MATH 392 Problem Set 10

Sam D. Olson

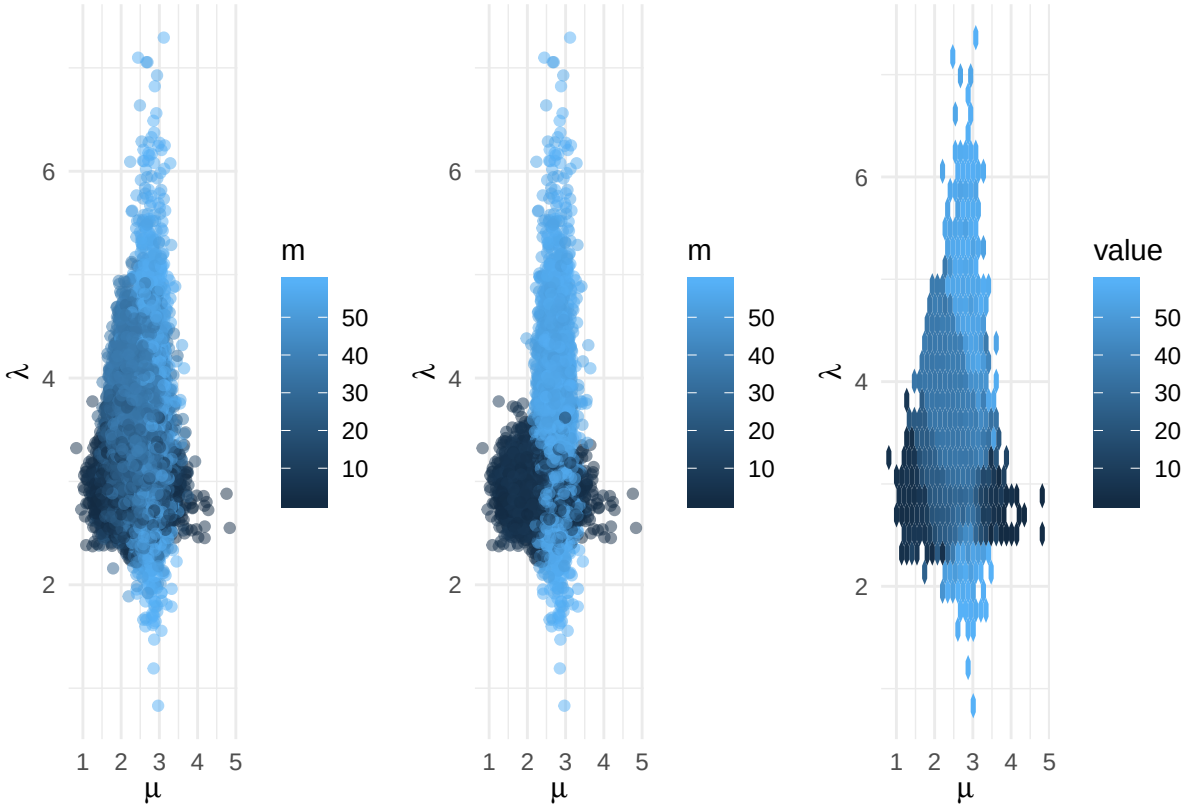
Part I: Changing the changepoint

You're encouraged to start these exercise by copying into your problem set code from the notes that implements the Gibbs Sampler with `set.seed(497)`.

1. Using the `post_samples` matrix (or corresponding dataframe `df`), investigate the shape of the joint posterior distribution by constructing several plots. After contructing these plots, summarize in a few sentences the relationship between the three parameters in the posterior. *Please put all three plots in a single 1 by 3 frame using `library(patchwork)`.*
    - A scatterplot of λ on μ with color mapped to the value of m.
    - The same scatterplot but where you only include samples with values of m that are less than 10 or greater than 50.
    - A hexplot where each hex is filled with either the mean or median value of m.

```
# Set Seed
set.seed(497)
# Initialize Data/Parameters
n <- 60
m <- 38
mu <- 2
lambda <- 4
y_mu <- rpois(m, lambda = mu)
y_lambda <- rpois(n - m, lambda = lambda)
y <- c(y_mu, y_lambda)
alpha <- 10
beta <- 4
nu <- 8
phi <- 2
# Post-Samples Matrix
it <- 50000
post_samples <- matrix(rep(NA, it * 3), ncol = 3)
colnames(post_samples) <- c("mu", "lambda", "m")
m_j <- 2 # initialize m
for (j in 1:it) {
  # sample mu
  mu_j      <- rgamma(1, alpha + sum(y[1:m_j]), beta + m_j)
  # sample lambda
  lambda_j  <- rgamma(1, nu + sum(y[(m_j+1):n]), phi + (n - m_j))
  # sample m
  m_vec <- rep(NA, n - 1)
  for (k in 1:(n - 1)) {
    m_vec[k] <-  mu_j^(alpha + sum(y[1:k]) - 1) *
      exp(-(beta + k) * mu_j) *
```

```
      lambda_j^(nu + sum(y[(k+1):n]) - 1) *
      exp(-(phi + n - k) * lambda_j)
  }
  p <- m_vec/sum(m_vec)
  m_j <- sample(1:(n - 1), size = 1, prob = p)
  # store results
  post_samples[j, "mu"]     <- mu_j
  post_samples[j, "lambda"] <- lambda_j
  post_samples[j, "m"]      <- m_j
}
df <- data.frame(post_samples) %>%
  mutate(index = 1:it)
# Visuals
# Part a
p4 <- ggplot(df, aes(x = mu, y = lambda, color=m)) +
  geom_point(alpha=0.5) +
  labs(y = expression(lambda), x = expression(mu)) +
  theme_minimal()
# p4
# Part b
# $m$ that are less than 10 or greater than 50.
df2 <- subset(df, m > 50 | m < 10,)
p5 <- ggplot(df2, aes(x = mu, y = lambda, color=m)) +
  geom_point(alpha=0.5) +
  labs(y = expression(lambda), x = expression(mu)) +
  theme_minimal()
# Part c
# A [hexplot](http://ggplot2.tidyverse.org/reference/stat_summary_2d.html) where each hex is filled wit
p6 <- ggplot(df, aes(x = mu, y = lambda, z = m)) +
  # geom_hex(fun = ~ quantile(m, 0.5)) +
  stat_summary_hex(fun = "quantile", fun.args = list(probs = 0.5)) +
  labs(y = expression(lambda), x = expression(mu)) +
  theme_minimal()
p4 + p5 + p6
```

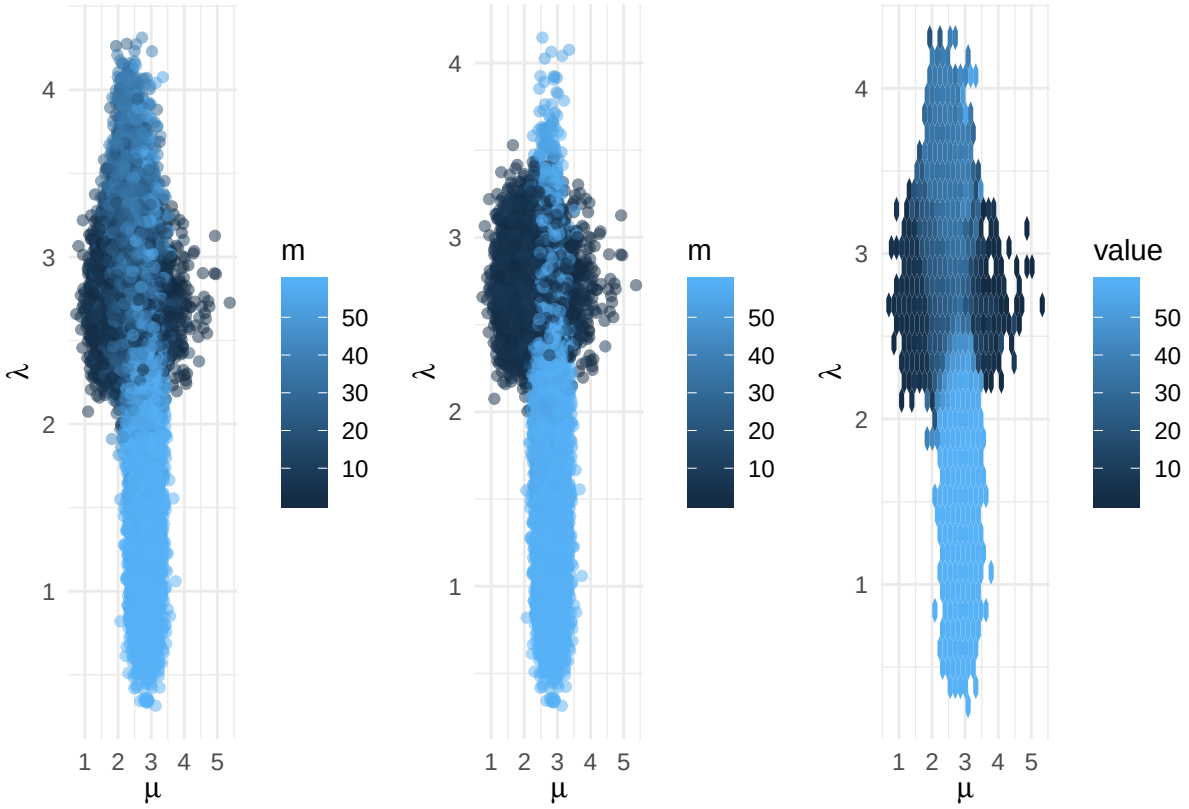


Commentary

2. Alter one of the prior distributions of the Poisson rate parameters so that they are much more flat. What is the effect on the joint posterior distribution?

```
# Set Seed
set.seed(497)
# Initialize Data/Parameters
n <- 60
m <- 38
mu <- 2
lambda <- 4
y_mu <- rpois(m, lambda = mu)
y_lambda <- rpois(n - m, lambda = lambda)
y <- c(y_mu, y_lambda)
alpha <- 10
beta <- 4
nu <- 8
# Modify the value of phi, decreasing the variance of the distribution
phi <- 6
# Post-Samples Matrix
it <- 50000
post_samples <- matrix(rep(NA, it * 3), ncol = 3)
colnames(post_samples) <- c("mu", "lambda", "m")
m_j <- 2 # initialize m
```

```r
for (j in 1:it) {
  # sample mu
  mu_j      <- rgamma(1, alpha + sum(y[1:m_j]), beta + m_j)
  # sample lambda
  lambda_j  <- rgamma(1, nu + sum(y[(m_j+1):n]), phi + (n - m_j))
  # sample m
  m_vec <- rep(NA, n - 1)
  for (k in 1:(n - 1)) {
    m_vec[k] <-  mu_j^(alpha + sum(y[1:k]) - 1) *
      exp(-(beta + k) * mu_j) *
      lambda_j^(nu + sum(y[(k+1):n]) - 1) *
      exp(-(phi + n - k) * lambda_j)
  }
  p <- m_vec/sum(m_vec)
  m_j <- sample(1:(n - 1), size = 1, prob = p)
  # store results
  post_samples[j, "mu"]     <- mu_j
  post_samples[j, "lambda"] <- lambda_j
  post_samples[j, "m"]      <- m_j
}
df <- data.frame(post_samples) %>%
  mutate(index = 1:it)
# Visuals
# Part a
p4 <- ggplot(df, aes(x = mu, y = lambda, color=m)) +
  geom_point(alpha=0.5) +
  labs(y = expression(lambda), x = expression(mu)) +
  theme_minimal()
# p4
# Part b
# $m$ that are less than 10 or greater than 50.
df2 <- subset(df, m > 50 | m < 10,)
p5 <- ggplot(df2, aes(x = mu, y = lambda, color=m)) +
  geom_point(alpha=0.5) +
  labs(y = expression(lambda), x = expression(mu)) +
  theme_minimal()
# Part c
# A [hexplot](http://ggplot2.tidyverse.org/reference/stat_summary_2d.html) where each hex is filled wit
p6 <- ggplot(df, aes(x = mu, y = lambda, z = m)) +
  # geom_hex(fun = ~ quantile(m, 0.5)) +
  stat_summary_hex(fun = "quantile", fun.args = list(probs = 0.5)) +
  labs(y = expression(lambda), x = expression(mu)) +
  theme_minimal()
p4 + p5 + p6
```



Commentary

Part II: The Metropolis algorithm

To validate the results from the Gibbs sampler, get a second estimate of the joint posterior (using the original priors) using the Metropolis algorithm. You're encouraged to start these exercise by copying into your problem set code from the notes that implements the Metropolis Algorithm. You will need to put some thought into how to build your symmetric proposal distribution since it combines continuous and discrete parameters.

1. Generate the same 3 x 1 plot as you did for the Gibbs sampler. Do these tell the same story? (these plots should represent your final conclusions after considering questions 2 and 3 below).

Notes

P.m.f. of Poisson:

For rate λ and support $k \in Z^+$:

$\frac{\lambda^k e^{-\lambda}}{k!}$

Parameters

$f \sim Pois(\mu)$

$g \sim Pois(\lambda)$

$m \sim Unif(1, n-1)$

Joint of $f(Y, \mu, \lambda, m) = \prod_{i=1}^{m} \frac{\lambda^m e^{-\lambda}}{m!} \prod_{j=m+1}^{n} \frac{\mu^{(n-m)} e^{-\mu}}{(n-m)!} \frac{1}{m} \frac{1}{n-m}$

2. What is your acceptance rate? Tinker with the variance of the proposal distribution to bring it within the target range.

```
library(patchwork)
library(tidyverse)
library(hexbin)

# Set Seed
set.seed(497)

# Set Parameters
n <- 60
m <- 38
mu <- 2
lambda <- 4
y_mu <- rpois(m, lambda = mu)
y_lambda <- rpois(n - m, lambda = lambda)
y <- c(y_mu, y_lambda)
alpha <- 10
beta <- 4
nu <- 8
phi <- 2

# Simulation
it <- 50000
post_samples <- matrix(rep(NA, it * 3), ncol = 3)
colnames(post_samples) <- c("mu", "lambda", "m")
m_j <- 2 # initialize m

# Prior Code

# Gibbs
it <- 50000
post_samples <- matrix(rep(NA, it * 3), ncol = 3)
colnames(post_samples) <- c("mu", "lambda", "m")
m_j <- 2 # initialize m
for (j in 1:it) {
  # sample mu
  mu_j      <- rgamma(1, alpha + sum(y[1:m_j]), beta + m_j)
  # sample lambda
  lambda_j  <- rgamma(1, nu + sum(y[(m_j+1):n]), phi + (n - m_j))
  # sample m
  m_vec <- rep(NA, n - 1)
  for (k in 1:(n - 1)) {
    m_vec[k] <-  mu_j^(alpha + sum(y[1:k]) - 1) *
      exp(-(beta + k) * mu_j) *
      lambda_j^(nu + sum(y[(k+1):n]) - 1) *
      exp(-(phi + n - k) * lambda_j)
```

```
  }
  p <- m_vec/sum(m_vec)
  m_j <- sample(1:(n - 1), size = 1, prob = p)
  # store results
  post_samples[j, "mu"]     <- mu_j
  post_samples[j, "lambda"] <- lambda_j
  post_samples[j, "m"]      <- m_j
}
# Office Hour Notes

n <- 50
m_prop <- sample(1:n, 1, replace=FALSE, prob = rep(1/n, n))

# Shrink variance of discrete distribution on 1 to n


# Adapted Metro-Hast
chain <- rep(NA, it + 1)
chain[1] <- mu
for (i in 1:it) {
  proposal <- rnorm(1, chain[i], lambda)
  p_move <- min(dgamma(proposal, alpha + sum(y[1:m_j]), beta + m_j)/
                  dgamma(chain[i],  nu + sum(y[(m_j+1):n]), phi + (n - m_j)),
                1)
  chain[i + 1] <- ifelse(runif(1) < p_move,
                         proposal,
                         chain[i])
}

# Metro-Hast
theta_0 <- 1.2
tau <- .3
it <- 50000
chain <- rep(NA, it + 1)
chain[1] <- theta_0
for (i in 1:it) {
  proposal <- rnorm(1, chain[i], tau)
  p_move <- min(dgamma(proposal, 2, 3)/
                  dgamma(chain[i], 2, 3),
                1)
  chain[i + 1] <- ifelse(runif(1) < p_move,
                         proposal,
                         chain[i])
}
```

```
# Burn-in & Other Considerations
burn_in <- 5000

data.frame(chain, index = 1:(it + 1)) %>%
  filter(index > burn_in) %>%
  ggplot(aes(x = chain)) +
  stat_function(fun = dgamma, args = list(2, 3)) +
  geom_density(col = "steelblue") +
```

```r
  theme_bw() +
  labs(x = expression(theta))


# Acceptance Rate
# Part 2
(acceptance <- 1 - mean(duplicated(chain[-(1:burn_in)])))


# Autocorrelation
# Part 3
df1 <- data.frame(x_0 = chain[1:(it-1)],
                  x_1 = chain[2:it])

chain_thinned <- data.frame(chain) %>%
  slice(seq(from = 1, to = it, by = 10)) %>%
  pull()

df2 <- data.frame(x_0 = chain_thinned[1:(length(chain_thinned)-1)],
                  x_1 = chain_thinned[2:length(chain_thinned)])

p1 <- ggplot(df1, aes(x = x_0, y = x_1)) +
  geom_point(alpha = .1) +
  theme_bw()

p2 <- ggplot(df2, aes(x = x_0, y = x_1)) +
  geom_point(alpha = .1) +
  theme_bw()

p1 + p2

# Plot Comparison
# Part 1
p4 <- ggplot(df2, aes(x = mu, y = lambda, color=m)) +
  geom_point(alpha=0.5) +
  labs(y = expression(lambda), x = expression(mu)) +
  theme_minimal()

df3 <- subset(df2, m > 50 | m < 10,)

p5 <- ggplot(df3, aes(x = mu, y = lambda, color=m)) +
  geom_point(alpha=0.5) +
  labs(y = expression(lambda), x = expression(mu)) +
  theme_minimal()

p6 <- ggplot(df2, aes(x = mu, y = lambda, z = m)) +
  # geom_hex(fun = ~ quantile(m, 0.5)) +
  stat_summary_hex(fun = "quantile", fun.args = list(probs = 0.5)) +
  labs(y = expression(lambda), x = expression(mu)) +
  theme_minimal()

p4 + p5 + p6
```

3. To understand the Markov dependence in this chain (or in fact any form of dependence in index), a helpful tool is an autocorrelation plot. Use the `autoplot()` function in `library(ggfortify)` to get a sense of the correlation in your chain (see example of a correlogram). How heavily do you need to thin the chain to diminish most of this dependence? Go ahead and do so to produce a final set of plots for question 1 above.

Part III: Rejection Sampling

In this part, you will apply to the same problem a third approach: rejection sampling. You may want to refer to past slides for more details, but in broad strokes rejection sampling for the changepoint problem will involve,

- Writing a function called `joint_post()` that takes as input values of the three parameters (and the data) and returns a value of the joint posterior density.

```
joint_post <- function(mu, lambda, m, data){
  joint_density <- dpois(data, lambda = mu) * dpois(data, lambda = lambda) * dunif(data, min = 0 , max =
  return(joint_density)
}
```

- Formulating a 3D proposal distribution that you can take random samples from that covers the support of the parameters (or the support with the highest density).
- Taking draws from the proposal distribution for your three parameters and retaining it with probability equal to the ratio of the proposal density at the point to the value of `joint_post()` at that point. Note that you will likely need to normalize either the proposal density or the joint density so that the proposal density is always greater than or equal to the joint density, ensuring acceptance probabilities less than or equal to 1.
- Repeat the previous step many many times until you have accepted a suitable number of draws from the joint posterior.

Implement such a rejection sampling scheme for the changepoint problem and create the same plots from part I with the original priors and the data from `set.seed(497)`. Is the structure in the new sample from the joint posterior the same as when we used a Gibbs Sampler (up to Monte Carlo variability)? What about the Metropolis Algorithm? What are the downsides, as you see it, to the rejection sampling approach? What about the upsides?

Note: this problem set has some serious computation in it, so I suggest cacheing your results by adding `cache = TRUE` to the r chunk options.