STAT 8310 Bayesian Data Analysis Final Project

Bayesian Analysis of Shopping Habits with Gamma-Poison Model

Department of Computer Science

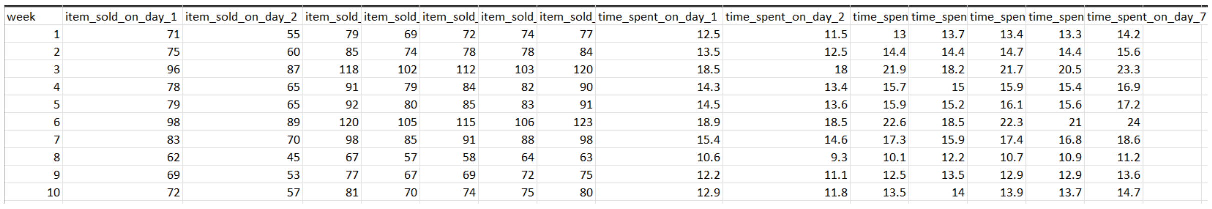
Abdul Afrid Mohammed, 002716253

Introduction:

This project aims to provide insights into the shopping habits of customers in a small business by utilizing Bayesian analysis and the Gamma-Poisson model. The objective is to estimate the parameters of the model and derive posterior distributions to understand the shopping patterns of customers on different days of the week.

The project uses the Metropolis-Hastings algorithm and the Gibbs sampling algorithm to select samples from the posterior distribution and report the posterior mean and 95% posterior interval for each variable. By analyzing the count data of items purchased by customers, this project intends to provide data-driven insights to small business owners, which could potentially inform their decision-making and improve their business strategies. The outcomes of this project could potentially help small business owners optimize their inventory, pricing, and promotional strategies to better cater to the shopping habits of their customers.

Dataset:



week	item_sold_on_day_1	item_sold_on_day_2	item_sold_	item_sold_	item_sold_	item_sold_	item_sold_	time_spent_on_day_1	time_spent_on_day_2	time_spen	time_spen	time_spen	time_spen	time_spent_on_day_7
1	71	55	79	69	72	74	77	12.5	11.5	13	13.7	13.4	13.3	14.2
2	75	60	85	74	78	78	84	13.5	12.5	14.4	14.4	14.7	14.4	15.6
3	96	87	118	102	112	103	120	18.5	18	21.9	18.2	21.7	20.5	23.3
4	78	65	91	79	84	82	90	14.3	13.4	15.7	15	15.9	15.4	16.9
5	79	65	92	80	85	83	91	14.5	13.6	15.9	15.2	16.1	15.6	17.2
6	98	89	120	105	115	106	123	18.9	18.5	22.6	18.5	22.3	21	24
7	83	70	98	85	91	88	98	15.4	14.6	17.3	15.9	17.4	16.8	18.6
8	62	45	67	57	58	64	63	10.6	9.3	10.1	12.2	10.7	10.9	11.2
9	69	53	77	67	69	72	75	12.2	11.1	12.5	13.5	12.9	12.9	13.6
10	72	57	81	70	74	75	80	12.9	11.8	13.5	14	13.9	13.7	14.7

My data has 52 weeks of data, where columns "item_sold_on_day1", ..,"item_sold_on_day 7", are the number of items sold in that week on day 1, day 2 till day 7. And the columns "time_spent_on_day1", .., and "time_spent_on_day7" is the average time spent by the customers to buy those products.

Now, I need the average time spent and the average number of products purchased for this project for each day of the week. I have calculated the average of products purchased and the time spent on each day of the week. The final data will be:

Day	1	2	3	4	5	6	7
Items sold	78	64	90	78	83	82	89
Time	14.1	13.2	15.4	14.9	15.6	15.2	16.6

Model:

I am building the Gamma-Poisson model for modeling the number of items purchased by customers in the store. The model assumes that the parameter of the Poisson distribution is proportional to the length of time the customer shops and uses Gamma distributions as priors for the parameters of the model.

The number of items purchased can be modeled by a Poisson distribution, with the parameter proportion to the length of time a customer shops:

$$y_j \sim Pois(\theta_j t_j)$$

In this case, appropriate model priors are:

$$\theta_j|\alpha, \beta \sim Gamma(\alpha, \beta)$$

$$\alpha \sim Exp(\lambda = 1)$$

$$\beta \sim Gamma(0.1, 1)$$

The procedure will go as follows:

1. Define priors: we are defining the prior distributions for the intercept parameter (alpha), the slope parameter (beta), and the error variance (sigma). The prior distribution for alpha is defined using the gamma distribution with shape parameter "a" and scale parameter "b". The prior distribution for beta is defined using the exponential distribution with the rate parameter "lambda". The prior distribution for sigma is defined using the gamma distribution.
2. Derive joint posteriors of $\theta 1, ..., \theta 7, \alpha, \beta|y$. and conditional posterior distributions for each parameter in the model.
   a. $\theta_j|\alpha, \beta, \mathbf{y}$
   b. $\alpha|\beta, \theta, \mathbf{y}$
   c. $\beta|\alpha, \theta, \mathbf{y}$
3. Define and use the Metropolis Hastings-within-Gibbs algorithm: This algorithm updates each parameter using its conditional posterior distribution. It starts with $\theta j = 0.1$, $\beta = 1$, and $\alpha = 1$ as the initial value, generates 10,000 samples for each parameter, calculates the posterior mean, and prints it. The posterior mean is a summary of the parameter's average value based on the posterior distribution. This helps understand the parameter's distribution and its relationship with the observed data, useful for making predictions.
4. Creating trace plots for all parameters and determining if there is evidence of convergence.
5. I will draw 10,000 samples from the conditional posterior distribution for each parameter and repeat this process each time using a different set of starting values for the parameters. The starting values for each iteration are provided:
   a) $\theta j = 5$, $\alpha = 0.5$, $\beta = 0.5$
   b) $\theta j = 1$, $\alpha = 10$, $\beta = 10$
   c) $\theta j = 0.5$, $\alpha = 1$, $\beta = 1$

d) θj = 0.1, α = 0.1, β = 0.1.

The goal of repeating the process with different starting values is to assess how sensitive the results are to the initial conditions. If the results are robust and consistent across different starting values, it provides evidence that the algorithm is converging and that the model is a good fit for the data.

6. Evaluating the convergence of each parameter by utilizing the Gelman-Rubin diagnostic across all five chains. The diagnostic compares the variance of the samples within each chain to the variance of the samples across all chains. If the chains have converged, the ratio of these variances should be close to 1.
7. To summarize the estimated posterior distribution of the model parameters, we can calculate the posterior mean and 95% posterior interval using the results from all five chains. After obtaining these samples, we can use them to make inferences about the data by estimating the posterior distribution of the model parameters.

Analysis:

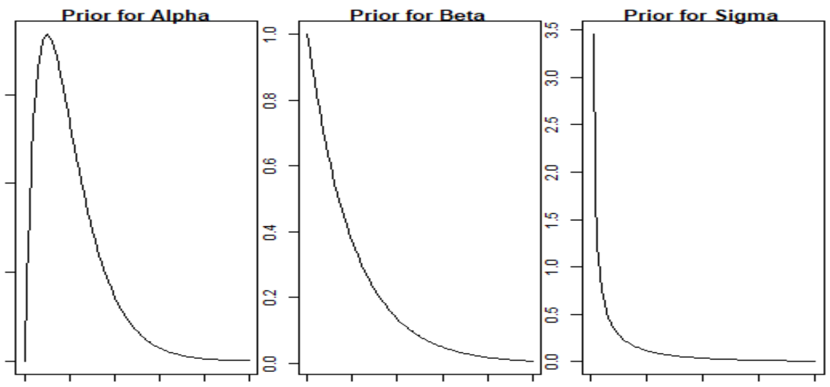
1.Initializing the parameters

```
# Define the hyperparameters for the priors

a <- 2  # Shape parameter for alpha gamma distribution
b <- 0.5   # Scale parameter for alpha gamma distribution
lambda <- 1   # Rate parameter for beta exponential distribution
```

2.Define the Prior and likelihood

```
# Define the prior distributions
prior_alpha <- function(alpha) {
  dgamma(alpha, shape = a, scale = b)
}
prior_beta <- function(beta) {
  dexp(beta, rate = lambda)
}
prior_sigma <- function(sigma) {
  dgamma(sigma, shape = 0.1, scale = 1)
}

likelihood <- function(alpha, beta, sigma, y, t) {
  lambda <- alpha * t
  log_likelihood <- sum(dpois(y, lambda, log = TRUE))
  return(exp(log_likelihood))
}
```

3.Plots of prior distributions.



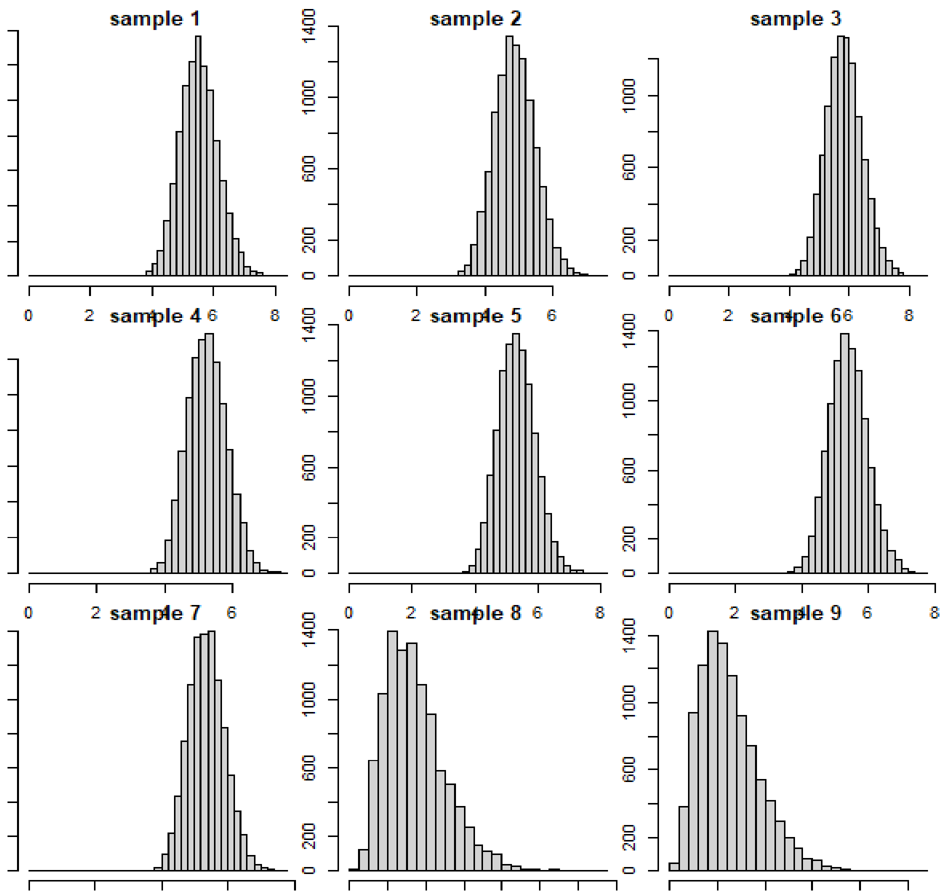
3.Define joint posterior and conditional posterior distributions.

```r
# Define the joint posterior
joint_posterior <- function(alpha, beta, sigma, y, t) {
  prior_alpha_dens <- prior_alpha(alpha)
  prior_beta_dens <- prior_beta(beta)
  prior_sigma_dens <- prior_sigma(sigma)
  lik <- likelihood(alpha, beta, sigma, y, t)

  posterior <- lik * prior_alpha_dens * prior_beta_dens * prior_sigma_dens
  return(posterior)
}
```

4.Define the Unnormalized posterior of alpha.

```r
alpha.posterior <- function(a, b, thetas) {
  beta <- b
  dens <- exp(-a)*prod(((thetas^(a-1))*(b^a))/gamma(a))
  return(dens)
}
```

5.Plots of Posterior distribution of the parameters.

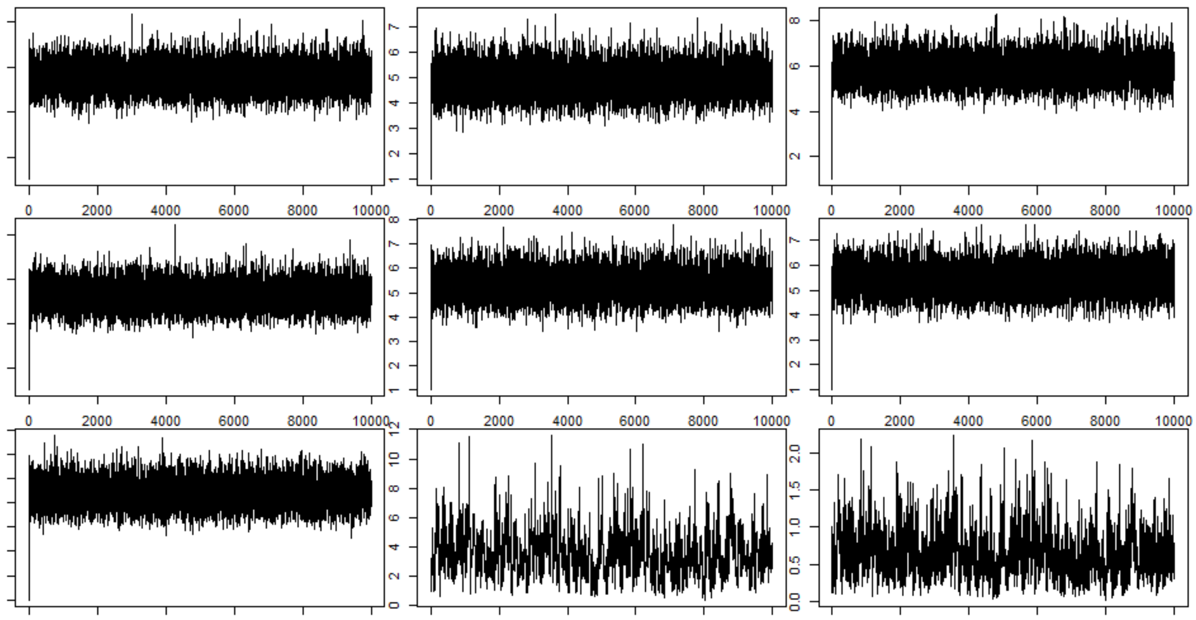


6.Gibbs Sampler

The function generates 10,000 samples for each parameter from its full conditional posterior distribution based on the input data and the initial parameter values. The resulting samples are stored in 'sample_1'.

```
> head(sample_1, n=10)
          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]      [,9]
 [1,] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.0000000
 [2,] 4.432527 4.932831 4.112466 4.429701 5.184998 4.745487 6.231658 1.000000 0.1337577
 [3,] 6.276944 3.678422 5.230651 4.978489 6.468001 5.044713 4.563526 1.000000 0.2247238
 [4,] 5.791018 5.089827 5.887235 6.125469 6.974283 4.586605 6.009500 1.000000 0.1115872
 [5,] 6.178565 5.327944 5.438261 6.472431 5.219561 5.051277 6.097653 1.218203 0.1528873
 [6,] 4.791671 4.482845 5.432845 6.064680 5.147743 4.769875 4.844312 1.218203 0.3198612
 [7,] 6.041704 5.198965 5.860107 5.674203 5.435757 4.704572 5.396713 1.218203 0.1485702
 [8,] 4.856473 5.555719 6.146682 5.732669 4.677517 5.964262 4.985784 1.218203 0.2106483
 [9,] 5.291454 5.152603 5.194107 5.149504 5.710822 6.026990 5.593145 1.218203 0.3454125
[10,] 5.233474 4.531070 5.393642 6.326204 5.057194 5.260027 5.251047 1.218203 0.2264850
```

7.Trace plots for all the parameters.

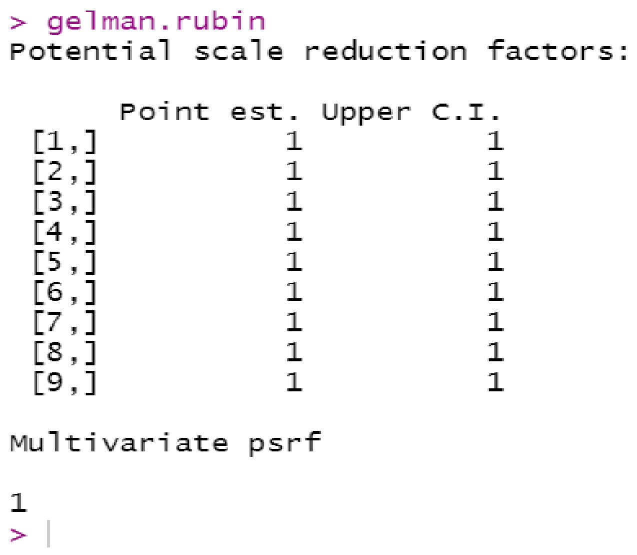


The trace plots indicate that all parameters have converged. A trace plot shows the values of a parameter over iterations. An ideal Markov Chain Monte Carlo algorithm should be irreducible, positive recurrent, and aperiodic, with a "hairy caterpillar" shape on the trace plot. Since there are no flat components in the plots, the sampled values appear to be randomly walking over their range, indicating convergence to their respective distributions.

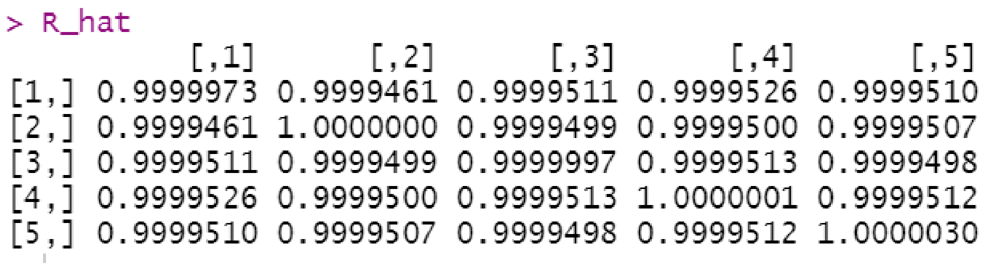
8.Gelman-Rubin Diagnosis:

The Gelman-Rubin diagnostic compares the variance between chains (W) to the variance within chains (B) to evaluate whether the chains have converged. It calculates a scale reduction factor (R_hat), which should be close to 1, to indicate convergence.

For each parameter specifically, the maximum upper bound of the psrf is 1.01, which is the case for ‘alpha’ and ‘beta’. The point estimate for α is also 1.01. We generally set a maximum threshold of 1.1 for individual psrf values, so, again, there is a strong case for convergence for each of the 9 parameters.



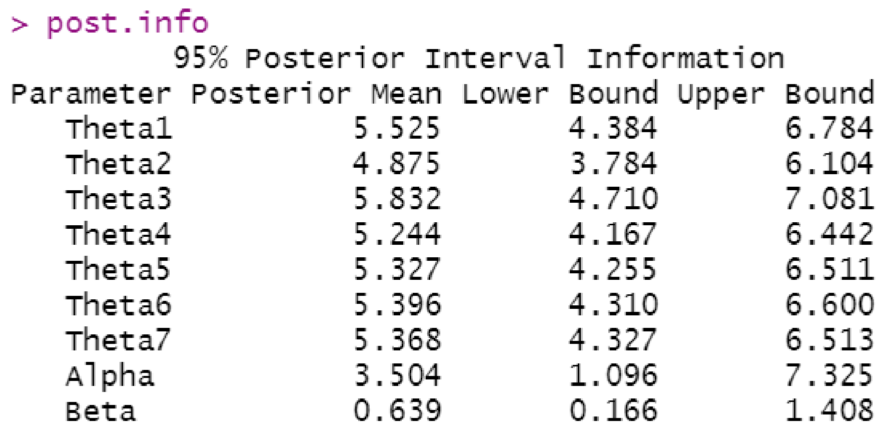
```
> gelman.rubin
Potential scale reduction factors:

      Point est. Upper C.I.
 [1,]          1          1
 [2,]          1          1
 [3,]          1          1
 [4,]          1          1
 [5,]          1          1
 [6,]          1          1
 [7,]          1          1
 [8,]          1          1
 [9,]          1          1

Multivariate psrf

1
> 
```



```
> R_hat
          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 0.9999973 0.9999461 0.9999511 0.9999526 0.9999510
[2,] 0.9999461 1.0000000 0.9999499 0.9999500 0.9999507
[3,] 0.9999511 0.9999499 0.9999997 0.9999513 0.9999498
[4,] 0.9999526 0.9999500 0.9999513 1.0000001 0.9999512
[5,] 0.9999510 0.9999507 0.9999498 0.9999512 1.0000030
```

The values close to 1 indicate that the chains have converged, and the posterior samples are mixing well.

9.Combining all the chains and summarizing the results



```
> post.info
         95% Posterior Interval Information
Parameter Posterior Mean Lower Bound Upper Bound
   Theta1          5.525       4.384       6.784
   Theta2          4.875       3.784       6.104
   Theta3          5.832       4.710       7.081
   Theta4          5.244       4.167       6.442
   Theta5          5.327       4.255       6.511
   Theta6          5.396       4.310       6.600
   Theta7          5.368       4.327       6.513
   Alpha           3.504       1.096       7.325
   Beta            0.639       0.166       1.408
```

The ‘Post.info’ matrix is containing the summary statistics of the posterior distribution of the parameters of interest.

The parameters of interest are:
α: the number of items a customer will purchase.
β: the mean waiting time until α items are purchased.
θ: the amount of time it will take a customer to purchase an item in the store.

The statistics indicate the posterior mean and 95% posterior interval for alpha, beta, and seven theta parameters (one for each day of the week) representing customer behavior. Alpha is estimated to be 3.504 with lower and upper bounds of 1.096 and 7.325, respectively. Beta is estimated to be 0.639 with lower and upper bounds of 0.166 and 1.408, respectively. The posterior mean and 95% posterior intervals are provided for each theta parameter, representing the amount of time it will take a customer to purchase an item in the store for each day of the week.

Conclusion:

Based on the above summary statistics, the average amount of time taken to purchase an item is 5 minutes and the average number of items a customer purchase is around 3. The time it takes to purchase an item does not change much for each day of the week which is shown by the overlapping intervals in the last two columns. However, I suggest the small business owner focus on new customers after 5-7 minutes since they are more likely to make a purchase than someone who has been in the store longer.

Code:

```
> library(MASS)
> y <- c(78, 64, 90, 78, 83, 82, 89)
> t <- c(14.1, 13.2, 15.4, 14.9, 15.6, 15.2, 16.6)
> t
[1] 14.1 13.2 15.4 14.9 15.6 15.2 16.6
> # Define the priors
> alpha <- 1
> beta <- 0.1
> sigma <- 1
> # Define the hyperparameters for the priors
> a <- 2  # Shape parameter for alpha gamma distribution
> b <- 0.5   # Scale parameter for alpha gamma distribution
> lambda <- 1   # Rate parameter for beta exponential distribution
> # Define the prior distributions
> prior_alpha <- function(alpha) {
+   dgamma(alpha, shape = a, scale = b)
+ }
> prior_beta <- function(beta) {
+   dexp(beta, rate = lambda)
+ }
> prior_sigma <- function(sigma) {
+   dgamma(sigma, shape = 0.1, scale = 1)
+ }
> likelihood <- function(alpha, beta, sigma, y, t) {
+   lambda <- alpha * t
+   log_likelihood <- sum(dpois(y, lambda, log = TRUE))
+   return(exp(log_likelihood))
+ }
> # Set up a range of values for each parameter
> alpha_seq <- seq(0, 5, length.out = 100)
> beta_seq <- seq(0, 5, length.out = 100)
> sigma_seq <- seq(0, 2, length.out = 100)
> # Compute the prior densities for each parameter
> prior_alpha_dens <- prior_alpha(alpha_seq)
> prior_beta_dens <- prior_beta(beta_seq)
> prior_sigma_dens <- prior_sigma(sigma_seq)
> # Plot the prior distributions
> par(mfrow = c(1, 3))
> plot(alpha_seq, prior_alpha_dens, type = "l", xlab = expression(alpha), ylab = "Density", main =
"Prior for Alpha")
> plot(beta_seq, prior_beta_dens, type = "l", xlab = expression(beta), ylab = "Density", main = "Prior
for Beta")
> plot(sigma_seq, prior_sigma_dens, type = "l", xlab = expression(sigma), ylab = "Density", main =
"Prior for Sigma")
> # Define the joint posterior
> joint_posterior <- function(alpha, beta, sigma, y, t) {
+   prior_alpha_dens <- prior_alpha(alpha)
+   prior_beta_dens <- prior_beta(beta)
+   prior_sigma_dens <- prior_sigma(sigma)
+   lik <- likelihood(alpha, beta, sigma, y, t)
+
+   posterior <- lik * prior_alpha_dens * prior_beta_dens * prior_sigma_dens
+   return(posterior)
+ }
> # Compute the conditional posterior distributions
> # Theta_j | alpha, beta, y
> theta_cond_posterior <- function(j, alpha, beta, sigma, y, t) {
+   prior_alpha_dens <- prior_alpha(alpha)
+   prior_beta_dens <- prior_beta(beta)
+   lik <- likelihood(alpha, beta, sigma, y, t)
+
+   marginal_sigma_dens <- integrate(function(log_sigma) {
+     sigma <- exp(log_sigma)
+     lik * prior_alpha_dens * prior_beta_dens * prior_sigma(sigma)
+   }, lower = -Inf, upper = Inf)$value
+
+   posterior_unnorm <- function(theta_j) {
+     theta <- c(alpha, beta, sigma)
+     theta[j] <- theta_j
+     lik_j <- likelihood(theta[1], theta[2], theta[3], y, t)
+     posterior <- lik_j * prior_alpha_dens * prior_beta_dens * prior_sigma(theta[3])
+     return(posterior)
+   }
+   theta_cond_posterior_dens <- function(theta_j) {
+     posterior_unnorm(theta_j) / marginal_sigma_dens
+   }
+
+   return(theta_cond_posterior_dens)
+ }
> # Alpha | beta, theta, y
> alpha_cond_posterior <- function(alpha, beta, sigma, y, t) {
+   prior_beta_dens <- prior_beta(beta)
+   lik <- likelihood(alpha, beta, sigma, y, t)
+   theta <- c(alpha, beta, sigma)
+
+   # Compute the normalizing constant (marginal likelihood) for sigma and alpha
+   marginal_sigma_alpha_dens <- integrate(function(log_sigma, alpha) {
+     sigma <- exp(log_sigma)
+     lik * prior_alpha(alpha) * prior_beta_dens * prior_sigma(sigma)
+   }, lower = -Inf, upper = Inf, alpha = alpha)$value
+
+   # Define the unnormalized posterior density for alpha
+   posterior_unnorm <- function(alpha) {
+     lik_alpha <- likelihood(alpha, theta[2], theta[3], y, t)
+     posterior <- lik_alpha * prior_alpha(alpha) * prior_beta_dens * prior_sigma(theta[3])
+     return(posterior)
+   }
+
+   # Compute the fully conditional posterior density for alpha
+   alpha_cond_posterior_dens <- function(alpha) {
+     posterior_unnorm(alpha) / marginal_sigma_alpha_dens
+   }
+
+   return(alpha_cond_posterior_dens)
+ }
> # Beta | alpha, theta, y
> beta_cond_posterior <- function(alpha, beta, sigma, y, t) {
+   prior_alpha_dens <- prior_alpha(alpha)
+   lik <- likelihood(alpha, beta, sigma, y, t)
+   theta <- c(alpha, beta, sigma)
+
+   # Compute the normalizing constant (marginal likelihood) for sigma and beta
+   marginal_sigma_beta_dens <- integrate(function(log_sigma, beta) {
+     sigma <- exp(log_sigma)
+     lik * prior_alpha_dens * prior_beta(beta) * prior_sigma(sigma)
+   }, lower = -Inf, upper = Inf, beta = beta)$value
+
+   # Define the unnormalized posterior density for beta
+   posterior_unnorm <- function(beta) {
+     lik_beta <- likelihood(theta[1], beta, theta[3], y, t)
+     posterior <- lik_beta * prior_alpha_dens * prior_beta(beta) * prior_sigma(theta[3])
+     return(posterior)
+   }
+
+   # Compute the fully conditional posterior density for beta
+   posterior_norm <- integrate(posterior_unnorm, lower = -Inf, upper = Inf)$value /
marginal_sigma_beta_dens
+   return(posterior_norm)
+ }
> # Define the unnormalized posterior of alpha
> alpha.posterior <- function(a, b, thetas) {# Data has thetas in column 1, betas in column 2
+   beta <- b
+   dens <- exp(-a)*prod(((thetas^(a-1))*(b^a))/gamma(a))
+   return(dens)
+ }
> #install.packages("truncnorm")
> library(truncnorm)
```

```
Warning message:
package ‘truncnorm’ was built under R version 4.2.3
>
> # Proposal distribution
> prop.dist.alpha <- function(a, prop.var) {
+   rtruncnorm(1, mean=a, sd=sqrt(prop.var), a=0)
+ }
> # Density of proposal
> prop.dist.alpha.dens <- function(a, a.mean, prop.var) {
+   dtruncnorm(a, mean = a.mean, sd=sqrt(prop.var), a=0)
+ }
> # Metropolis-Hastings Algorithm
> metrop <- function(param, thetas, b, alpha.posterior, prop.dist.alpha, prop.dist.alpha.dens,
prop.var, n.iter) {
+   # Store sampled alpha values
+   alphas <- c()
+   # Initialize model
+   param.t <- param
+   for(t in 1:n.iter) {
+     # Draw proposed value of alpha
+     param.new <- prop.dist.alpha(param.t, prop.var)
+     # Calculate acceptance probability
+     u <- runif(1, 0, 1)
+     prob.accept <- min(1, (alpha.posterior(param.new, b, thetas)*
+                            (prop.dist.alpha.dens(param.t, param.new, prop.var)))/(alpha.posterior(param.t, b,
thetas)
+                            *(prop.dist.alpha.dens(param.new, param.t, prop.var))))
+
+     if(u < prob.accept) {
+       value <- param.new
+     } else {
+       value <- param.t
+     }
+     alphas <- c(alphas, value)
+     param.t <- value
+   }
+
+   # Modification for MH-within-Gibbs sampling --> if only drawing one sample, return the sampled
value.
+   # If drawing multiple samples, return the list of all samples
+   if (length(alphas) == 1) {
+     return(alphas[1])
+   } else {
+     return(alphas)
+   }
+ }
>
> gibbs <- function(initial, y, t, n.iter) {
+   # Initialize variables
+   J <- length(y)
+   l <- length(initial)
+   results <- matrix(NA, n.iter, l)
+   results[1,] <- initial
+
+   for(i in 2:n.iter) {
+     thetas <- results[i-1, 1:7] # Stores all 7 theta_j values
+     a <- results[i-1,8]
+     b <- results[i-1,9]
+
+     # Draw theta_j samples
+     for(j in 1:J) {
+
+       # Find alpha, beta parameters for theta_j's gamma posterior distribution
+       alpha.theta <- y[j] + a
+       beta.theta <- t[j] + b
+
+       # Store singular theta_j sample using parameters calculated above
+       results[i,j] <- rgamma(1, alpha.theta, beta.theta)
+     }
+
+     # Find alpha, beta parameters for beta's gamma posterior distribution using theta sample
+     alpha.beta <- J*a + 0.1
+     beta.beta <- 1 + sum(results[i, 1:7])
+
+     # Store singular beta sample from its gamma posterior distribution
+     results[i, 9] <- rgamma(1, alpha.beta, beta.beta)
+
+     # Use Metropolis-Hastings algorithm from above to draw singular alpha sample
+     results[i, 8] <- metrop(a, results[i, 1:7], results[i, 9], alpha.posterior, prop.dist.alpha,
prop.dist.alpha.dens, 4, 1)
+   }
+   return(results)
+ }
>
> set.seed(5000) # Set seed for reproducability
>
>
> # Store prior data
> yhat <- c(78, 64, 90, 78, 83, 82, 89)
> t <- c(14.1, 13.2, 15.4, 14.9, 15.6, 15.2, 16.6)
>
>
> # Initialize first set of sampling values
> initial_1 <- c(rep(.1, 7), 1, 1)
> # Draw samples
> sample_1 <- gibbs(initial_1, yhat, t, 10000)
>
> par(mfrow=c(3,3))
> for (i in 1:9) {
+   hist(sample_1[,i], breaks= 30, main = paste0("sample ", i))
+ }
> means <- matrix(NA, 9, 1)
> for (i in 1:9) {
+   means[i,1] <- mean(sample_1[,i])
+ }
> rownames(means) <- c("Theta1", "Theta2", "Theta3", "Theta4", "Theta5","Theta6","Theta7",
"Alpha", "Beta")
> colnames(means) <- c("Posterior Mean")
> means <- as.table(means)
> means
       Posterior Mean
Theta1      5.5327379
Theta2      4.8806227
Theta3      5.8284910
Theta4      5.2315078
Theta5      5.3255287
Theta6      5.4011312
Theta7      5.3639737
Alpha       3.4411400
Beta        0.6263803
> head(sample_1, n=5)
         [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
 [1,] 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
 [2,] 5.122584 5.836837 5.819962 3.670906 4.813397 5.275224 5.263498
 [3,] 5.899408 4.894206 6.027335 4.676994 4.708066 5.026438 6.148848
 [4,] 3.968249 5.398127 5.513905 3.883546 5.367584 5.922404 6.444208
 [5,] 5.464872 4.715590 5.904113 6.128883 6.301530 5.854783 5.576875
         [,8]      [,9]
 [1,] 1.000000 1.0000000
 [2,] 1.000000 0.1278500
 [3,] 1.000000 0.2144906
 [4,] 1.000000 0.1061247
 [5,] 1.713104 0.2241926
> par(mfrow = c(3, 3))
> par(mar = c(1, 1, 1, 1))
> for (i in 1:9) {
+   plot(sample_1[,i], type="l")
+ }
> # Set initial values and run MCMC
> initial_1 <- c(1, 1, 1, 1, 1, 1, 1, 1, 1)
> sample_1 <- gibbs(initial_1, yhat, t, 10000
> initial_2 <- c(rep(5, 7), 0.5, 0.5)
> sample_2 <- gibbs(initial_2, yhat, t, 10000)
> initial_3 <- c(rep(1, 7), 10, 10)
> sample_3 <- gibbs(initial_3, yhat, t, 10000)
> initial_4 <- c(rep(0, 7), 1, 1)
> sample_4 <- gibbs(initial_4, yhat, t, 10000)
> initial_5 <- c(rep(0, 7), 0.1, 0.1)
> sample_5 <- gibbs(initial_5, yhat, t, 10000)
> # Store MCMC chain
> chains <- list(sample_1, sample_2, sample_3, sample_4, sample_5)
> # Run Gelman-Rubin diagnostic
> n_chains <- length(chains)
> n_samples <- nrow(chains[[1]])
> n_params <- ncol(chains[[1]])
> B <- n_samples * var(sapply(chains, function(chain) rowMeans(chain)))
> W <- mean(sapply(chains, function(chain) var(chain))) * n_samples
> var_hat <- ((n_samples - 1) / n_samples) * W + (1 / n_samples) * B
> R_hat <- sqrt(var_hat / W)
> R_hat
          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 0.9999973 0.9999461 0.9999511 0.9999526 0.9999510
[2,] 0.9999461 1.0000000 0.9999499 0.9999500 0.9999507
[3,] 0.9999511 0.9999499 0.9999997 0.9999513 0.9999498
[4,] 0.9999526 0.9999500 0.9999513 1.0000001 0.9999512
[5,] 0.9999510 0.9999507 0.9999498 0.9999512 1.0000030
> #install.packages("MCMCpack")
> library(MCMCpack)
> chain_1 <- mcmc(sample_1)
> chain_2 <- mcmc(sample_2)
> chain_3 <- mcmc(sample_3)
> chain_4 <- mcmc(sample_4)
> chain_5 <- mcmc(sample_5)
> # Store MCMC chain
> combined.chains <- mcmc.list(chain_1, chain_2, chain_3, chain_4, chain_5)
> # Run Gelman-Rubin diagnostic
> gelman.rubin <- gelman.diag(combined.chains)
> gelman.rubin
Potential scale reduction factors:

      Point est. Upper C.I.
 [1,]          1          1
 [2,]          1          1
 [3,]          1          1
 [4,]          1          1
 [5,]          1          1
 [6,]          1          1
 [7,]          1          1
 [8,]          1          1
 [9,]          1          1

Multivariate psrf

1
>
> # Combine results from all 5 samples
> total.samples <- rbind(sample_1, sample_2, sample_3, sample_4, sample_5)
> post.info <- matrix(NA, 9, 3)
> for (i in 1:9) {
+   post.info[i, 1] <- round(mean(total.samples[,i]), 3)
+   post.info[i, 2] <- round(quantile(total.samples[,i], 0.025), 3)
+   post.info[i, 3] <- round(quantile(total.samples[,i], 0.975), 3)
+ }
> rownames(post.info) <- c("Theta1", "Theta2", "Theta3",
+                       "Theta4", "Theta5","Theta6","Theta7", "Alpha", "Beta")
> colnames(post.info) <- c("Posterior Mean", "Lower Bound", "Upper Bound")
> post.info <- as.table(post.info)
> names(dimnames(post.info)) <- c("Parameter", "95% Posterior Interval Information")
> post.info
        95% Posterior Interval Information
Parameter Posterior Mean Lower Bound Upper Bound
   Theta1          5.525       4.384       6.784
   Theta2          4.875       3.784       6.104
   Theta3          5.832       4.710       7.081
   Theta4          5.244       4.167       6.442
   Theta5          5.327       4.255       6.511
   Theta6          5.396       4.310       6.600
   Theta7          5.368       4.327       6.513
   Alpha           3.504       1.096       7.325
   Beta            0.639       0.166       1.408
>
```