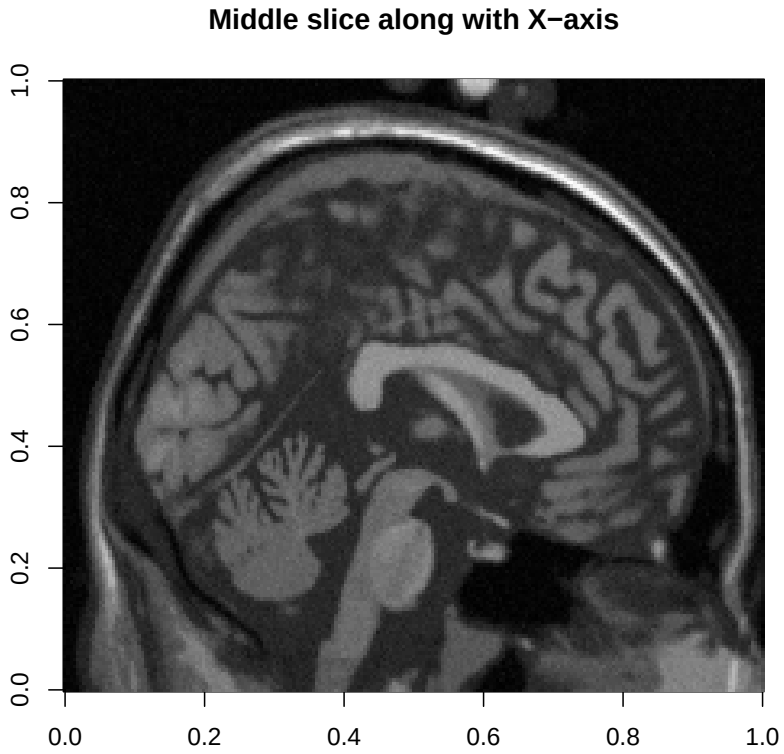
STAT7400 HW6, 2017

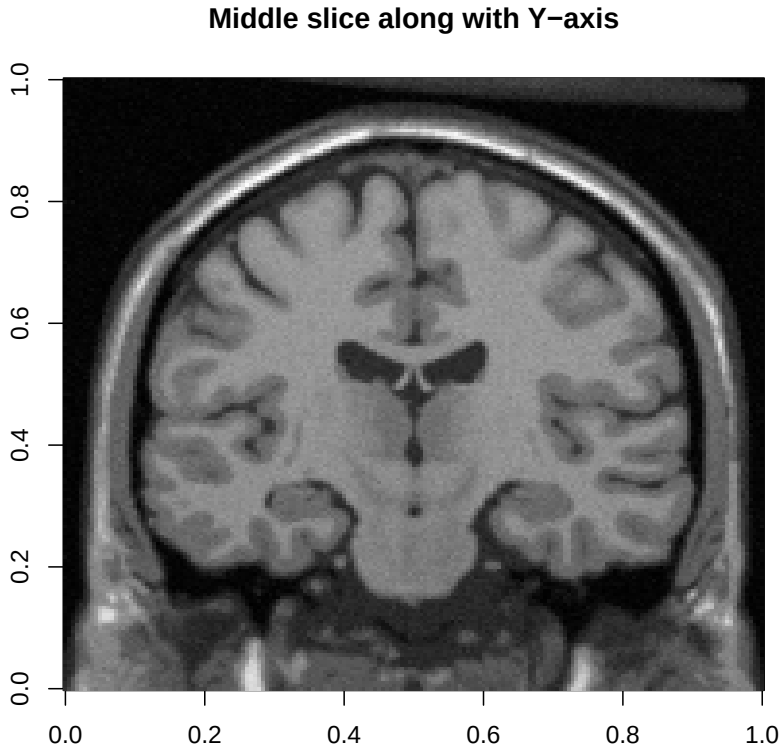
Yiheng Liu

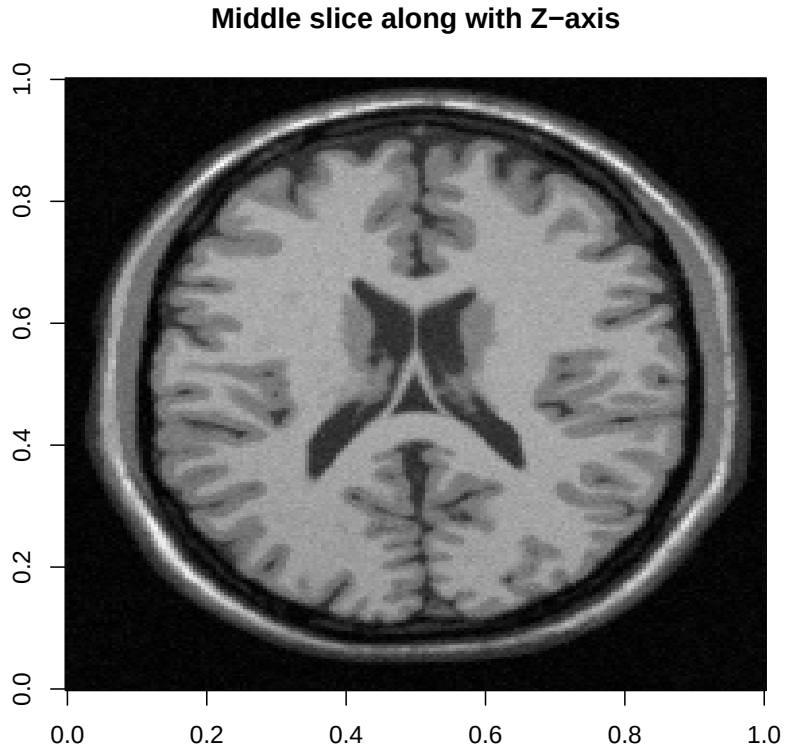
Mar 5, 2017

Problem 2

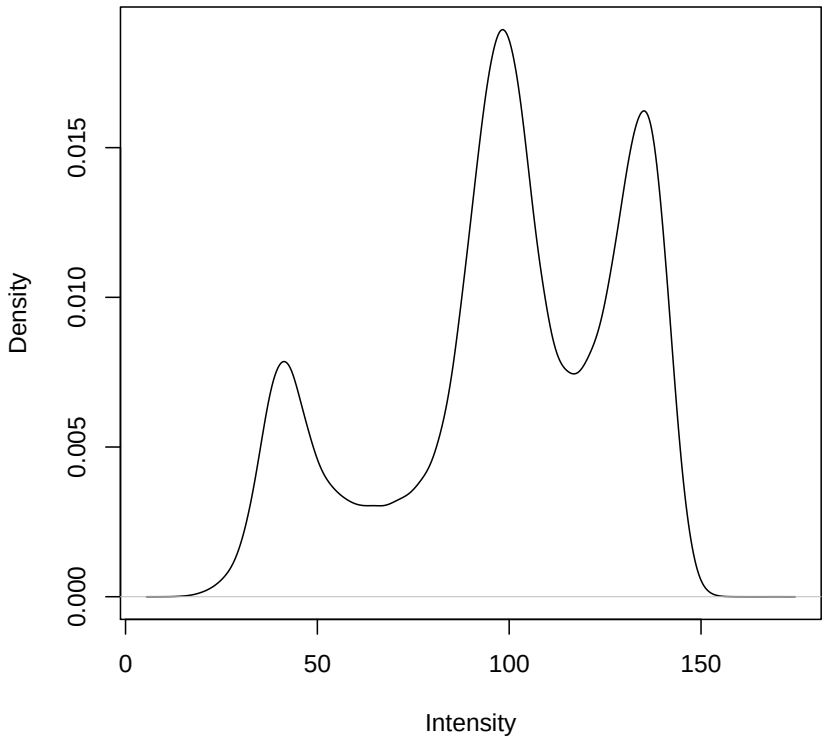
(a) Three middle slice graphs along different axes are given below.




Middle slice along with Y−axis
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0



(b) The plot of the estimate of the density of the image intensities for voxels in the brain is given below.



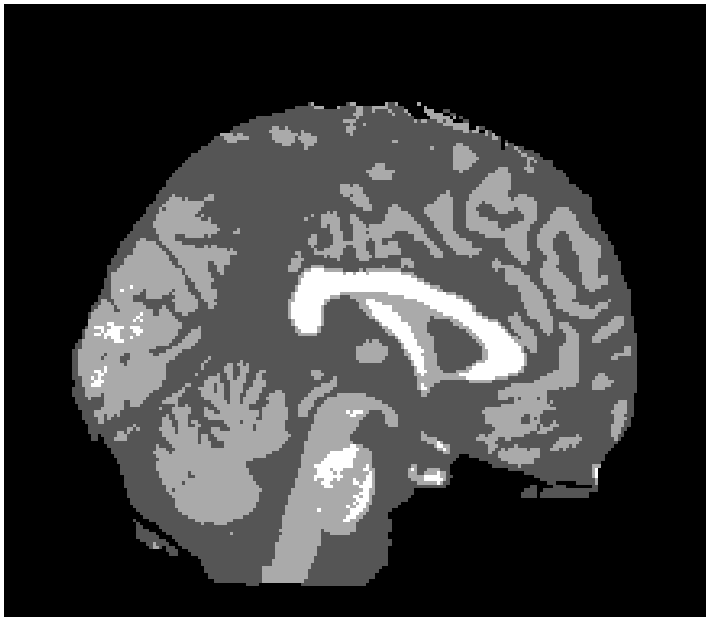
(c) The initial valuse are set by function `initOtsu`in package `mritc`.

```
> EMmix1 <- function(x, theta){
+     mu <- theta$mu
+     sigma <- theta$sigma
+     p <- theta$p
+     M <- length(mu)
+
+     ## E step
+     Ez <- outer(x, 1 : M, function(x, i) p[i] * dnorm(x, mu[i], sigma[i]))
+     Ez <- sweep(Ez, 1, rowSums(Ez), "/")
+     colSums.Ez <- colSums(Ez)
+
+     ## M step
+     xp <- sweep(Ez, 1, x, "*")
+     mu.new <- colSums(xp) / colSums.Ez
+
+     sqRes <- outer(x, mu.new, function(x, m) (x - m) ^ 2)
+     sigma.new <- sqrt(colSums(Ez * sqRes) / colSums.Ez)
+
```

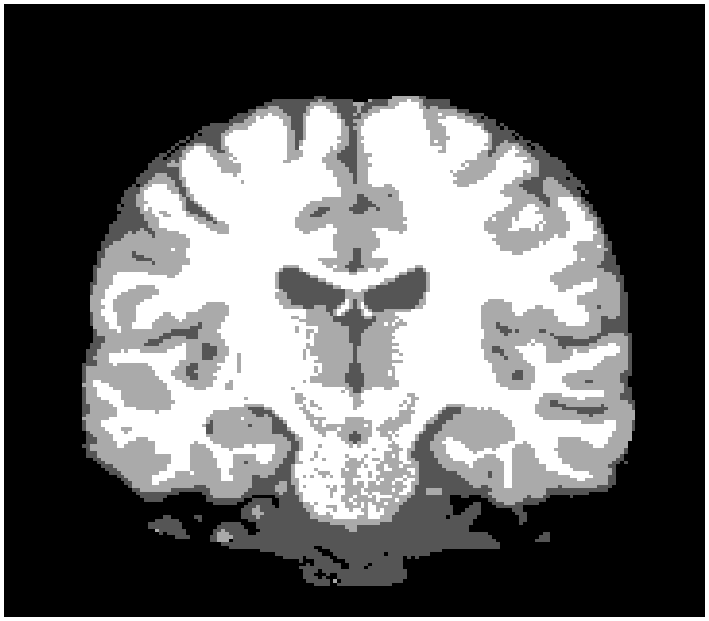
```
+     p.new <- colSums.Ez / sum(colSums.Ez)
+     ## pack up result
+     list(mu = mu.new, sigma = sigma.new, p = p.new)
+ }
> diff <- function(x, y){
+     max(abs(x - y)) / (1 + max(abs(x), abs(y)))
+ }
> converge.check <- function(theta.new, theta,
+                            err, iter, iter.max){
+     diff.mu <- diff(theta.new$mu, theta$mu)
+     diff.sigma <- diff(theta.new$sigma, theta$sigma)
+     diff.p <- diff(theta.new$p, theta$p)
+     flag <- 0
+     if (diff.mu < err[1] && diff.sigma < err[2]
+         && diff.p < err[3] || iter > iter.max)
+         flag <- 1
+     flag
+ }
> EMmix <- function(x, theta, err, iter.max){
+     iter <- 0
+     repeat {
+         iter <- iter + 1
+         theta.new <- EMmix1(x, theta)
+         flag <- converge.check(theta.new, theta, err, iter, iter.max)
+         if (flag) break
+         theta <- theta.new
+     }
+     list(mu = theta.new$mu, sigma = theta.new$sigma, p = theta.new$p)
+ }
> library(mritc)
> init <- initOtsu(T1.mask, 2)
> theta <- list(mu=init$mu, sigma = init$sigma, p = init$prop)
> err <- rep(1e-08, 3)
> iter.max <- 200
> theta.est1 <- EMmix(T1.mask, theta, err, iter.max)
> data.frame(mu = theta.est1$mu, sigma = theta.est1$sigma, p = theta.est1$p)

         mu     sigma         p
1  45.76594 10.223615 0.1766328
2  98.98909 13.719448 0.5502270
3 133.95977  6.460461 0.2731402
```

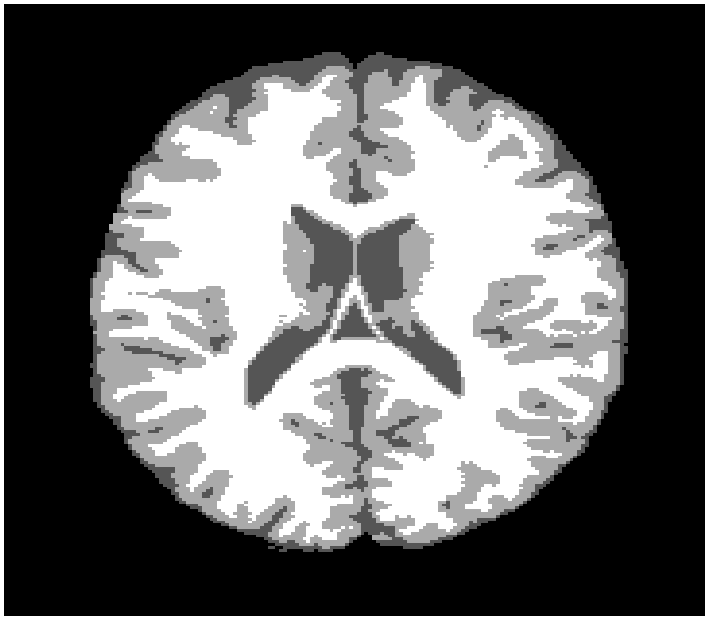
(d) Based on the classfication results, three graphs are shown below.

Middle slice along with
X−axis



Middle slice along with
Y−axis





(e) By `summaryRprof`, we can see that most time is spent on `EMmix1` and `outer`.

```
> Rprof(tmp <- tempfile())
> theta.est1 <- EMmix(T1.mask, theta, err, iter.max)
> Rprof()
> summaryRprof(tmp)

$by.self
                self.time self.pct total.time total.pct
".External"         57.84    34.88      57.84     34.88
"array"             24.32    14.67      24.32     14.67
"dnorm"             19.70    11.88      77.54     46.76
"aperm.default"     11.84     7.14      11.84      7.14
"-"                 11.08     6.68      11.08      6.68
"outer"              9.30     5.61     108.80     65.61
"sweep"              8.20     4.95      50.72     30.59
"FUN"                7.62     4.60      99.50     60.00
"rowSums"            5.80     3.50       5.80      3.50
"*"                  4.58     2.76       4.58      2.76
"colSums"            3.30     1.99       6.30      3.80
```

```
"^"                  1.68     1.01       1.68      1.01
"aperm"              0.52     0.31      36.68     22.12
"%%"                 0.02     0.01       0.02      0.01
"parent.frame"       0.02     0.01       0.02      0.01

$by.total
                total.time total.pct self.time self.pct
"<Anonymous>"       165.82    100.00      0.00     0.00
"doTryCatch"        165.82    100.00      0.00     0.00
"EMmix"             165.82    100.00      0.00     0.00
"EMmix1"            165.82    100.00      0.00     0.00
"eval"              165.82    100.00      0.00     0.00
"evalFunc"          165.82    100.00      0.00     0.00
"try"               165.82    100.00      0.00     0.00
"tryCatch"          165.82    100.00      0.00     0.00
"tryCatchList"      165.82    100.00      0.00     0.00
"tryCatchOne"       165.82    100.00      0.00     0.00
"withVisible"       165.82    100.00      0.00     0.00
"outer"             108.80     65.61      9.30     5.61
"FUN"                99.50     60.00      7.62     4.60
"dnorm"              77.54     46.76     19.70    11.88
".External"          57.84     34.88     57.84    34.88
"sweep"              50.72     30.59      8.20     4.95
"aperm"              36.68     22.12      0.52     0.31
"array"              24.32     14.67     24.32    14.67
"aperm.default"      11.84      7.14     11.84     7.14
"-"                  11.08      6.68     11.08     6.68
"colSums"             6.30      3.80      3.30     1.99
"rowSums"             5.80      3.50      5.80     3.50
"*"                   4.58      2.76      4.58     2.76
"is.data.frame"       3.00      1.81      0.00     0.00
"^"                   1.68      1.01      1.68     1.01
"%%"                  0.02      0.01      0.02     0.01
"parent.frame"        0.02      0.01      0.02     0.01
"match.fun"           0.02      0.01      0.00     0.00

$sample.interval
[1] 0.02

$sampling.time
[1] 165.82
```

(f) Since the intensities can take on only a small number of distinct values, we should only compute the values for unique intensity values and then multiply these values by the counts of each unique intensity value. The improved EM function is as

follows.

```
> EMmix1c <- function(x.c, c, theta){
+     mu <- theta$mu
+     sigma <- theta$sigma
+     p <- theta$p
+     M <- length(mu)
+     ## E step
+     Ez <- outer(x.c, 1 : M, function(x, i) p[i] *
+                     dnorm(x, mu[i], sigma[i]))
+     Ez <- sweep(Ez, 1, rowSums(Ez), "/")
+     Ez.c <- sweep(Ez, 1, c, "*")
+     colSums.Ez.c <- colSums(Ez.c)
+     ## M step
+     xp.c <- sweep(Ez.c, 1, x.c, "*")
+     mu.new <- colSums(xp.c) / colSums.Ez.c
+     sqRes.c <- outer(x.c, mu.new, function(x, m) (x - m) ^ 2)
+     sigma.new <- sqrt(colSums(Ez.c * sqRes.c) / colSums.Ez.c)
+     p.new <- colSums.Ez.c / sum(colSums.Ez.c)
+     ## pack up result
+     list(mu = mu.new, sigma = sigma.new, p = p.new)
+ }
> EMmixc <- function(x, theta, err, iter.max){
+     iter <- 0
+     c <- table(x)
+     x.c <- sort(unique(x))
+     repeat {
+         iter <- iter + 1
+         theta.new <- EMmix1c(x.c, c, theta)
+         flag <- converge.check(theta.new, theta, err, iter, iter.max)
+         if (flag) break
+         theta <- theta.new
+     }
+     theta.new
+ }
```

Then we fit the normal mixture model again with the improve EM algorithm, the results are exactly the same as before.

```
> init <- initOtsu(T1.mask, 2)
> theta <- list(mu=init$mu, sigma = init$sigma, p = init$prop)
> err <- rep(1e-08, 3)
> iter.max <- 200
> theta.est2 <- EMmixc(T1.mask, theta, err, iter.max)
> data.frame(mu.c = theta.est2$mu, sigma.c = theta.est2$sigma,
+            p.c = theta.est2$p)
```

```
       mu.c   sigma.c       p.c
1  45.76594 10.223615 0.1766328
2  98.98909 13.719448 0.5502270
3 133.95977  6.460461 0.2731402
```

By summaryRprof, we can see that the improved algorithm is quite faster than before.

```
> Rprof(tmp <- tempfile())
> theta.est2 <- EMmixc(T1.mask, theta, err, iter.max)
> Rprof()
> summaryRprof(tmp)

$by.self
                 self.time self.pct total.time total.pct
"as.character"        0.30    57.69       0.30     57.69
"table"               0.08    15.38       0.44     84.62
"match"               0.04     7.69       0.04      7.69
"unique.default"      0.04     7.69       0.04      7.69
"sweep"               0.02     3.85       0.06     11.54
"any"                 0.02     3.85       0.02      3.85
"match.fun"           0.02     3.85       0.02      3.85

$by.total
                 total.time total.pct self.time self.pct
"<Anonymous>"          0.52    100.00      0.00     0.00
"doTryCatch"           0.52    100.00      0.00     0.00
"EMmixc"               0.52    100.00      0.00     0.00
"eval"                 0.52    100.00      0.00     0.00
"evalFunc"             0.52    100.00      0.00     0.00
"try"                  0.52    100.00      0.00     0.00
"tryCatch"             0.52    100.00      0.00     0.00
"tryCatchList"         0.52    100.00      0.00     0.00
"tryCatchOne"          0.52    100.00      0.00     0.00
"withVisible"          0.52    100.00      0.00     0.00
"table"                0.44     84.62      0.08    15.38
"factor"               0.36     69.23      0.00     0.00
"as.character"         0.30     57.69      0.30    57.69
"sweep"                0.06     11.54      0.02     3.85
"EMmix1c"              0.06     11.54      0.00     0.00
"match"                0.04      7.69      0.04     7.69
"unique.default"       0.04      7.69      0.04     7.69
"aperm.default"        0.04      7.69      0.00     0.00
"aperm"                0.04      7.69      0.00     0.00
"order"                0.04      7.69      0.00     0.00
"unique"               0.04      7.69      0.00     0.00
```

```
"any"                   0.02      3.85      0.02      3.85
"match.fun"             0.02      3.85      0.02      3.85
"lapply"                0.02      3.85      0.00      0.00
"sort"                  0.02      3.85      0.00      0.00
"unlist"                0.02      3.85      0.00      0.00

$sample.interval
[1] 0.02

$sampling.time
[1] 0.52
```