior_colors <- c("darkgreen", "green", "lightgreen")
for (i in seq_along(seeds)) {
  lines(posterior_densities[[as.character(seeds[i])]]$x,
posterior_densities[[as.character(seeds[i])]]$y,
        col = posterior_colors[i], lwd = 4)
}

# Plotting the Bayes Monte Carlo posterior distributions for each of the three seeds
bmc_posterior_colors <- c("purple", "violet", "magenta")
for (i in seq_along(seeds)) {
  lines(bmc_posterior_densities[[as.character(seeds[i])]]$x,
        bmc_posterior_densities[[as.character(seeds[i])]]$y,
        col = bmc_posterior_colors[i], lwd = 4)
}

# Adding a vertical dashed line to denote the expected value of available fuel
abline(v = plot_data[[i]]$expected, col = "red", lwd = 3, lty = 2)

# Adding a vertical dashed line to denote the most likely value of available fuel
abline(v = plot_data[[i]]$most_likely, col = "darkorange", lwd = 3, lty = 3)
```

```
# Adding a set of vertical dashed lines to denote the standard deviation of the digital fuel sensor
abline(v = c(fuel_sensor_reading - 20, fuel_sensor_reading + 20), col = "darkgray", lwd = 3, lty
= 3)

# Adding text to label the +1 and -1 standard deviations of the digital fuel sensor
text(fuel_sensor_reading - 20, max(plot_data[[i]]$y) * 0.7, "-1 SD", pos = 2, col = "darkgray")
text(fuel_sensor_reading + 20, max(plot_data[[i]]$y) * 0.7, "+1 SD", pos = 4, col = "darkgray")

# Adding vertical dashed lines to denote the bounds for the proper prior
segments(x0 = minimum_fuel_tank_capacity, y0 = 0, x1 = minimum_fuel_tank_capacity, y1 =
prior_height, col = "black", lwd = 3, lty = 2)
segments(x0 = total_fuel_tank_capacity, y0 = 0, x1 = total_fuel_tank_capacity, y1 =
prior_height, col = "black", lwd = 3, lty = 2)

# Adding a horizontal dashed line for the proper prior between 0 liters and 182 liters to create a
rectangle
segments(x0 = minimum_fuel_tank_capacity, y0 = prior_height, x1 = total_fuel_tank_capacity,
y1 = prior_height, col = "black", lwd = 3, lty = 2)

# Adding a legend to the plot to label the key elements including the likelihood function for each
seed, the proper prior, the posterior for each seed, the expected value (mean), the most likely
value (mode), and the fuel sensor standard deviation
legend("topright", legend = c("Likelihood (Seed 3)", "Likelihood (Seed 5)", "Likelihood (Seed
9)", "Proper Prior", "M-H Posterior (Seed 3)", "M-H Posterior (Seed 5)", "M-H Posterior (Seed
9)", "BMC Posterior (Seed 3)", "BMC Posterior (Seed 5)", "BMC Posterior (Seed 9)", "Expected
Value (Mean)", "Most Likely Value (Mode)", "Fuel Error SD"),
       col = c("darkblue", "blue", "lightblue", "black", "darkgreen", "green", "lightgreen","purple",
"violet", "magenta", "red", "orange", "darkgray"), lwd = c(4, 4, 4, 2, 4, 4, 4, 4, 4, 4, 3, 3, 3), lty =
c(1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 3, 3, 3), cex = 0.75, inset = 0.02)
#_____
__
#### PRINTING A TABLE DETAILING THE REQUIRED NUMBER OF RUNS NEEDED
FOR CONVERGENCE FOR THE METROPOLIS-HASTING ALGORITHM WITH THE
BAYES MONTE CARLO METHOD FOR EACH OF THE SEEDS ####

# Loading the gridExtra and grid libraries to produce a well-formatted table of the convergence
results
library(gridExtra)
library(grid)

# Opening a new blank page in the PDF file to print the table
plot.new()

# Stating the three fixed seeds for the table
seeds <- c(3, 5, 9)
```

```
# Specifying the iteration for convergence for each of the three seeds for the Metropolis-Hasting algorithm
metropolis_hasting_convergence <- c(552000, 446000, 695000)

# Specifying the iteration for convergence for each of the three seeds for the Bayes Monte Carlo method
bayes_monte_carlo_convergence <- c(202000, 279000, 148000)

# Creating a data frame with properly formatted column names
convergence_df <- data.frame(
  Seed = seeds,
  "M-H Required Iterations for Convergence" = metropolis_hasting_convergence,
  "BMC Required Iterations for Convergence" = bayes_monte_carlo_convergence,
  # Ensuring that the column names do not have periods in place of spaces between the words
  check.names = FALSE
)

# Adding a title to the top of this last page in the PDF file
grid.text("Table of Required Iterations for Convergence for Metropolis-Hastings and Bayes Monte Carlo",
          x = 0.5, y = 0.9, gp = gpar(fontsize = 11, fontface = "bold"))

# Printing the table onto the PDF page
grid.table(convergence_df)
#_____________________________________________________________________________
__
# Closing the PDF file
dev.off()
```