Project 4_G16

Arjun Adapalli, Brian Lanier, Christine Mulcahy, Brett Scroggins

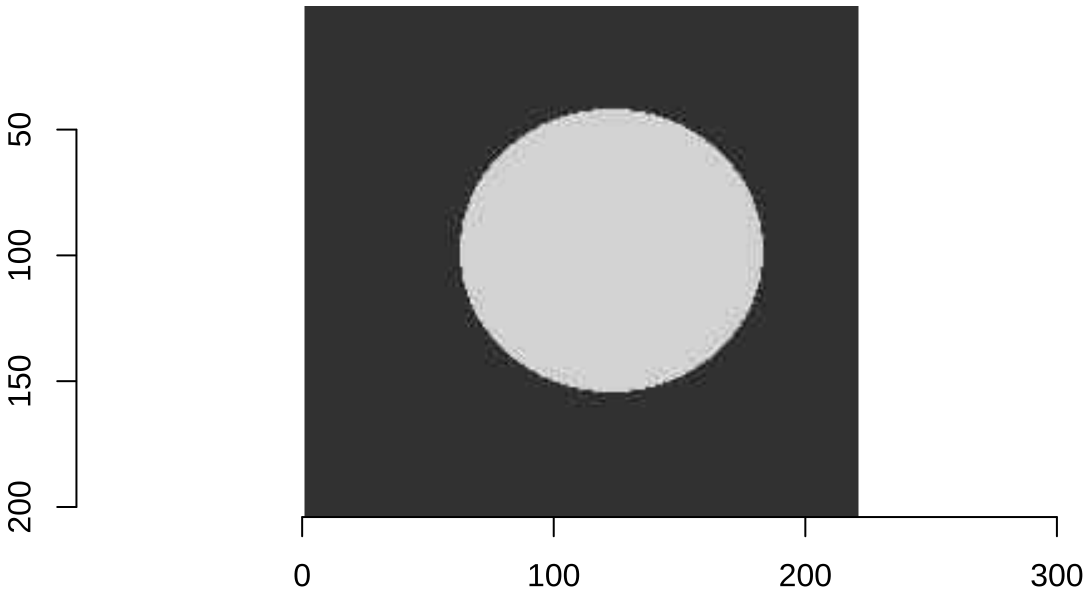
Due 4/11/2018

Part 1

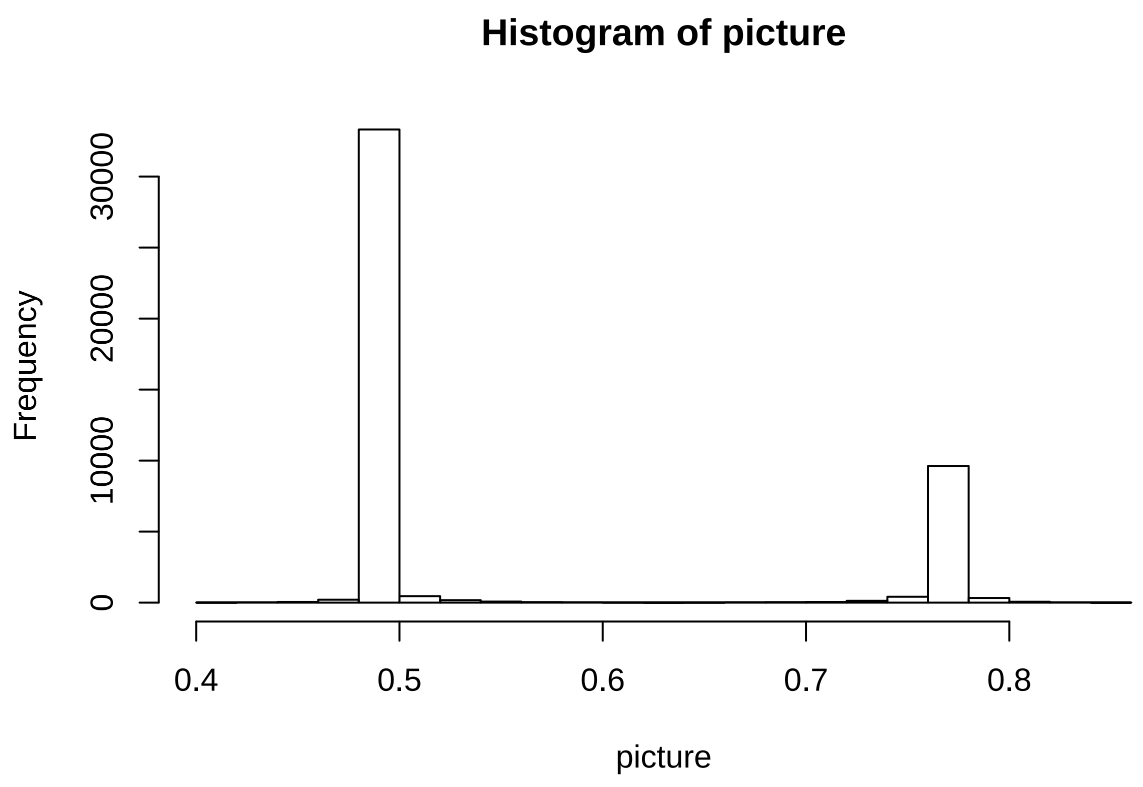
```
pick_threshold = function(picture){
  histogram = hist(picture)
  freq = histogram$counts

  derivatives = diff(sign(diff(freq)))==2
  midpoint = which(derivatives)+1

  thresh = histogram$breaks[midpoint]
  return(mean(thresh))
}
```

Pictures in Black-and-White

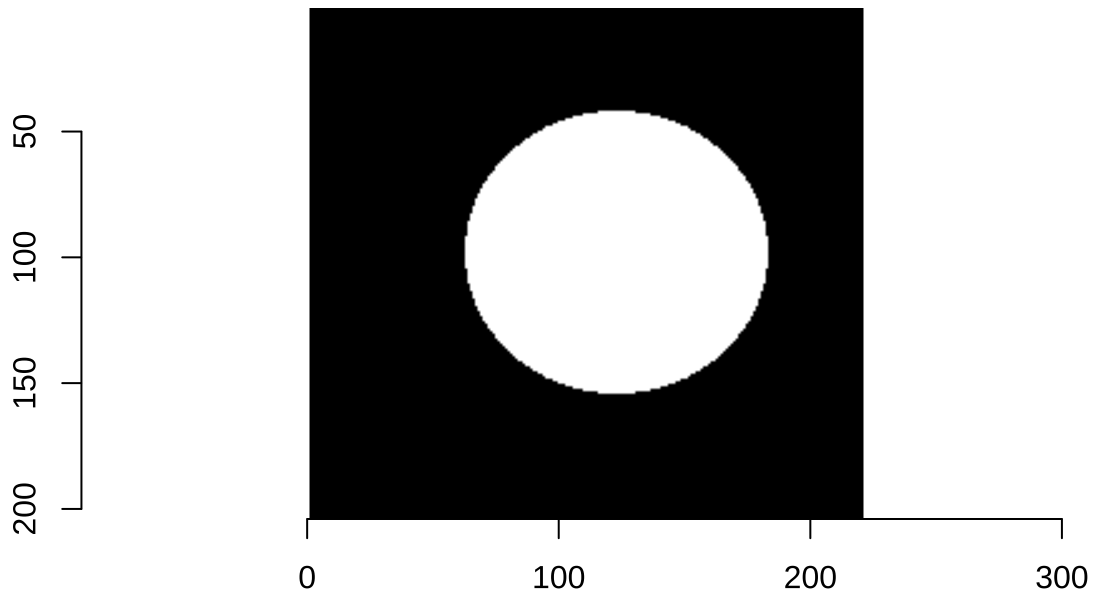
```
pic1 = load.image('Pic1.jpg')
plot(pic1)
```



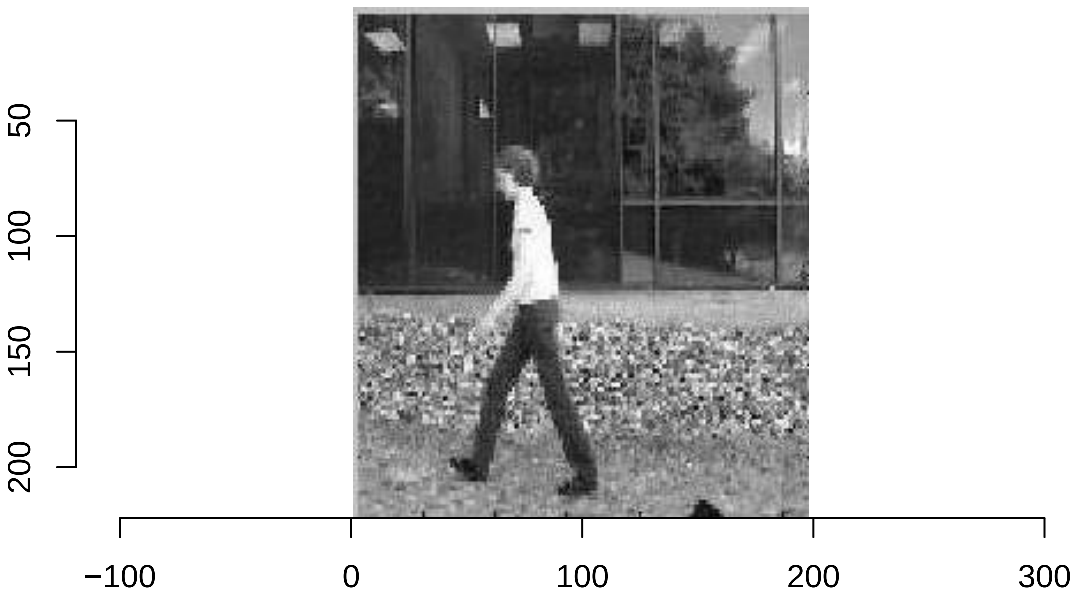
```
trial = pick_threshold(pic1)
```



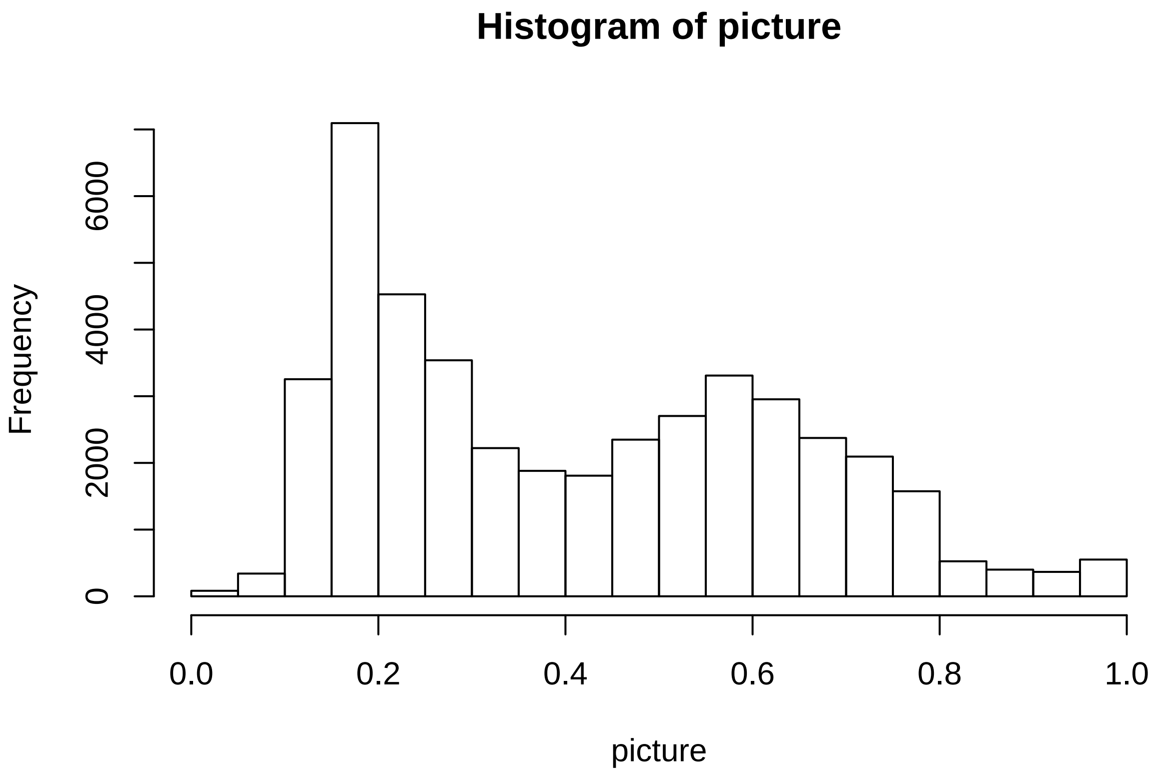
```
plot(pic1>trial)
```



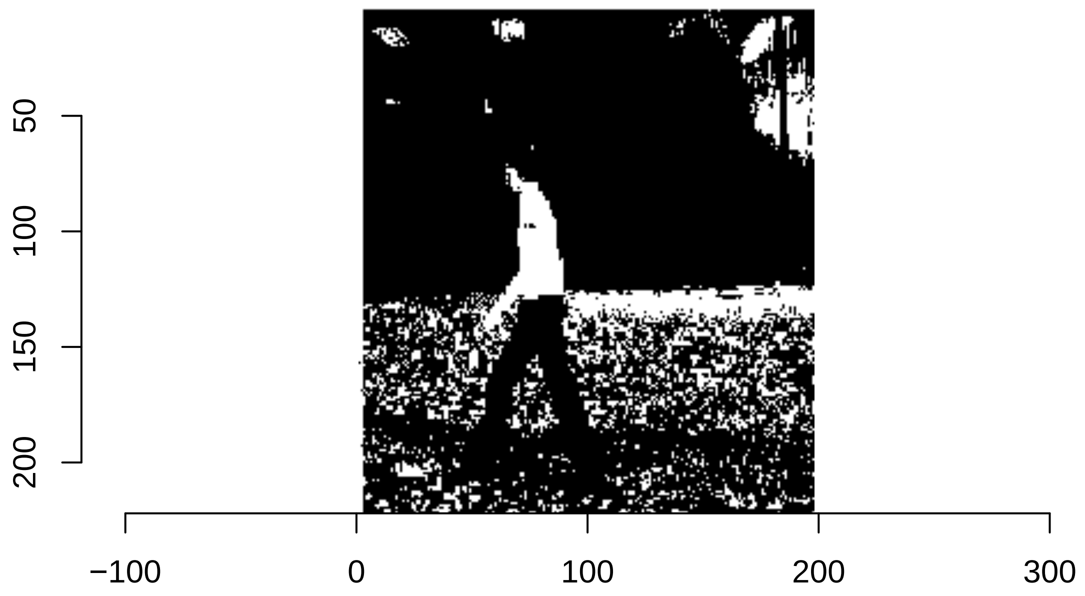
```
pic2 = load.image('Pic2.jpg')
plot(pic2)
```



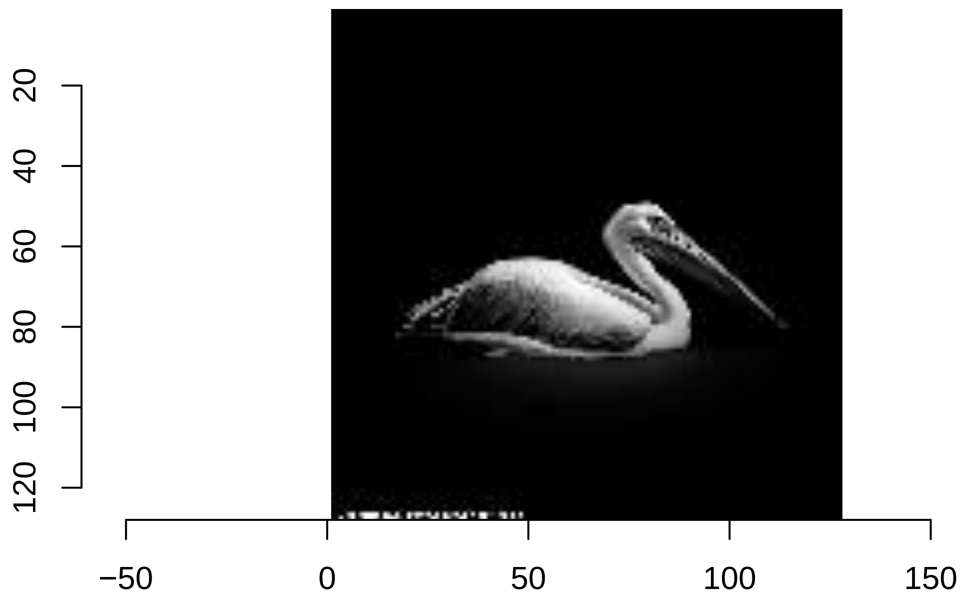
```r
trial2 = pick_threshold(pic2)
```



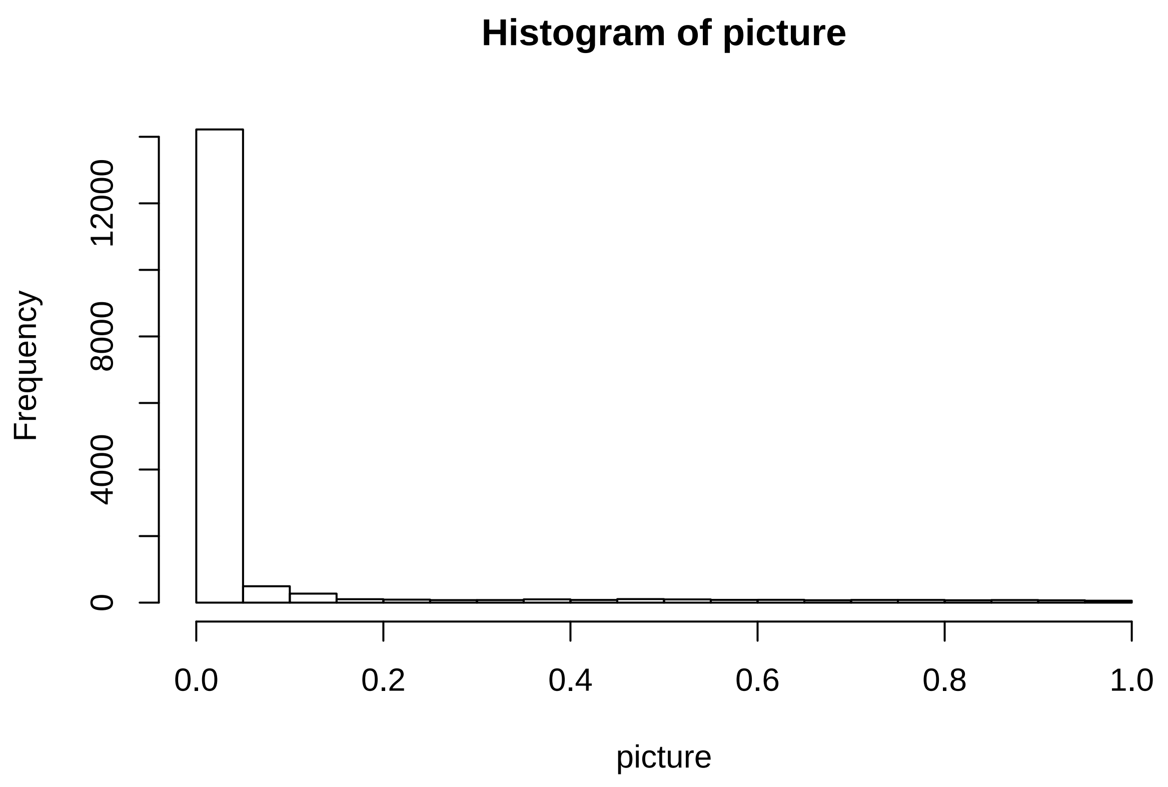
```r
plot(pic2>trial2)
```



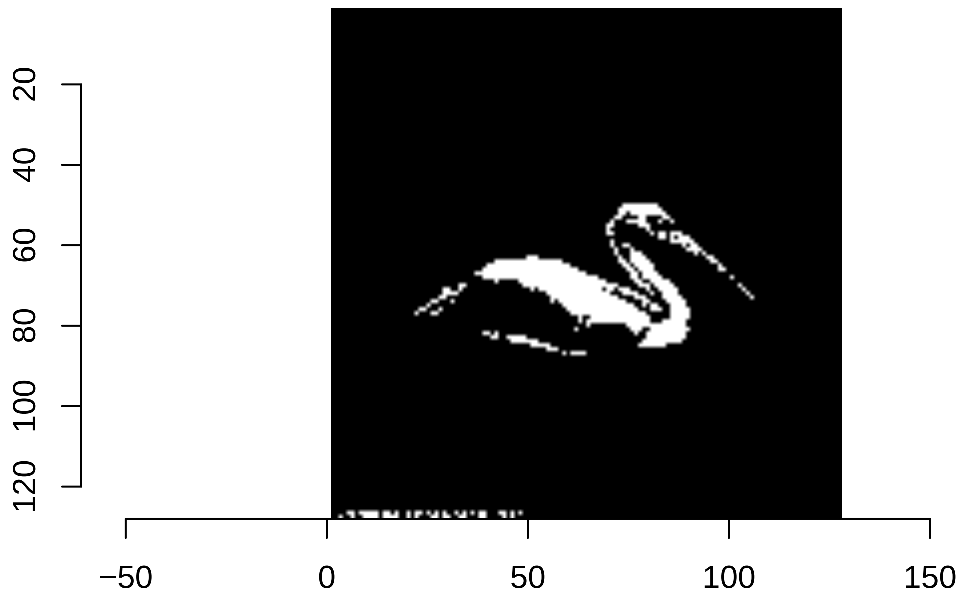
```
pic3 = load.image('Pic3.jpg')
plot(pic3)
```



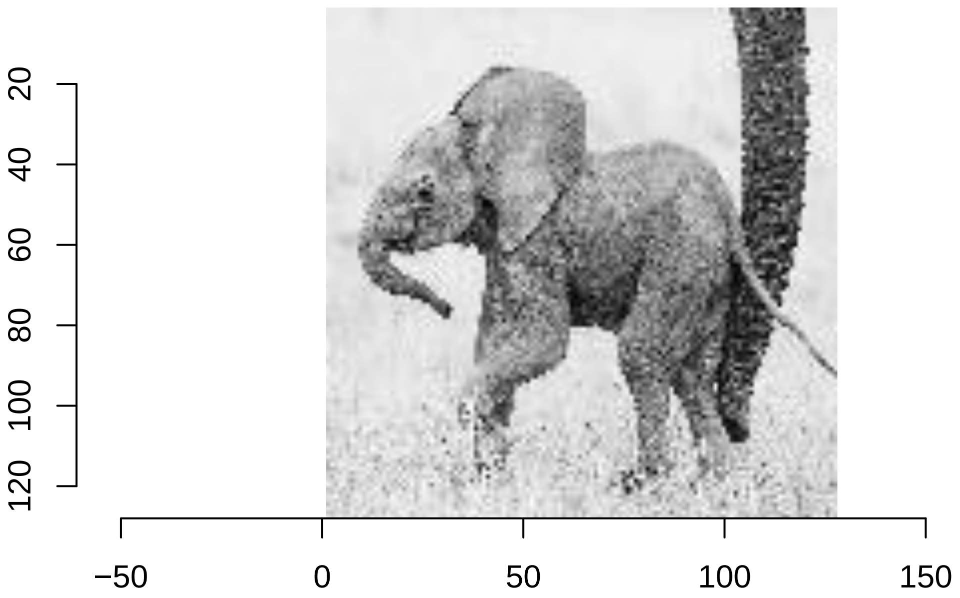
```
trial3 = pick_threshold(pic3)
```



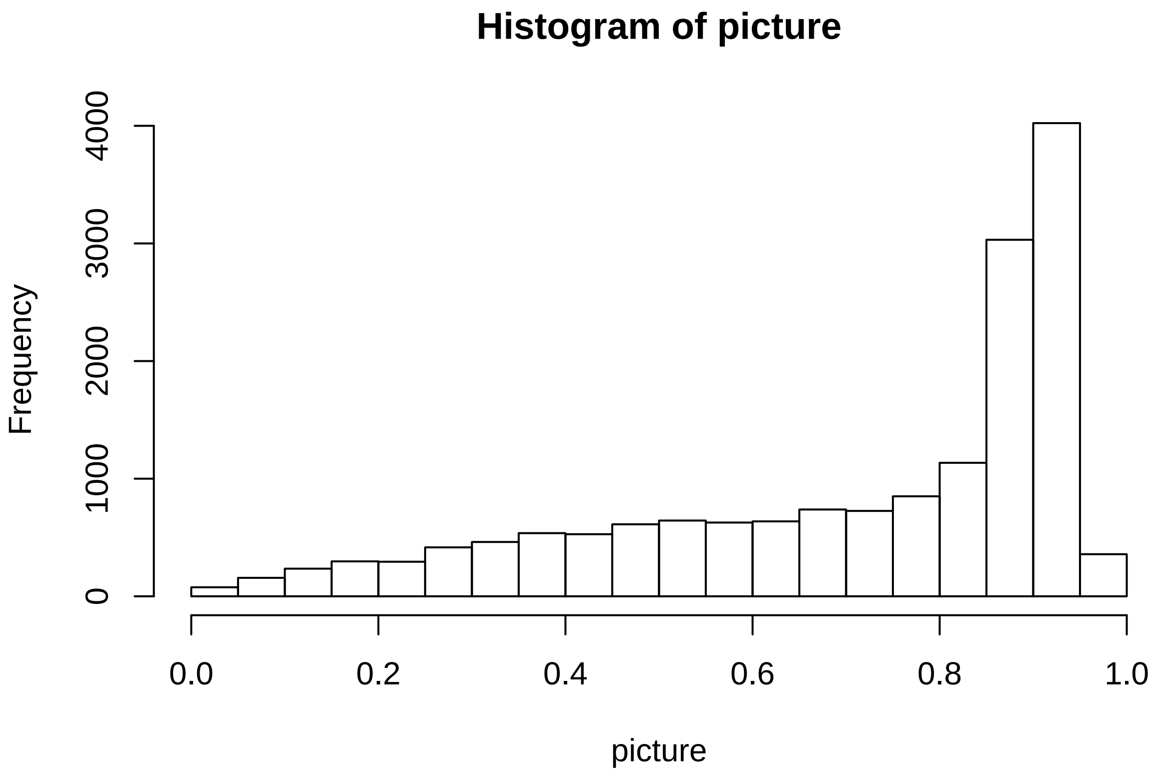
```r
plot(pic3>trial3)
```



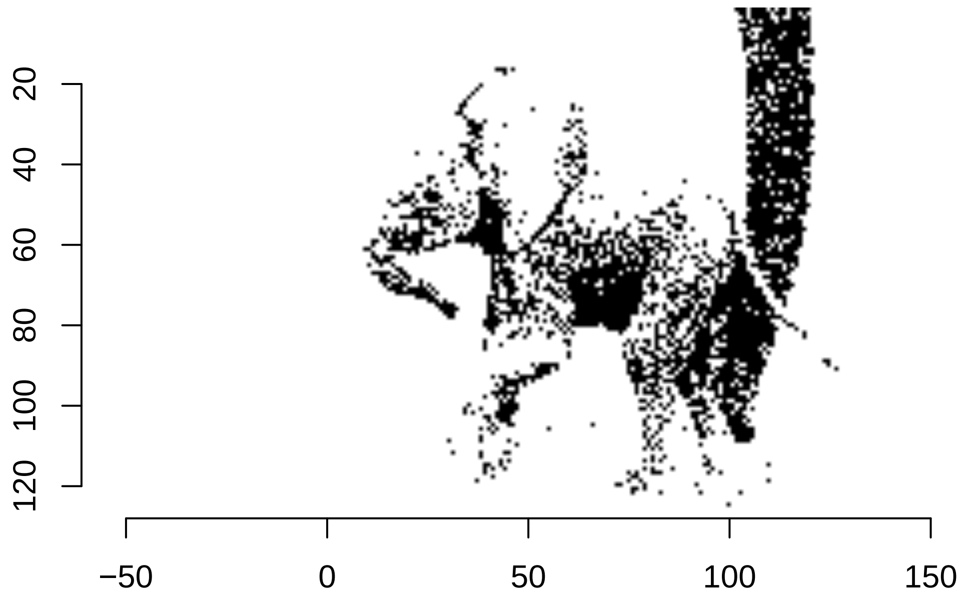
```r
pic4 = load.image('Pic4.jpg')
plot(pic4)
```



```
trial4 = pick_threshold(pic4)
```



```
plot(pic4>trial4)
```



Part 2

```
image_segmentation = function(picture){

  pic = resize(picture,128,128)

  xrange = c(10,120)
  yrange = c(40,128)
  tempforeground = imsub(pic,x %inr% xrange, y %inr% yrange)
  tempbackground = imsub(pic, !(x %inr% xrange), !(y %inr% yrange))

  #compute the mean for each subimage
  p_bar_f = mean(c(tempforeground))
  p_bar_b = mean(c(tempbackground))


  #make the a and b lists
  picture_matrix = matrix(pic, 128,128)

  a_vector = c()
  b_vector = c()

  for (p_i in c(pic)){
    a_i = -log(abs(p_i-p_bar_f)/(abs(p_i-p_bar_f)+abs(p_i-p_bar_b)))
    a_i = min(a_i,1000000)

    b_i = -log(abs(p_i-p_bar_b)/(abs(p_i-p_bar_f)+abs(p_i-p_bar_b)))
    b_i = min(b_i,1000000)


    a_vector= c(a_vector,a_i)
    b_vector= c(b_vector,b_i)
  }

  # get the column pairs
  x = c(1:(dim(picture_matrix)[1]*dim(picture_matrix)[2]))

  # adjacency matrix
  adjacent_matrix = matrix(x, dim(picture_matrix)[1], dim(picture_matrix)[2], byrow=TRUE)
  f1=head(adjacent_matrix, -1)
  t1 = tail(adjacent_matrix, 127)

  # transposed adjacency matrix
  transpose_adjacent = t(adjacent_matrix)
  f2 = head(transpose_adjacent, -1)
  t2 = tail(transpose_adjacent,127)

  # top portion of dataframe of to and from
  to = c(c(t2), c(t1))
  from = c(c(f2), c(f1))
  top_df = data.frame(from, to)

  # Initialize before loop
  pic_list = c(picture_matrix)
  c_vec = c()
  K = .01
  sigma = 1

  # Find capacities
  for (i in 1:nrow(top_df)){

    row = top_df[i,]

    p_i = pic_list[row$from]
    p_j = pic_list[row$to]
    numer = -((p_i-p_j)^2)
    denom = (sigma^2)

    cij = K*exp(numer/denom)

    c_vec = c(c_vec, cij)

  }

  # add capacity to top portion of dataframe
  top_df$capacity = c_vec

  # Create and stack bottom dataframe
  bottom_df = data.frame(top_df$to,top_df$from,top_df$capacity)
  names(bottom_df) = c("from", "to","capacity")

  # Initialize
  lower_df = rbind(top_df,bottom_df)
```

```
  # add source to nodes and nodes to sink
  nodes=seq(1,128^2)
  s=rep(1,128^2)
  t=rep(128^2,128^2)

  nodes_from=c(s,nodes)
  nodes_to=c(nodes,t)

  weights=c(a_vector,b_vector)
  upper_df=data.frame(nodes_from,nodes_to,weights)
  names(upper_df) <- c("from", "to","capacity")
  final_df<-rbind(upper_df, lower_df)

  graph=graph_from_data_frame(final_df, directed = TRUE, vertices = NULL)
  maxflow=max_flow(graph, 1, 128^2)

  new_img = rep(0,128^2)
  new_img[maxflow$partition1]=0
  new_img[maxflow$partition2]=1
  new_img = matrix(new_img,128,128)
  new_img = array(new_img, c(128,128,1,1))
  new_img = as.cimg(new_img)
  color_scale <- scales::gradient_n_pal(c("red","blue"),c(1,0))
  par(mfrow=c(1,2))
  plot(pic,main = 'Input')
  plot(new_img,main='Segmented', colourscale=color_scale,rescale=FALSE)
}
```

Image Segmentation

Image 1

```
image_segmentation(pic1)
```

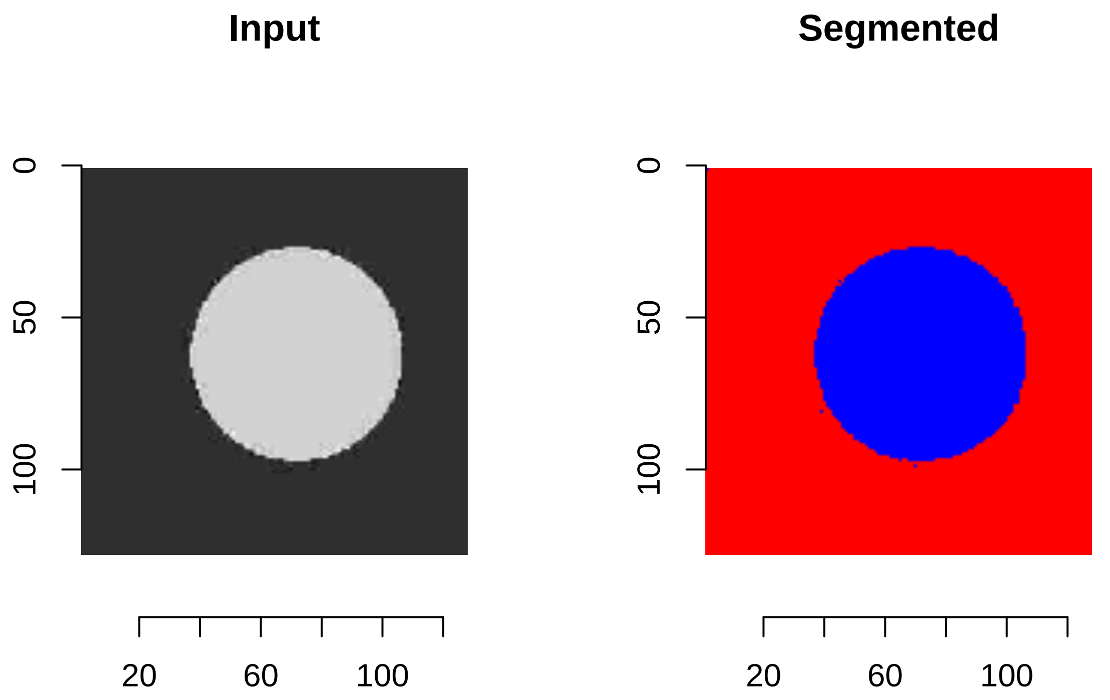


Image 2

```
image_segmentation(pic2)
```

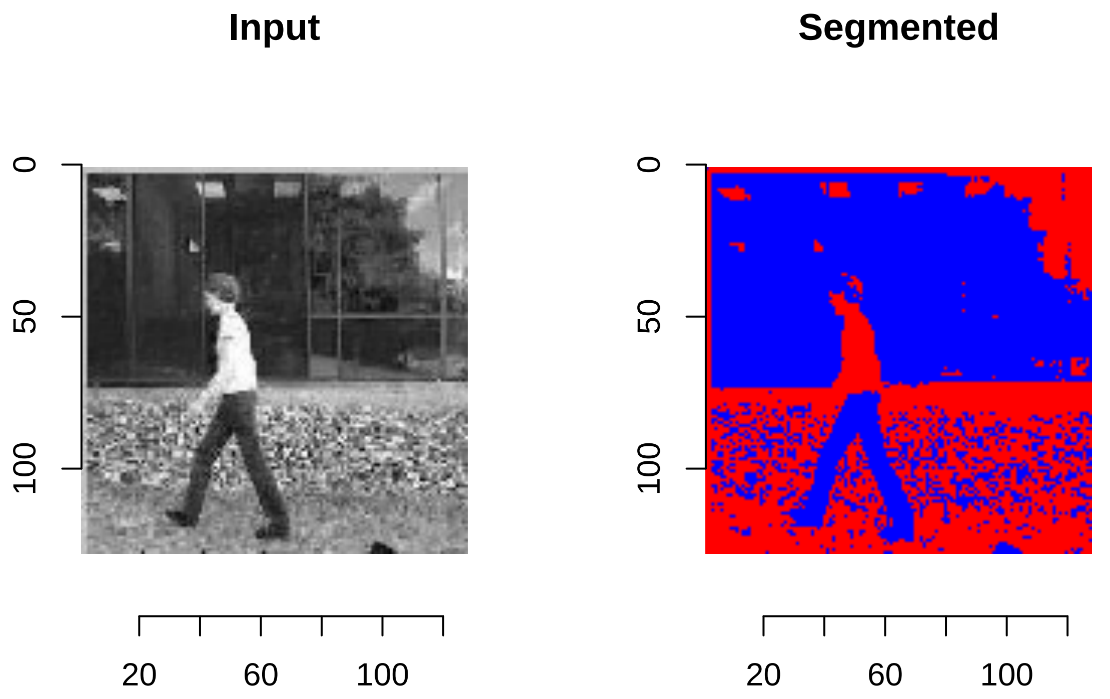


Image 3

```
image_segmentation(pic3)
```

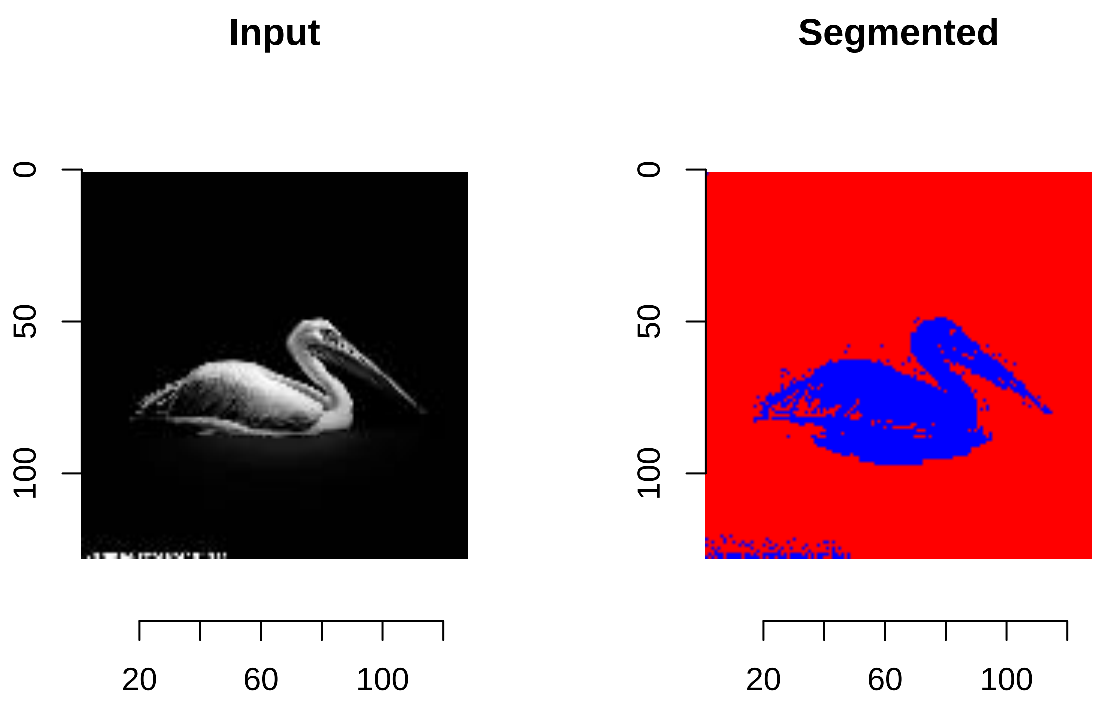
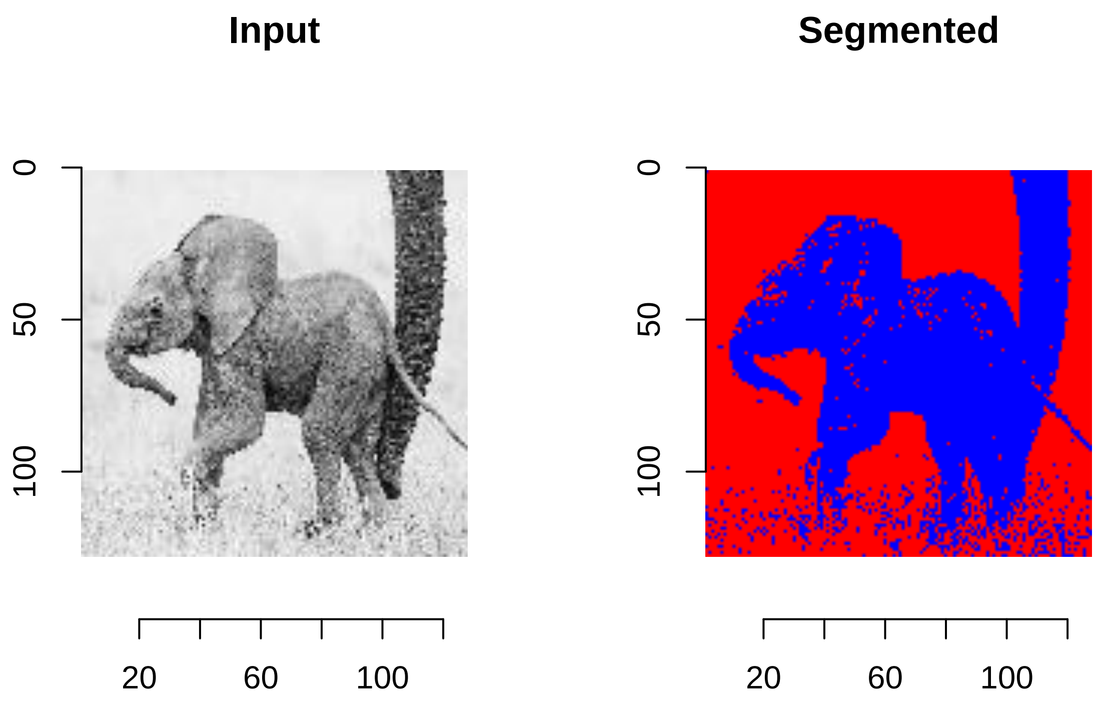


Image 4

```r
image_segmentation(pic4)
```


Input
Segmented