202C-HW3

Jiahao Tian

2022-11-05

```
library(tidyverse)
library(dplyr)
library(ggplot2)
library(invgamma)
library(mgcv)
library(grid)
library(MASS)
library(plotly)
library(mvtnorm)
```

Question 1

1a) Solution:

log Posterior:

$$logf(\mu, \sigma^2|y) \propto -log(\sigma) - \frac{\sum_{i=1}^{N}(yi-\mu)^2}{2\sigma^2} - \frac{(\mu-\mu_0)^2}{2\sigma_0^2} - (a+1)log(\sigma^2) - \frac{\beta}{\sigma^2}$$

Then we can calculate $\hat{\mu}$ and $\hat{\sigma}^2$:

$$\frac{\partial logf}{\partial \mu} = -\frac{\sum_{i=1}^{N}(\mu-yi)}{\sigma^2} - \frac{(\mu-\mu_0^2)}{\sigma_0^2}$$

$$\frac{\partial logf}{\partial \sigma^2} = -(\frac{n}{2}+\alpha+1)(\sigma^2)^{-1} + (\frac{\sum_{i=1}^{N}(yi-\mu)^2}{2}+\beta)(\sigma^2)^{-2}$$

```
N <- 100 # more data
y <- rnorm(N, mean = 50, sd = sqrt(10))

mu0 <- 0
sig20 <- 100

alpha <- 2
beta  <- 10

maxIts <- 1000
mus    <- rep(0, maxIts) # chains
mus[1] <- mu0
sigma2s <- rep(0, maxIts)
sigma2s[1] <- sig20
```

```
for (i in 2 : maxIts) {
  # mu update
  Var <- 1/(1 / sig20 + N / sigma2s[i-1])
  Mean <- (mu0 / sig20 + sum(y) / sigma2s[i-1]) * Var
  mus[i] <- rnorm(n = 1, mean = Mean, sd = sqrt(Var))

  # sigma2 update
  resids <- y - mus[i]
  sigma2s[i] <- 1/rgamma(n = 1,
                         shape = alpha + N/2,
                         rate = beta + t(resids) %*% resids / 2)

}
y
```

```
##   [1] 51.23065 47.88001 43.49521 55.36108 46.12629 40.67168 49.66313 46.23777
##   [9] 52.86822 49.61376 52.24517 48.59096 46.67704 51.83633 49.31813 47.94657
##  [17] 45.83530 43.97001 51.07097 52.25155 52.75665 50.44316 49.63058 46.74409
##  [25] 48.30984 53.60567 53.53199 50.33506 49.76184 45.19008 51.68605 47.55823
##  [33] 46.62319 54.32928 52.77344 48.92104 48.64978 55.23335 43.69083 47.71851
##  [41] 50.13907 52.33791 50.56849 57.73588 53.21603 53.73376 51.08420 51.39057
##  [49] 46.80096 44.57657 45.06060 49.41016 53.34453 48.00161 45.08627 48.32154
##  [57] 49.50198 46.31822 47.45942 52.36878 47.12745 48.58316 47.20665 49.87286
##  [65] 52.70274 51.48555 50.54602 54.87733 51.10519 49.66036 50.17918 55.87881
##  [73] 47.33081 51.45085 50.52958 49.86809 48.88426 49.77580 46.15813 44.40726
##  [81] 49.65504 51.41291 42.95229 49.48945 48.25267 52.42864 48.41673 49.33183
##  [89] 49.18625 47.28446 47.85608 49.12907 49.65466 55.92338 46.03765 52.50085
##  [97] 52.90019 45.95861 47.65633 48.26935
```

```
muhat <- mean(mus)
sighat <- mean(sigma2s)
```

We get $\hat{\mu}$ and $\hat{\sigma}^2$ as follow:

```
c(muhat, sighat)
```

```
## [1] 49.38621 10.14387
```

By negative inverse we get the following covariance matrix:

```
var1 = -N / (sighat) - 1 / sig20
var2 = (N * muhat - sum(y)) / (sighat^2)
var3 = (N / 2 + alpha + 1) / (sighat^2) - (sum((muhat - y)^2) + 2 * beta) / (sighat^3)

mav <- rbind(c(-(var1), -(var2)), c(-(var2), -(var3)))
mav <- solve(mav)
mav
```

```
##             [,1]        [,2]
## [1,]  0.10155579 -0.02201097
## [2,] -0.02201097  2.20325129
```

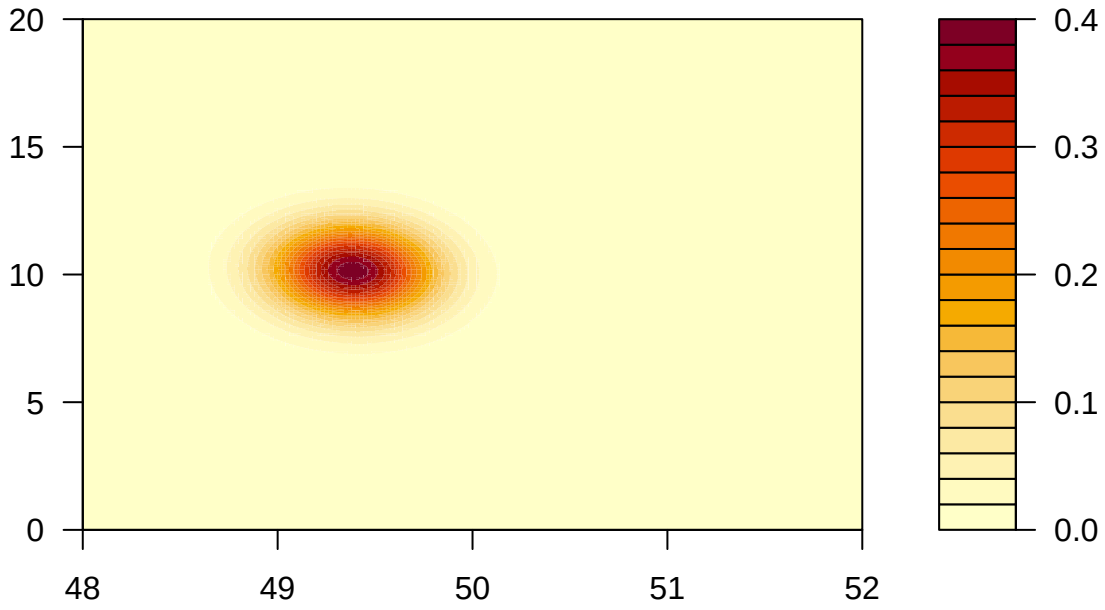
2D contour Gaussian Approximation

```
xs <- seq(48,52, by = 0.02)
ys <- seq(0,20, by = 0.1)

xy <- expand.grid(xs, ys)

mu1a <- c(muhat, sighat)
cov1a <- rbind(c(0.09206337, -0.01720958), c(-0.01720958, 1.77900006))
mulpdf <- dmvnorm(xy, mean = mu1a, sigma = cov1a, log = FALSE)
Zmat <- matrix(mulpdf, nrow = 201)

filled.contour(xs, ys, Zmat)
```



1b) Solution

By using

$$q(\mu, \sigma^2 | m^{(L)}, v^{(L)2}, b^{(L)}) = \ Normal(\mu | m^{(L)}, v^{(L)2}) * InverGamma(\sigma^2 | 1, b^{(L)})$$

And follow the lecture notes, calculate as follow:

```
m <- mu0
v <- sqrt(sig20)
b <- beta
a <- 1

c(m, v, b, mu0, sig20, alpha, beta)
```

```
## [1]   0  10  10   0 100   2  10
```

2D contour Variational inference (Bayes Approximation)

```
N2 = 1000

for(i in N2){
  m = (sum(y) + b/a * (mu0 / sig20)) / (N + b / (a * sig20))
  v = sqrt((b / a) / (N + b / (a * sig20)))
  b = a * (N * v^2 + sum((y - m)^2) + 2 * beta) / (2 * alpha + N)
  }

c(m,v,b)
```

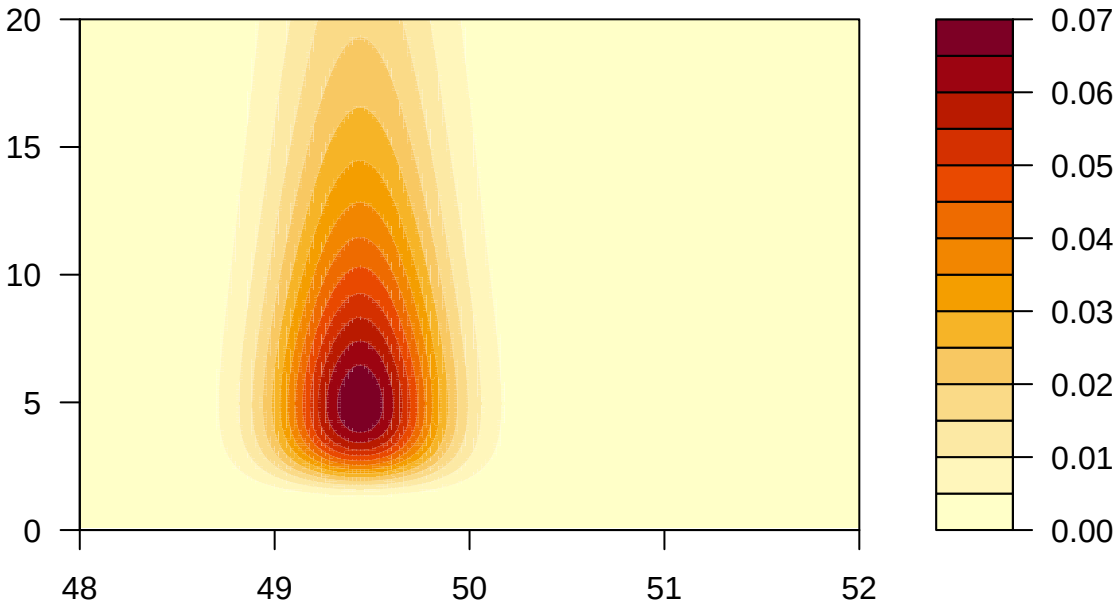
```
## [1] 49.4382173  0.3160698  9.8231330
```

```
xs1 <- seq(48, 52, by = 0.02)
ys1 <- seq(0, 20, by = 0.1)
xys1 <- data.frame(xs1, ys1)

xys11 <- xys1 %>%
  mutate(px = dnorm(xs1, mean = m, sd = v)) %>%
  mutate(py = dinvgamma(ys1, a, b))

xys11 <- xys11 %>%
  mutate(pz = outer(px, py, "*"))

filled.contour(xs1, ys1, xys11$pz)
```



Question 2

2a) Solution

First follow the hint:

- Sample N = 200 sample points from the mixture distribution.
- Generate w1, . . . , w200 from Bernoulli(0.5).

Then by using algorithem from lecture notes (ADF):

$$
\begin{aligned}
m_\theta &= 0 \\
v_\theta^o &= 100 \\
for\ n &= 1, ..., N \\
total\ function : s &= s^{-n} z_n \\
probability\ of\ estimates : \pi_n &= 1 - \frac{w}{z_n} * N(y_n | 0, 10 I_D) \\
m_\theta &= m_\theta^{-n} + v_\theta^{-n} + \pi_n \left(\frac{y_n - m_\theta^{-n}}{v_\theta^{-n} + 1} \right) \\
v_\theta &= v_\theta^{-n} - \pi_n \left(\frac{(v_\theta^{-n})^2}{v_\theta^{-n} + 1} \right) + \pi_n (1 - \pi_n)(v_\theta^{-n})^2 \left(\frac{||x_n - m_\theta||^2}{D(v_\theta^{-n} + 1)^2} \right)
\end{aligned}
$$

```
NN <- 200
theta_0 <- rbind(5, 5)
pdfb <- rbernoulli(n = NN, p = 0.5)
mati <- rbind(c(1, 0), c(0, 1))

pdf1 <- rmultinom(mati, size = sum(pdfb), prob = theta_0, log = FALSE)
pdf2 <- rmultinom(mati, size = NN - sum(pdfb), prob = rbind(0, 0), log = FALSE)

pdf12 <- cbind(pdf1, pdf2)

for(n in NN){
  a1 = (1 - 1 / 2) * dmvnorm(
    pdf12, mean = m, cov=(v + 1) * (2)) + 1 / 2 * dmultinom(
      pdf12, mean = 2, cov=10)
  pi = 1 - 1 / 2 / a1 * dmultinom(pdf12, mean = n2, cov = 10)
  newmu = m + v * pi * (pdf12 - m) / (v + 1)
  newvar = v - pi * (v^2) / (v + 1) + pi * (1 - pi) * (v^2 * sum((pdf12 - m)^2)) / (2*(v + 1)^2)
}
```

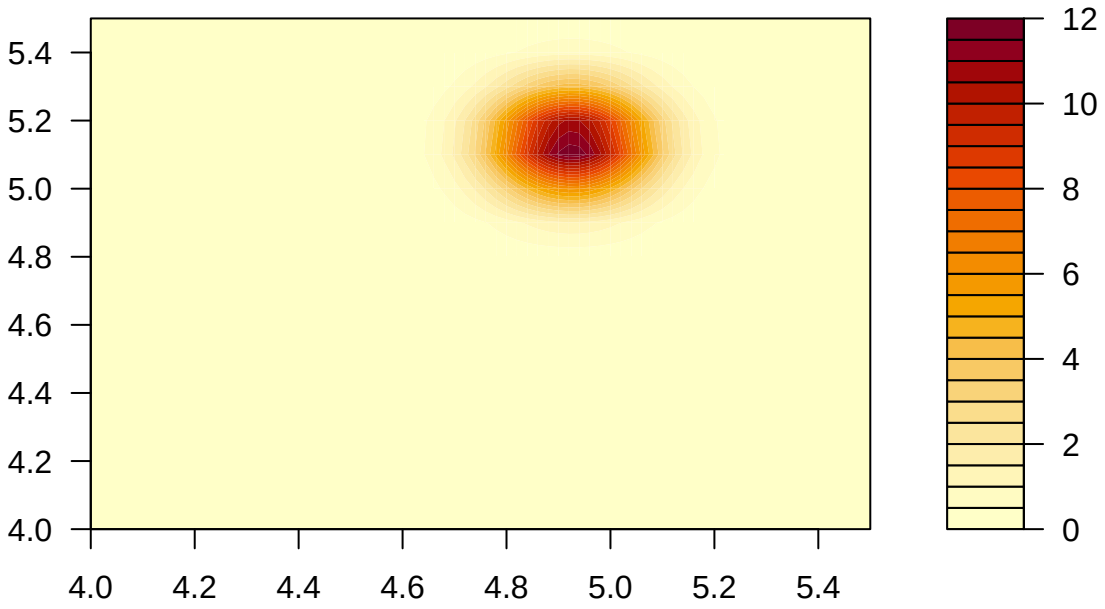
2D contour Gaussian Approximation by ADF

```
xs2 <- seq(4,5.5, by = 0.02)
ys2 <- seq(4,5.5, by = 0.1)

xy2 <- expand.grid(xs2, ys2)

mulpdf1 <- dmvnorm(xy2, mean = newmu, sigma = newvar, log = FALSE)
Zmat1 <- matrix(mulpdf1, nrow = 76)

filled.contour(xs2, ys2, Zmat1)
```



2b) Solution

First initialize prior terms, then initialize m_θ and v_θ

Then by following the steps of EP:

$$
\begin{aligned}
m_\theta &= m_0 \\
v_\theta &= v_0 \\
for\ n &= 1, ..., N \\
remove\ r_n &\propto 1\ to\ get\ the\ old\ posterior \\
(v_\theta^{-n})^{-1} &= v_\theta^{-1} - v_n^{-1} \\
m_\theta^{-n} &= m_\theta + \frac{v_\theta^{-n}}{n}(m_\theta - m_n)
\end{aligned}
$$

Then follow the same ADF steps to recompute $(m_\theta, v_\theta, z_n)$

$$
\begin{aligned}
probability\ of\ estimates: \pi_n &= 1 - \frac{w}{z_n} * N(y_n|0, 10I_D) \\
m_\theta &= m_\theta^{-n} + v_\theta^{-n} + \pi_n\left(\frac{y_n - m_\theta^{-n}}{v_\theta^{-n} + 1}\right) \\
v_\theta &= v_\theta^{-n} - \pi_n\left(\frac{(v_\theta^{-n})^2}{v_\theta^{-n} + 1}\right) + \pi_n(1 - \pi_n)(v_\theta^{-n})^2\left(\frac{||x_n - m_\theta||^2}{D(v_\theta^{-n} + 1)^2}\right) \\
z_n &= (1 - w)N(y_n|m_\theta^{-n}, (v_\theta^{-n} + 1)I_D) + wN(y_n|0, 10I_D)
\end{aligned}
$$

```
## Initialize prior term
m0 <- rep(0, 2)
v0 <- 100
s0 <- (2 * pi * v0)^(-1)
w0 <- 0.5

## Initialize data term
mm <- rep(0, 2)
vv <- Inf
```

```
ss <- 1

## Initialize target variables
mnew <- m0
mold <- m0
vnew <- v0
vold <- v0
sold <- s0
eps <- 1e-5
iter <- 0

while (norm(matrix(m - mold), type = "2") > eps |
       abs(v - vold) > eps |
       abs(s - sold) > eps) {
  iter = iter + 1
  mold = m; vold = v; sold = s
  for (n in 1:NN) {
    v_theta = 1 / ((1 / vnew) - (1 / vv))
    mean_theta = mew + (vnew * (1 / vv) * (mnew - 0))
    Zn = ((1 - w) * dnorm(X, mean_theta, (v_theta + 1))) + (w * dnorm(
      X, pi_0 = 1 - ((w / Zn) * dnorm(X, c(0,0), 10))))
    est_mean = mean_theta + (pi_0 * v_theta * (X - mean_theta) / (v_theta + 1))
    est_var = v_theta - (pi_0 * (v_theta^2) / (v_theta + 1)) +
      (pi_0 * (1 - pi_0) * (v_theta^2) * (((abs(X - mean_theta - mean_theta[1]))^2)) + (D * ((v_theta +
    pi = 1 - 1/2 / z * dmultinom(y, mean = rbind(2, 0),cov = 10)
    m_new = m + v * pi * (y - m) / (v + 1)
    v_new = v - pi * (v^2) / (v + 1) + pi * (1 - pi) * (v^2*sum((y-m)^2))(2*(v+1)**2)
    v_data = 1 / (1 / v_new - 1 / v)
    m_data = m + (v_data + v) / v * (m_new - m)

  }

}
```

```
m_new
```

```
## [1] 4.994564 5.234581
```

```
v_new
```

```
##             [,1]       [,2]
## [1,] 0.01262977 0.00000000
## [2,] 0.00000000 0.01262977
```

2D contour Gaussian Approximation by EP

```
xs3 <- seq(4, 5.5, by = 0.02)
ys3 <- seq(4, 5.5, by = 0.1)

xy3 <- expand.grid(xs3, ys3)
```

```
mulpdf2 <- dmvnorm(xy3, mean = m_new, sigma = v_new, log = FALSE)
Zmat2 <- matrix(mulpdf2, nrow = 76)

filled.contour(xs3, ys3, Zmat2)
```