Homework 7
Computer Intensive Statistics, Spring 2017

Alex Zajichek

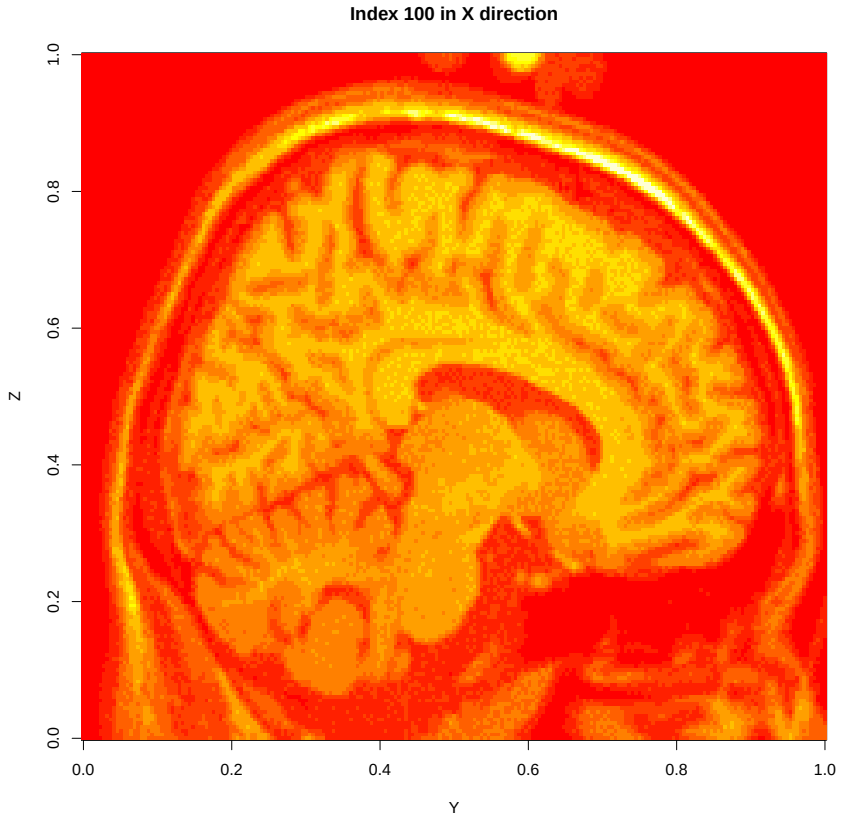
Due March 5, 2017

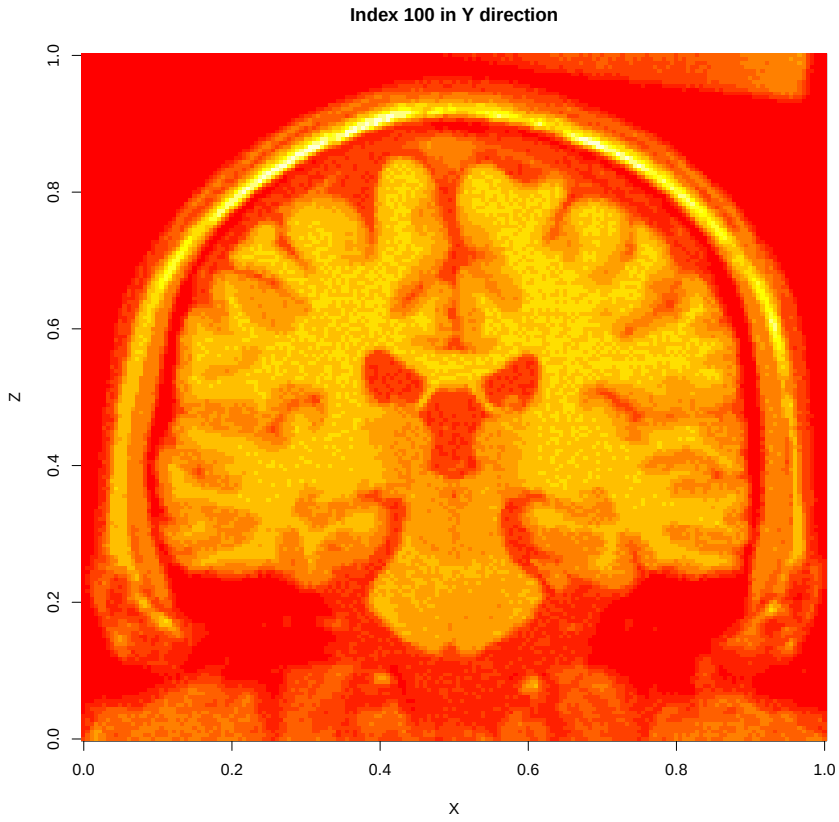
*All R code is in the separate file `Brains.R`

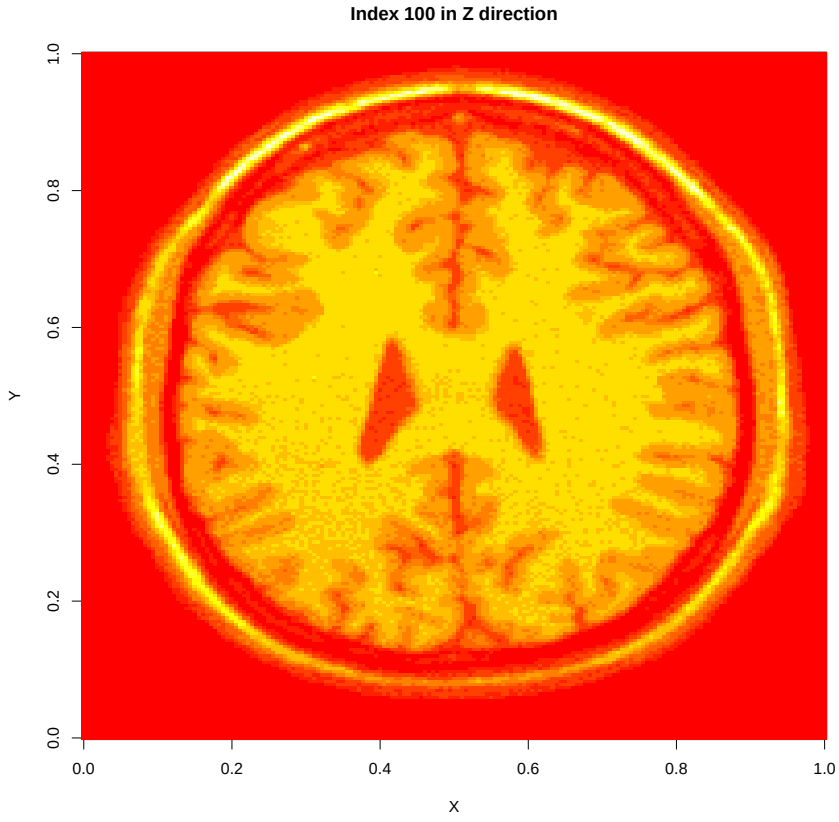
Normal mixture model on brain image via EM algorithm

(a)

Cross-sections at the 100^{th} index of each dimension was chosen to display a slice of the brain image.

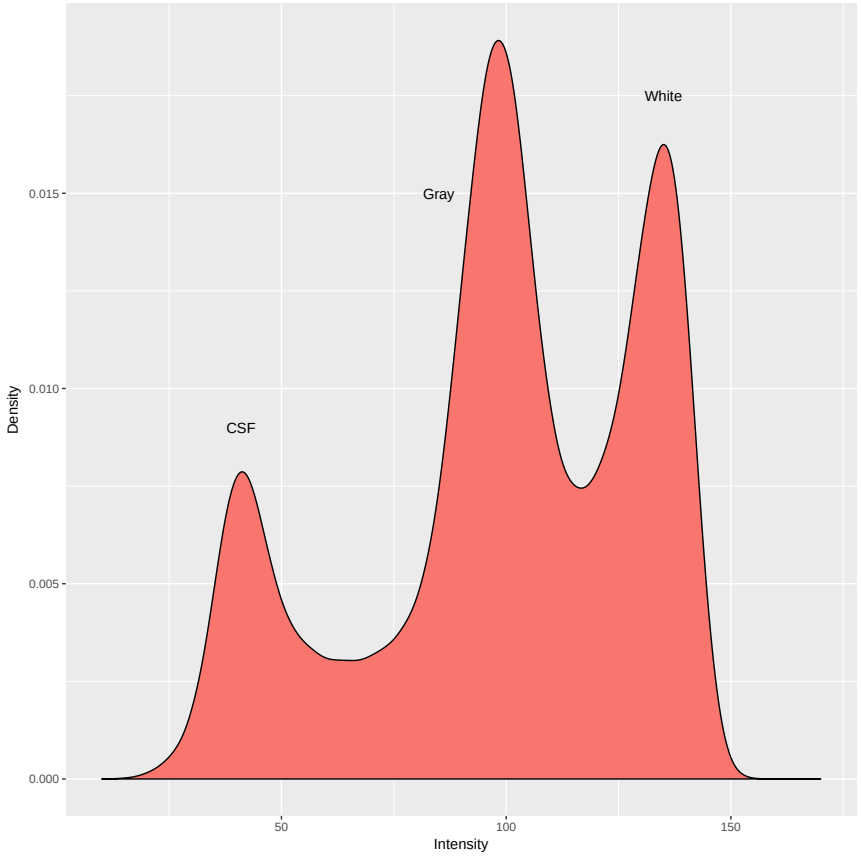



Index 100 in Y direction
Z
X
0.0
0.2
0.4
0.6
0.8
1.0


Index 100 in Z direction
Y
X
0.0
0.2
0.4
0.6
0.8
1.0

(b)

`ggplot2` was used to plot a density of the intensities of brain matter.



(c)

The `EMmix1` function from the class notes was used to obtain parameter estimates. The above density plot was examined to input reasonable starting values. The table below gives the results:

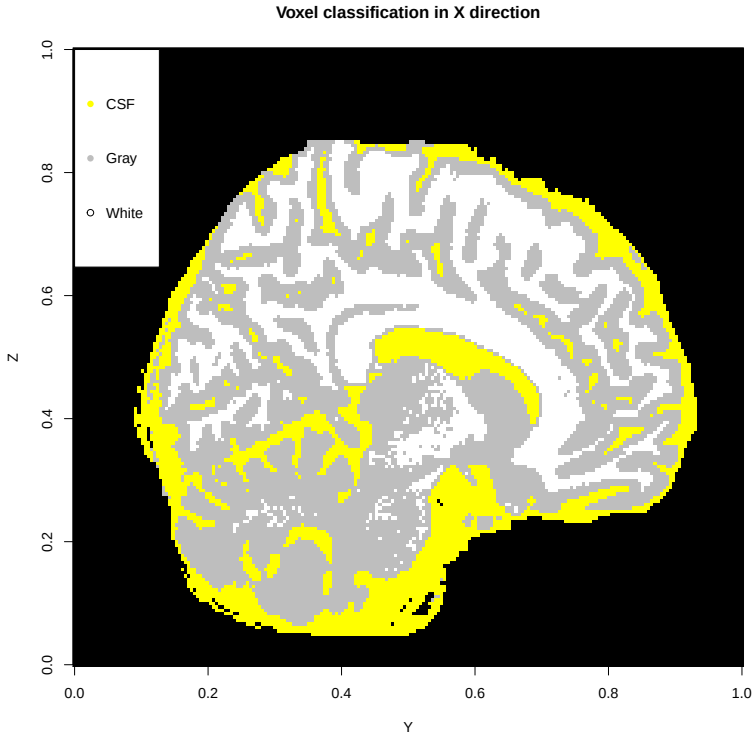
	CSF	Gray	White
$\hat{\mu}$	47.497	97.228	131.826
$\hat{\sigma}$	11.565	10.715	7.961
$\hat{p}$	0.192	0.478	0.331

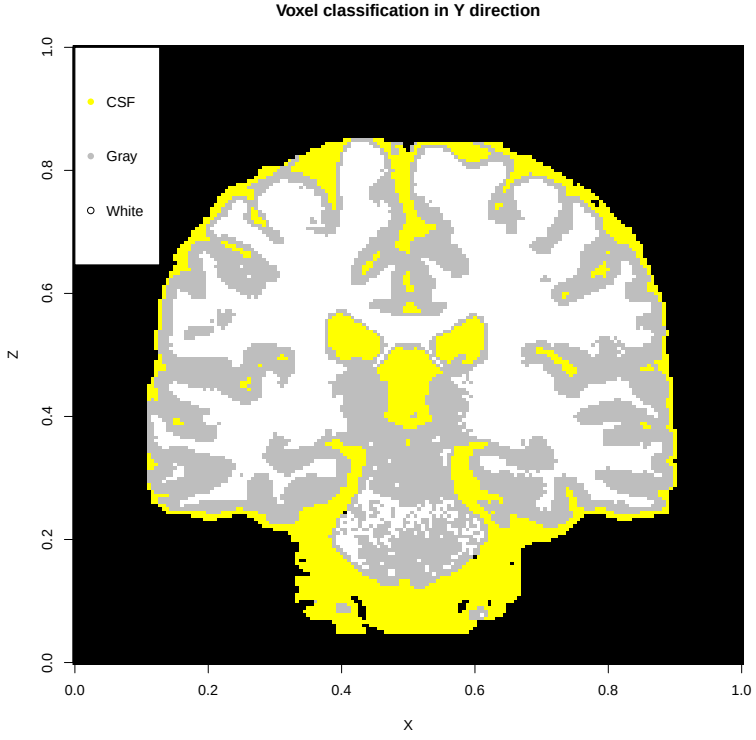
(d)

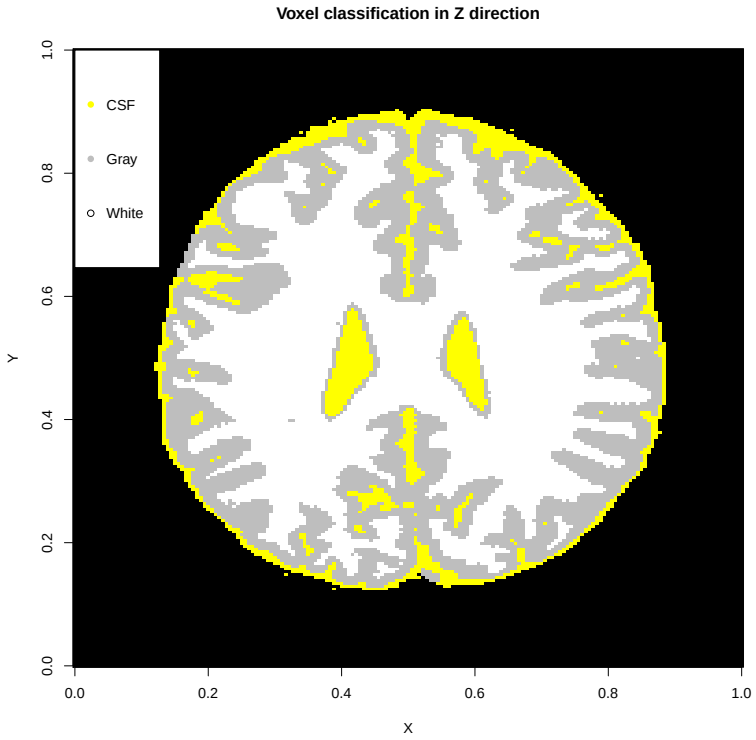
The function written is called `classify_brain`, and is in the corresponding `.R` file (as well as pasted below).

```
EM <- list(mu = c(47.49701, 97.22827, 131.82621),
sigma = c(11.564902, 10.715485, 7.961161), p = c(0.1916802, 0.4775411, 0.3307788))

classify_brain <- function(x, theta) {
  which.max(theta$p*dnorm(x, theta$mu, theta$sigma))
}

classifications <- sapply(brain_only, classify_brain, theta = EM)
```

It classifies a given intensity into a tissue type by finding the maximum of each class probability multiplied by the corresponding normal density evaluated at the intensity. Once classifications were obtained, the were placed back into the original array structure of the brain image in order to plot the results using the `image` function. The images below show the same cross-sections as in part (a), but the colors now correspond to the predicted type of brain matter from the normal mixture model.







(e)

By using the `Rprof` and `summaryRprof` functions, we could determine the `dnorm` calls were taking the most time to complete while running the `EMmix1` function.

```
Rprof()
EMmix1(brain_only,theta)
summaryRprof() #Spending most time in 'dnorm'
```

(f)

Yes. In the $E - Step$ of the algorithm, instead of calculating the product of the class probabilities and the normal density for every observation in the data, we can simply just do that for the unique values. We can keep track of how many of each unique value there are, and then multiply each weight by that value. This will significantly decrease the number of the calculations being made in the process.