Assignment 8

Sam Olson

Problem Description

Consider a problem of conducting a Bayesian analysis with a one-sample gamma model. Assume that random variables $Y_1, \ldots, Y_n$ are independent and identically distributed with common probability density function (for $\alpha > 0$ and $\beta > 0$):

$$f(y \mid \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} y^{\alpha-1} \exp(-\beta y), \quad y > 0.$$

Suppose that we will assign a joint prior to α and β in product form, with particular values of $A > 0$, $\gamma_0 > 0$, and $\lambda_0 > 0$:

$$\pi_\alpha(\alpha) = \frac{1}{A} I(0 < \alpha < A), \quad \pi_\beta(\beta) = \frac{\lambda_0^{\gamma_0}}{\Gamma(\gamma_0)} \beta^{\gamma_0 - 1} e^{-\lambda_0 \beta}, \quad \beta > 0.$$

Recall that in the analysis of an actual data set, A, γ_0 and λ_0 will be given specific numerical values. Since this is a simulated example and we have no actual prior information, use the following hyperparameters:

$$A = 20, \quad \gamma_0 = 0.5, \quad \lambda_0 = 0.1.$$

This gives prior expectation of 5.0 and prior variance of 50. The prior does focus probability on smaller values, but still has $Pr(\beta > 10) = 0.16$.

1.

Consider using a Metropolis–Hastings algorithm with independent random-walk proposals for α and β.

Suppose that our current values are (α_m, β_m), and that the proposal (α^*, β^*) has been generated from

$$q(\alpha, \beta \mid \alpha_m, \beta_m)$$

which is the product of independent random walks.

Identify the appropriate acceptance probability for the jump proposal (α^*, β^*).

Answer

I believe this question is being asked in the abstract, i.e., not for the specific dataset in question. Under that pretense:

The target distribution for the Metropolis–Hastings algorithm is the joint posterior:

$$\pi(\alpha, \beta \mid y) \propto L(y \mid \alpha, \beta)\, \pi_\alpha(\alpha)\, \pi_\beta(\beta)$$

where L denotes the likelihood (not log-transformed), and with the likelihood for independent $Y_i \sim \text{Gamma}(\alpha, \beta)$ given by:

$$L(y \mid \alpha, \beta) = \prod_{i=1}^{n} \frac{\beta^\alpha}{\Gamma(\alpha)} y_i^{\alpha-1} e^{-\beta y_i}$$

If the proposal distribution is a product of independent random walks, then it must satisfy the relation:

$$q(\alpha^*, \beta^* \mid \alpha_m, \beta_m) = q_\alpha(\alpha^* \mid \alpha_m) q_\beta(\beta^* \mid \beta_m)$$

The acceptance probability is therefore

$$a\big((\alpha_m, \beta_m) \to (\alpha^*, \beta^*)\big) = \min\left\{1,\ \frac{\pi(\alpha^*, \beta^* \mid y)}{\pi(\alpha_m, \beta_m \mid y)}\right\} = \min\left\{1,\ \frac{L(y \mid \alpha^*, \beta^*)\, \pi_\alpha(\alpha^*)\, \pi_\beta(\beta^*)}{L(y \mid \alpha_m, \beta_m)\, \pi_\alpha(\alpha_m)\, \pi_\beta(\beta_m)}\right\}$$

Because $\pi_\alpha(\alpha)$ is uniform on $(0, A)$, this implies that any proposal with $\alpha^* \notin (0, A)$ or $\beta^* \leq 0$ is automatically rejected ($a = 0$). However, for our purposes we would want to use the log-likelihoods ℓ, as it will enforce positive values for α, β respectively (and generally seems to "stabilize" the sampling method).

Knowing the data model comes from a one-sample Gamma, we know it comes from the exponential dispersion family. This is used primarily to motivate knowing the form of the sufficient statistics, $S_1 = \sum_{i=1}^{n} \log y_i$ and $S_2 = \sum_{i=1}^{n} y_i$. With this, we can write:

$$\log r = n\left[\alpha^* \log \beta^* - \log \Gamma(\alpha^*) - \alpha_m \log \beta_m + \log \Gamma(\alpha_m)\right] + (\alpha^* - \alpha_m) S_1 - (\beta^* - \beta_m) S_2 + (\gamma_0 - 1)\left[\log \beta^* - \log \beta_m\right] - \lambda_0(\beta^* - \beta_m)$$

Such that we have the acceptance probability for the Metropolis–Hastings update of (α, β) (exactly) as:

$$a = \min\{1,\ \exp(\log r)\}$$

2.

On the course web page is a data set called `gammadat_bayes.txt`.

Program a Metropolis–Hastings algorithm and simulate 50,000 values from the joint posterior of α, β, and $\mu = \alpha/\beta$.

Provide (with supporting evidence if appropriate):

- Information on how you selected a burn-in period. *NOTE: I do not expect you to compute Gelman-Rubin scale reduction factors for this assignment.*
- Information on how you tuned the algorithm for acceptance rate, including the random-walk variances and the final acceptance rate.
- Summaries of the marginal posterior distributions of α and β and $\mu = \alpha/\beta$, including histograms and five-number summaries, 95% central credible intervals, and correlation between α and β in the Markov chain.

Answer

To start, while not affecting the priors (in distribution or parameter value), we do want to look at the data to get our "initial" values.

Based on method of moments estimators, we have:

$\alpha_0 = \frac{\bar{y}}{s_y^2}, \beta_0 = \frac{\alpha_0}{\bar{y}}$

Where $\bar{y}$ and s_y^2 are the sample mean and variance, respectively.

For both α_0 and β_0, we will "impose" that any estimates or updates retain $\alpha_0, \beta_0 > 0$.

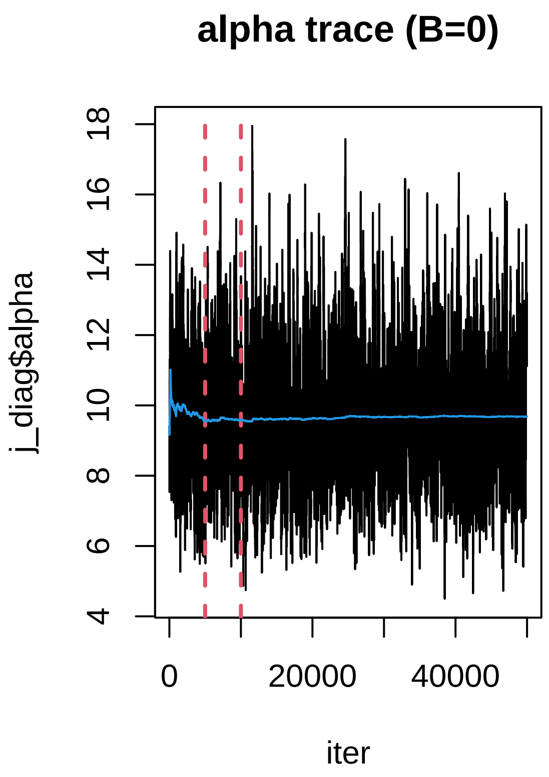
Additionally, given the hyperparameters to this problem, we also "impose" that $\alpha_0 < 20$.

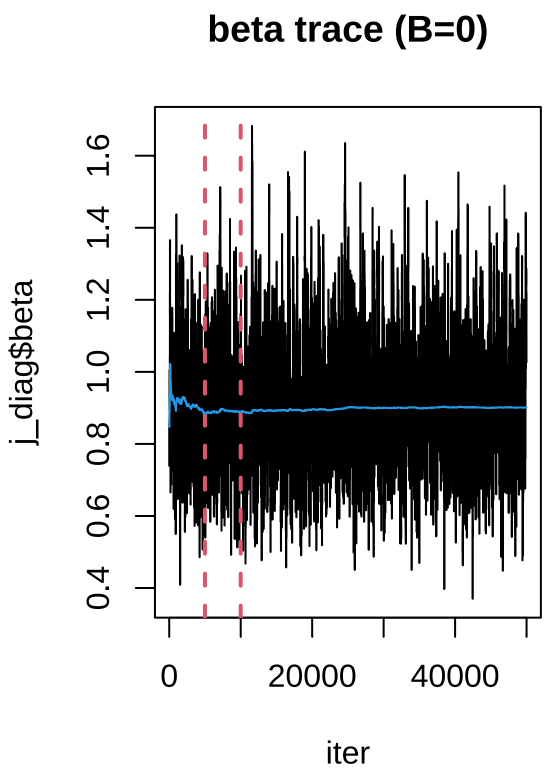
We run a sampling procedure without any tuning, first to determine a suitable Burn-In period.

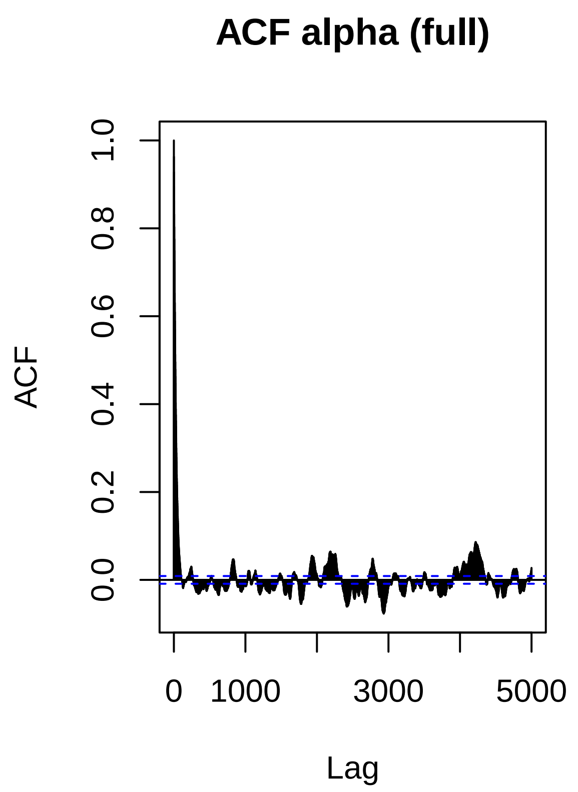
Note: Throughout the sampling procedure, we simulate α and β simultaneously, which is important to keep later estimates of μ more comparable ("apples-to-apples"). Furthermore, the actual sampling is done using log likelihood, again, for the purpose of stability. However, we built-in a check for "invalid" parameter proposals (negative-valued); during the procedure, after identifying a "candidate", if the proposed parameter is negative-valued, it is "reverted" to the prior value. This ensures appropriateness of the method.

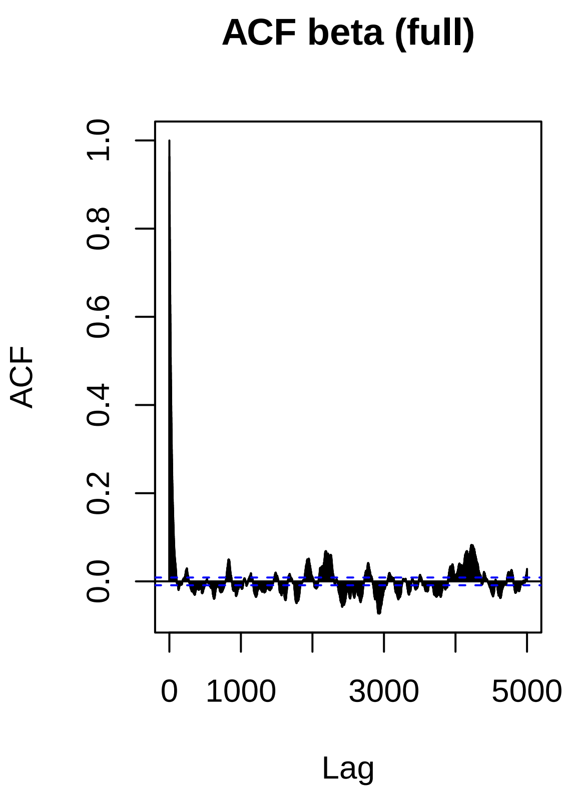
We also consider a handful of starting values for α, β to ensure the trace plots achieve stability around the same number of iterations.

Our sample values provide a basis for determining our Burn-In Period. We use our Metropolis-Hastings algorithm for a rather large number (50,000) without *any* Burn-In, and then look at both the trace plots and autocorrelation plots (ACFs). With that, we have:


alpha trace (B=0)
j_diag$alpha
18
16
14
12
10
8
6
4
0
20000
40000
iter


beta trace (B=0)
j_diag$beta
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
20000
40000
iter





To that end, we use a "running mean" (blue) to get a sense of when variability in the parameter "stabilizes" in the Trace Plot graphs. We see that by iteration 5,000, α and β samples tend to stabilize. Additionally, the ACFs decay rapidly (well before the 5,000); to stay on the conservative side, we double this and set that equal to our Burn-In period.

Ultimately, we settle on a Burn-In period of 10,000.

We then have some additional tuning. The general procedure is to "scale" the jump variance by some proportion of the original sample variance. To begin with, we settle on multipling the variance by some proportion, initially around 0.25 to 2, and then after iterating deciding on 0.5, 0.75, 1.0, 1.25, and 1.5. The acceptance probabilities for each of these scales are reported, using the Burn-In of 10,000.

scale	valpha	vbeta	accept
0.50	0.8473	0.0073	0.4264
0.75	1.9063	0.0165	0.2917
1.00	3.3891	0.0293	0.2166
1.25	5.2954	0.0458	0.1677
1.50	7.6254	0.0660	0.1338

Our goal in tuning is to reach a Metropolis-Hastings acceptance rate somewhere in the range of 20-60%, which is a heuristic noted in the Chapter 7 notes on Simulation. Ultimately, we decided on using a "scale" of 1.25, which given the seed and method noted, gives us an acceptance probability around 16.5%. Note: While this is on the lower end, I ultimately wanted to scale the jump variance somewhat higher than the sample to allow for greater exploration of a wider range of possible values for the "jumps".

This then leads us to run the whole simulation procedure again, now using both the Burn-In period of 5,000 and the tuned parameters.

Table 2: Posterior summaries: five-number statistics and 95 central credible intervals

		Posterior Summary						
Method	Parameter	Min	Q1	Median	Q3	Max	CI.2.5	CI.97.5
MH	alpha	4.4454	8.2681	9.3900	10.6457	17.4904	6.3353	13.3748
MH	beta	0.3904	0.7641	0.8763	0.9926	1.6698	0.5804	1.2527
MH	mu	8.8090	10.4279	10.7508	11.0919	13.2251	9.8048	11.8102

Table 3: 95 central credible intervals

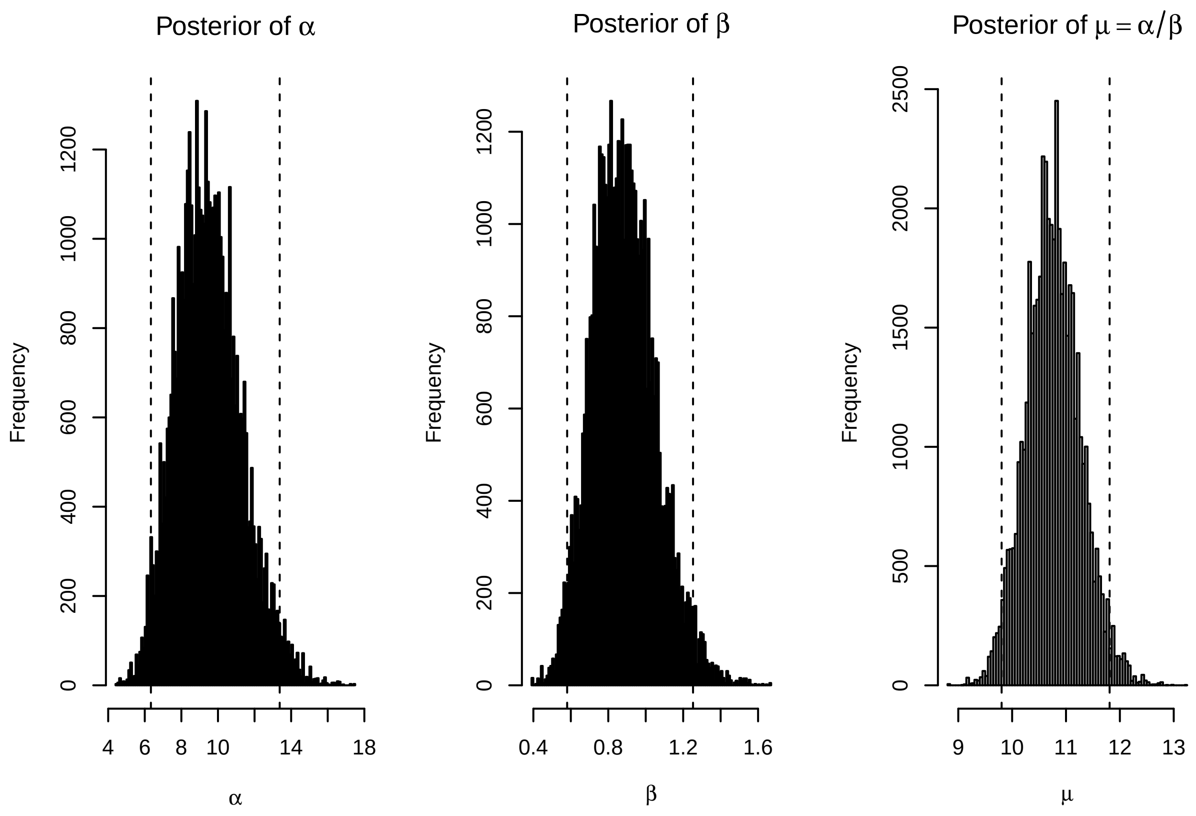
Method	Parameter	Lower 2.5	Upper 97.5
MH	alpha	6.3353	13.3748
MH	beta	0.5804	1.2527
MH	mu	9.8048	11.8102

Table 4: Acceptance rates

Component	Acceptance.Rate
Overall	16.8%

Table 5: Correlation matrix for alpha and beta

Parameter	alpha	beta
alpha	1.0000	0.9713
beta	0.9713	1.0000


Posterior of α
Posterior of β
Posterior of μ = α/β
Frequency
α
β
μ

3.

Using both the 75th percentile and the range as data characteristics of potential interest, compute posterior predictive p-values from 10,000 posterior predictive datasets.

Answer

We then do additional posterior predictive checks to validate the results of our simulation. Using the kept MH draws from the prior Question, we simulated 10,000 posterior predictive datasets.

Table 6: Posterior predictive p-values (upper tail) from 10,000 replicates

Statistic	Observed	PPP
75th percentile	12.9317	0.4387
Range	14.9500	0.5707

Generally, we want posterior predictive p-values that are not too large and not too small (so somewhere in the range of 0.2 to 0.7, or so); the values we observe for the statistics of interest seem suitable.

4.

Now consider the use of a Gibbs Sampling algorithm to simulate from the joint posterior of α and β and μ. Derive full conditional posterior densities for α and β Using these distributions, program a Gibbs Sampling algorithm and simulate 50,000 values from the joint posterior. Provide (with supporting evidence if appropriate),

- information on how you selected a burn-in period. Again, there is no need to compute Gelman-Rubin scale reduction factors for this assignment.
- summaries of the marginal posterior distributions of α and β, including histograms and five-number summaries, 95% central credible intervals, and correlation between α and β in the Markov chain.

Answer

Let $Y_i \overset{\text{iid}}{\sim} \text{Gamma}(\alpha, \beta)$ with density

$f(y \mid \alpha, \beta) = \dfrac{\beta^\alpha}{\Gamma(\alpha)}, y^{\alpha-1} e^{-\beta y}$ for $y > 0$.

Priors: $\alpha \sim \text{Uniform}(0, A)$ and $\beta \sim \text{Gamma}(\gamma_0, \lambda_0)$ in using the rate parametrization.

Denote $S_1 = \sum_{i=1}^n \log y_i$ and $S_2 = \sum_{i=1}^n y_i$.

The joint posterior (up to a constant) is

$$\pi(\alpha, \beta \mid y) \propto \beta^{n\alpha} e^{-\beta S_2} \frac{e^{(\alpha-1)S_1}}{[\Gamma(\alpha)]^n} \beta^{\gamma_0 - 1} e^{-\lambda_0 \beta} \mathbb{1}_{(0,A)}(\alpha)$$

Collecting terms in β gives the kernel

$$\beta^{n\alpha + \gamma_0 - 1} \exp\big(-(\lambda_0 + S_2)\beta\big)$$

so

$$\beta \mid \alpha, y \ \sim\ \text{Gamma}\,(\ \gamma_0 + n\alpha,\ \lambda_0 + S_2\ )$$

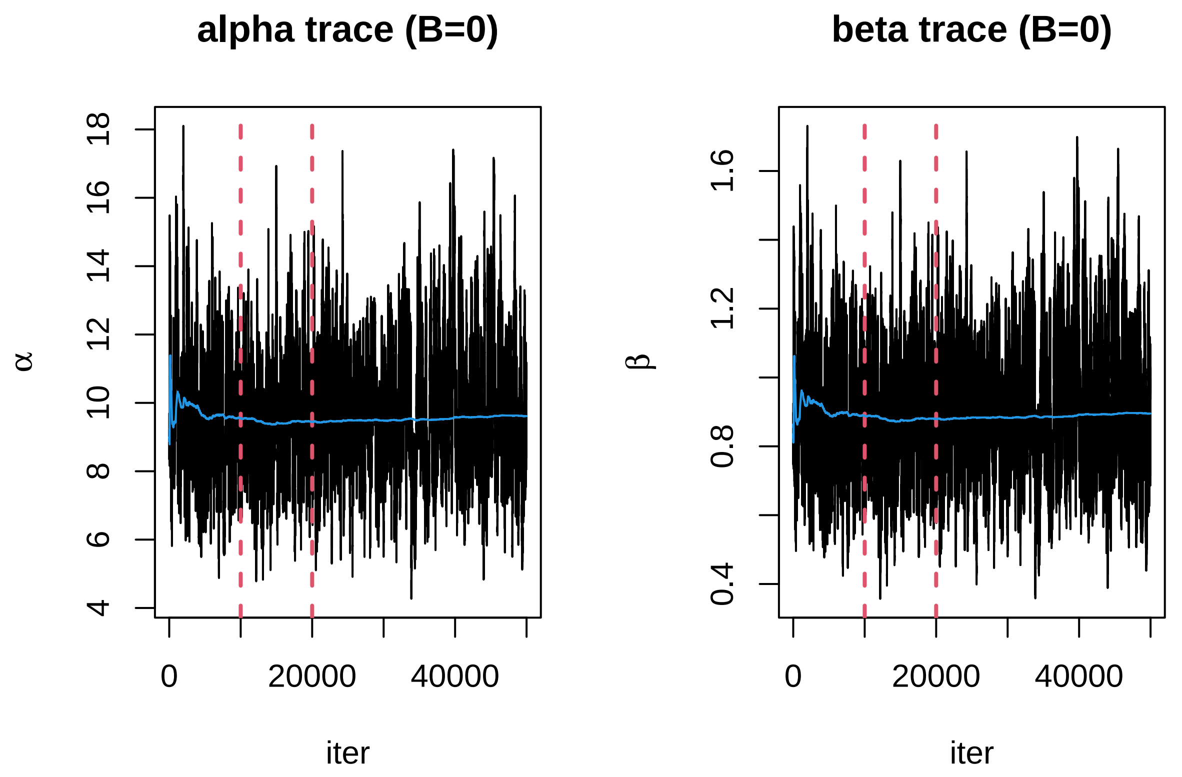
Collecting terms in α yields

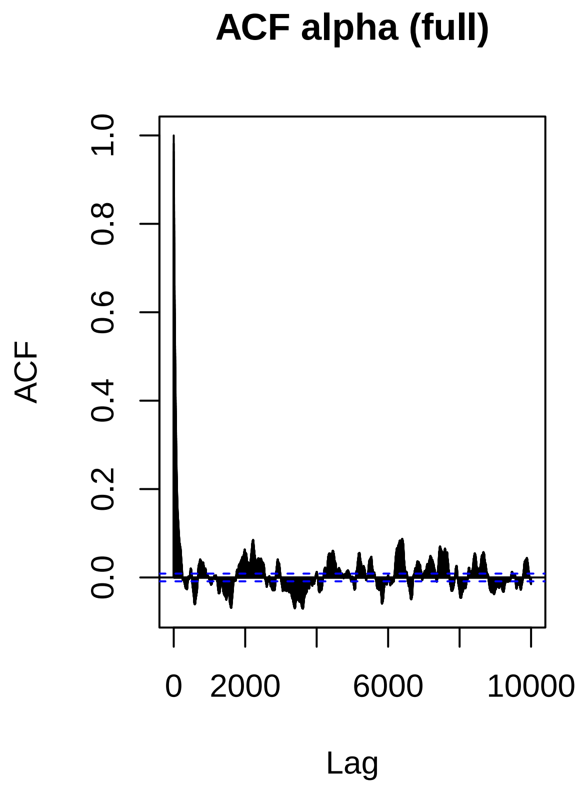
$$\pi(\alpha \mid \beta, y)\ \propto \exp\Big(n\alpha \log \beta - n \log \Gamma(\alpha) + (\alpha - 1)S_1\Big)\mathbf{1}_{(0,A)}(\alpha)$$

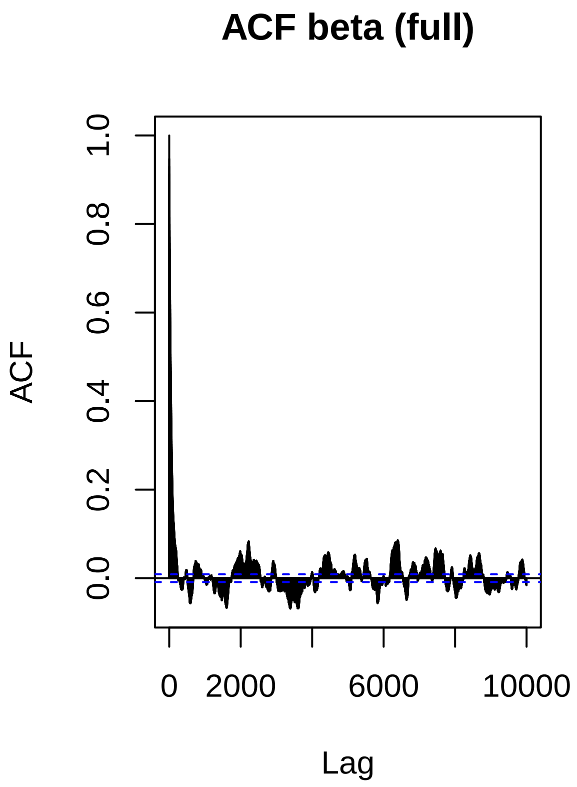
which is not a standard family (because of $\log \Gamma(\alpha)$). Therefore α is updated by a random-walk Metropolis step inside the Gibbs sampler (Metropolis-within-Gibbs), with proposals constrained to $(0, A)$.

Now, with that proof out the way, we can begin assessing a suitable Burn-In and tuning (which, given the priors, will be done solely on α given conditional conjugacy).

Similar to the MH method, we start by looking at trace plots and ACF plots with zero burn-in. Again, the fine details of the method are noted in the Appendix, but primarily focus on simulating α and β simmultaneously, and working with the log-likelihood.


alpha trace (B=0)
beta trace (B=0)
α
β
iter
iter
0
20000
40000
4
6
8
10
12
14
16
18
0.4
0.8
1.2
1.6





Similar to Question 2, we use a "running mean" (blue) to get a sense of when variability in the parameter "stabilizes". And again, similar (but different!) to Question 2, by iteration 10,000, the ACFs has decayed rapidly, and the trace plot has generally stabilized; to stay on the conservative side, we double this and set that equal to our Burn-In period, settling on a Burn-In period of 20,000.

For tuning then, we again scale our α variance in increments of 0.25, considering the same range of scales, and targeting an acceptance probability around 20-60%. However, given this is using the Gibbs sampling method, I am cautious about not properly "mixing", so apriori I will prioritize lower acceptance proabilities (we will accept more sampling that don't make a "jump", where by comparisons the MH algorithm will actively discard unsuitable "jump" candidates).

scale	valpha	acc_alpha
0.50	0.4766	0.5625
0.75	1.0723	0.4366
1.00	1.9063	0.3539
1.25	2.9787	0.2896
1.50	4.2893	0.2483
1.75	5.8382	0.2168
2.00	7.6254	0.1909

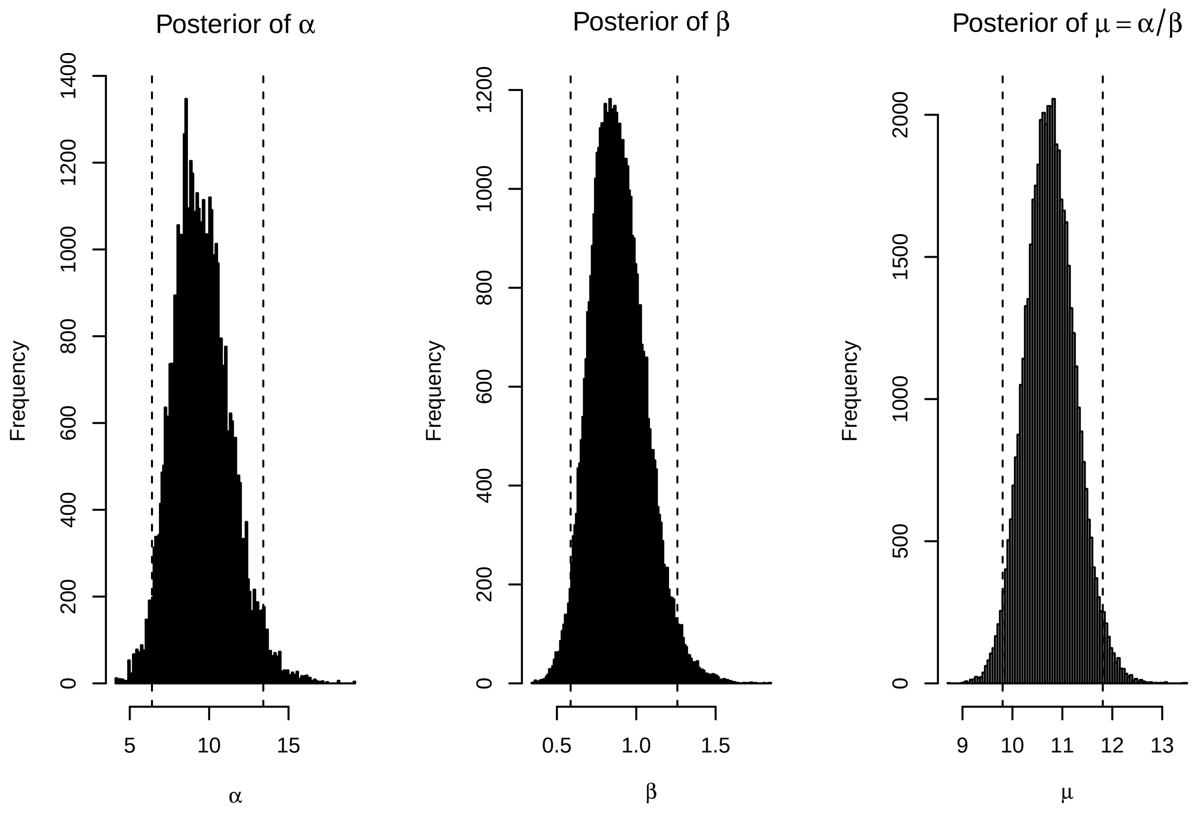
Given the scaling of 2.0, with acceptance probability around 19.1%, we now have both a suitable Burn-In Period and tuned parameters for our "full simulation" via Gibbs.

Table 8: Posterior summaries for alpha and beta: five-number statistics and 95 central credible intervals

	Posterior Summary						
Parameter	Min	Q1	Median	Q3	Max	CI 2.5\%	CI 97.5\%
alpha	4.1562	8.2506	9.3828	10.6414	19.1854	6.3985	13.4101
beta	0.3475	0.7650	0.8738	0.9948	1.8442	0.5866	1.2588

Table 9: Correlation matrix for alpha and beta (from Markov chain)

Parameter	alpha	beta
alpha	1.0000	0.9719
beta	0.9719	1.0000



5.

Using both the 75th percentile and the range as data characteristics of potential interest, compute posterior predictive p-values from 10,000 simulated posterior predictive data sets.

Answer

We then do additional posterior predictive checks to validate the results of our simulation. Using the Gibbs draws from the prior Question, we simulated 10,000 posterior predictive datasets.

Table 10: Posterior predictive p-values (upper tail) from 10,000 replicated datasets

	Posterior Predictive Check	
Statistic	Observed	PPP
75th percentile	12.9317	0.4416
Range	14.9500	0.5720

Generally, we want posterior predictive p-values that are not too large and not too small (so somewhere in the range of 0.2 to 0.7, or so); the values we observe for the statistics of interest seem suitable, though potentially a bit large. Further comparison and analysis is discussed in Question 6.

6.

Compare your results from the use of Metropolis-Hastings and Gibbs Sampling.

Answer

The two sampling approaches—Metropolis–Hastings (MH) and Gibbs with a Metropolis-within-Gibbs update—produce highly consistent posterior inferences for all parameters of interest. Both recover nearly identical posterior locations and spreads for α, β, and the derived parameter $\mu = \alpha/\beta$.

For the MH sampler, the 95% central credible intervals were

$$\alpha \in [6.32,\ 13.72], \quad \beta \in [0.578,\ 1.294], \quad \mu \in [9.823,\ 11.788],$$

while the Gibbs sampler yielded

$$\alpha \in [6.40,\ 13.41], \quad \beta \in [0.587,\ 1.259], \quad \mu \in [9.810,\ 11.792].$$

These results indicate essentially identical central tendencies and dispersion, showing that both samplers successfully explored the same posterior region.

The pairwise posterior correlation between α and β is also very similar—approximately (0.97) for both methods—which is expected since the parameters are not independent: μ is well identified even though α and β are highly positively associated.

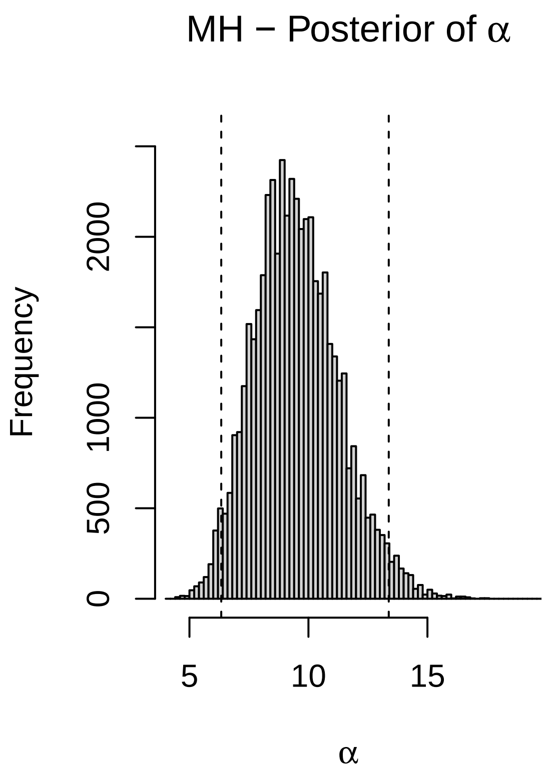
From a qualitative perspective, the MH posterior histograms appear slightly more "jagged," reflecting that its random-walk updates move both parameters jointly and can yield higher autocorrelation. The Gibbs sampler, which draws β directly from its conjugate full conditional and updates α by a univariate Metropolis step, produces smoother histograms and traces due to its more localized updates and higher effective sample size per iteration.

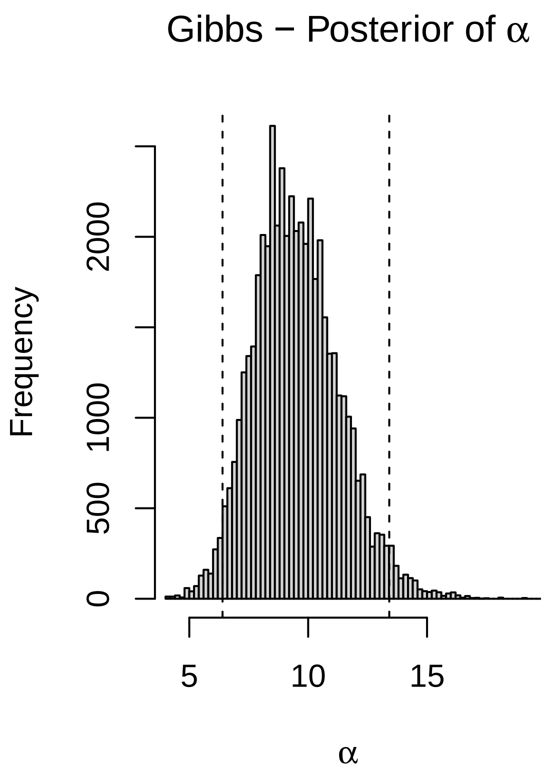
Posterior predictive checks (PPCs) are also consistent across methods. Both yield one-sided posterior predictive p-values between approximately 0.4 and 0.6 for both the 75th percentile and range statistics:

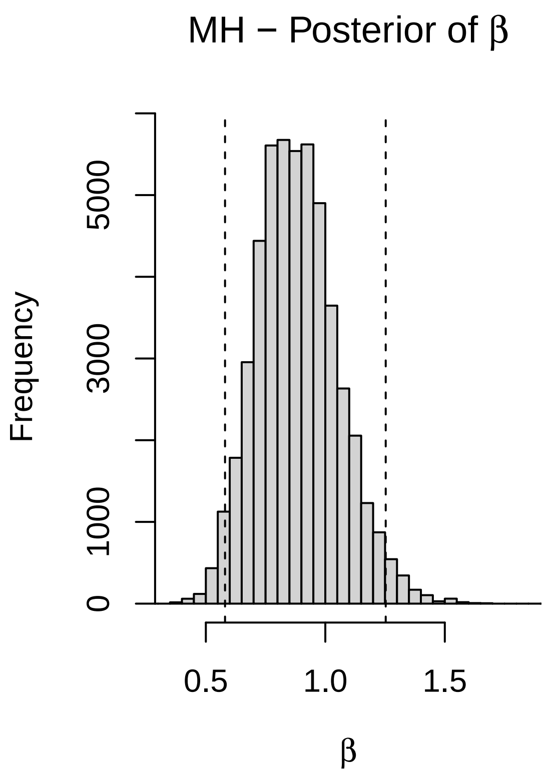
$$p_{\text{ppp}}^{(Q_{0.75})} \approx 0.44, \quad p_{\text{ppp}}^{(\text{range})} \approx 0.57$$

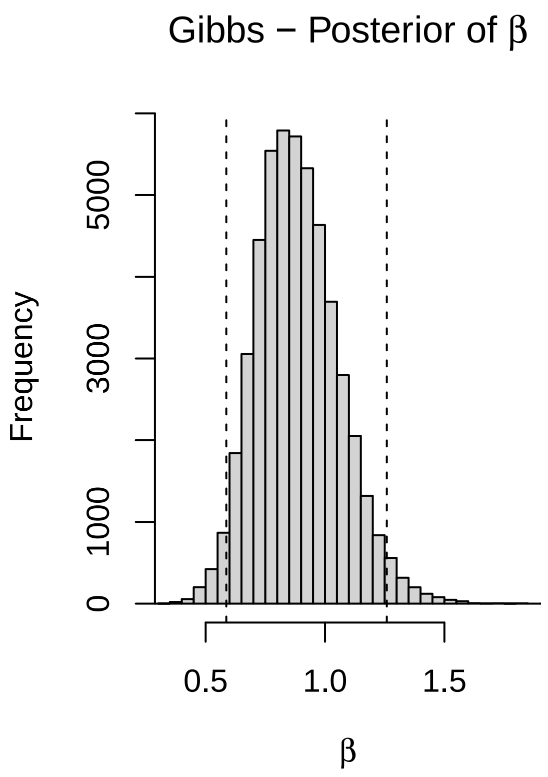
These are comfortably midrange, indicating that simulated datasets from each posterior reproduce the empirical features of the data without systematic bias (values near 0 or 1 would imply under- or over-dispersion).

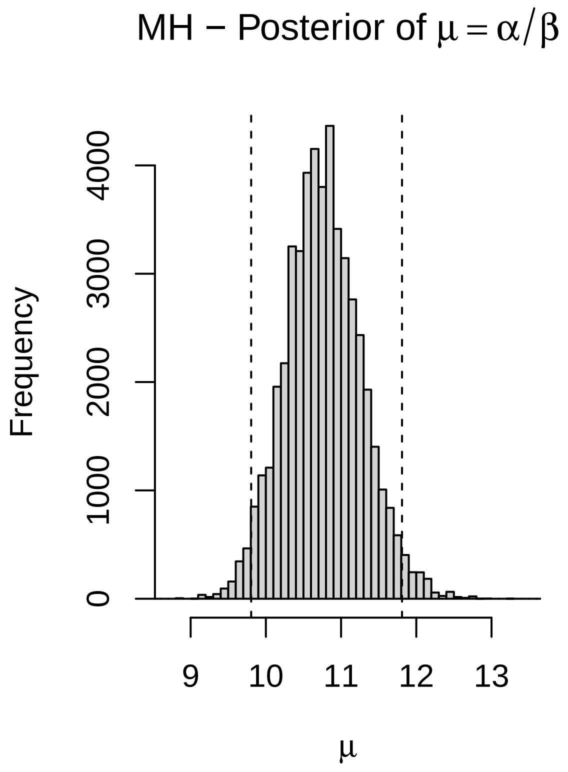
In short, both algorithms converge to effectively the same posterior. The MH method shows slightly rougher marginal behavior but comparable central estimates, while the Gibbs sampler—with its conjugate β-updates and well-tuned step for α—achieves smoother mixing and a visually steadier chain. After conservative burn-in and careful tuning in both cases, the two methods give nearly indistinguishable posterior summaries and predictive performance, confirming that each appropriately captures the underlying posterior distribution.

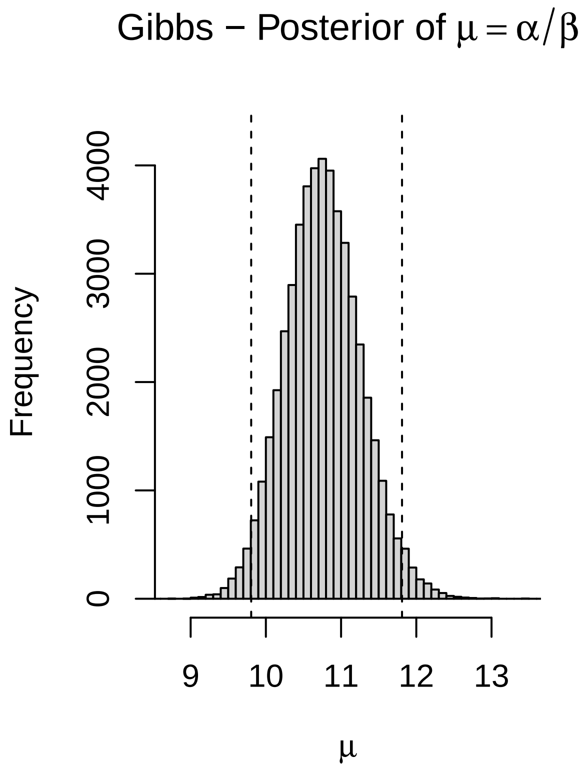

MH − Posterior of α
Frequency
0
500
1000
2000
5
10
15
α


Gibbs − Posterior of α
Frequency
0
500
1000
2000
5
10
15
α


MH − Posterior of β
Frequency
0
1000
3000
5000
0.5
1.0
1.5
β


Gibbs − Posterior of β
Frequency
0
1000
3000
5000
0.5
1.0
1.5
β


MH − Posterior of $\mu = \alpha/\beta$
Frequency
0
1000
2000
3000
4000
9
10
11
12
13
μ


Gibbs − Posterior of $\mu = \alpha/\beta$
Frequency
0
1000
2000
3000
4000
9
10
11
12
13
μ

7.

On this particular assignment, attach your R code for functions you programmed to do the necessary computations as an APPENDIX – not part of the body of your answer.

Metropolis

```r
metropforgamma <- function(dat, start, priorpars, jumpvars, B, M){
# Metropolis for a one-sample Gamma(shape = alpha, rate = beta) model
# with joint prior being the product of marginals
# alpha ~ Uniform(0, A)
# beta ~ Gamma(gamma0, lambda0)
#
# Inputs:
# dat : vector of observed positive data (y_i > 0)
# start : starting values for (alpha, beta) c(alpha0, beta0)
# priorpars: prior parameters c(gamma0, lambda0, A)
#   gamma0, lambda0 are shape/rate of prior on beta
#   A is the upper bound for alpha's Uniform(0, A) prior
# jumpvars : proposal variances for random-walk jumps c(valpha, vbeta)
# B : burn-in iterations
# M : number of kept draws
#
# Additional Notes
# Likelihood: Y_i ~ Gamma(alpha, beta)
#   f(y | alpha, beta) = beta^alpha / Gamma(alpha) * y^(alpha-1) * exp(-beta*y)
# Prior on alpha: Uniform(0, A)
# Prior on beta: Gamma(gamma0, lambda0) also using 'rate' parameterization
# Proposals: independent Gaussian random walks on alpha and beta, consistent
#   Adjust invalid proposals (negative parameter values) by reverting to current values
#   I.e., use same method as the Normal example from Kaiser for sig2

# Starting inputs
calpha <- start[1]
cbeta <- start[2]
gamma0 <- priorpars[1]
lambda0 <- priorpars[2]
A <- priorpars[3]
valpha <- jumpvars[1]
vbeta <- jumpvars[2]

# Initialize
alphas <- NULL
betas <- NULL
mus <- NULL

acceptind <- 0
cnt <- 0

# Calculate sufficient statistics for the Gamma likelihood
# Based on initial data input
n <- length(dat)
```

```r
sumlogy <- sum(log(dat))
sumy <- sum(dat)

repeat{
  cnt <- cnt + 1
  alphastar <- proposealpha(calpha, valpha, A)
  betastar <- proposebeta(cbeta, vbeta)

  # log-likelihood
  # log f(alpha, beta | y) = n * (alpha * log(beta) - log(Gamma(alpha))) + (alpha - 1) * sum(log(y)) -
  lfcur <- n * (calpha * log(cbeta) - lgamma(calpha)) +
    (calpha - 1) * sumlogy - cbeta * sumy
  lfstar <- n * (alphastar * log(betastar) - lgamma(alphastar)) +
    (alphastar - 1) * sumlogy - betastar * sumy

  # log-prior for alpha: Uniform(0, A)
  #   log pi(alpha) = 0 for alpha in (0, A), -Inf otherwise
  lpi_alpha_cur <- if(calpha > 0 && calpha < A) 0 else -Inf
  lpi_alpha_star <- if(alphastar > 0 && alphastar < A) 0 else -Inf

  # log-prior for beta: Gamma(gamma0, lambda0), rate parameterization
  # log pi(beta) = gamma0 * log(lambda0) - log(Gamma(gamma0)) + (gamma0 - 1) * log(beta) - lambda0 * be
  lpi_beta_cur <- gamma0 * log(lambda0) - lgamma(gamma0) +
    (gamma0 - 1) * log(cbeta)  - lambda0 * cbeta
  lpi_beta_star <- gamma0 * log(lambda0) - lgamma(gamma0) +
    (gamma0 - 1) * log(betastar) - lambda0 * betastar

  lpicur  <- lpi_alpha_cur + lpi_beta_cur
  lpistar <- lpi_alpha_star + lpi_beta_star

  # Metropolis acceptance calculations
  astar <- min(exp((lfstar + lpistar) - (lfcur + lpicur)), 1)
  ustar <- runif(1, 0, 1)

  newalpha <- calpha; newbeta <- cbeta
  if(ustar <= astar){
    # update
    newalpha <- alphastar
    newbeta <- betastar
    acceptind <- acceptind + 1
  }

  if(cnt > B){
    alphas <- c(alphas, newalpha)
    betas <- c(betas, newbeta)
    # Posterior samples of mu
    # Generate mu based on the new alpha AND beta
    mus <- c(mus, newalpha / newbeta)
  }

  calpha <- newalpha
  cbeta <- newbeta
  if(cnt == (B + M)) break
}

cat("acceptprob:", acceptind / M, fill = TRUE)
res <- data.frame(alpha = alphas, beta = betas, mu = mus)
attr(res, "acceptprob") <- acceptind / M
return(res)
}

proposealpha <- function(calpha, valpha, A){
  # propose jump from random walk for alpha (shape)
  # enforce support (0, A)
  # if invalid, revert to current
  z <- rnorm(1, 0, sqrt(valpha))
  alphastar <- calpha + z
  if(alphastar <= 0 || alphastar >= A) alphastar <- calpha
  return(alphastar)
}

proposebeta <- function(cbeta, vbeta){
  # propose jump from random walk for beta (rate)
  # enforce positivity
  # if invalid, revert to current
  z <- rnorm(1, 0, sqrt(vbeta))
  betastar <- cbeta + z
  if(betastar <= 0) betastar <- cbeta
  return(betastar)
}
```

Gibbs

```
gibbsforgamma <- function(dat, start, priorpars, B, M, valpha){
# Gibbs sampler for one-sample Gamma(shape = alpha, rate = beta) model
# with alpha ~ Uniform(0, A)
# beta ~ Gamma(gamma0, lambda0)
# Inputs: Very similar in structure to MH method
# dat : vector of observed positive data (y_i > 0)
# start : c(alpha0, beta0)
# priorpars : c(gamma0, lambda0, A)
# B : burn-in
# M : number of kept draws
# valpha : proposal variance for MH step on alpha (b/c Metropolis-within-Gibbs)
#
# Notes on full conditionals and conjugacy:
# Conditional for beta | alpha, y is Gamma(gamma0 + n*alpha, lambda0 + sum(y))
#   (shape/rate parametrization) this is conjugate, so we can sample beta directly
# Conditional for alpha | beta, y is NOT standard:
#       p(alpha | beta, y) proportional to [beta^(n*alpha) / Gamma(alpha)^n] *
#       (prod(y_i))^(alpha - 1) * I(0 < alpha < A)
# Use a random-walk MH step for alpha inside the Gibbs loop
#   (Metropolis-within-Gibbs)
#   mirroring the reference Gibbs code structure
```

```r
# Starting inputs
calpha <- start[1]
cbeta <- start[2]
gamma0 <- priorpars[1]
lambda0 <- priorpars[2]
A <- priorpars[3]

# Initialize
alphas <- NULL
betas <- NULL
mus <- NULL

cnt <- 0
accept_alpha <- 0

# Same sufficient statistics as before
n <- length(dat)
sumlogy <- sum(log(dat))
sumy <- sum(dat)

repeat{
    cnt <- cnt + 1

    # 1. Sample beta | alpha, y  (conjugate Gamma)
    # where
    # shape = gamma0 + n*alpha
    # rate = lambda0 + sum(y)
    newbeta <- rgamma(1, shape = gamma0 + n * calpha, rate = lambda0 + sumy)

    # 2. Sample alpha | beta, y  (Metropolis within Gibbs)
    # target log-density up to constant:
    # log p(alpha | beta, y) = n*alpha*log(beta) - n*log(Gamma(alpha)) + (alpha - 1)*sum(log(y))
    # for 0<alpha<A
    astep <- sampalpha_mh(calpha, newbeta, valpha, sumlogy, n, A)
    newalpha <- astep$alpha
    accept_alpha <- accept_alpha + astep$acc

    if(cnt > B){
      alphas <- c(alphas, newalpha)
      betas  <- c(betas,  newbeta)
      mus    <- c(mus,    newalpha / newbeta)
    }
    calpha <- newalpha; cbeta <- newbeta

    if(cnt == (B + M)) break
  }

  cat("alpha_acceptprob (within Gibbs):", accept_alpha / M, fill = TRUE)
  res <- data.frame(alpha = alphas, beta = betas, mu = mus)
  return(res)
}

sampalpha_mh <- function(calpha, beta, valpha, sumlogy, n, A){
```

```r
# One-step random-walk MH update for alpha (shape) given beta and y.
# Returns a list(alpha = ..., acc = 0/1)

# target log-density (up to constant in alpha):
# log f(alpha | beta, y) = n*alpha*log(beta) - n*log(Gamma(alpha)) + (alpha - 1)*sumlogy
# with support 0 < alpha < A
  z <- rnorm(1, 0, sqrt(valpha))
  alphastar <- calpha + z
  if(alphastar <= 0 || alphastar >= A){
    # If proposal is invalid
    # revert to prior value
    return(list(alpha = calpha, acc = 0))
  }

  # log target at current and proposed
  lfcur <- n * calpha   * log(beta) - n * lgamma(calpha)   +
    (calpha   - 1) * sumlogy
  lfstar <- n * alphastar * log(beta) - n * lgamma(alphastar) +
    (alphastar - 1) * sumlogy

  a <- min(exp(lfstar - lfcur), 1)
  u <- runif(1, 0, 1)
  if(u <= a) return(list(alpha = alphastar, acc = 1))
  return(list(alpha = calpha, acc = 0))
}
```