ISYE 6420: Homework 4

aelhabr3

1. Metropolis: The Bounded Normal Mean.

Instructions

Suppose that we have information that the normal mean θ is bounded between $-m$ and m, for some known number m. In this case it is natural to elicit a prior on θ with the support on interval $[-m, m]$. A prior with interesting theoretical properties supported on $[-m, m]$ is Bickel-Levit prior,

$$\pi(\theta) = \frac{1}{m}\cos^2\left(\frac{\pi\theta}{2m}\right), \quad -m \leq \theta \leq m.$$

Assume that a sample $[-2, -3, 4, -7, 0, 4]$ is observed from normal distribution

with a known precision $\tau = 1/4$. Assume also that the prior on θ is Bickel-Levit, with $m = 2$. This combination likelihood/prior does not result in an explicit posterior (in terms of elementary functions). Construct a Metropolis algorithm that will sample from the posterior of θ.

(a) Simulate 10,000 observations from the posterior, after discarding first 500 observations (burn-in), and plot the histogram of the posterior.
(b) Find Bayes estimator of θ, and 95% equitailed Credible Set based on the simulated observations.

Suggestions:
(i) Take uniform distribution on $[-m, m]$ as a proposal distribution since it is easy to sample from. This is an independence proposal, the proposed θ' does not depend on the current value of the chain, θ.
(ii) You will need to calculate $\sum_{i=1}^{n}(y_i - \theta)^2$ for current θ and $\sum_{i=1}^{n} i = 1(y_i - \theta')^2$ for the proposed θ', prior to calculating the Metropolis ratio.

$$f(y|\theta) \propto \sqrt{\tau}\, \exp\{\tfrac{\tau}{2}(y-\theta)^2\},$$

Response

a

As noted in the instructions, the posterior cannot be written easily in terms of “elementary” functions. Alternatively, the Metropolis algorithm can be used to approximate a posterior distribution.

First, let’s construct the target density distribution. This is proportional to the posterior, which is the product of the prior and the likelihood. Using the Bickel-Levit prior $\pi(\theta)$ with $m = 2$ and the likelihood distribution $f(y|\theta)$ with $\tau = \frac{1}{4}$, , we formulate the target density function as follows.

$$
\begin{aligned}
\text{posterior} &\propto \text{prior} \times \text{likelihood} \\
\pi(y|\theta) &\propto \pi(\theta) \times f(y|\theta) \\
&= (\tfrac{1}{(2)}\cos^2\left(\tfrac{\pi\theta}{2(2)}\right))(\sqrt{(1/4)}\,\exp\{\tfrac{(1/4)}{2}(y-\theta)^2\})
\end{aligned}
$$

(This is still a relatively complex expression, but code can help us solve for given values of θ and τ.)

Using the suggestion and given $m = 2$, we define the proposal density as $\mathcal{U}(-2, 2)$.

Then we have the acceptance probability ρ as

$$
\rho\left(\theta_n, \theta'\right) = \min\left\{1, \frac{\pi(\theta')}{\pi(\theta_n)} \frac{q(\theta_n|\theta')}{q(\theta'|\theta_n)}\right\} = \min\left\{1, z\right\}.
$$

Note that we implement the Metropolis algorithm in the code as follows.

1. Start with arbitrary θ. (Here, 0 is a fine choice.) [1]
2. At stage n, generate proposal θ' from $q(y|x_n)$.
3. Define

$$
\theta_{n+1} = \begin{cases} \theta' & \text{with probability} \quad \rho(\theta_n, \theta'), \\ \theta_n & \text{with probability} \quad 1 - \rho(\theta_n, \theta'). \end{cases}
$$

(Generate $U = \mathcal{U}(0, 1)$ and accept proposal θ' if $U \leq \rho(\theta_n, \theta')$.)

4. Increment n and go to Step 2.

See the code below for the implementation. (Note that `theta_prop` below represents θ' in the formulations above.)

```
set.seed(42)
library(tidyverse)

# Data, constants, hyperparameters, and helper functions.
y <- c(-2, -3, 4, -7, 0, 4)

n_burnin_q1 <- 500L
n_mcmc_q1 <- 10000L + n_burnin_q1

m <- 2
tau <- 1 / 4



.f_likelihood_1 <- function(y, theta, tau) {
  mu <- mean(y)
  sqrt(tau) * exp(-0.5 * tau * (y - theta)^2)
}

.f_likelihood_2 <- function(y, theta, tau) {
  sqrt(tau) * exp(-0.5 * tau * sum(y - theta)^2)
}

bl <- function(theta, m) {
  stopifnot(theta <= m || theta >= -m)
  (1 / m) * (cos((pi * theta) / (2 * m))) ^ 2
}

f_prior <- function(theta, .m = m) {
  bl(theta, m = .m)
}

f_proposal <- function(.n = 1, .m = m) {
  runif(n = .n, min = -.m, max = .m)
}

f_likelihood_1 <- function(theta, .y = y, .tau = tau) {
  sum(.f_likelihood_1(theta, y = y, tau = tau))
}

f_likelihood_2 <- function(theta, .y = y, .tau = tau) {
  .f_likelihood_2(theta, y = y, tau = tau)
}

f_likelihood <- f_likelihood_2

is_likeinteger <- function(x, tol = .Machine$double.eps^0.5) {
  abs(x - round(x)) < tol
}
```

```r
# Main function.
.do_mcmc_mh_q1 <- function(..., n_mcmc = 10000L, n_burnin = 1000L, theta_init = 0) {

  # n_mcmc = n_mcmc_q1; n_burnin = n_burnin_q1; theta_init = 0

  stopifnot(is_likeinteger(n_mcmc))
  if(!is.null(n_burnin)) {
    stopifnot(is_likeinteger(n_burnin))
    stopifnot(n_burnin < n_mcmc)
  }
  cols_mat_mcmc <-
    c(
      'theta_current',
      'theta_prop',
      'q_current',
      'q_prop',
      'pi_current',
      'pi_prop',
      'ratio_num',
      'ratio_den',
      'ratio',
      'rho',
      'u',
      'is_accepted'
    )
  mat_mcmc <- matrix(nrow = n_mcmc, ncol = length(cols_mat_mcmc))
  colnames(mat_mcmc) <- cols_mat_mcmc

  # Step 1 achieved with `.theta_init`.
  theta_current <- theta_init
  # theta_current <- f_proposal()
  # theta_current <- 1
  for (i in 1:n_mcmc) {
    #  i <- 1

    # Step 2.
    theta_prop <- f_proposal()

    # Step 3.
    q_current <- f_prior(theta = theta_current)
    q_prop <- f_prior(theta = theta_prop)

    pi_current <- f_likelihood_2(theta = theta_current)
    pi_prop <- f_likelihood_2(theta = theta_prop)

    # ratio_num <- pi_prop * q_current
    ratio_num <- pi_prop * q_prop
    # ratio_den <- pi_current * q_prop
    ratio_den <- pi_current * q_current
    ratio <- ratio_num / ratio_den
    # ratio <- ratio_den / ratio_num

    u <- runif(n = 1, min = 0, max = 1)
```

```
    rho <- min(1, ratio)

    # Step 4.
    is_accepted <- 0
    if(u <= rho) {
      theta_current <- theta_prop
      is_accepted <- 1
    } else {
      # NULL
      # theta_current <- theta_current
    }
    mat_mcmc[i, ] <-
      c(
        theta_current,
        theta_prop,
        q_current,
        q_prop,
        pi_current,
        pi_prop,
        ratio_num,
        ratio_den,
        ratio,
        rho,
        u,
        is_accepted
      )
  }
  res_mcmc <-
    mat_mcmc %>%
    as_tibble() %>%
    mutate(idx = row_number()) %>%
    select(idx, everything()) # %>%
    # mutate_at(vars(is_accepted), ~ifelse(. == 1, TRUE, FALSE))

  if(!is.null(n_burnin)) {
    idx_slice <- (n_burnin + 1):nrow(mat_mcmc)
    res_mcmc <- res_mcmc %>% slice(idx_slice)
  }

  res_mcmc
}

do_mcmc_mh_q1 <- function(..., .n_mcmc = n_mcmc_q1, .n_burnin = n_burnin_q1) {
  .do_mcmc_mh_q1(n_mcmc = .n_mcmc, n_burnin = .n_burnin, ...)
}
```

```
res_mcmc_mh_q1 <- do_mcmc_mh_q1()
res_mcmc_mh_q1
```

```
## # A tibble: 10,000 x 13
##      idx theta_current theta_prop q_current  q_prop pi_current  pi_prop
##    <int>         <dbl>      <dbl>     <dbl>   <dbl>      <dbl>    <dbl>
##  1   501     -0.01791    1.393       0.4999 0.1052     0.07523 2.552e-9
##  2   502     -0.01791    1.279       0.4999 0.1438     0.07523 1.985e-8
##  3   503     -0.003920  -0.003920    0.4999 0.5000     0.07523 6.927e-2
##  4   504     -0.003920   0.2164      0.5000 0.4857     0.06927 1.496e-2
##  5   505     -1.057     -1.057       0.5000 0.2276     0.06927 2.518e-1
##  6   506     -0.3141    -0.3141      0.2276 0.4702     0.2518  2.858e-1
##  7   507     -0.3141    -1.393       0.4702 0.1052     0.2858  4.648e-2
##  8   508     -0.3141    -1.333       0.4702 0.1252     0.2858  6.786e-2
##  9   509     -0.3141    -1.556       0.4702 0.05846    0.2858  1.427e-2
## 10   510     -0.3141     1.194       0.4702 0.1750     0.2858  8.580e-8
## # ... with 9,990 more rows, and 6 more variables: ratio_num <dbl>,
## #   ratio_den <dbl>, ratio <dbl>, rho <dbl>, u <dbl>, is_accepted <dbl>
```

```
y_mean <- y %>% mean()
y_mean
```

```
## [1] -0.6666667
```

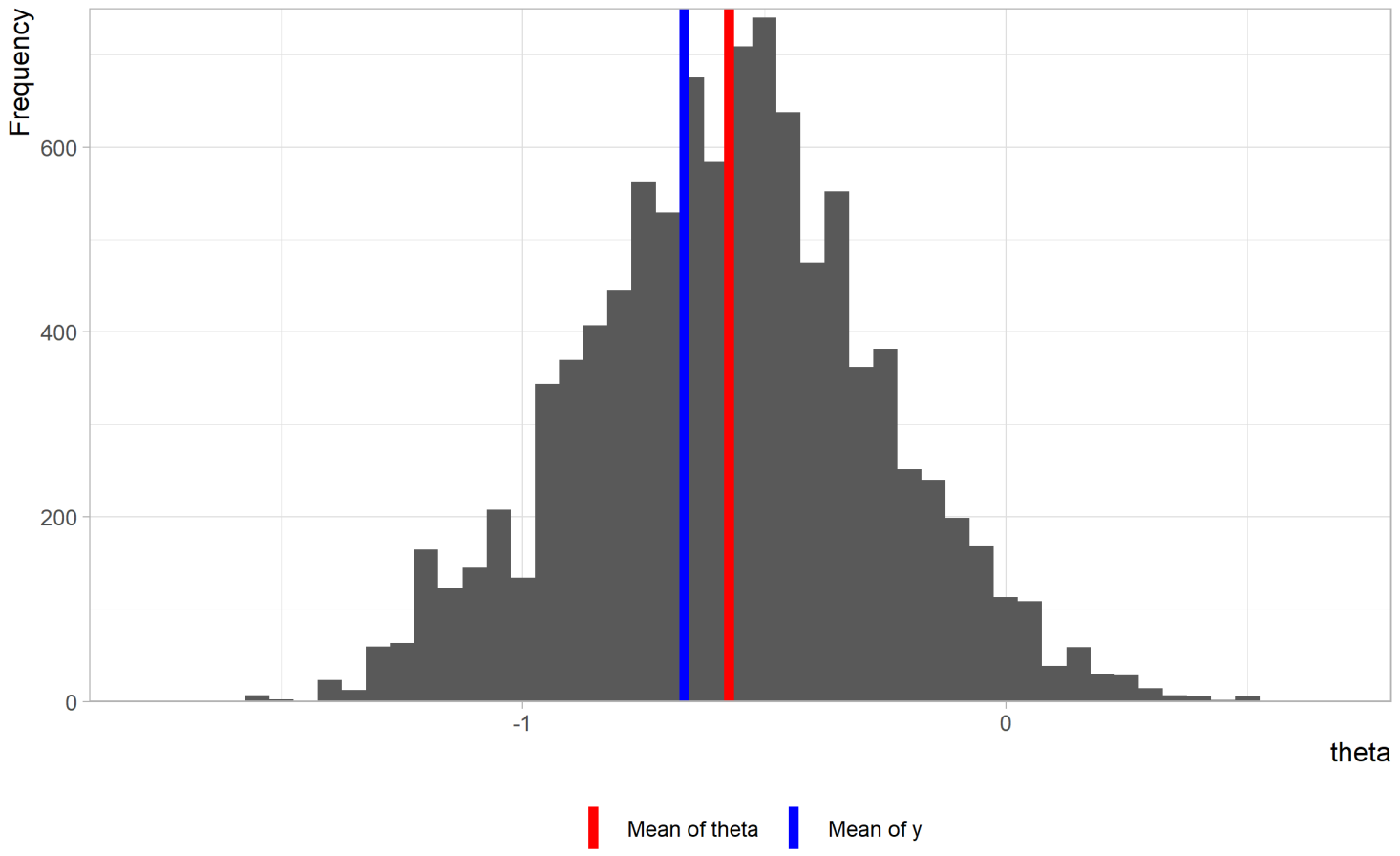
```
theta_mean <-
  res_mcmc_mh_q1 %>%
  summarise_at(vars(theta_current), mean) %>%
  pull(1)
theta_mean
```

```
## [1] -0.57319
```

```
theme_custom <- function(...) {
  theme_light() +
    theme(
      legend.position = 'bottom',
      legend.title = element_blank(),
      axis.title.x = element_text(hjust = 1),
      axis.title.y = element_text(hjust = 1),
      ...
    )
}
```

```
viz_mcmc_mh_q1 <-
  res_mcmc_mh_q1 %>%
  ggplot() +
  aes(x = theta_current) +
  geom_histogram(binwidth = 0.05) +
  geom_vline(aes(xintercept = y_mean, color = 'Mean of y'), size = 2) +
  geom_vline(aes(xintercept = theta_mean, color = 'Mean of theta'), size = 2) +
  scale_color_manual(values = c('red', 'blue')) +
  guides(color = guide_legend(override.aes = list(size = 2))) +
  scale_y_continuous(labels = scales::comma, expand = c(0, 0, 0, 10)) +
  # coord_cartesian(expand = FALSE) +
  theme_custom() +
  labs(
    x = 'theta',
    y = 'Frequency'
  )
```

```
viz_mcmc_mh_q1
```



Note that the posterior mean $\hat{\theta}$ = -0.573 is close to the mean of the observations $\hat{y}$ = -0.667.

2. Gibbs Sampler and High/Low Protein Diet in Rats.

Instructions

Armitage and Berry (1994, p. 111) report data on the weight gain of 19 female rats between 28 and 84 days after birth. The rats were placed in randomized manner on diets with high (12 animals) and low (7 animals) protein content.

High protein	Low protein
134	70
146	118
104	101
119	85
124	107
161	132
107	94
83	
113	
129	
97	
123	

We want to test the hypothesis on dietary effect: Did a low protein diet result in a significantly lower weight gain? The classical t test against the one sided alternative will be significant at 5% significance level, but we will not go in there. We will do the test Bayesian way using Gibbs sampler. Assume that high-protein diet measurements y_{1j}, $i = 1, \ldots, 12$ are coming from normal distribution $\mathcal{N}(\theta_1, 1/\tau_1)$, where τ_1 is the precision parameter,

$$f(y_{1i}|\theta_1, \tau_1) \propto \tau_1^{1/2} \exp\{-\frac{\tau_1}{2}(y_{1i} - \theta_1)^2\}, \quad i = 1, \ldots, 12.$$

The low-protein diet measurements y_{2i}, $i = 1, \ldots, 7$ are coming from normal distribution $\mathcal{N}(\theta_2, 1/\tau_2)$,

$$f(y_{2i}|\theta_2, \tau_2) \propto \tau_2^{1/2} \exp\{-\frac{\tau_2}{2}(y_{2i} - \theta_2)^2\}, \quad i = 1, \ldots, 7.$$

Assume that θ_1 and θ_2 have normal priors $\mathcal{N}(\theta_{10}, 1/\tau_{10})$ and $\mathcal{N}(\theta_{20}, 1/\tau_{20})$, respectively. Take prior means as $\theta_{10} = \theta_{20} = 110$ (apriori no preference) and precisions as $\tau_{10} = \tau_{20} = 1/100$.
Assume that τ_1 and τ_2 have the gamma $\mathcal{G}a(a_1, b_1)$ and $\mathcal{G}a(a_2, b_2)$ priors with shapes $a_1 = a_2 - 0.01$ and rates $b_1 = b_2 = 4$.
(a) Construct Gibbs sampler that will sample $\theta_1, \tau_1, \theta_2$, and τ_2 from their posteriors.
(b) Find sample differences $\theta_1 - \theta_2$. Proportion of positive differences approximates the posterior probability of hypothesis $H_0 : \theta_1 > \theta_2$. What is this proportion if the number of simulations is 10,000, with burn-in of 500?
(c) Using sample quantiles find the 95% equitailed credible set for $\theta_1 - \theta_2$. Does this set contain 0?

Hint: No WinBUGS should be used (except maybe to check your results). Use Octave (MATLAB), or R, or Python here. You may want to consult Handout GIBBS.pdf from the course web repository.

Response

a

See the code and its output below.

```
# Data, constants, hyperparameters, and functions.
y_1 <- c(134, 146, 104, 119, 124, 161, 107, 107, 83, 113, 129, 97, 123)
y_2 <- c(70, 118, 101, 85, 107, 132, 94)

n_burnin_q2 <- 500L
n_mcmc_q2 <- 10000L + n_burnin_q2

theta_1_0 <- 110
theta_2_0 <- theta_1_0
tau_1_0 <- 1 / 100
tau_2_0 <- tau_1_0
a_1_0 <- 0.01
a_2_0 <- a_1_0
b_1_0 <- 4
b_2_0 <- b_1_0

.compute_mu_new <- function(mu_i, tau_i, mu_0, tau_0, y_sum, n_obs) {
  mu_tau_0 <- mu_0 * tau_0
  mu_rnorm_num <- tau_i * y_sum + mu_tau_0
  mu_rnorm_den <- tau_0 + n_obs * tau_i
  mu_rnorm <- mu_rnorm_num / mu_rnorm_den
  sigma2_rnorm <- 1 / (tau_0 + n_obs * tau_i)
  sigma_rnorm <- sqrt(sigma2_rnorm)
  rnorm(1, mu_rnorm, sigma_rnorm)
}

y_sum_1 <- sum(y_1)
y_sum_2 <- sum(y_2)
n_obs_1 <- length(y_1)
n_obs_2 <- length(y_2)
compute_mu_new_1 <-
  function(mu_i,
           tau_i,
           .mu_0 = theta_1_0,
           .tau_0 = tau_1_0,
           .y_sum = y_sum_1,
           .n_obs = n_obs_1) {
    .compute_mu_new(
      mu_i,
      tau_i,
      mu_0 = .mu_0,
      tau_0 = .tau_0,
      y_sum = .y_sum,
      n_obs = .n_obs
    )
  }

compute_mu_new_2 <-
  function(mu_i,
           tau_i,
           .mu_0 = theta_2_0,
           .tau_0 = tau_2_0,
```

```
           .y_sum = y_sum_2,
           .n_obs = n_obs_2) {
    .compute_mu_new(
      mu_i,
      tau_i,
      mu_0 = .mu_0,
      tau_0 = .tau_0,
      y_sum = .y_sum,
      n_obs = .n_obs
    )
  }

.compute_tau_new <- function(mu_new, y, a_0, b_0, n_obs) {
  shape_rgamma <- a_0 + 0.5 * n_obs
  rate_rgamma  <- b_0 + 0.5 * sum((y - mu_new) ^ 2)
  rgamma(1, shape = shape_rgamma, rate = rate_rgamma)
}

compute_tau_new_1 <-
  function(mu_new,
           .y = y_1,
           .a_0 = a_1_0,
           .b_0 = b_1_0,
           .n_obs = n_obs_1) {
    .compute_tau_new(
      mu_new = mu_new,
      y = .y,
      a_0 = .a_0,
      b_0 = .b_0,
      n_obs = .n_obs
    )
  }

compute_tau_new_2 <-
  function(mu_new,
           .y = y_2,
           .a_0 = a_2_0,
           .b_0 = b_2_0,
           .n_obs = n_obs_2) {
    .compute_tau_new(
      mu_new,
      y = .y,
      a_0 = .a_0,
      b_0 = .b_0,
      n_obs = .n_obs
    )
  }

theta_1_init <- mean(y_1) # theta_1_0
theta_2_init <- mean(y_2) # theta_1_i
tau_1_init <- 1 / sd(y_1) # tau_1_0
tau_2_init <- 1 / sd(y_2) # tau_1_i
```

```
# Main function.
.do_mcmc_gibbs_q2 <-
  function(...,
           n_mcmc = 10000,
           n_burnin = 1000,
           theta_1_init = 0,
           theta_2_init = 0,
           tau_1_init = 1,
           tau_2_init = 1) {
    stopifnot(is_likeinteger(n_mcmc))
    if (!is.null(n_burnin)) {
      stopifnot(is_likeinteger(n_burnin))
      stopifnot(n_burnin < n_mcmc)
    }
    stopifnot(is.numeric(theta_1_init))
    stopifnot(is.numeric(theta_2_init))
    stopifnot(is.numeric(tau_1_init))
    stopifnot(is.numeric(tau_2_init))
    cols_mat_mcmc <- c('theta_1', 'theta_2', 'tau_1', 'tau_2')
    mat_mcmc <- matrix(nrow = n_mcmc, ncol = length(cols_mat_mcmc))

    colnames(mat_mcmc) <- cols_mat_mcmc
    theta_1_i <- theta_1_init
    theta_2_i <- theta_2_init
    tau_1_i <- tau_1_init
    tau_2_i <- tau_2_init
    y_1_sum <- sum(y_1)
    y_2_sum <- sum(y_2)
    n_1_obs <- length(y_1)
    n_2_obs <- length(y_2)
    for (i in 1:n_mcmc) {
      theta_1_new <-
        compute_mu_new_1(
          mu_i = theta_1_i,
          tau_i = tau_1_i
        )

      theta_2_new <-
        compute_mu_new_2(
          mu_i = theta_2_i,
          tau_i = tau_2_i
        )

      tau_1_new <-
        compute_tau_new_1(
          mu_new = theta_1_new
        )

      tau_2_new <-
        compute_tau_new_2(
          mu_new = theta_2_new
        )
      mat_mcmc[i,] <-
```

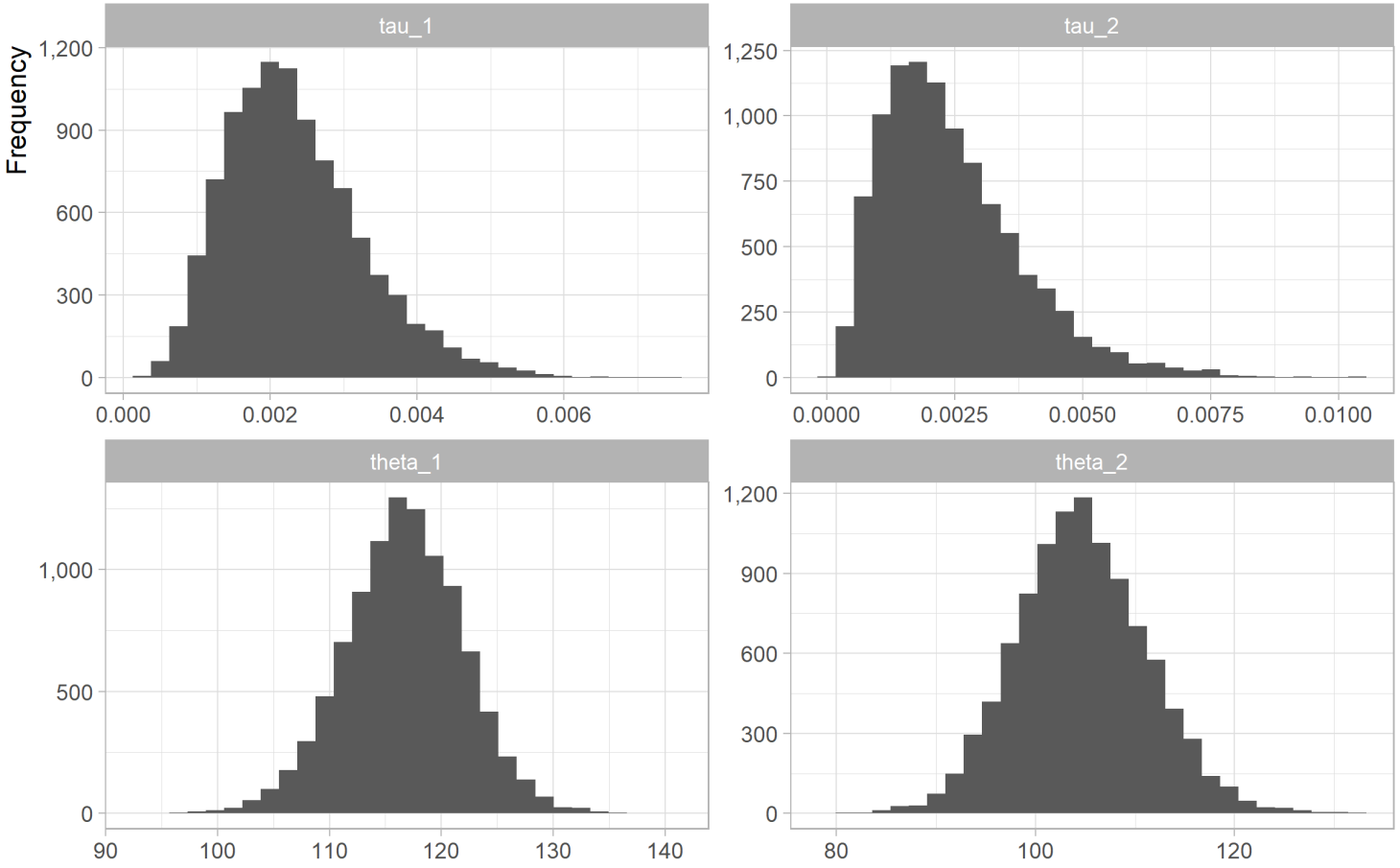
```
      c(theta_1_new, theta_2_new, tau_1_new, tau_2_new)

    theta_1_i <- theta_1_new
    tau_1_i <- tau_1_new
    theta_2_i <- theta_2_new
    tau_2_i <- tau_2_new

  }
  res_mcmc <-
    mat_mcmc %>%
    as_tibble() %>%
    mutate(idx = row_number()) %>%
    select(idx, everything())

  if (!is.null(n_burnin)) {
    idx_slice <- (n_burnin + 1):nrow(mat_mcmc)
    res_mcmc <- res_mcmc %>% slice(idx_slice)
  }

  res_mcmc
}

do_mcmc_gibbs_q2 <-
  function(...,
           .n_mcmc = n_mcmc_q2,
           .n_burnin = n_burnin_q2,
           .theta_1_init = theta_1_init,
           .theta_2_init = theta_2_init,
           .tau_1_init = tau_1_init,
           .tau_2_init = tau_2_init) {
    .do_mcmc_gibbs_q2(
      n_mcmc = .n_mcmc,
      n_burnin = .n_burnin,
      theta_1_iniit = .theta_1_init,
      theta_2_init = .theta_2_init,
      tau_1_init = .tau_1_init,
      tau_2_init = .tau_2_init,
      ...
    )
  }
```

```
res_mcmc_gibbs_q2 <- do_mcmc_gibbs_q2()
res_mcmc_gibbs_q2
```

```
## # A tibble: 10,000 x 5
##      idx theta_1 theta_2    tau_1    tau_2
##    <int>   <dbl>   <dbl>    <dbl>    <dbl>
##  1   501   111.0  113.4  0.003211 0.002151
##  2   502   116.9  106.9  0.002621 0.003411
##  3   503   120.4   93.45 0.001919 0.004457
##  4   504   117.1  100.5  0.002827 0.003930
##  5   505   107.2   98.30 0.001275 0.003104
##  6   506   119.7  110.0  0.002853 0.001182
##  7   507   120.1  101.3  0.002505 0.003208
##  8   508   131.2  100.9  0.003106 0.002914
##  9   509   114.0  109.1  0.004564 0.001845
## 10   510   111.6   92.53 0.001908 0.005517
## # ... with 9,990 more rows
```

```
viz_mcmc_gibbs_q2 <-
  res_mcmc_gibbs_q2 %>%
  gather(key = 'key', value = 'value', -idx) %>%
  ggplot() +
  aes(x = value) +
  geom_histogram() +
  scale_y_continuous(labels = scales::comma) +
  theme_custom() +
  facet_wrap(~key, scales = 'free') +
  labs(
    x = NULL,
    y = 'Frequency'
  )
```

```
viz_mcmc_gibbs_q2
```



As a quick check of the validity of our results, we can compare the posterior means of θ_1 and θ_2 (from the Gibbs sampling results) with those of the observations y_1 and y_2 and the priors $\theta_{10} = \theta_{20} = 110$. We find that $\hat{\theta_1}$ = 116.7, which is between the observed mean $\hat{y_1}$ = 119 and the prior 110; and we find that $\hat{\theta_2}$ = 104.7, which is between the observed mean $\hat{y_2}$ = 101 and 110. This is what we should have expected.

b

We can calculate the proportion as follows.

```
res_mcmc_gibbs_q2_calc <-
  res_mcmc_gibbs_q2 %>%
  mutate(
    theta_diff = theta_1 - theta_2
  ) %>%
  mutate(
    h_0 = ifelse(theta_diff > 0, 1, 0)
  )
```

```
h_0_q2 <-
  res_mcmc_gibbs_q2_calc %>%
  summarise(h_0 = sum(h_0), n = n()) %>%
  mutate(frac = h_0 / n)
h_0_q2
```

```
## # A tibble: 1 x 3
##     h_0     n   frac
##   <dbl> <int>  <dbl>
## 1  9195 10000 0.9195
```

We find that the proportion of positive differences is 0.919 (i.e. 92.0%).

c

We can calculate the credible set as follows.

```
equi_credible_set_q2 <-
  res_mcmc_gibbs_q2_calc %>%
  summarise(
    mu = mean(theta_diff),
    sd = sd(theta_diff)
  ) %>%
  mutate(
    l = qnorm(0.025, mu, sd),
    u = qnorm(0.975, mu, sd)
  )
equi_credible_set_q2
```

```
## # A tibble: 1 x 4
##      mu    sd      l     u
##   <dbl> <dbl>  <dbl> <dbl>
## 1 12.02 8.473 -4.584 28.63
```

We find that the 95% equitailed credible set is [-4.58, 28.63]. Indeed, this set does include 0.

---

1. We don't really have an idea of the form of the support of the target; otherwise, we would try to initialize θ to be like the target.↩