Portefolio 2

Astrid Olling, Daniel Christensen, Karoline Klitgaard, Pernille Jensen, Søren Orm

18/2/2020

```
library(jpeg)
```

##1. Linear regression Participant 372 from the sleepstudy has the following data:

```
#defining data
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)
```

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

```
#making a constant vector of 1's
con_vec <- c(rep(1, length(reaction372)))
```

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

```
#finding the inner product
dot1 <- days372 %*% con_vec

dot1
```

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

- The inner product for the two matrices is 45.

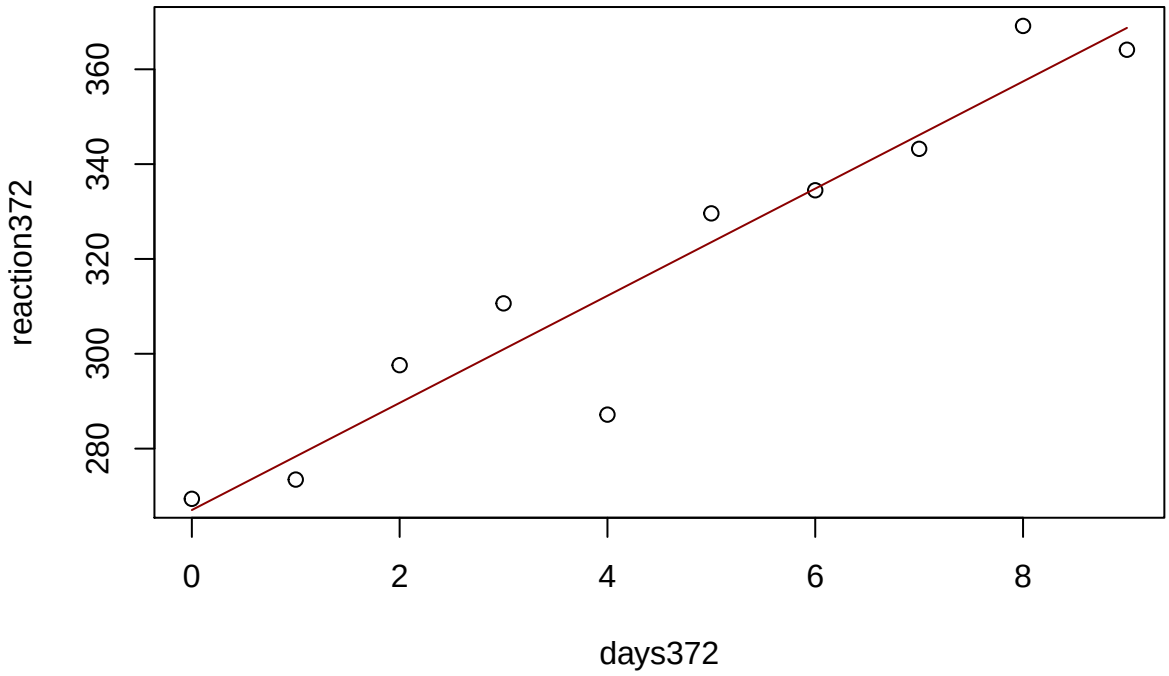
1.c: What does the dot product say about the possibility of finding an optimal linear regression? - The two vectors are not othogonal, meaning they're not completely un-correlated and could be explaning some of same variance, which should be kept in mind when making the regression.

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.

```
#creating a matrix
x <- matrix(c(days372, con_vec), ncol = 2)

#finding the coefficients
beta <- solve(t(x) %*% x) %*% t(x) %*% reaction372
beta
```

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

```r
#making a plot
plot(days372, reaction372, type = 'p')

#drawing the line
lines(
  c(days372[1], days372[10]),
  c(beta[2] + beta[1] * days372[1], beta[2] + beta[1] * days372[10]),
  col = 'darkred'
)
```



- The optimal slope is found to have a value of 11.298, and the optimal intercept is found to have a value of 267.044.

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

```r
#making the linear model to check if the coefficients are the same
checking <- lm(reaction372~0+x)

summary(checking)
```

```
##
## Call:
## lm(formula = reaction372 ~ 0 + x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -25.066  -4.182   1.007   7.489  11.712
##
## Coefficients:
##    Estimate Std. Error t value Pr(>|t|)
## x1   11.298      1.242   9.093 1.72e-05 ***
## x2  267.044      6.633  40.261 1.59e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.29 on 8 degrees of freedom
## Multiple R-squared:  0.999,  Adjusted R-squared:  0.9988
## F-statistic:  4009 on 2 and 8 DF,  p-value: 9.874e-13
```

- The coeficients of the two linear models are found to be equal.

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? (we will return to why this happened in a later class, but if you are curious, you can check this website out: https://www.theanalysisfactor.com/center-on-the-mean/)

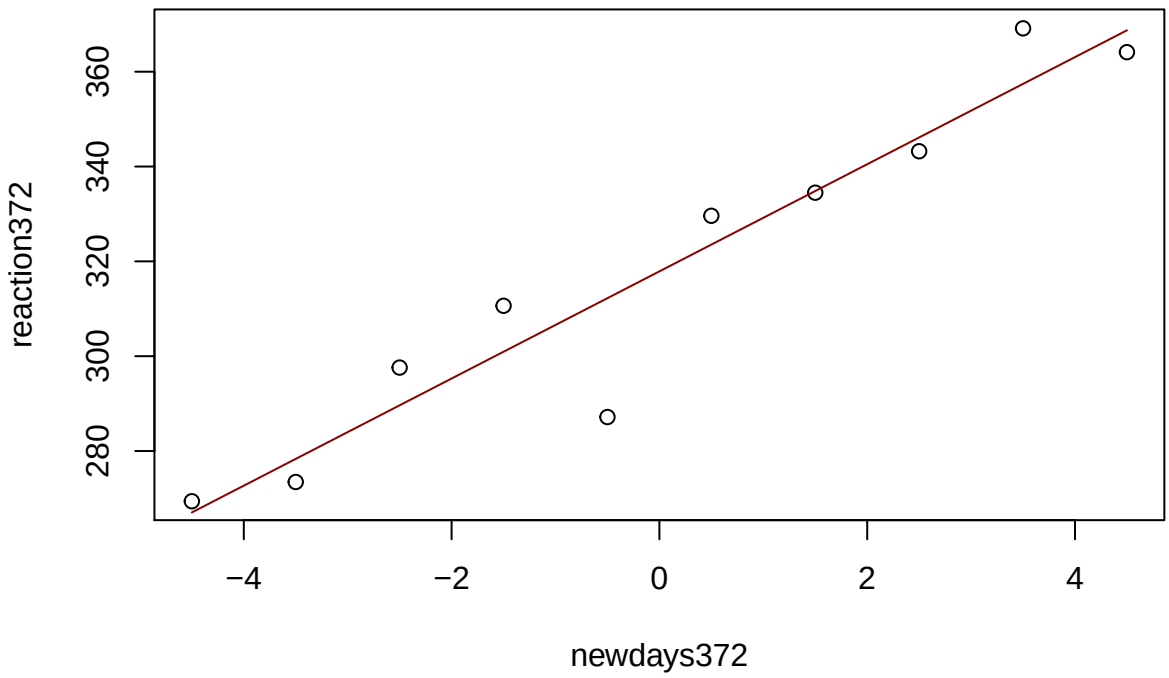
```
#removing the mean from the days
newdays372 <- days372 - mean(days372)

#creating a new matrix
x2 <- matrix(c(newdays372, con_vec), nrow = 10)

#finding the new coefficients
beta2 <- solve(t(x2) %*% x2) %*% t(x2) %*% reaction372
beta2
```

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

The slope did not change, but the intercept changed from 267.044 to 317.885.

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

```
#making the plot
plot(newdays372, reaction372, type = 'p')

#drawing the line
lines(
  c(newdays372[1], newdays372[10]),
  c(
    beta2[2] + beta2[1] * newdays372[1],
    beta2[2] + beta2[1] * newdays372[10]
  ),
  col = 'darkred'
)
```



##2. Images and matrices Load the data:

```
#Loading the image
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?

```
#finding the maximum value of the matrix
max(matrix)
```

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

```
#finding the minimum values of the matrix
min(matrix)
```

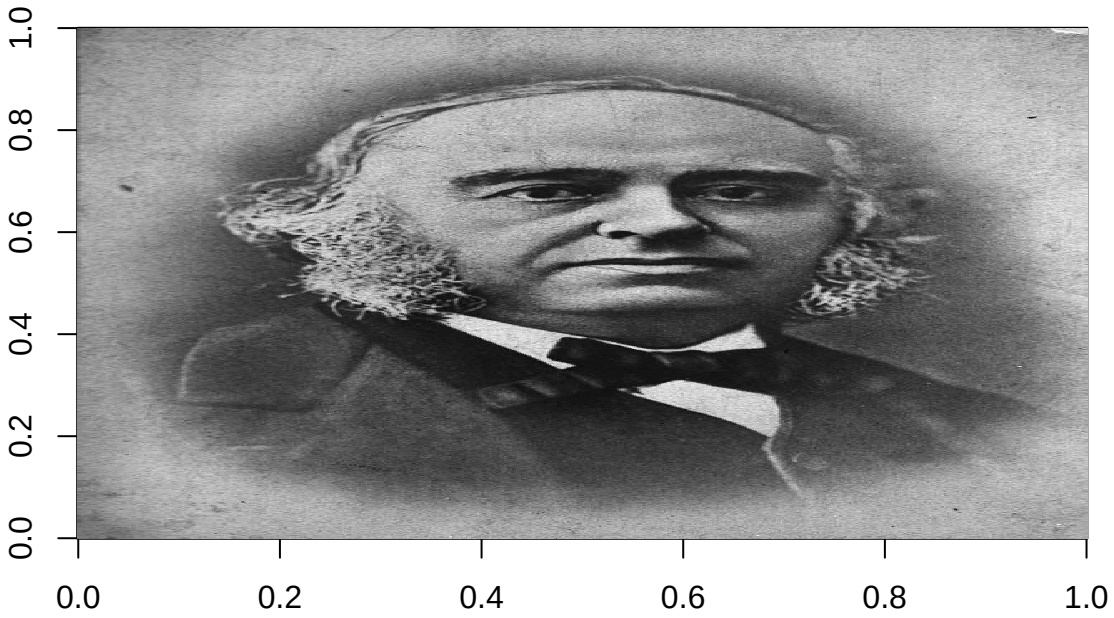
```
## [1] 0.0627451
```

```
#finding the dimensions of the matrix
dim(matrix)
```

```
## [1] 900 606
```

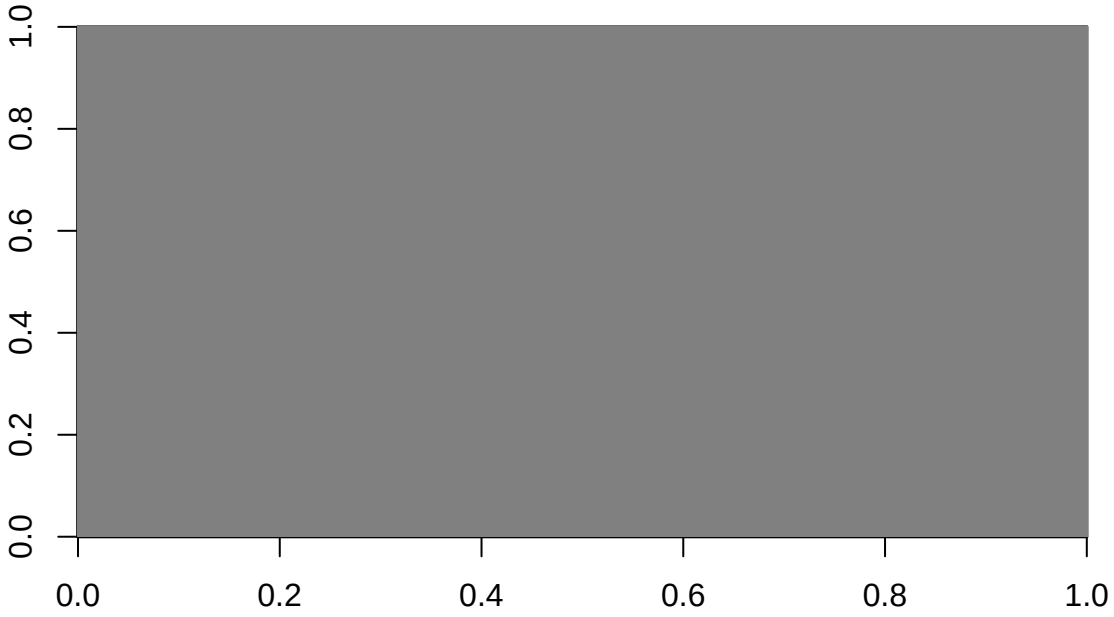
2.b: Make an image of the loaded matrix. Be sure to rotate the image into the correct orientation. The functions needed are found the in lecture slides. Furthermore, grey scale the picture with gray(1:100/100) - this will color values near 0 black/dark and values near 1 white/light.

```r
#defining the rotate function
rotate <- function(y)
  t(apply(y, 2, rev))
rot_matrix <- rotate(matrix)

#drawing the image from the matrix
image(rot_matrix, col = gray(1:100 / 100))
```



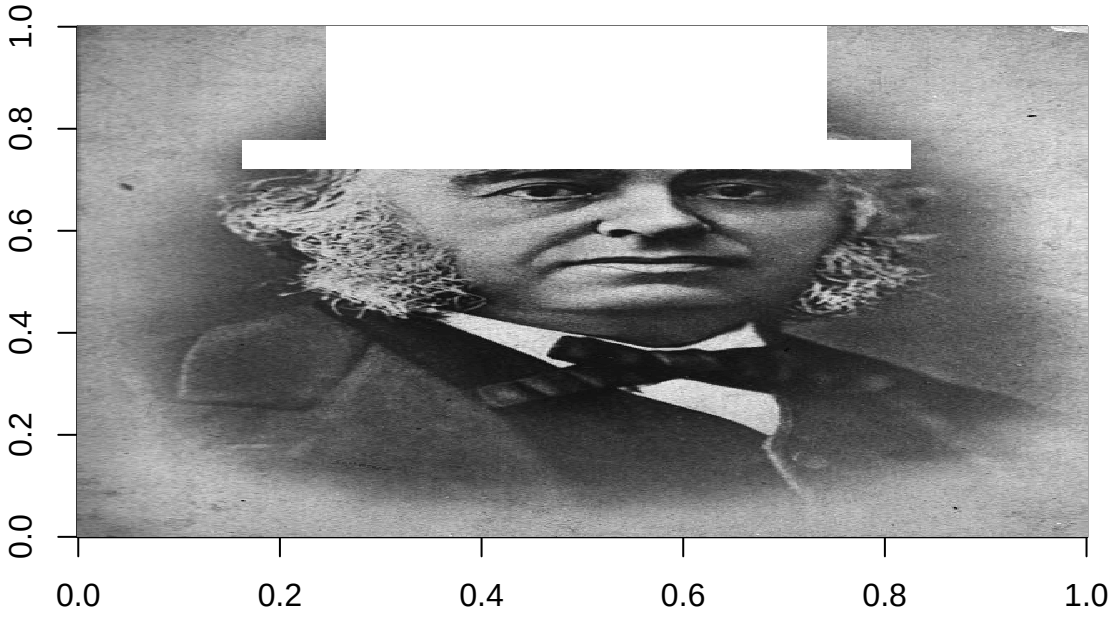
2.c: Draw an image with the same dimensions as that from 2.b. But this image should be completely black (hint: use zeros).

```r
#creating a matrix of all 0's with the same dimentions as the Broca-matrix
black <- matrix(c(rep(0, 900*606)), nrow = 900)

#drawing the image
image(black, col = gray(1:100 / 100))
```



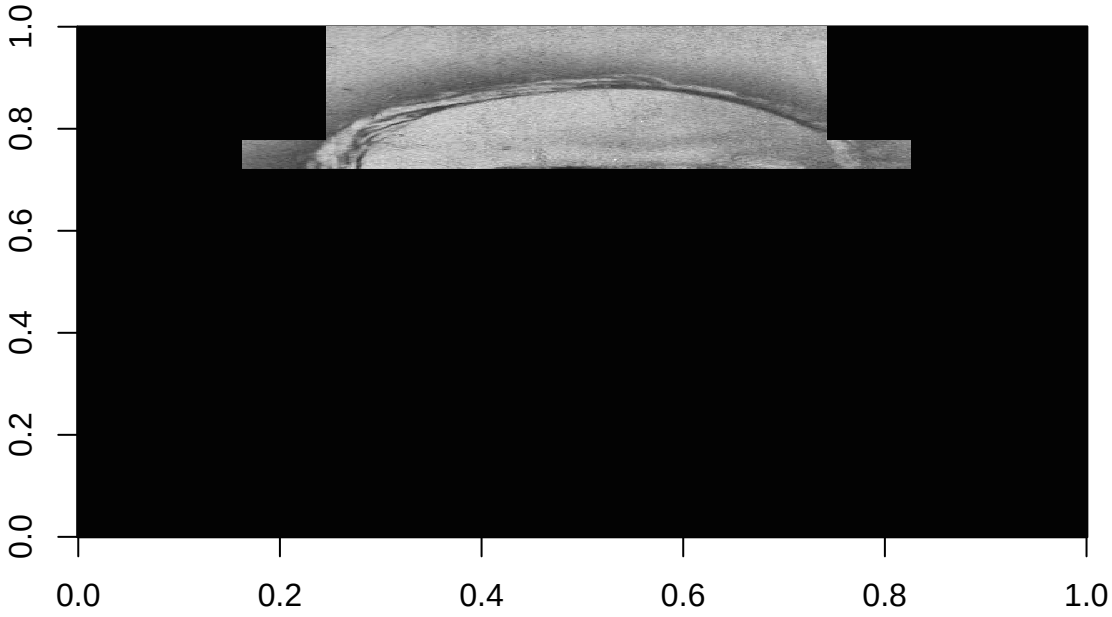
2.d: Draw a white hat on the image from 2.b (hint: use ones).

```
#resetting the image (good for drawing of a hat, if you don't know where the hat should be)
rot_matrix <- rotate(matrix)

#the different shapes of the hat
rot_matrix[100:500,650:700] <- 1
rot_matrix[150:450,700:900] <- 1

#drawing the image
image(rot_matrix, col = gray(1:100 / 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.

```
#getting the right dimensions of the black image
rot_black <- rotate(black)
#resetting the picture of Broca (removing the hat)
rot_matrix <- rotate(matrix)

#getting the parts covered by the hat on the black image
rot_black[100:500,650:700] <- rot_matrix[100:500,650:700]
rot_black[150:450,700:900] <- rot_matrix[150:450,700:900]

#drawing the image
image(rot_black, col = gray(1:100 / 100))
```

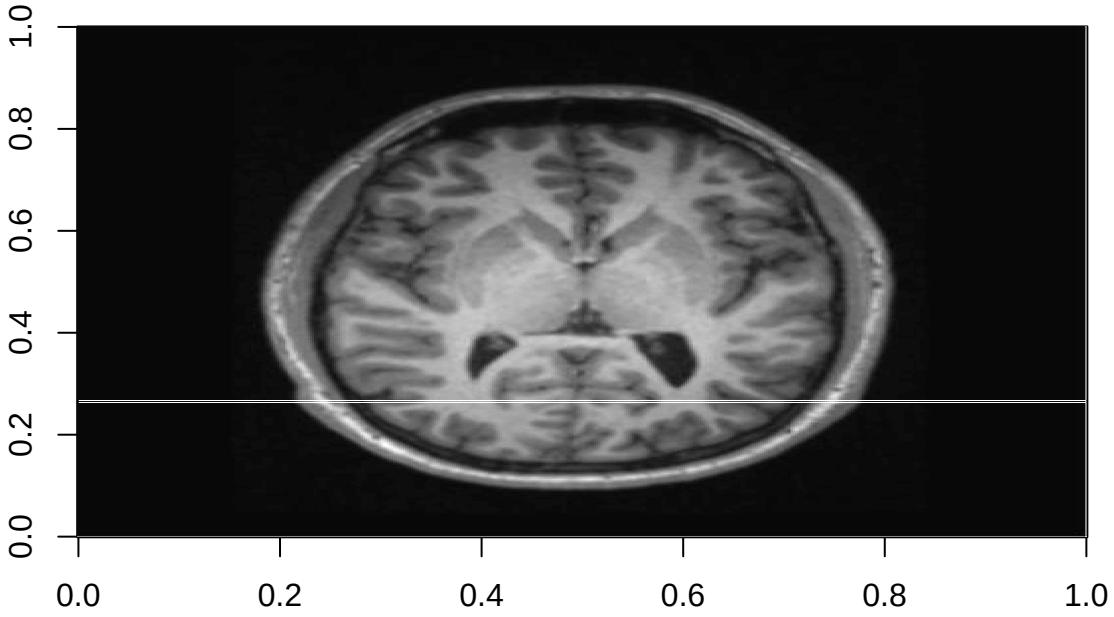


3. Brains and matrices Load the brain data using something like:

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

3.a: Make an image of the brain.

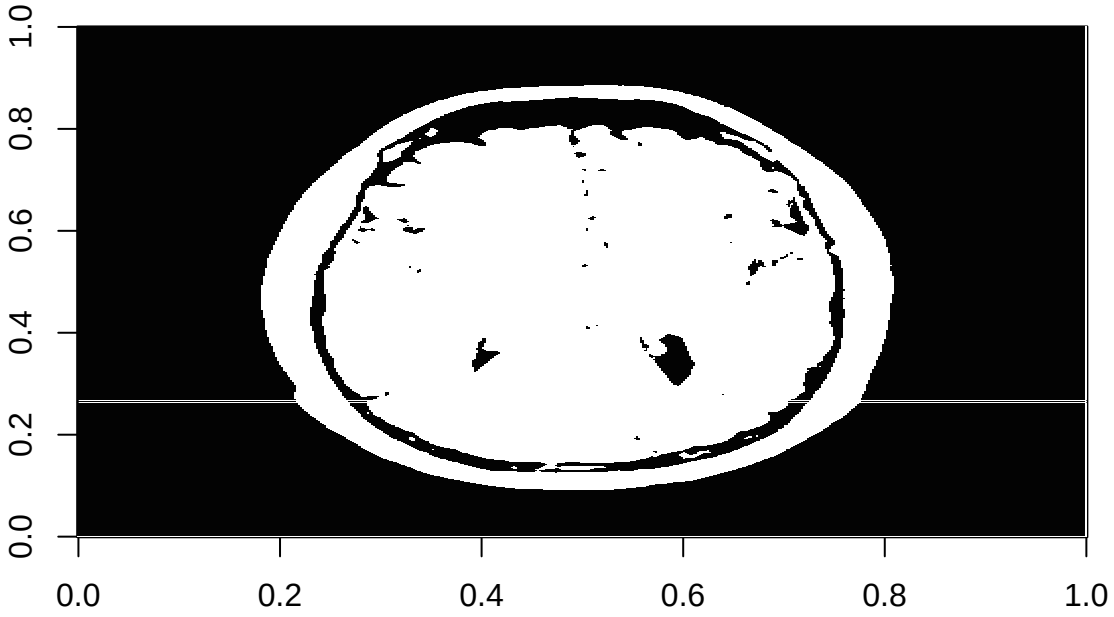
```
#rotating the brain image
rot_brain<-rotate(brain)

#draing the image
image(rot_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. To do this we will create two masks, one that filters all darker areas away, and one that filters the white matter away. The masks will work by having zeros at the areas we want to fliter away, and ones at the interesting areas. Thus, the mask will have the intended effect if we do element-wise mutiplication of it with the brain matrix.

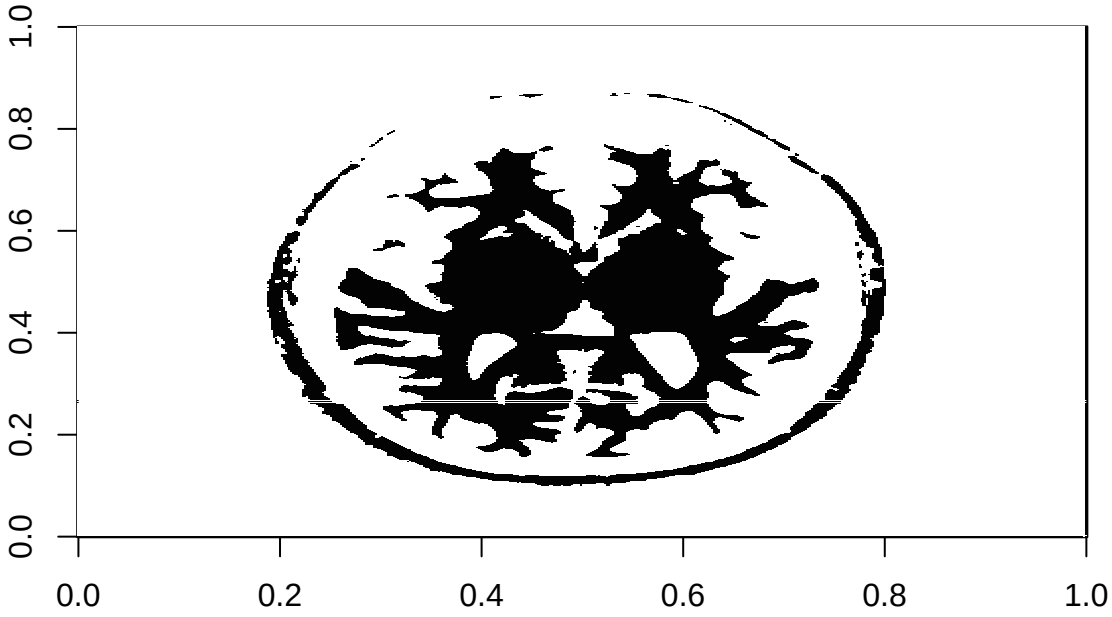
Start by making an image which is white (have ones) where the pixel values of the brain image are larger than the mean value of the whole image. Let the image be black (have zeros) everywhere else. Call this matrix mask1.

```
#defining the mean value of the brain-matrix
mean_brain <- mean(brain)

#creating the first mask (logic statement output is T/F - T is 1, F is 0)
mask1 <- brain > mean_brain

#rotating the mask
rot_mask1 <- rotate(mask1)

#drawing the mask
image(rot_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

```
#creating the second mask
mask2 <- brain < mean_brain*2.5

#rotating the mask
rot_mask2 <-rotate(mask2)

#drawing the mask
image(rot_mask2, col=gray(1:100/100))
```



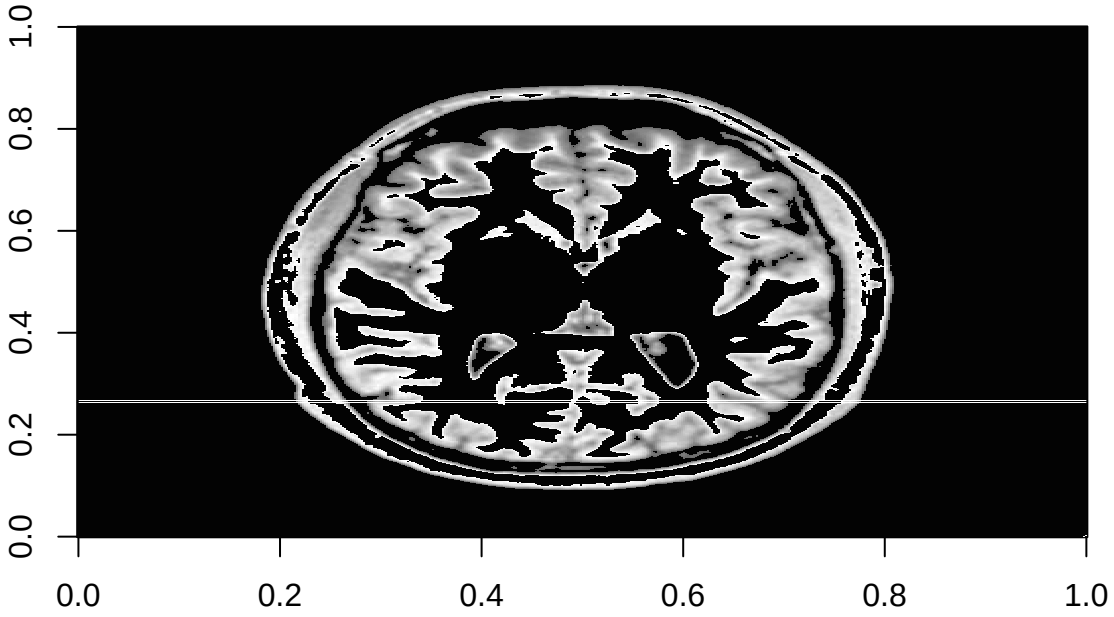
3.d: Convert mask1 and mask2 into one mask with ones where the two masks overlap and zeros everywhere else. What type mathematical procedure can be used to produce this?

```
#overlapping the masks
mask <- mask1 == mask2
```

A logic statement can be used to convert mask1 and mask2 into one mask with ones where the two masks overlap and zeros everywhere else.

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?

```
#laying the combined mask over the brain-image
masked_brain <- ifelse(mask == 1, brain, 0)

#rotating the mask
rot_masked_brain <- rotate(masked_brain)

#drawing the image
image(rot_masked_brain, col=gray(1:100/100))
```



- If gray matter is defined as being areas with values above the mean value and below 2.5 times the mean, then yes - however it looks like we have included some of the cranium as well.

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

```
#finding the number of white (TRUE) pixels
sum(mask)
```

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

There are 50004 pixels, which are white.

4. Two equations with two unknowns Two linear equations with two unknowns can be solved using matrix inversion. For example, see here: https://www.mathsisfun.com/algebra/matrix-inverse.html

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?

```
#making the text into equations
#3/4 * men + 1/4 * women = 116
#1/3 * men + 2/3 * women = 92

#defining the first matrix
A <- matrix(c(3/4, 1/4, 1/3, 2/3), ncol = 2)
A <- t(A)
```

```r
#visualizing the matrix (to avoid errors)
A
```

```
##           [,1]      [,2]
## [1,] 0.7500000 0.2500000
## [2,] 0.3333333 0.6666667
```

```r
#defining the second matrix
b <- matrix(c(116, 92), ncol = 1)

#visualizing the matrix (to avoid errors)
b
```

```
##      [,1]
## [1,]  116
## [2,]   92
```

```r
#finding the result
x <- solve(A) %*% b
x
```

```
##       [,1]
## [1,] 130.4
## [2,]  72.8
```

- 130.4 men and 72.8 women attended the bar acording to the equations (which are based on pridictive statistics, hence the not whole result).