Portfolio 2, Study Group 10

Jesper Fischer, Lasse Hansen & Sarah Hvid

10/2/2020

```
pacman::p_load(tidyverse, jpeg)
```

1.a Make a constant vector of the same length as the data, consisting of ones.

```
#First we will type in the first two vectors of the dataset
Reaction372<-c(269.41, 273.47, 297.60, 310.63, 287.17, 329.61, 334.48, 343.22, 369.14, 364.12)
Days372<-c(0,1,2,3,4,5,6,7,8,9)
#Thereafter using the rep() function we will make a new vector that contains 10 number 1's
constant <- rep(1,10)
```

1.b: Report the inner product (aka dot product) of the days vector and the constant vector.

```
#The inner product of two vectors is calculated like this:
sum(Days372 * constant)
```

```
## [1] 45
```

Therefore the inner product of the days and constant vector = 45.

1.c: What does the dot product say about the possibility of finding an optimal linear regression?

The dot product is the part of the vectors magnitudes multiplied, multiplied by cos(angle between them). Therefore the value of the cross product speaks to how correlated our two vectors are. Since the dot product is larger than 1 the two vectors are somewhat correlated. This means dot product > 0 -> angle less than 90 degrees between the two vectors. Therefore there is a reason to believe that we can make an optimal linear regression.

1.d: Create a 10x2 matrix called X with the days vector and constant vector as columns and use the least squares method manually to find the optimal coefficients (i.e. slope and intercept) to reaction time.

```
#First we make a matrix of the constant and the number of days
x <- matrix(c(constant, Days372), ncol = 2)

#Then we calculate the coefficients, with the formula of the least squares method
coefficients <- solve(t(x) %*% x) %*% t(x) %*% Reaction372
coefficients
```

```
##         [,1]
## [1,] 267.044
## [2,]  11.298
```

The optimal coefficients to reaction time as a function of days of sleep deprivation; Intercept = 267,044, Slope = 11,298.

1.e: Check result using lm(). Use the formula lm(Reaction372~0+X) - the zero removes the default constant.

```
#A linear model is made using the lm() function
coef.calc <- lm(Reaction372~0+x)
coef.calc
```

```
##
## Call:
## lm(formula = Reaction372 ~ 0 + x)
##
## Coefficients:
##    x1     x2
## 267.0   11.3
```

As can be seen from the method above, the coefficients calculated from the lm() method are the same rounded up.

1.f: Subtract the mean of Days372 from the Days372 vector. Replace the days vector with the new vector in X and redo the linnear regression. Did the coefficients change?

```
#First we substract the mean of days from the days vector
new.days <- Days372 - mean(Days372)
#Then a new matrix is made with the substractet mean of days vector
x1 <- matrix(c(constant, new.days), ncol = 2)
#Then the coefficients are calculated using the formula of the least squares method
coefficients1 <- solve(t(x1) %*% x1) %*% t(x1) %*% Reaction372
coefficients1
```

```
##          [,1]
## [1,] 317.885
## [2,]  11.298
```

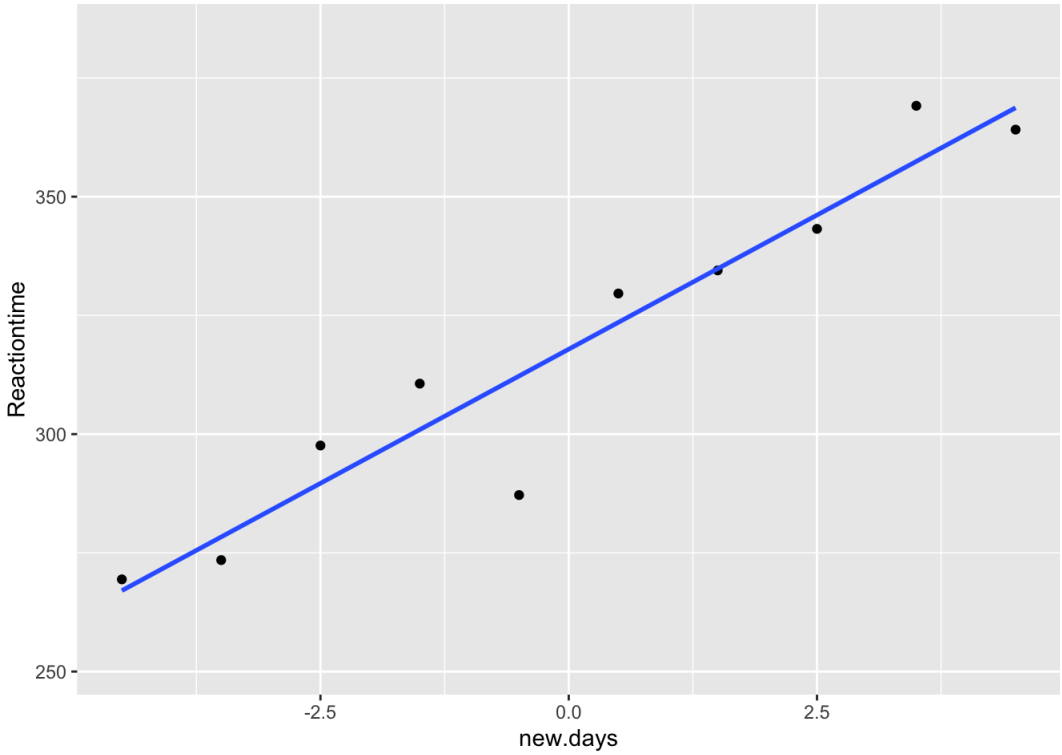
```
coef.calc1 <- lm(Reaction372~0+x1)
coef.calc1
```

```
##
## Call:
## lm(formula = Reaction372 ~ 0 + x1)
##
## Coefficients:
##   x11    x12
## 317.9   11.3
```

The slope remained the same, however the intercept changed, because we centered the days around 0. the intercept is now the mean of reaction time, and the value of x is no longer in days.

1.g: Make a scatter plot with the mean-centered days covariate against response time and add the best fitted line.

```
#First we will make a dataframe with the new data
new.df <- as.data.frame(new.days)
new.df$Reactiontime <- Reaction372
#Then we will plot the mean-centered days as a function of the response time using ggplot
ggplot(new.df, aes(x = new.days, y = Reactiontime))+geom_point()+geom_smooth(method = "lm", alpha=0)
```



2. Images and matrices

Loading the data imaga:

```
matrix <- readJPEG('portfolio_assignment2_matrices_data.jpg', native = FALSE)
```

2.a: report how many rows and how many columns the matrix has. What are the maximun, minimum and

mean pixel values?

```
#Columnns and rows
attributes(matrix)
```

```
## $dim
## [1] 900 606
```

```
#Mean
mean(matrix)
```

```
## [1] 0.5118474
```

```
#Max
max(matrix)
```

```
## [1] 1
```

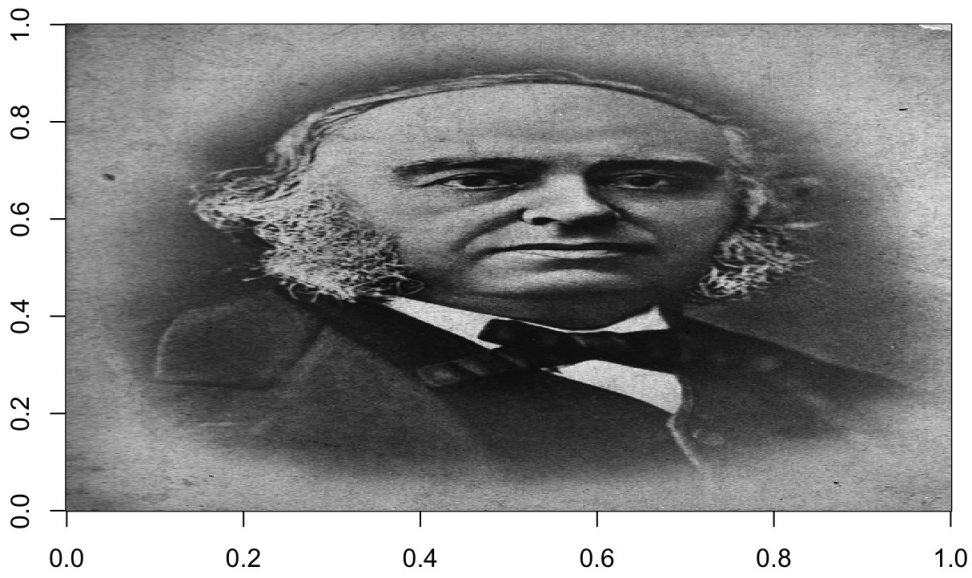
```
#Min
min(matrix)
```

```
## [1] 0.0627451
```

The matrix has 900 rows and 606 columns. Furthermore, the mean value of the pixels are 0.51 (rounded), the max value is 1 and the minimum value is 0.06 (rounded).

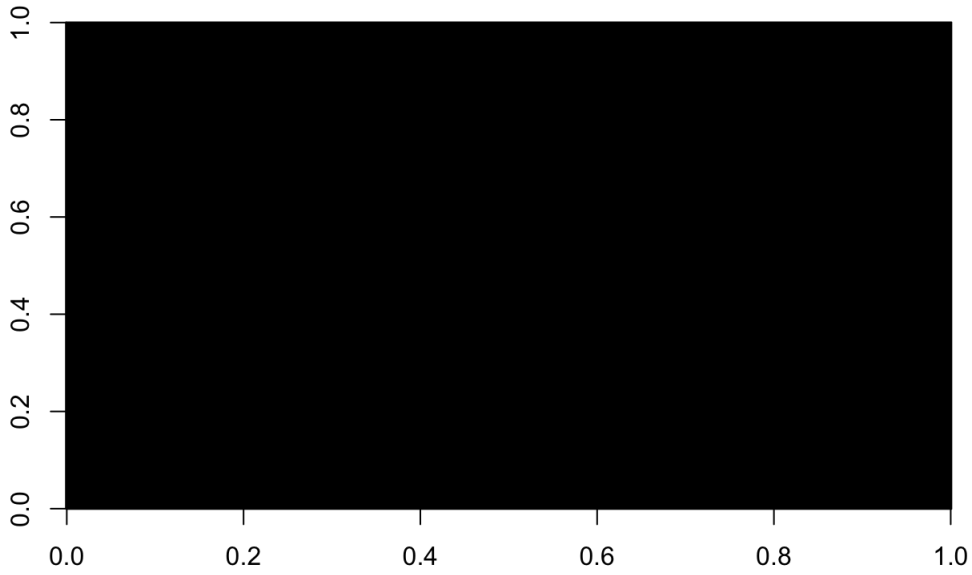
2.b: Make an image of the loaded matrix. Be sure to rotate the image into the correct orientation. Furthermore, grey scale the picture with gray(1:100/100) - this will color values near 0 black/dark and values near 1 white/light.

```
#Creating a function to rotate a matrix, so the image will be shown correctly
rotate <- function(x) t(apply(x, 2, rev))
#Rotating the matrix
matrix2 <- rotate(matrix)

#Converting rotated matrix into an image, using gray colours
image(matrix2,col=gray(1:100/100))
```



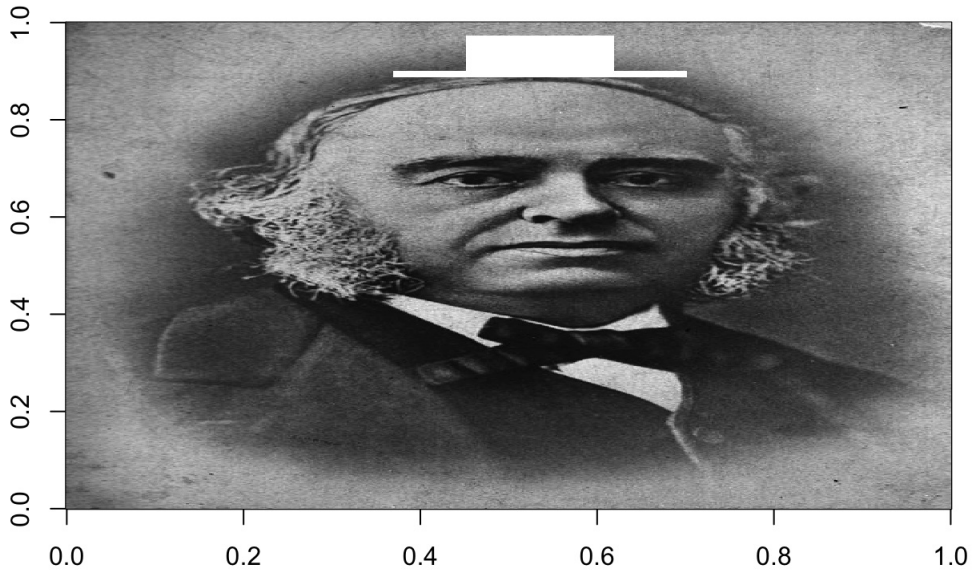
2.c: Draw an image with the same dimensions as that from 2.b. But this image should be completely black

```
# setting the greyscale to colour everything black
image(matrix2,col=gray(0))
```



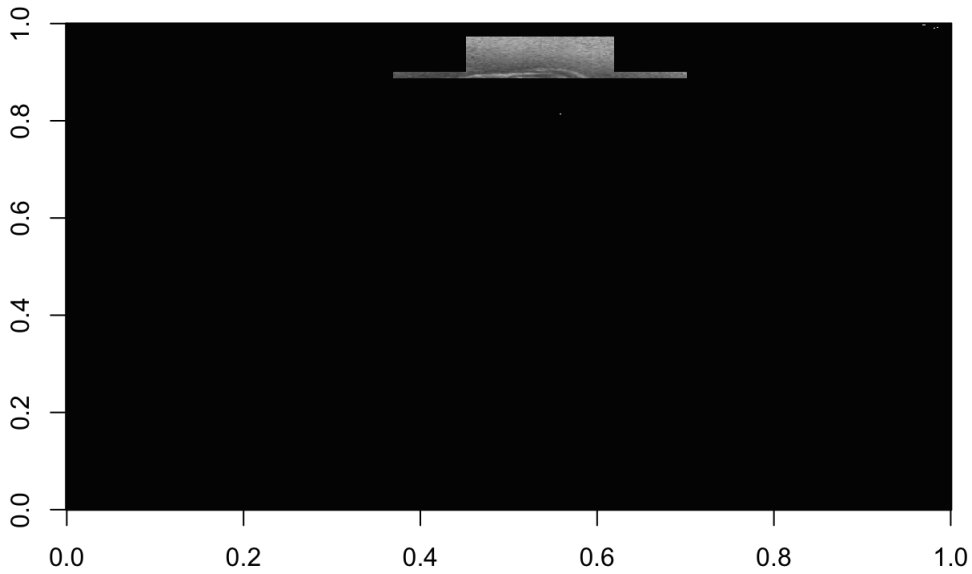
2.d: Draw a white hat on the image from 2.b (hint: use ones).

```
# creating a new matrix
matrix3 <- rotate(matrix)

# calling on specific cells in the matrix to be renamed to 1 (white, when image is drawn)
matrix3[275:375, 800:875] <- 1
matrix3[225:425, 800:810] <- 1

# imaging the new matrix with the hat
image(matrix3,col=gray(0:100/100))
```



2.e: Make an image which has the same dimensions as 2.b., and which only contains the parts which was hidden behind the hat in 2.d. The rest should be black so the hat has to be grayscale original pic.

```
# Changing pixels to black around our hat drawing, so if the pixel value is not 1 make it 0.
img.zero <- ifelse(matrix3 < 1, 0, matrix3)
img.yes <- rotate(matrix)

#Multiplying the black and white hat image with the original image, so that pixels will say 1 * x = x (where x is the original pixel value).
image(img.zero*img.yes, col = gray(1:100/100))
```

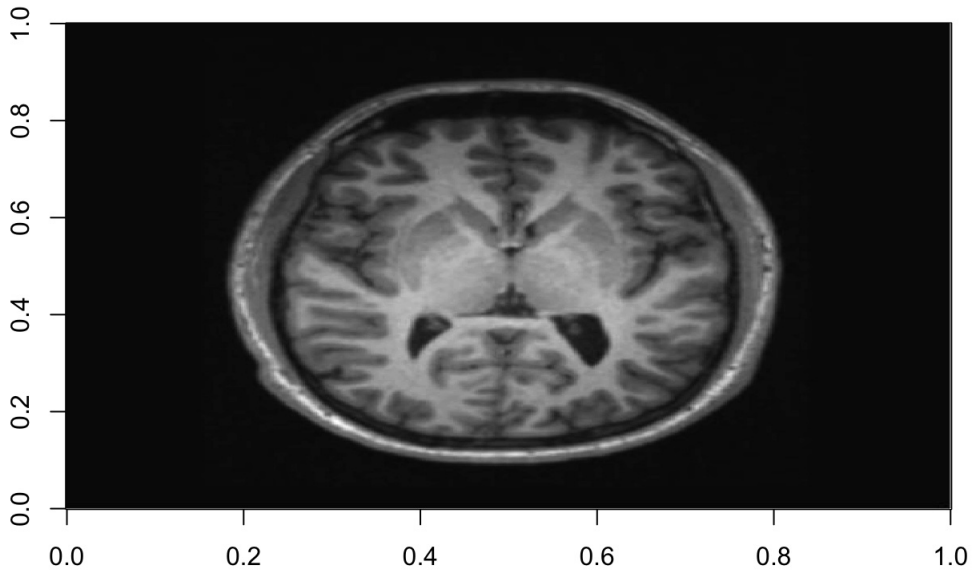


3

```
#Loading the picture of the brain
Brain <- readJPEG('portfolio_assignment2_matrices_data2.jpg', native = FALSE)
```

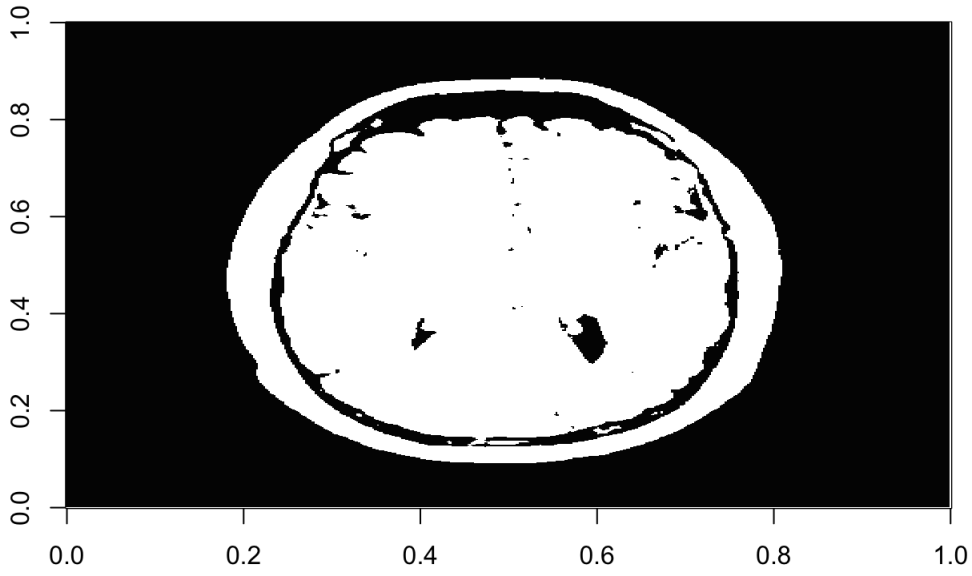
3.a: Make an image of the brain.

```
# Rotating matrix and creating image
brain <- rotate(Brain)
image(brain, col = gray(1:100/100))
```



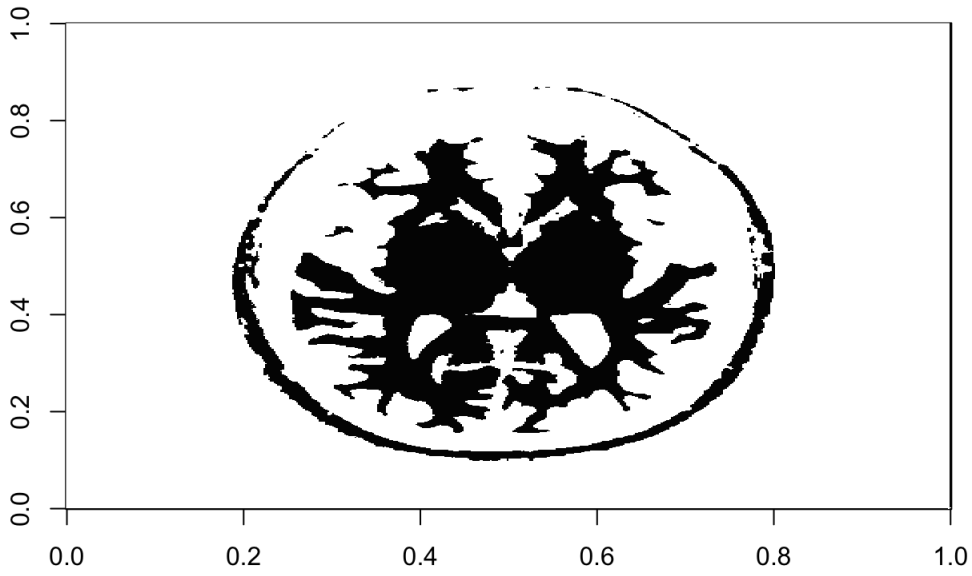
3.b: We will attempt to find the interesting areas of this brain image, e.g. only areas with gray matter.

```
# Mask1 says that if the pixel value is bigger than the mean value of the pixels, make it white, otherwise m
ake it black.
mask1 <- ifelse(brain > mean(brain), 1, 0)
image(mask1, col = gray(1:100/100))
```



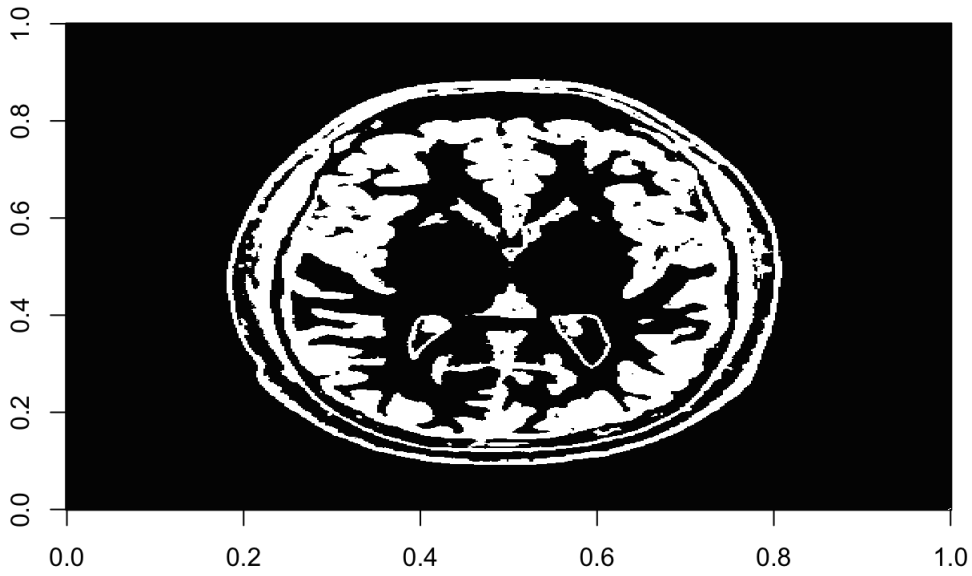
3.c: Make an image which is white where the pixel values of the brain image are smaller than 2.5 times the mean value of the whole image. Call this matrix mask2

```
# Mask2 says that if the pixel value is smaller than the mean pixel value * 2,5 then make it white, otherwis
e make it black
mask2 <- ifelse(brain < 2.5 * mean(brain), 1, 0)
image(mask2, col = gray(1:100/100))
```



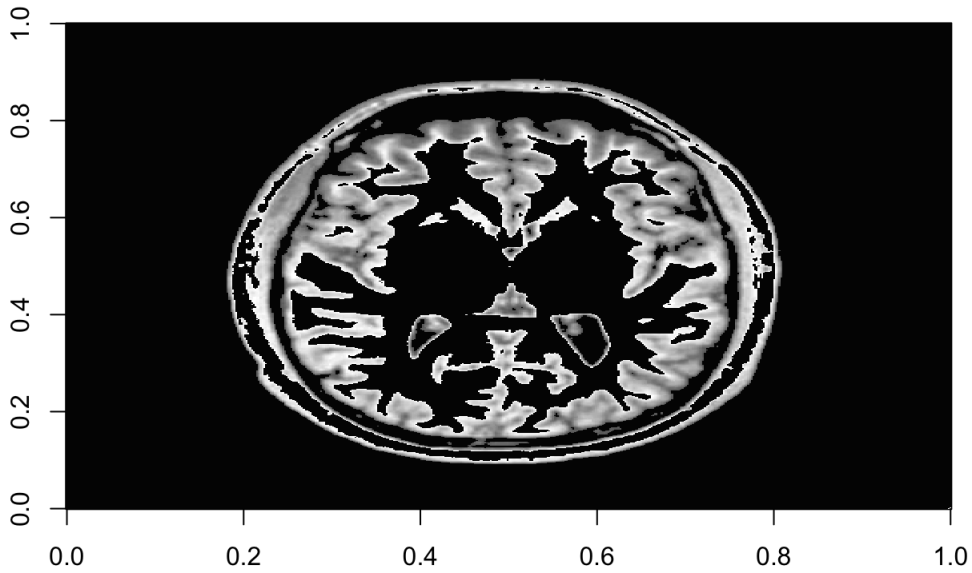
3.d: Convert mask1 and mask2 into one mask with ones where the two masks overlap and zeros everywhere else. What type mathematical procedure can be used to produce this?

```
# The final mask says that if the pixel value is 1 in mask1 and mask2 which is calculated by multiplying the
m, they will stay 1. however, if there is a 0 value, that pixel will be 0.
mask <- mask1 * mask2
image(mask, col = gray(1:100/100))
```



3.e. Use the combined mask on the brain image to give you an image with only the image values where the mask has ones, and zeros everywhere else. Did we successfully limit our image to only contain gray matter?

```
#We will time the combination of mask1 and mask2 with the image of the brain. To limir our image to only con
tain gray matter
final <- mask * brain
image(final, col = gray(1:100/100))
```



Yes we succesfully did it.

3.e: Count the number of pixels in the combined mask.

```
sum(mask)
```

```
## [1] 50004
```

there are 50004 pixels left with color in them.

4.a: In the friday bar, men were three times as likely as women to buy beer. A total of 116 beers were sold. Women were twice as likely as men to buy wine. 92 glasses of wine were sold. How many men and women attended the Friday bar?

```
#First we will start by turning the two statemaents into two equation one for beers and one for wine:
#Beers: 116 = 3M+W
#Wine: 92 = 2W+M

#Afterwards i will put this into a matrix and a vector:
M1 <- matrix(c(3,1,1,2),ncol = 2)
M2 <- c(116,92)
#This can be solved:
x <- solve(M1)%*%M2
x
```

```
##      [,1]
## [1,]   28
## [2,]   32
```

This means that there are 28 men and 32 women.