hw2 ISYE-8803

2a) Import libraries and read and show the image

```
library(imager)
```

```
## Loading required package: magrittr
```

```
##
## Attaching package: 'imager'
```

```
## The following object is masked from 'package:magrittr':
##
##     add
```

```
## The following objects are masked from 'package:stats':
##
##     convolve, spectrum
```

```
## The following object is masked from 'package:graphics':
##
##     frame
```

```
## The following object is masked from 'package:base':
##
##     save.image
```

```
library(pracma)
```

```
##
## Attaching package: 'pracma'
```

```
## The following objects are masked from 'package:magrittr':
##
##     and, mod, or
```

```
library(R.matlab)
```

```
## R.matlab v3.6.2 (2018-09-26) successfully loaded. See ?R.matlab for help.
```

```
##
## Attaching package: 'R.matlab'
```

```
## The following objects are masked from 'package:base':
##
##     getOption, isOpen
```

```r
library(cluster)
library(factoextra)
```

```
## Loading required package: ggplot2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```r
library(profvis)
library(magick)
```

```
## Linking to ImageMagick 6.9.12.3
## Enabled features: cairo, freetype, fftw, ghostscript, heic, lcms, pango, raw, rsvg, webp
## Disabled features: fontconfig, x11
```

```r
#install.packages("BiocManager")
#BiocManager::install("EBImage")
library(EBImage)
```

```
##
## Attaching package: 'EBImage'
```

```
## The following objects are masked from 'package:imager':
##
##     channel, dilate, display, erode, resize, watershed
```

```r
horse_I = load.image("./horse1-2.jpg")
plot(horse_I)
```


```
#N = 256
#hist( x = I, breaks = N, xlab = '', ylab = '',ylim= c(0,5000), main='Histogram')
```

b) What is the size of the original image? Resize the original image to a third of its size.What is the size of the new image?

```
dim(horse_I)[1:2]
```

```
## [1] 1000  563
```

The dimensions of the image are 1000 x 563 pixels.

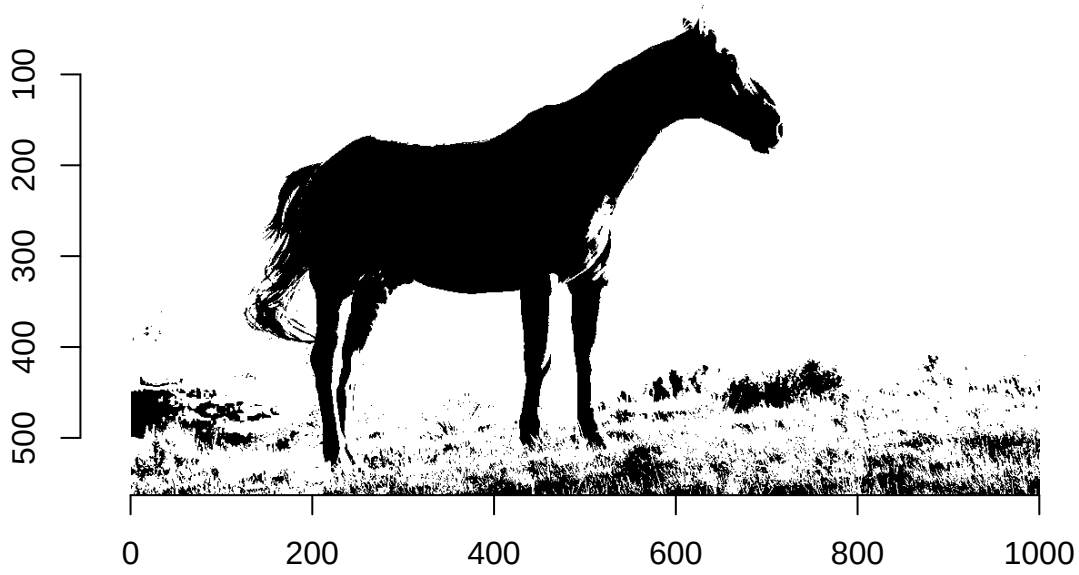
```
horse_resized <- resize(horse_I, dim(horse_I)[1]/3)
dim(horse_resized)[1:2]
```

```
## [1] 333 188
```

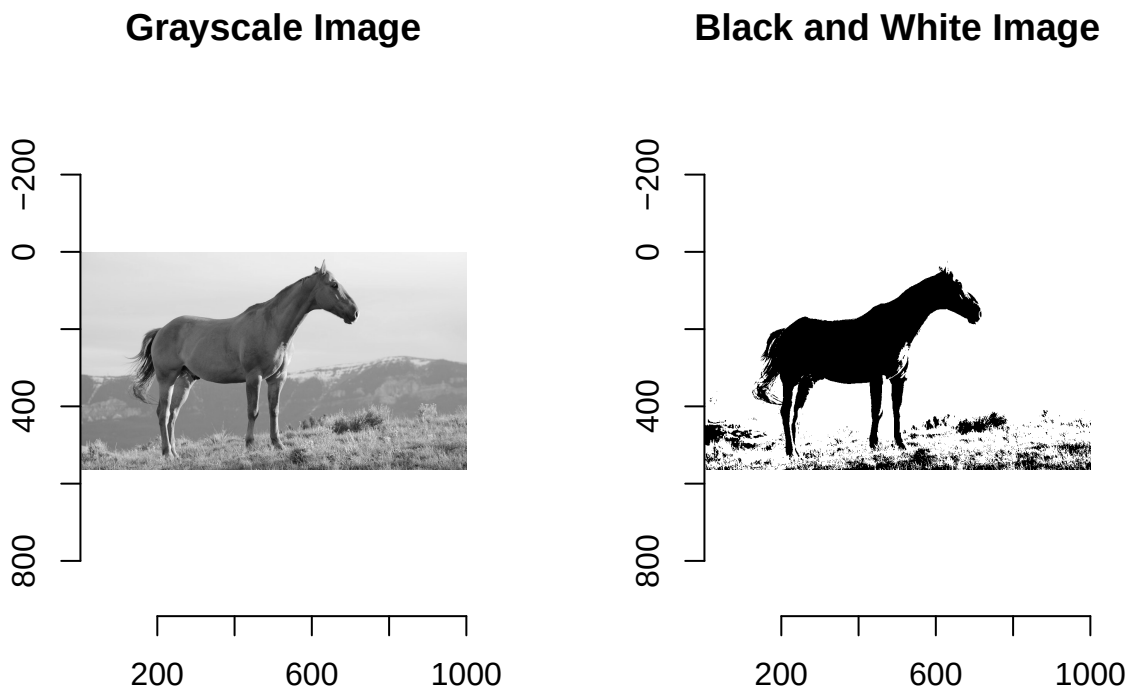
The new image is now 333 by 188, which is exactly 1/3 of the original image size.

2c) c) Convert the original image to a gray image and to a black and white image (using alevel of 0.5).

```
horse_grayscale <- grayscale(horse_I)
horse_bw <- threshold(horse_grayscale, thr = 0.5, approx = TRUE, adjust = 1)
plot(horse_bw)
```



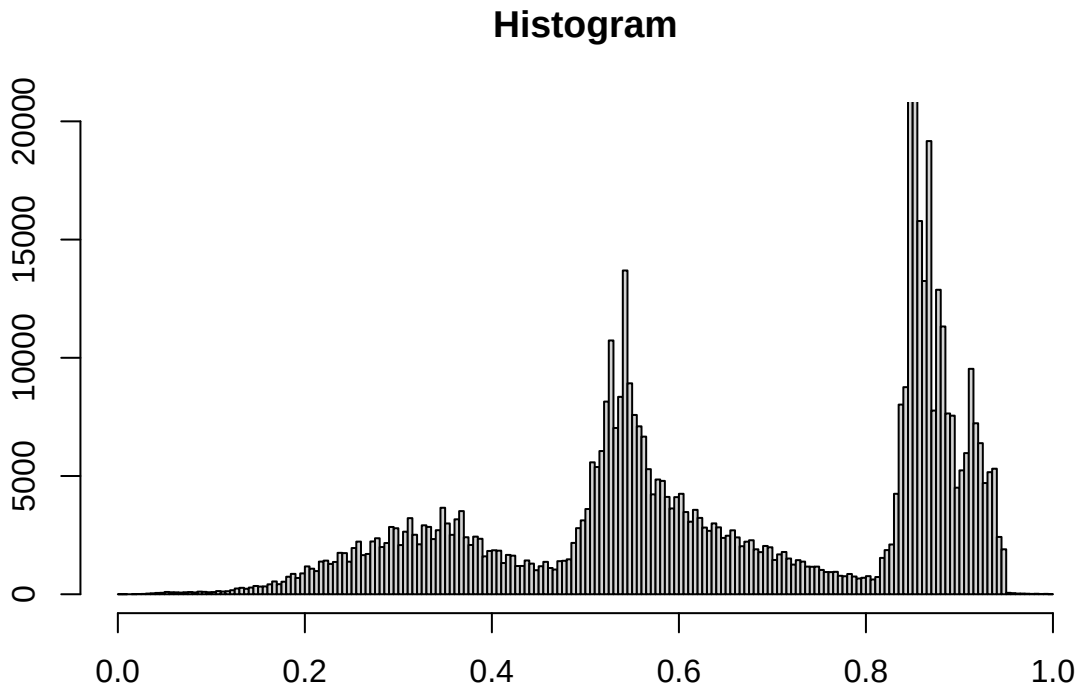

```
par(mfrow=c(1,2))
plot(horse_grayscale,main='Grayscale Image')
plot(horse_bw,main = 'Black and White Image')
```




```
frink <- image_read("https://jeroen.github.io/images/frink.png")
```

d) Show the histogram of the gray image. What observations can be made from this histogram?

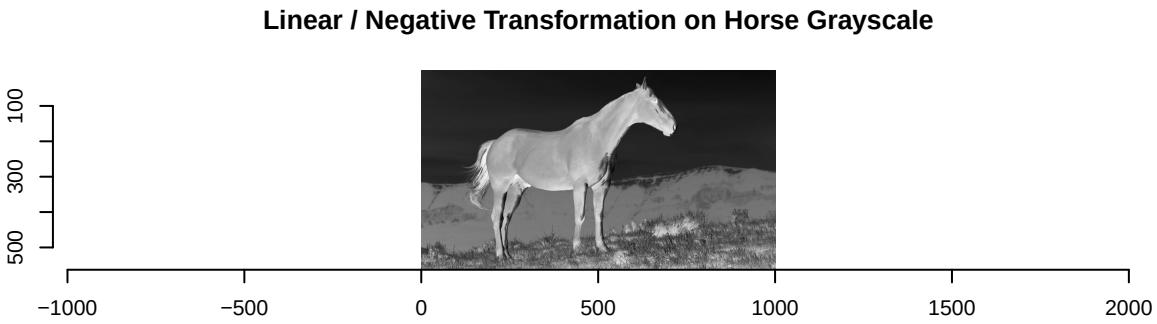
```
hist( x = horse_grayscale, breaks = 256, xlab = '', ylab = '',ylim= c(0,20000), main='Histogram')
```

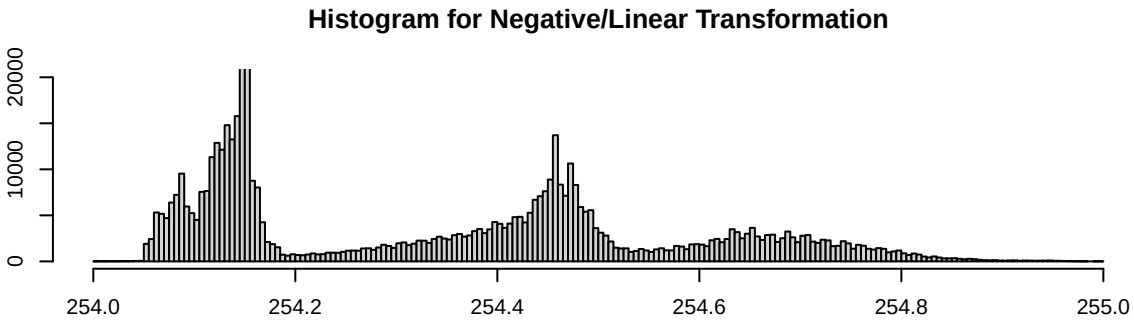


When observing the histogram, we can see that the darker portion of the images vary in intensity as well as provide much lower densities as seen on the far left of the histogram. The middle part of the histogram likely represents the grass / ground in front of the horse, the distribution varies much less compared to the other two groups of the image. This background of the image can be seen with the right part of the histogram, where the frequency is much higher since it takes up a majority of the image, the distribution is much thinner because the background image color does not vary a lot.

2e i) Linear Transformation

```
horse_grayscale_rescaled <- horse_grayscale*255
layout(matrix(c(1,1,2,2), ncol = 1, byrow = TRUE))
linear = (256-1)-horse_grayscale
plot(linear,main = 'Linear / Negative Transformation on Horse Grayscale')
hist( x = linear, breaks = 256, xlab = '', ylab = '',ylim= c(0,20000), main='Histogram for Negative/Lin
```

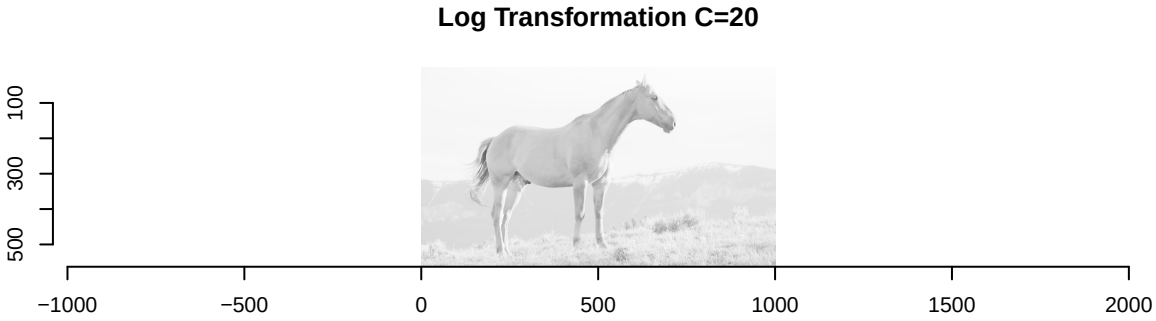


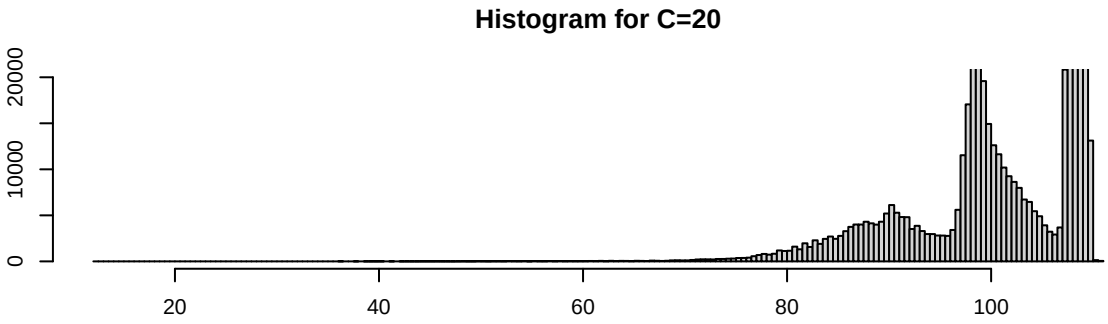


The negative linear transformation will essentially invert the pixel to the opposite representation, so if it's a dark pixel it will now become lighter, etc. You can see this reflected in the histogram, as it's now the exact opposite compared to the original grayscale histogram.

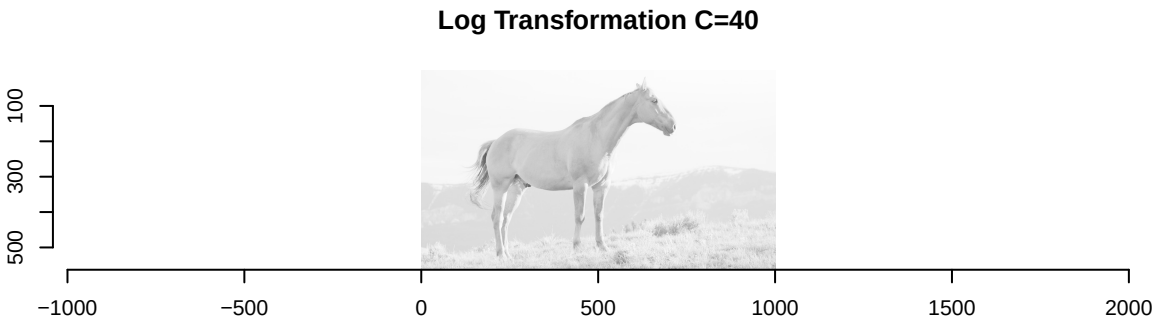
ii) Log Transformation c= 20

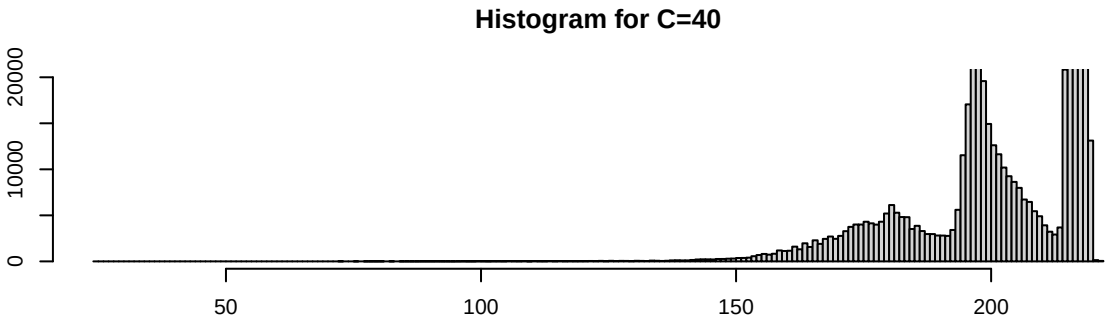
```
layout(matrix(c(1,1,2,2), ncol = 1, byrow = TRUE))
log_20 = 20*log(horse_grayscale_rescaled+1)
plot(log_20,main = 'Log Transformation C=20')
hist( x = log_20, breaks = 256, xlab = '', ylab = '',ylim= c(0,20000), main='Histogram for C=20')
```






```
layout(matrix(c(1,1,2,2), ncol = 1, byrow = TRUE))
log_40 = 40*log(horse_grayscale_rescaled+1)
plot(log_40,main = 'Log Transformation C=40')
hist( x = log_40, breaks = 256, xlab = '', ylab = '',ylim= c(0,20000), main='Histogram for C=40')
```

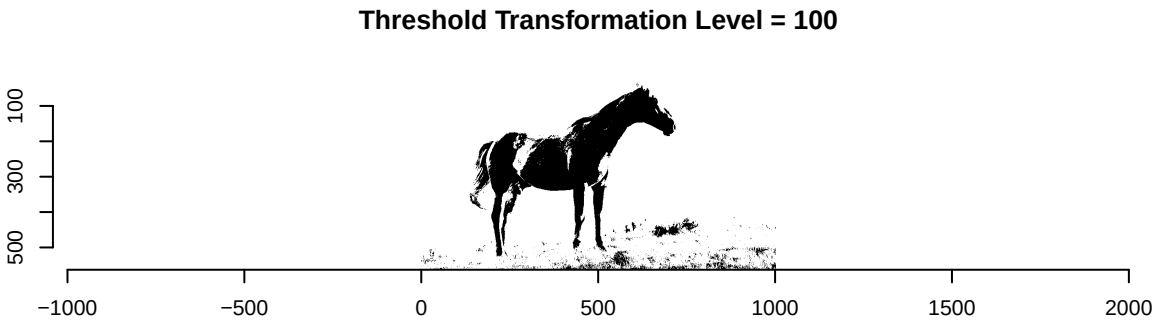


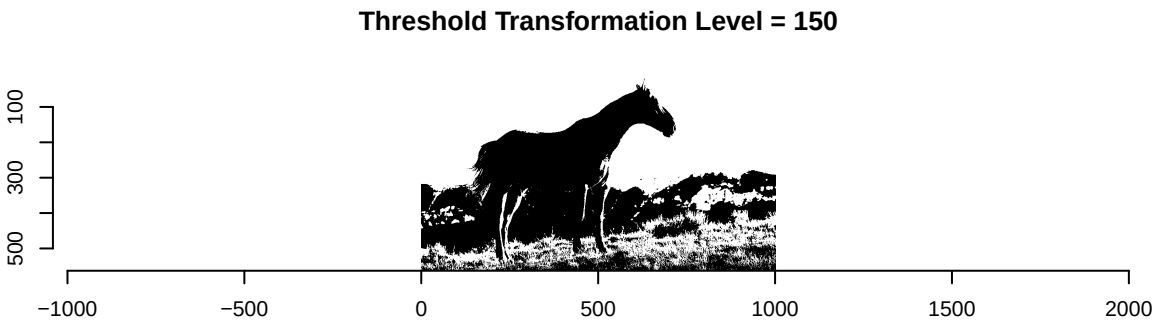


From the outputs of the log transformation, we can see that it's doing something similar to a shift, where we're seeing lower pixel densities get shifted to the right and distribution for all 3 groups is much more compact. Therefore the resulting image is much brighter for both transformations. You shouldn't see much of a difference between C=20 and C=40.

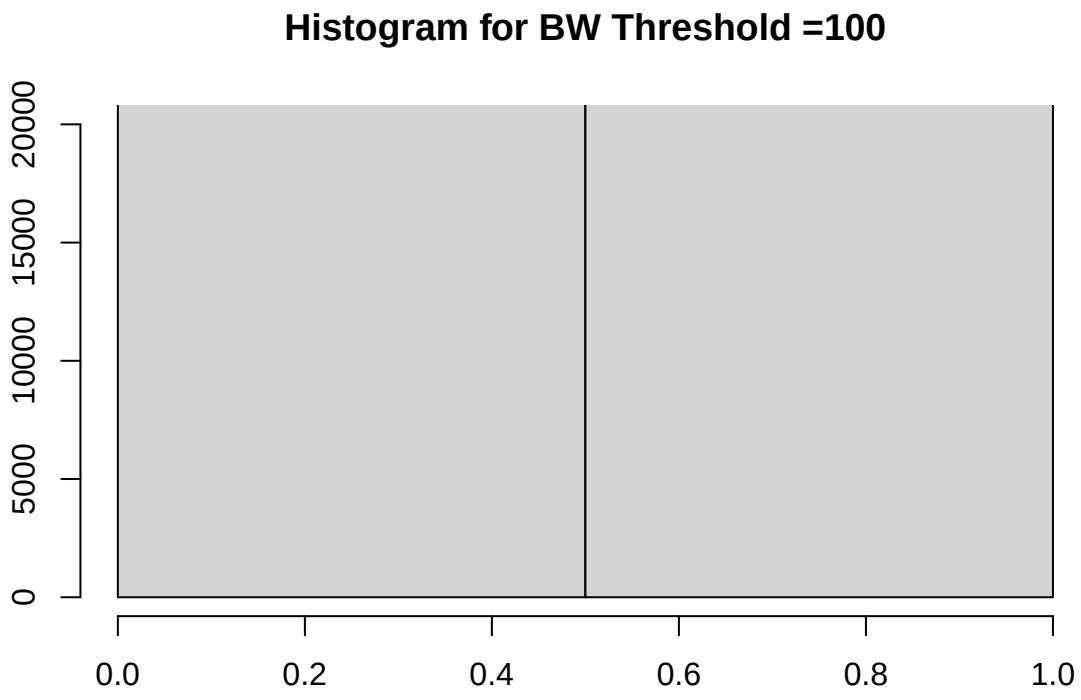
iii) Threshold 100 and 150

```
layout(matrix(c(1,1,2,2), ncol = 1, byrow = TRUE))
level=100
BW_100=horse_grayscale_rescaled
BW_100[horse_grayscale_rescaled>level] = 1
BW_100[horse_grayscale_rescaled<=level] = 0
plot(BW_100,main = 'Threshold Transformation Level = 100')
level=150
BW_150=horse_grayscale_rescaled
BW_150[horse_grayscale_rescaled>level] = 1
BW_150[horse_grayscale_rescaled<=level] = 0
plot(BW_150,main = 'Threshold Transformation Level = 150')
```

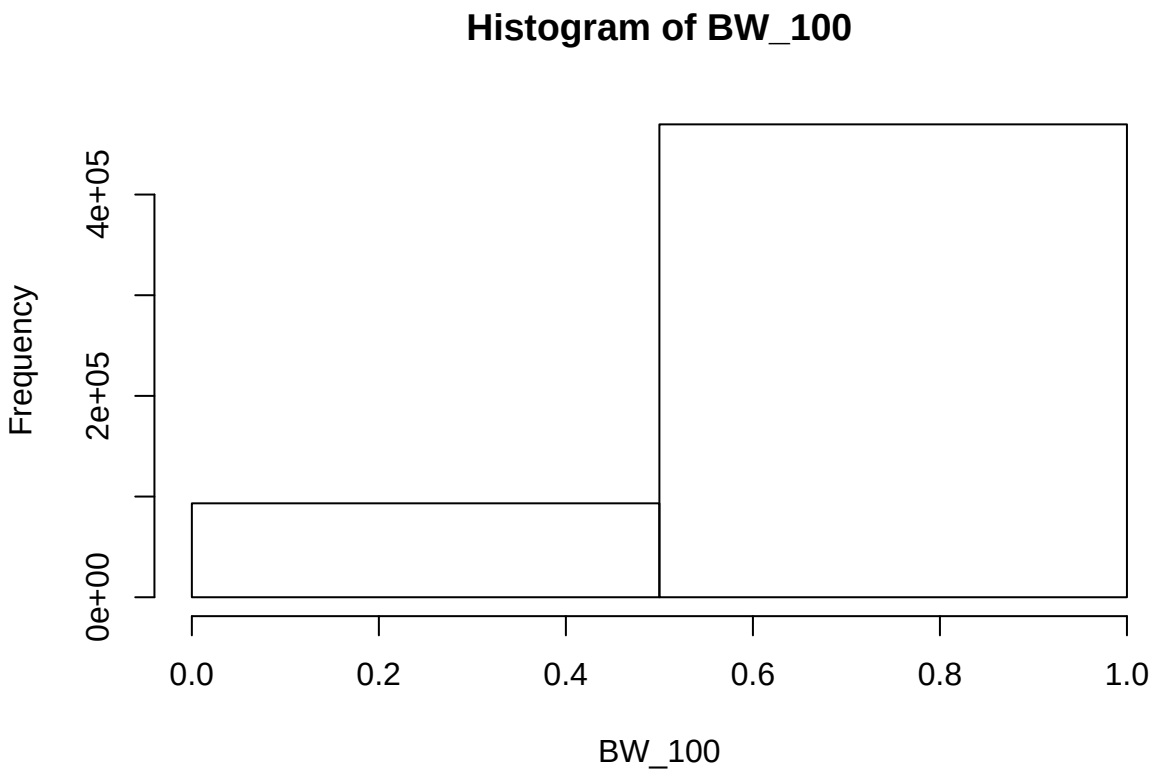




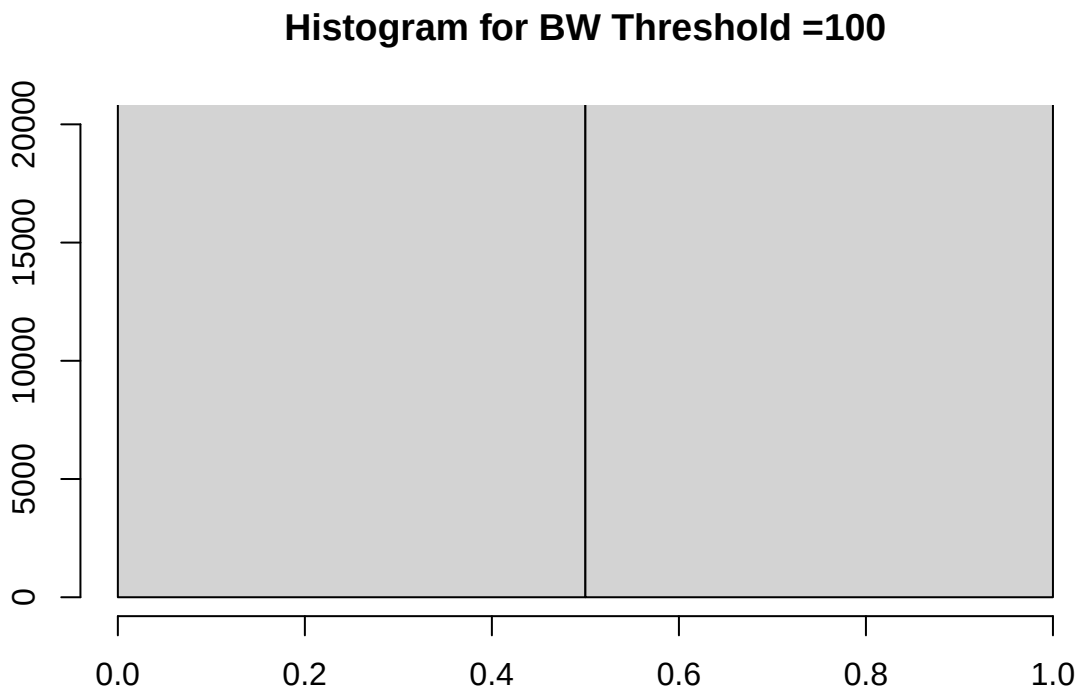

```
#layout(matrix(c(1,1), ncol = 1, byrow = TRUE))
bw100_hist <- hist( x = BW_100, breaks = 2, xlab = '', ylab = '',ylim= c(0,20000), main='Histogram for
```



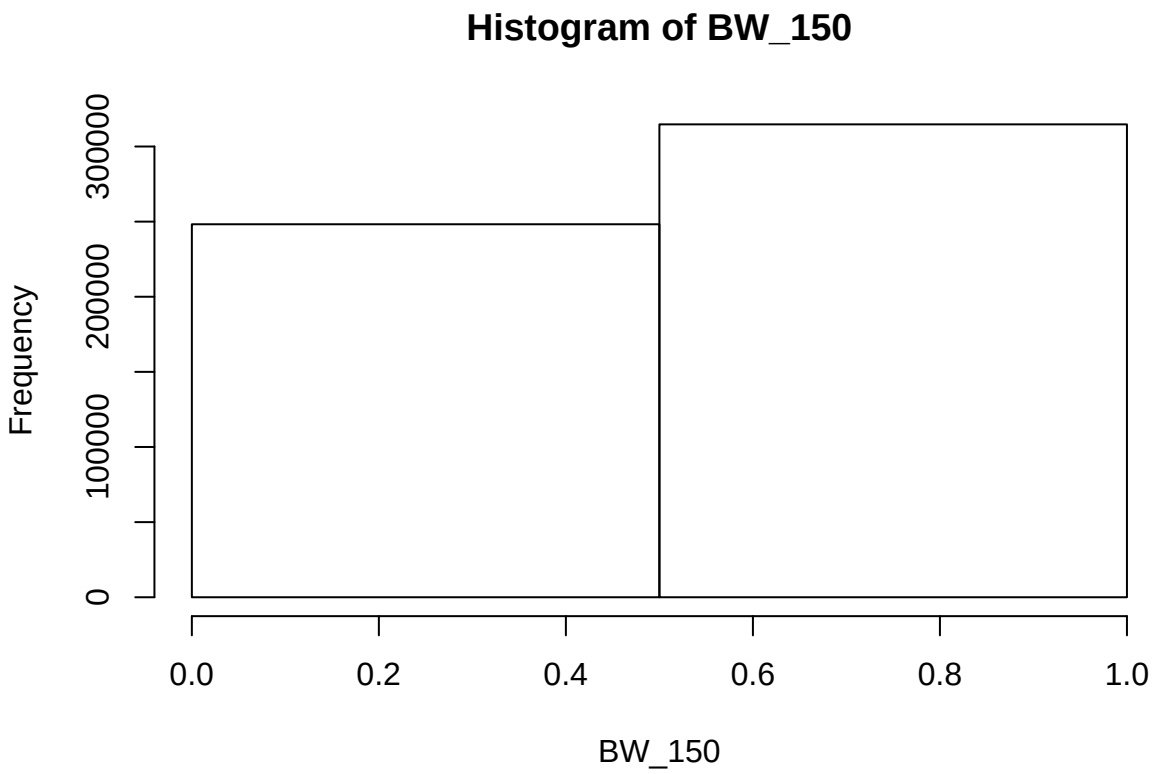

```
plot(bw100_hist)
```




```
layout(matrix(c(1,1), ncol = 1, byrow = TRUE))
bw150_hist <- hist( x = BW_150, breaks = 2, xlab = '', ylab = '',ylim= c(0,20000), main='Histogram for
```



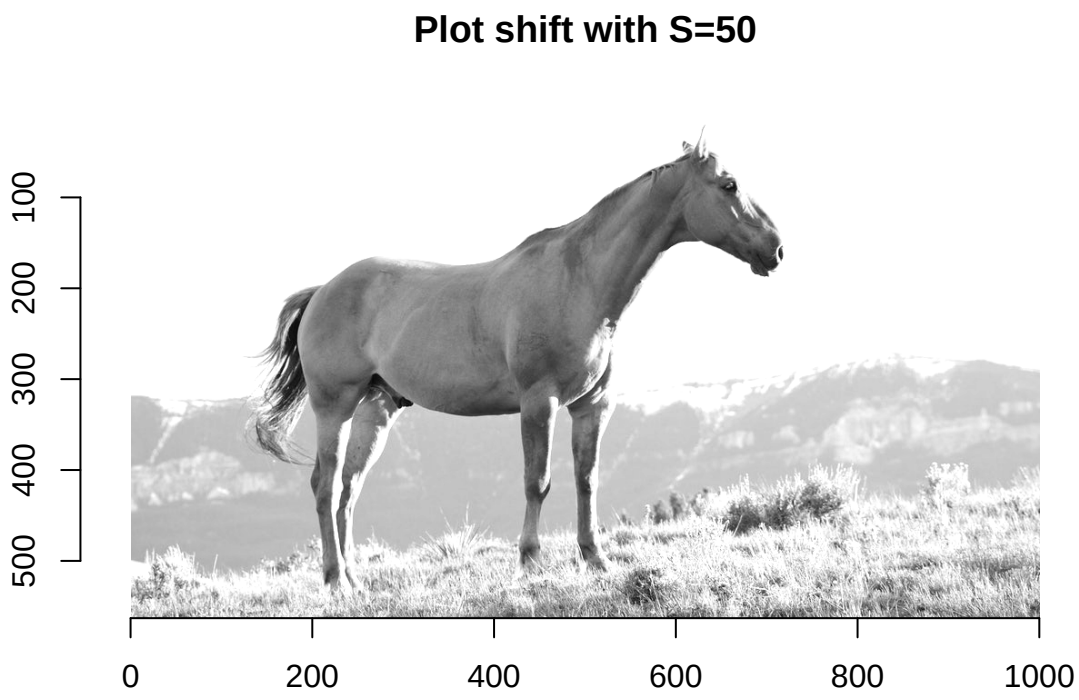

```
plot(bw150_hist)
```



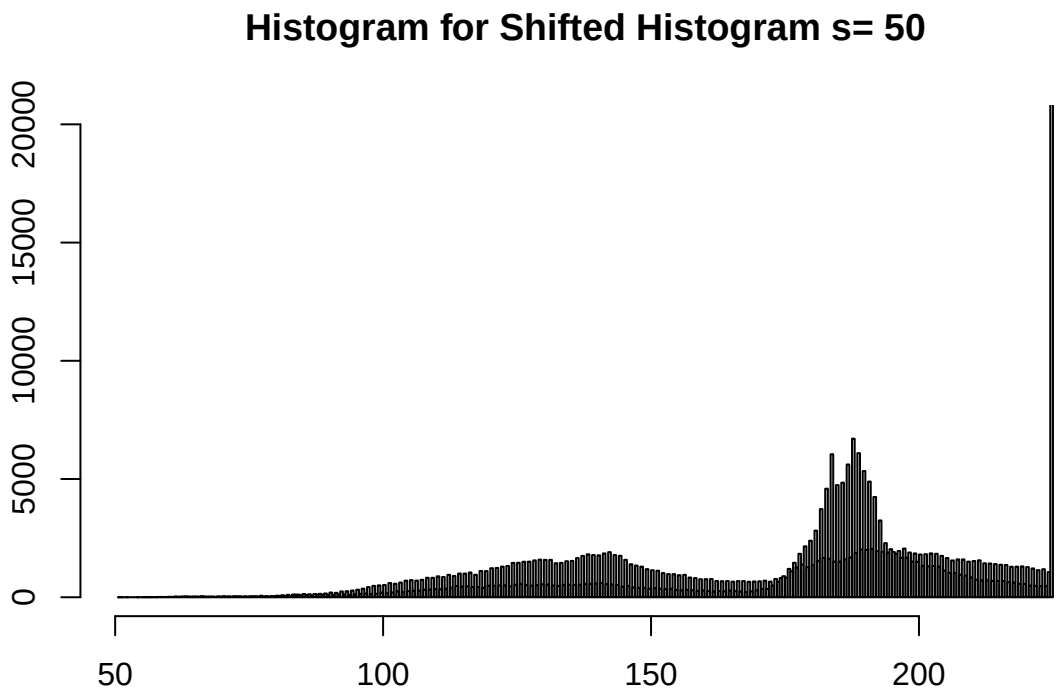
Provided is the histogram showing the output of the black and white images produced from a threshold value of 100 and 150, respectively. We can see from the first result that the count for the white pixels is much higher, than as the threshold is increased to 150, the count goes up significantly as represented in the first bar.

iv) Histogram Shift

```
#layout(matrix(c(1,1,2,2), ncol = 1, byrow = TRUE))
U=225
L=25
s=50
shifted_image=horse_grayscale_rescaled+50
shifted_image[horse_grayscale_rescaled>U-s] = U
shifted_image[horse_grayscale_rescaled<L-s] = L
plot(shifted_image, main='Plot shift with S=50')
```



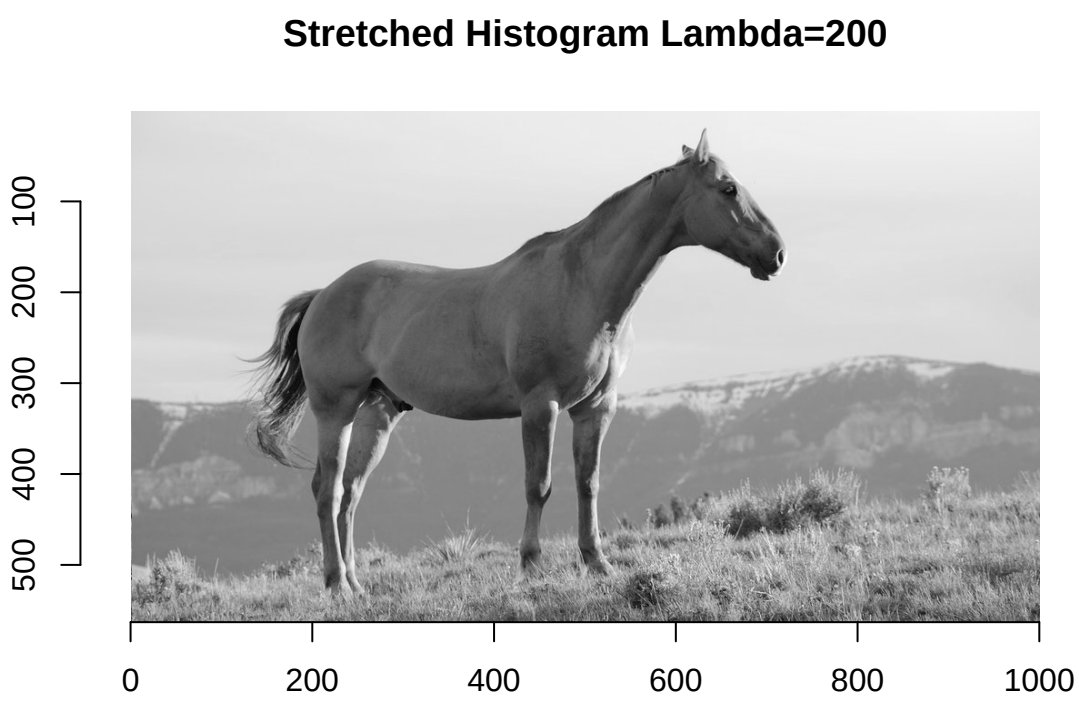

```
hist( x = shifted_image, breaks = 256, xlab = '', ylab = '',ylim= c(0,20000), main='Histogram for Shift
```



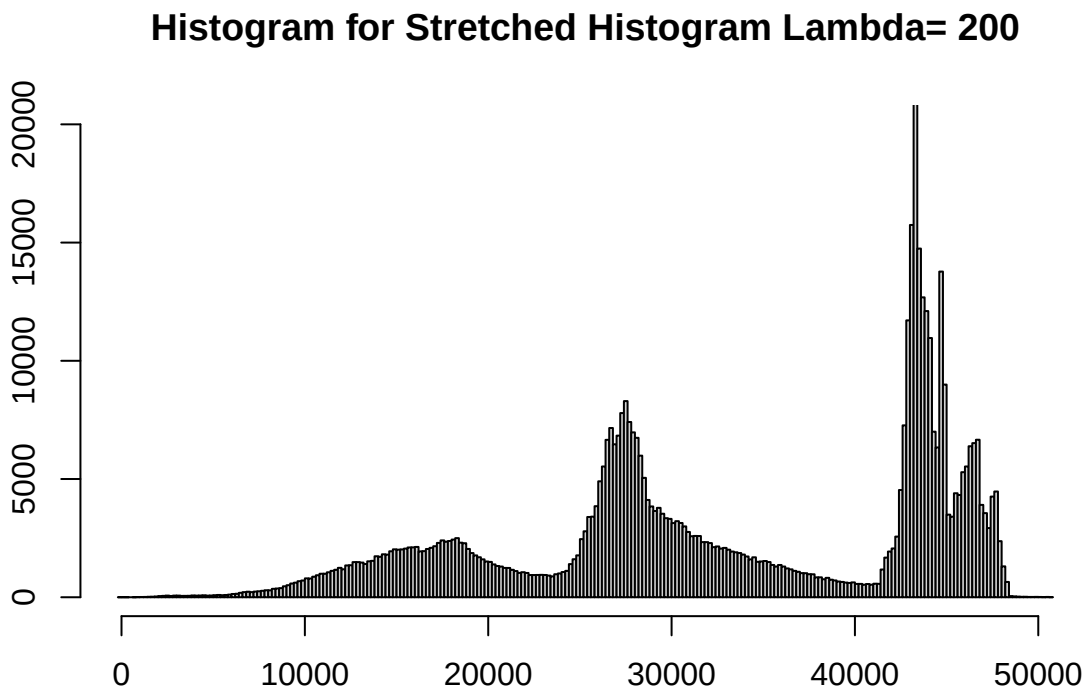

```
#layout(matrix(c(1,1,2,2), ncol = 1, byrow = TRUE))
lambda=200
stretched_image=horse_grayscale_rescaled
min <- min(horse_grayscale_rescaled)
max <- max(horse_grayscale_rescaled)
stretched_image[(horse_grayscale_rescaled-min/max-min)*lambda<0]=0
stretched_image[(horse_grayscale_rescaled-min/max-min)*lambda>0]=(horse_grayscale_rescaled-min/max-min)
```

```
## Warning in NextMethod(): number of items to replace is not a multiple of
## replacement length
```

```
plot(stretched_image, main='Stretched Histogram Lambda=200')
```

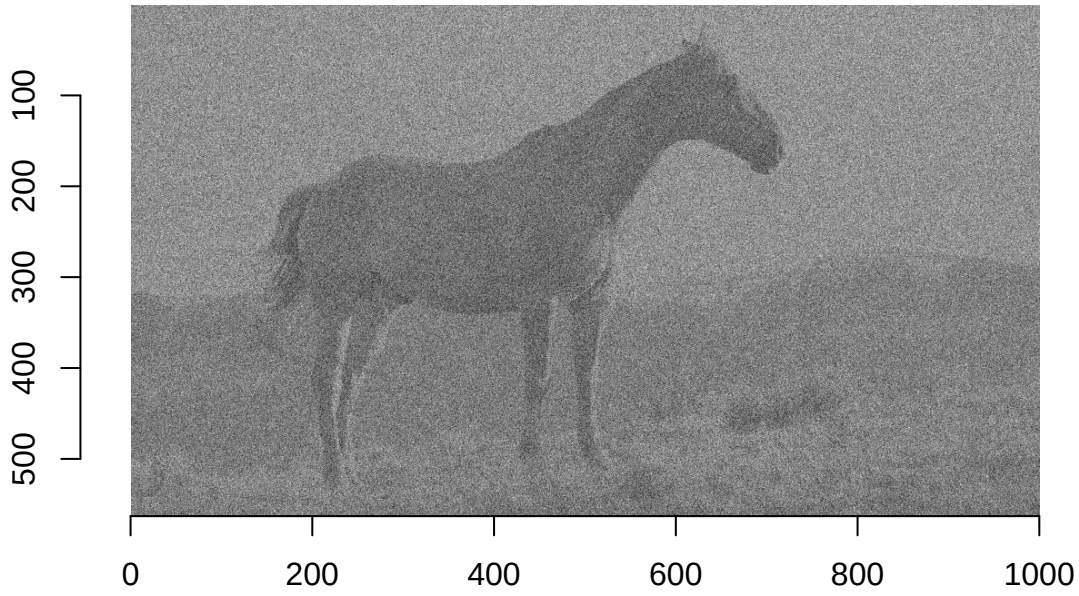



```
hist( x = stretched_image, breaks = 256, xlab = '', ylab = '',ylim= c(0,20000), main='Histogram for Str
```

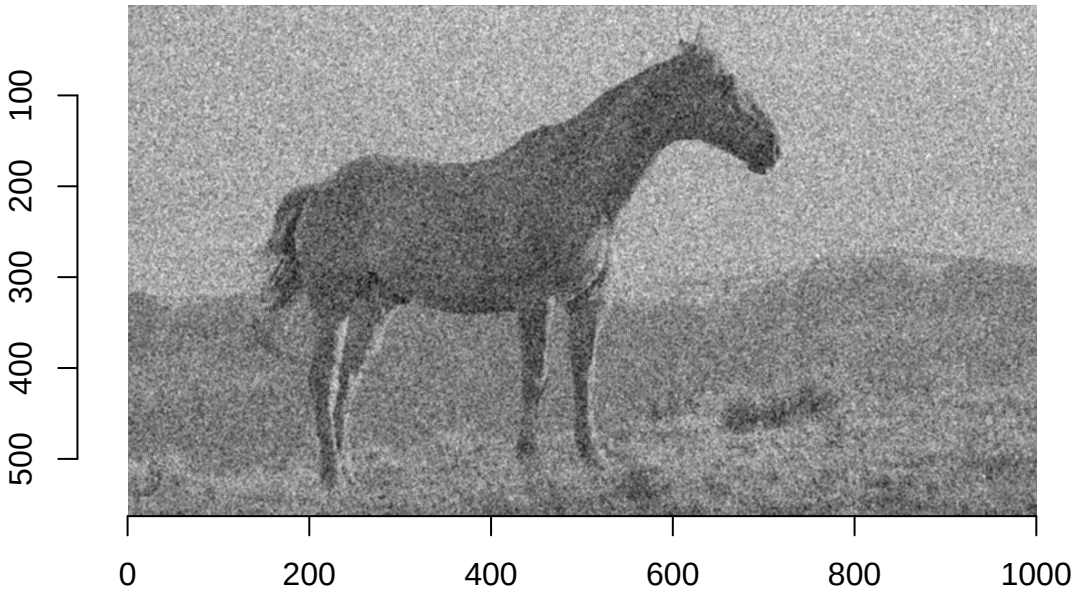


We can see the results when shifting the image that the lower density values from 0 to 50 get completely cut off, and therefore the image appears to be much brighter. For stretching we see that the densities get widened out.

```
#layout(matrix(c(1,1,2,2), ncol = 1, byrow = TRUE))
#@U=225
#noisy_matrix <- as.cimg(matrix(rnorm(n=dim(horse_grayscale_rescaled)[1]*dim(horse_grayscale_rescaled)[
noisy_matrix <- imnoise(dim(horse_grayscale_rescaled)[1],dim(horse_grayscale_rescaled)[2],mean=0,sd=100
noisy_image =horse_grayscale_rescaled+noisy_matrix
plot(noisy_image)
```



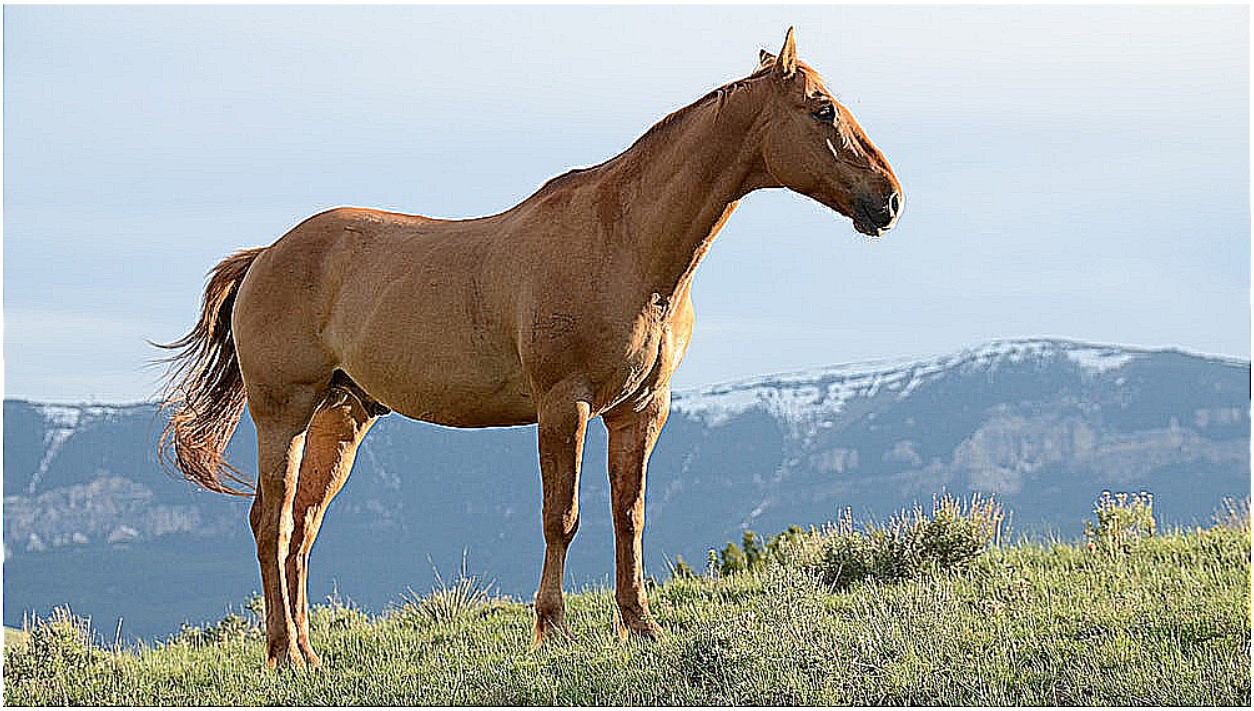

```
K <- array(0, dim=c(3,3,1))
K[,,1] <- matrix(c(1/9,1/9,1/9,1/9,1/9,1/9,1/9,1/9,1/9), nrow = 3, ncol = 3)
Yk <- filter2(as.matrix(noisy_image),K[,,1])
plot(as.cimg(Yk))
```



After applying the transformation matrix to the noisy image, we can see that the clarity is much better compared to the noisy image.

```
horse_im <- readImage('./horse1-2.jpg')
plot(horse_im)
```

```
K <- array(0, dim=c(3,3,1))
K[,,1] <- matrix(c(-1,-1,-1, -1,9,-1, -1,-1,-1), nrow = 3, ncol = 3)
Yk <- filter2(horse_im,K[,,1])
plot(Yk)
```



This is the results for the sharpened image.