Portfolio 2

Emma-Louise, Nicoline, Julie, Nanna

2/10/2020

1 - LINEAR REGRESSION

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

```
vector1 <- c(1,1,1,1,1,1,1,1,1,1)
```

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

```
Days372 <- c(0,1,2,3,4,5,6,7,8,9)
Reaction372 <- c(269.41, 273.47, 297.60, 310.63, 287.17, 329.61, 334.48, 343.22, 369.14, 364.12)
sum(vector1*Days372)
```

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

The inner product of the days vector and the constant vector is 45.

1.c: What does the dot product say about the possibility of finding an optimal linear regression? If the inner product = 0, the two vectors are orthogonal (angle = 90) and there is no correlation. An inner product of 45 therefore implies a possibility of a correlation. Had the inner product been negative, we would have known that the vectors were "anti"-correlated, as the angle then would have been larger than 90 degrees.

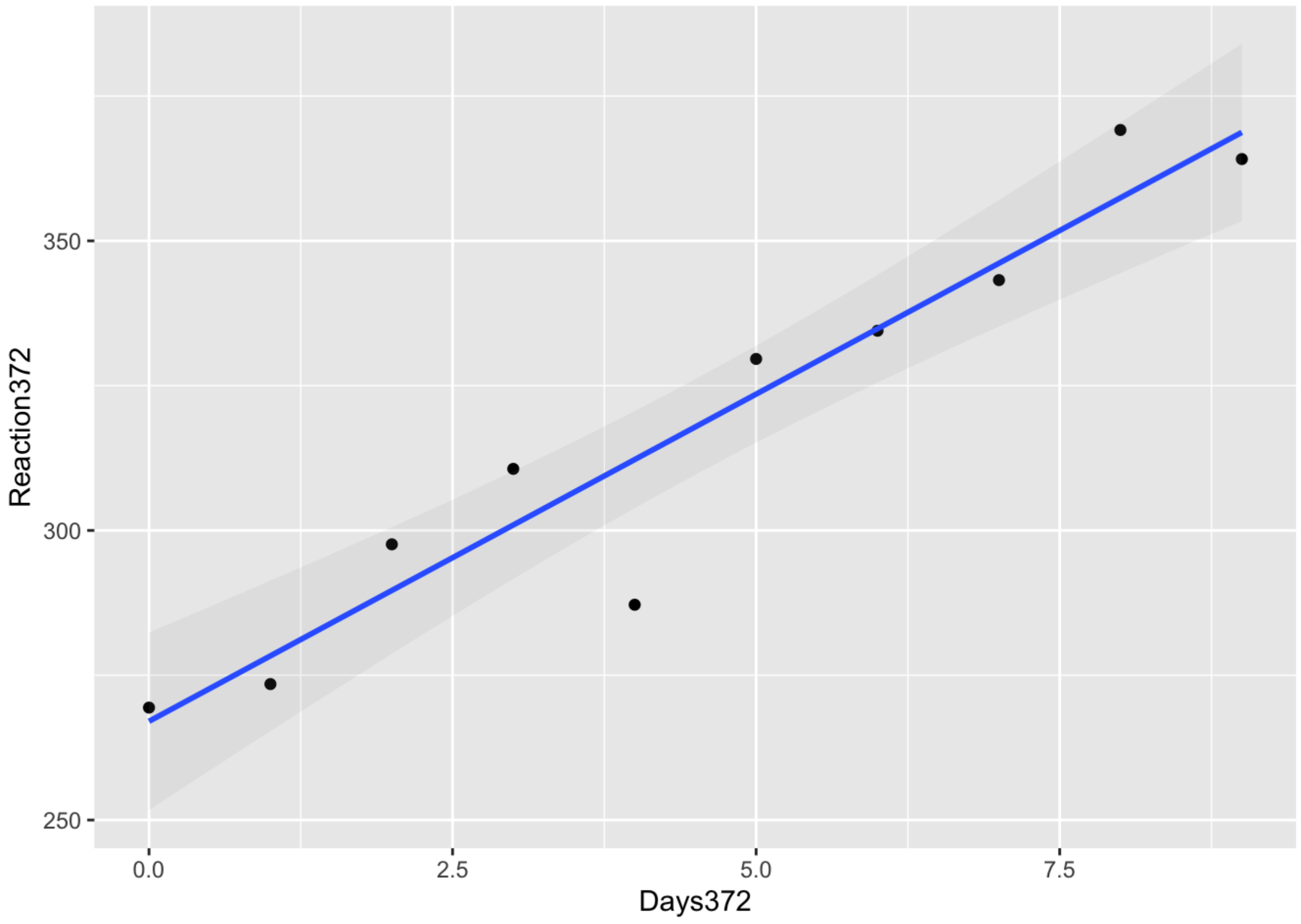
1.d: Create a 10x2 matrix called X with the days vector and constant vector as columns and use the least squares method manually to find the optimal coefficients (i.e. slope and intercept) to reaction time.

```
X <- matrix(c(Days372,vector1), ncol = 2)
beta <- solve(t(X)%*%X)%*%t(X)%*%Reaction372
beta
```

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

Our beta coefficients tell us that the intercept b=267 and the slope a=11.3 as seen in the plot below.

```
df_X <- data.frame(Days372,Reaction372)

ggplot(df_X, aes(Days372, Reaction372))+
  geom_point()+geom_smooth(method = lm, alpha = 0.1)
```



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

```
lm(Reaction372~0+X)
```

```
##
## Call:
## lm(formula = Reaction372 ~ 0 + X)
##
## Coefficients:
##    X1     X2
##  11.3  267.0
```

Using lm() we get the same results.

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/ (https://www.theanalysisfactor.com/) center-on-the-mean/)

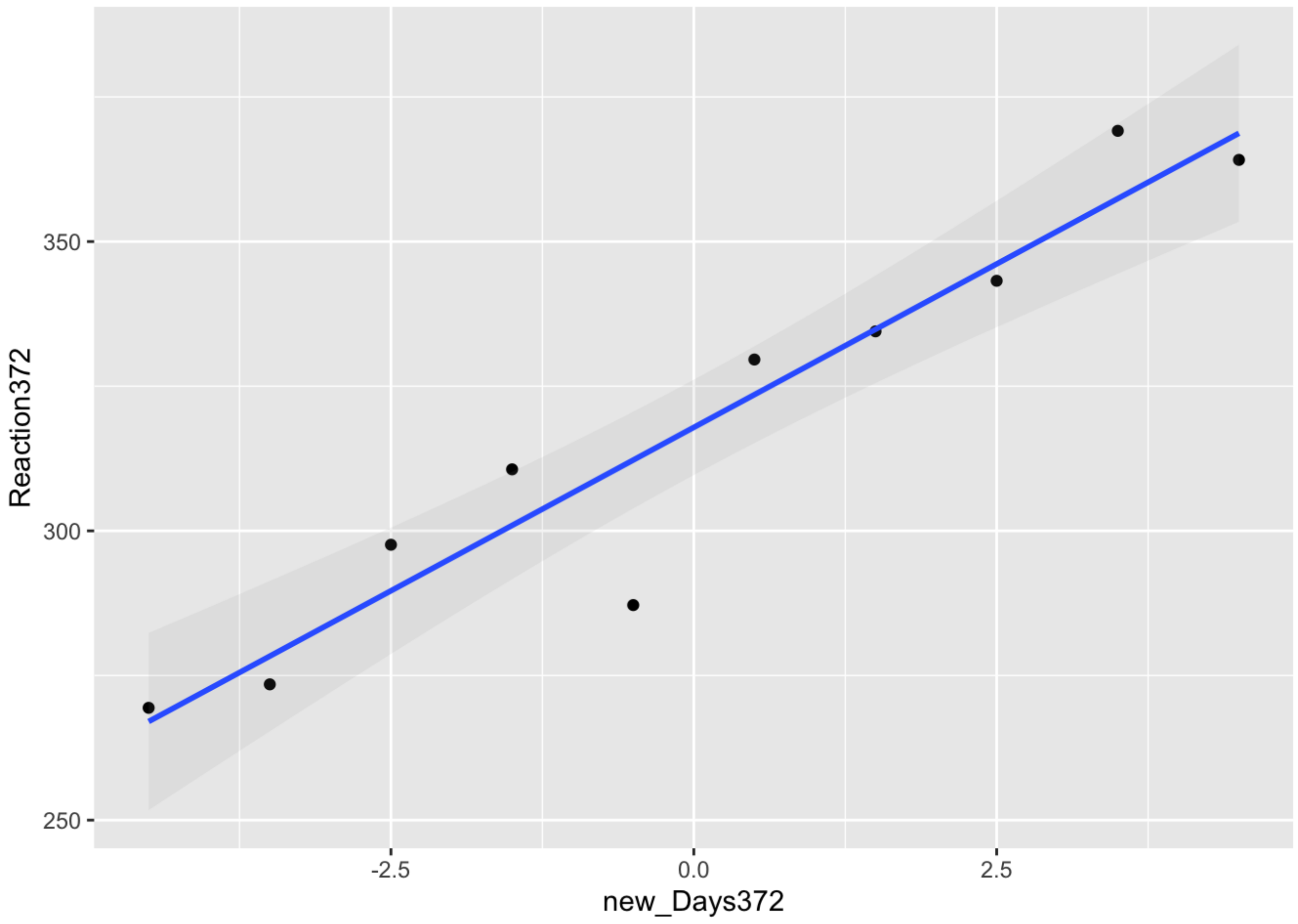
```
new_Days372 <- c(Days372-mean(Days372))

X1 <- matrix(c(new_Days372, vector1), ncol = 2)
lm(Reaction372~0+X1)
```

```
##
## Call:
## lm(formula = Reaction372 ~ 0 + X1)
##
## Coefficients:
##   X11    X12
##  11.3  317.9
```

The intercept changed, while the slope stayed the same.

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

```
df_X1 <- data.frame(X1)

ggplot(df_X1, aes(new_Days372, Reaction372))+
  geom_point()+geom_smooth(method = lm, alpha = 0.1)
```



2 - IMAGES AND MATRICES Setting up

```
library(jpeg)

matrix <- readJPEG('data1_matrices.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?

```
nrow(matrix)
```

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

```
ncol(matrix)
```

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

```
max(matrix)
```

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

```
min(matrix)
```

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

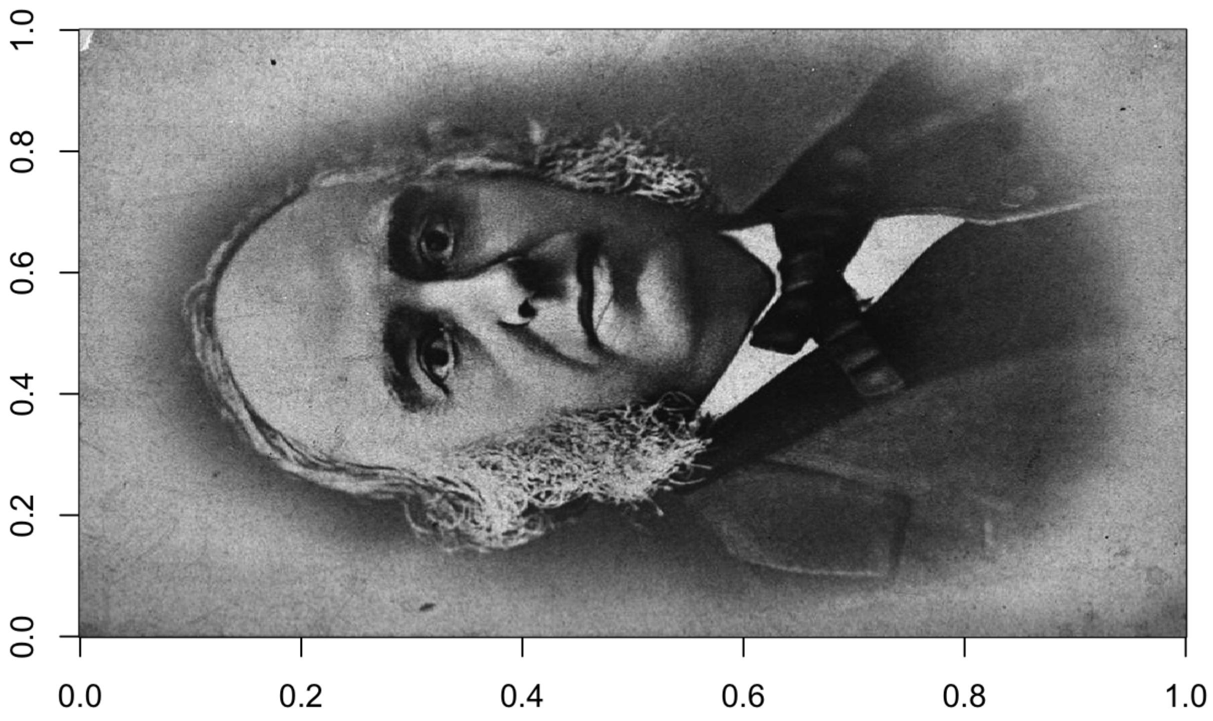
```
mean(matrix)
```

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

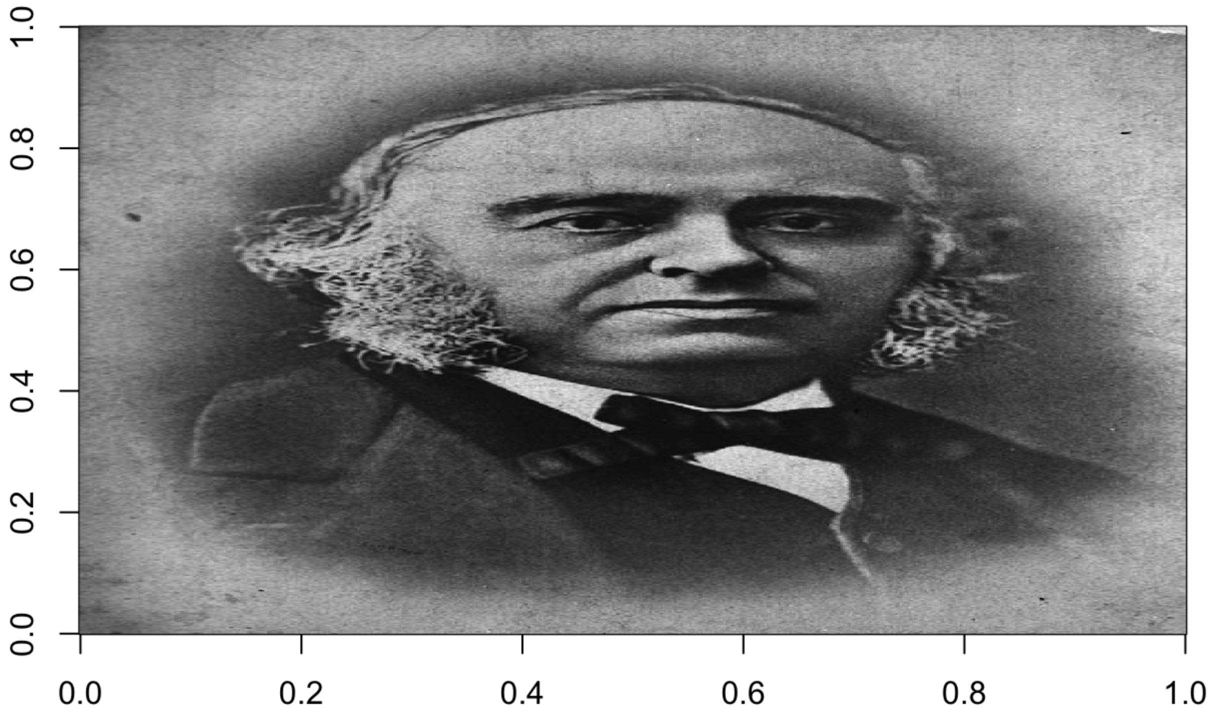
900 rows, 606 columns. Maximum pixel value = 1 Minimum pixel value = 0.06 Mean pixel value = 0.51

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.

```
image(matrix, col = gray(1:100/100))
```



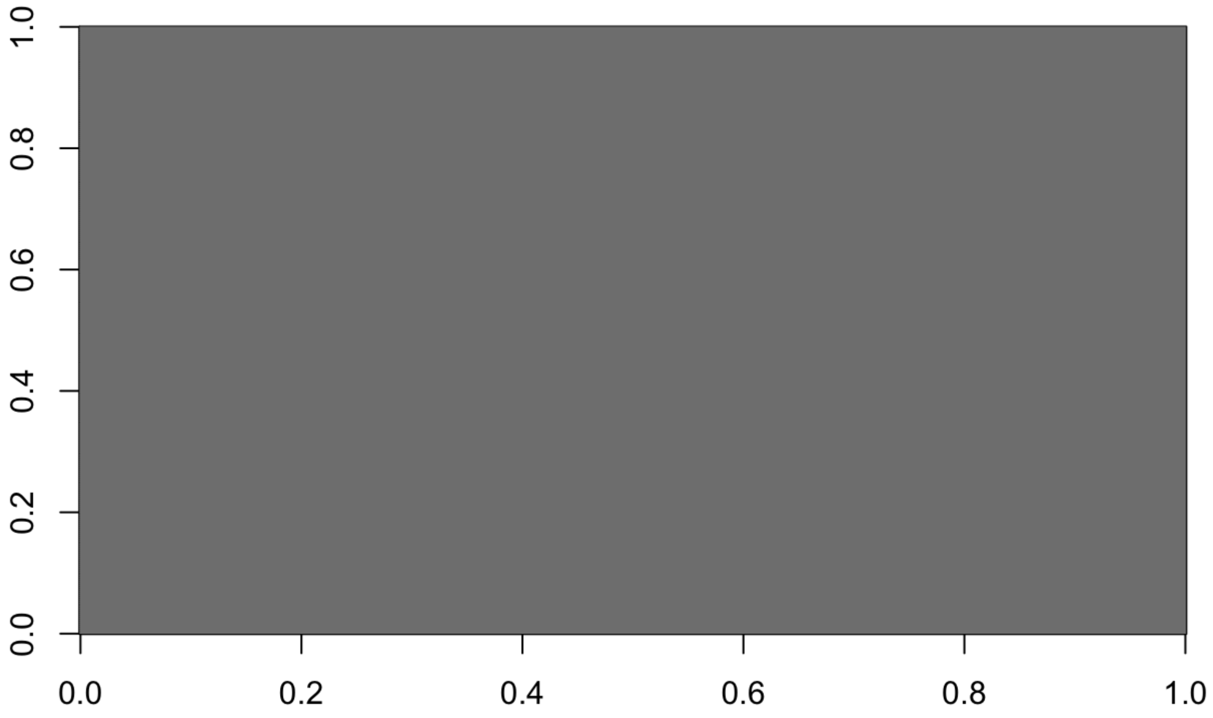
```
rotate <- function(x) t(apply(x, 2, rev))

matrix_rot <- rotate(matrix)

image(matrix_rot, col = gray(1:100/100))
```



Per default the image function is 90 degrees rotated, so we make a function that rotates the matrix back in order for us to get the image upright rather than "lying down".

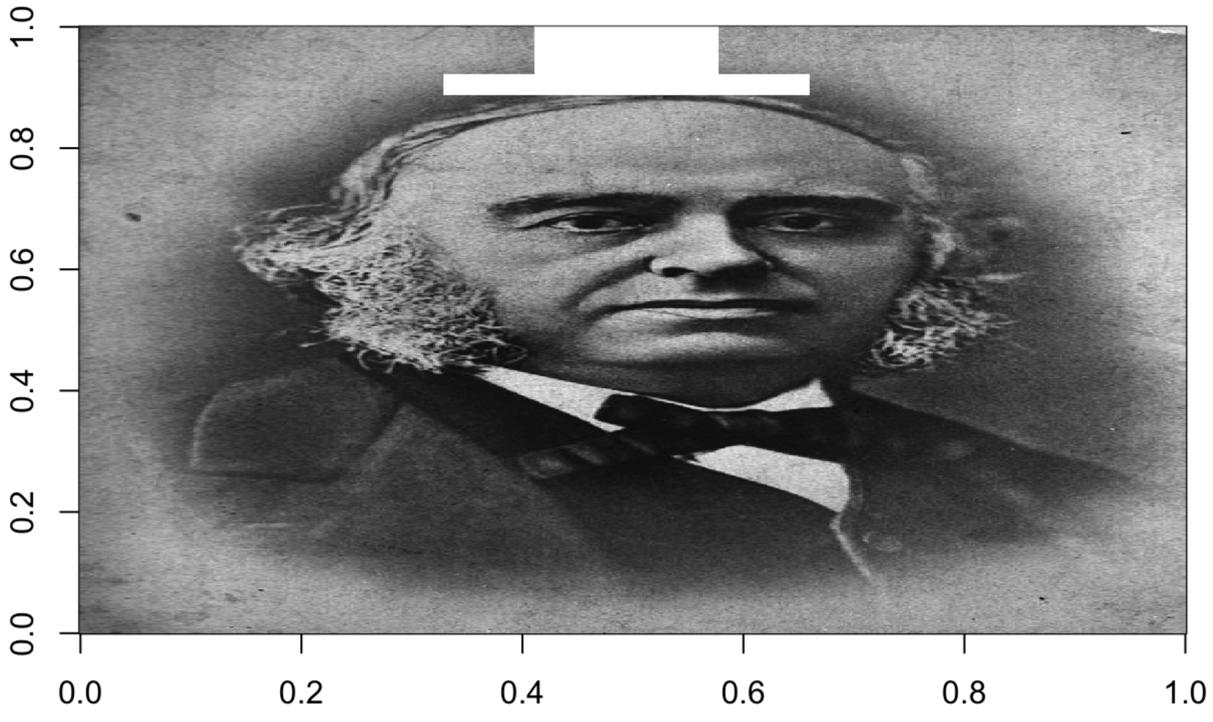
2.c: Draw an image with the same dimensions as that from 2.b. But this image should be completely black (hint: use zeros).

```
#Create new matrix with all values = 0
blackimage <- ifelse(matrix_rot > 0, 0, matrix_rot)
image(blackimage, col = gray(1:100/100))
```



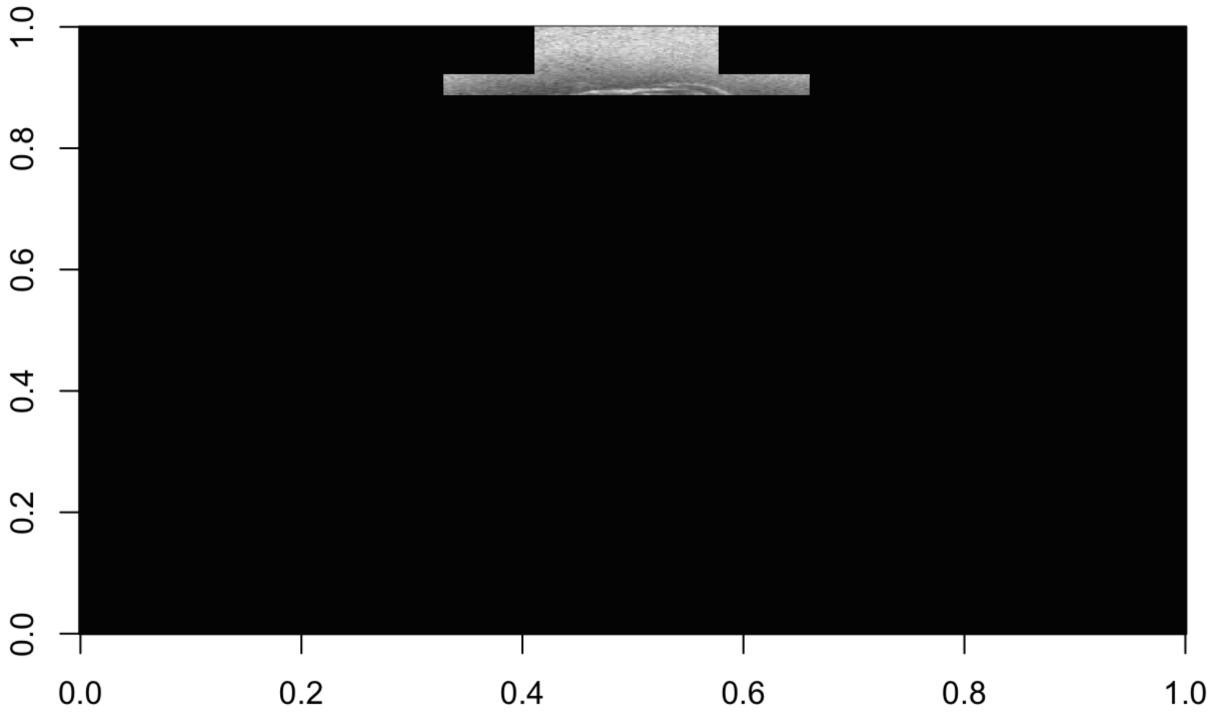
2.d: Draw a white hat on the image from 2.b (hint: use ones).

```
matrix_rot[250:350,800:900] <- 1
matrix_rot[200:400,800:830] <- 1

image(matrix_rot, 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.

```
blackimage[250:350,800:900] <- 1
blackimage[200:400,800:830] <- 1

matrix_broca_rot <- rotate(matrix)

matrix_2e <- blackimage*matrix_broca_rot
image(matrix_2e, col = gray(1:100/100))
```

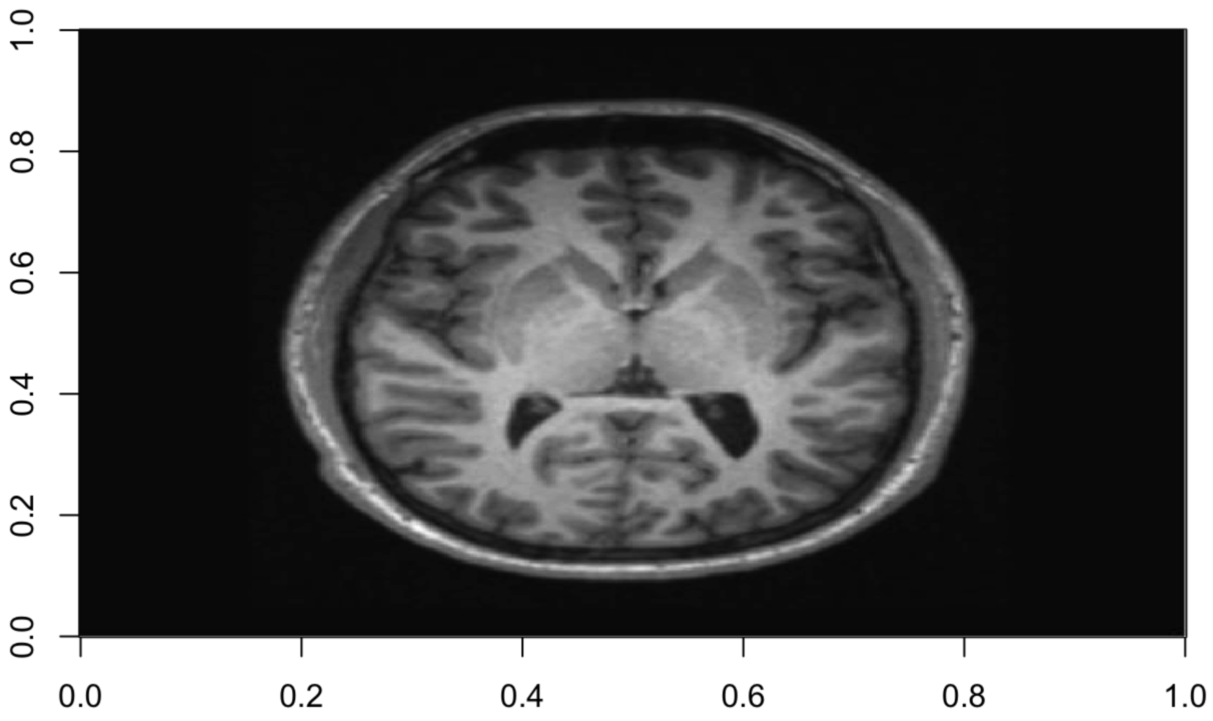


3 - BRAINS AND MATRICES Setting up

```
library(jpeg)
brain <- readJPEG('data2_matrices.jpg', native = FALSE)
```

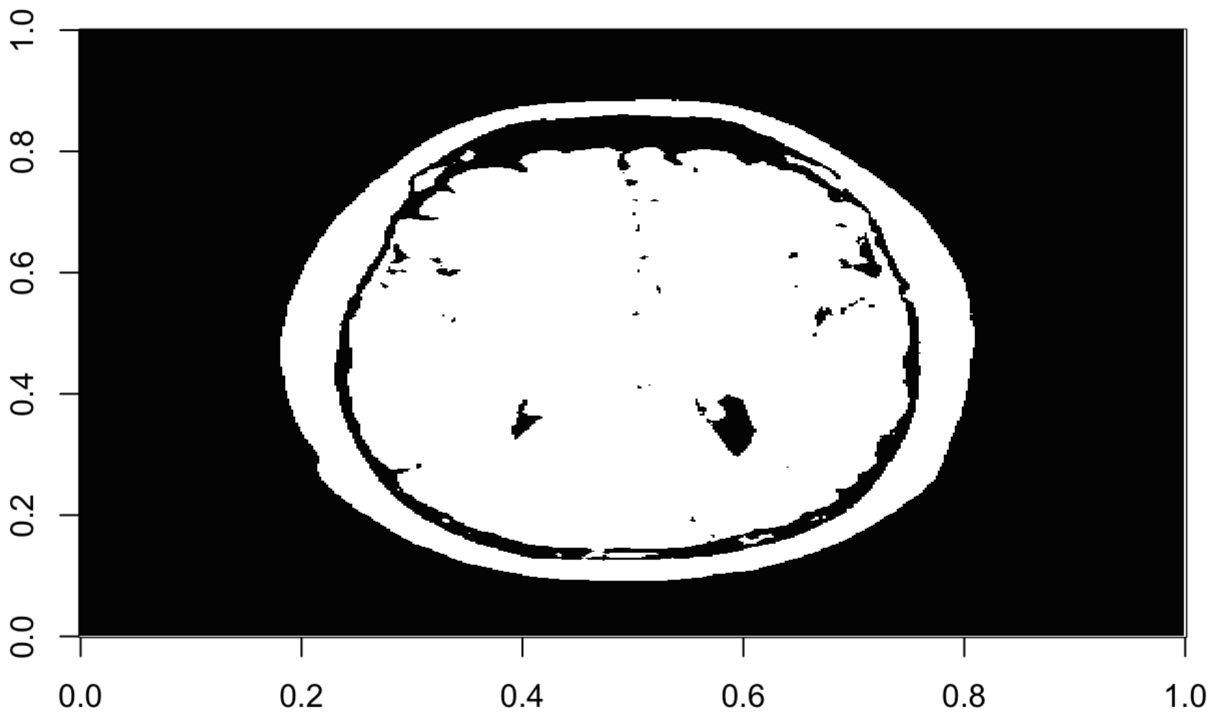
3.a: Make an image of the brain.

```
brain_rotated <- rotate(brain)

image(brain_rotated, col = gray(1:100/100))
```



3.b: 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.

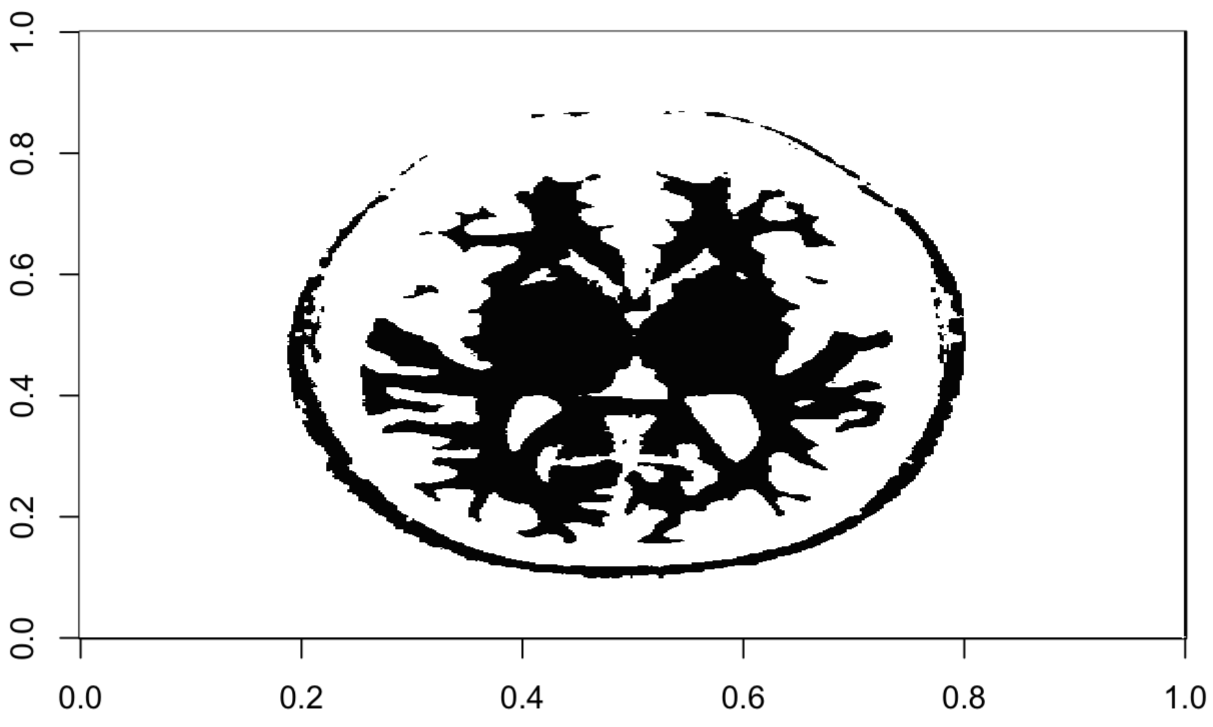
```
bmean <- mean(brain)
brow <- nrow(brain_rotated)
bcol <- ncol(brain_rotated)

black_brain_2 <- matrix(0, ncol = bcol, nrow = brow)
black_brain_2[brain_rotated > bmean] <- 1
image(black_brain_2, col = gray(1:100/100))
```



OBS: We have called mask1 “black_brain_2”

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

```
black_brain_3 <- matrix(0, ncol = bcol, nrow = brow)
black_brain_3[brain_rotated < 2.5*bmean] <- 1
image(black_brain_3, col = gray(1:100/100))
```



OBS: Mask2 is called black_brain_3

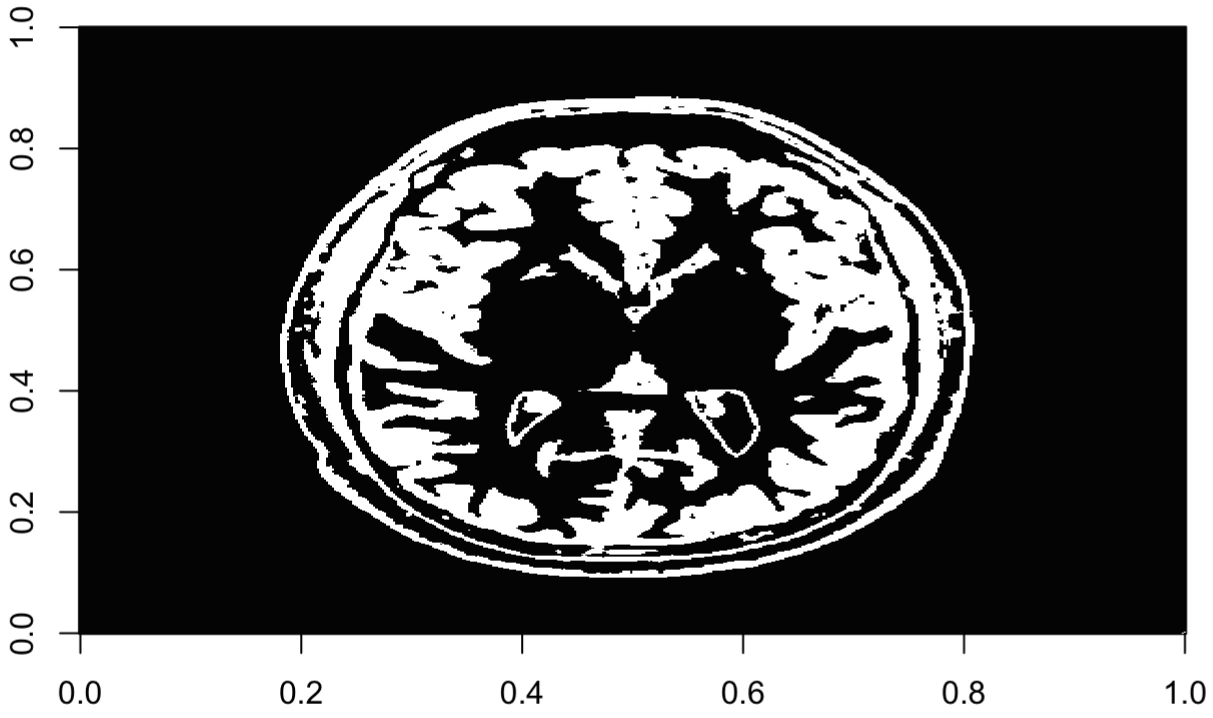
3.d: Convert mask1 and mask2 into one mask with ones where the two masks overlap and zeros everywhere else. What type mathematical procedure can be used to produce this?

```
final_mask <- black_brain_2*black_brain_3
```

We multiply the two masks cell by cell, so that ones only appear where there is ones in both masks.

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?

```
image(final_mask, col = gray(1:100/100))
```



When we compare the image above with the brain picture from 3.a, it seems that only the grey matter is white, so it seems that we limited it succesfully.

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

```
sum(final_mask == 1)
```

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

The number of pixels in the combined mask is 50004.

4 - TWO EQUATIONS WITH TWO UNKNOWNS 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?

```
friday_bar <- matrix(c(1,3,2,1), nrow = 2)

inv_fridaybar <- solve(friday_bar, c(92,116))
print(inv_fridaybar)
```

```
## [1] 28 32
```

There were 32 women and 28 men at the friday bar.