STA 141A Final Project

Group members: Ruochen Zhong, Chloe Liu, Zhuocheng Li, Jixian Fu

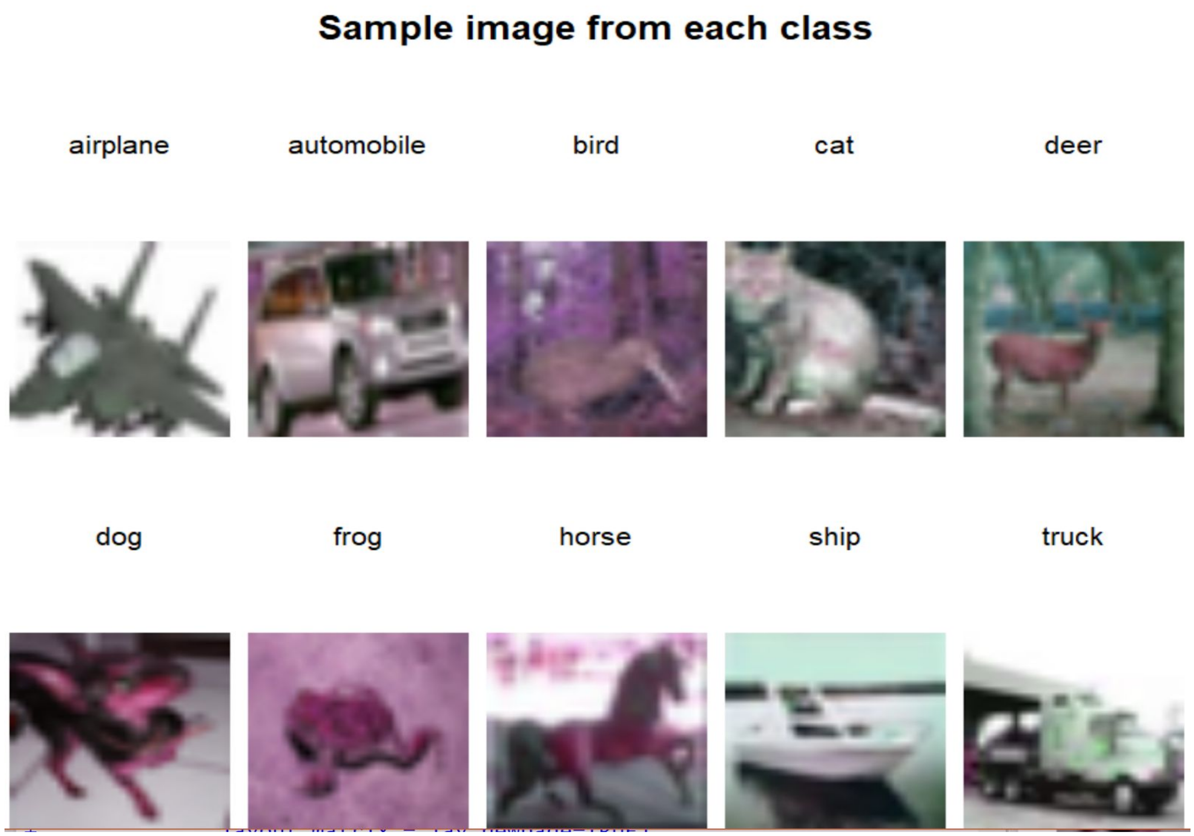
Question 1, 2

Defined functions and detailed comments in the Code Appendix.

Question 3

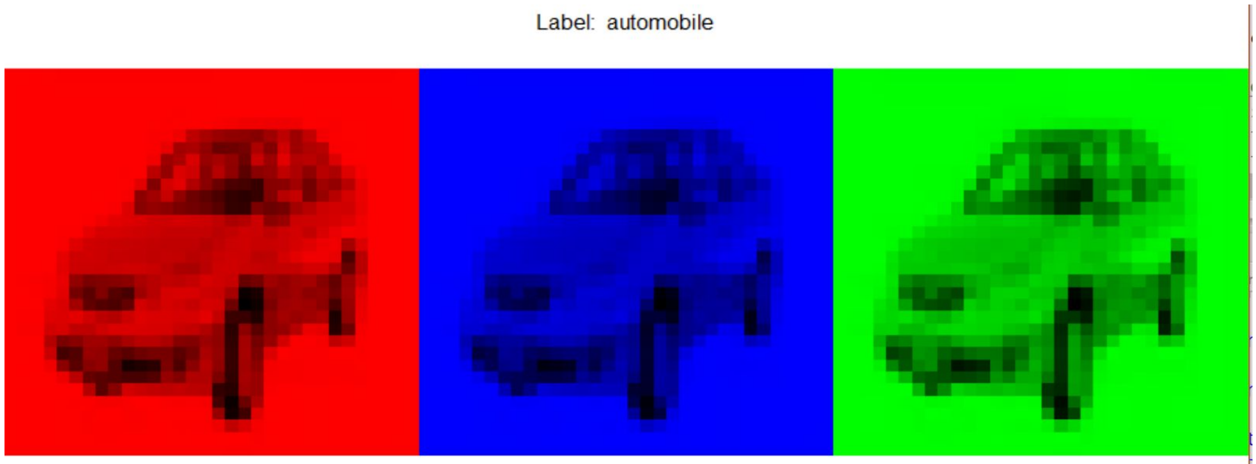
*Reference: http://home.wlu.edu/~lambertk/classes/101/Images.pdf

We have randomly selected one image from each of the 10 classes as displayed below.

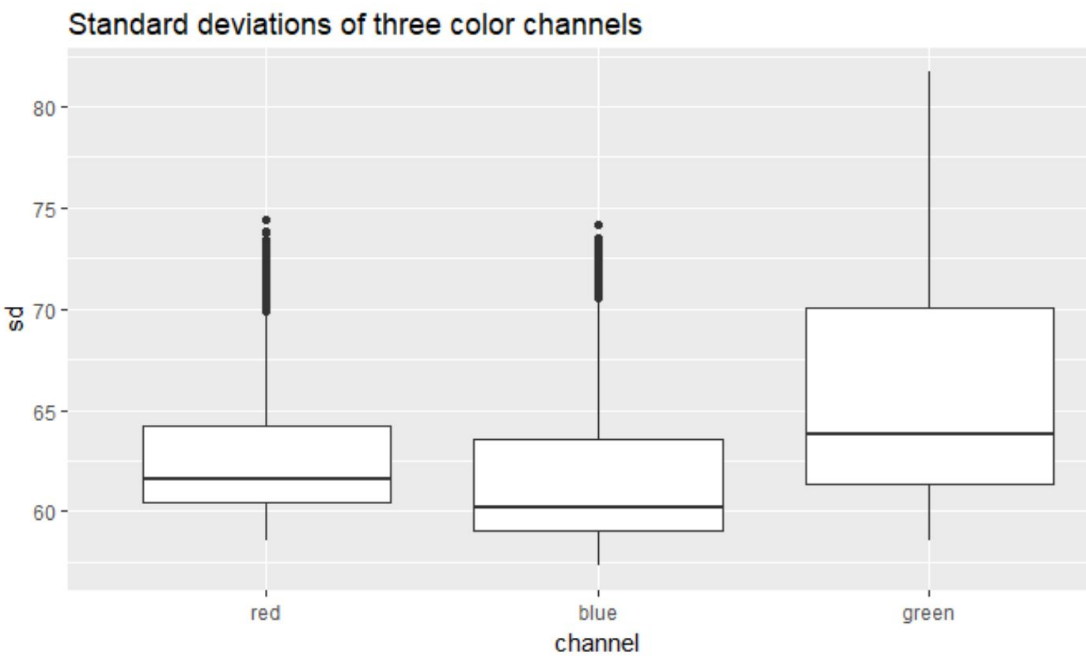


The classified images seem to differ a lot not only between classes but also within each class. To analyze the variability of images in the dataset, we did some research online and discovered that each pixel in our 32×32-pixel image represents a color in the RGB system with three components, and each component can take 256 possible values ranging from 0 to 255.

To illustrate, we separated the R, G, and B color channels of an example image in the dataset as shown below. The variability of values in three channels seem to be different.



After calculating the standard deviations of each pixel for all observations in the training dataset and comparing them among three color channels, it is clearly shown in the boxplots below that green channel values have higher standard deviations and larger variability within the channel, while the blue channel values have lower standard deviations. The results indicate that the green channel might contain the most information about the image and the blue channel contains the least information.



To classify an image into a category, the key features that contribute to image classification seem to be captured from similarity in RGB values. Thus, the key pixels containing these common characteristics would have small variabilities as they are more clustered around the median.

In the table below, we report 5 pixels with the largest and the smallest standard deviations for the whole dataset and for each color channel. When mapping the pixel values in a 2-d 32×32 grid, the first entry of this RGB vector becomes the pixel in the upper left corner, and the last entry would be at the lower right corner, so the pixel positions are as following:

	Ordered by	5 Pixel positions in the 32*32 grid
Whole dataset	Max s.d.	(0,1), (1,1), (1,0), (0,2), (0,31) → all from Green channel
	Min s.d.	(11,15), (11,19), (11,16), (11,20), (12,16) → all from Blue channel
Red channel	Max s.d.	(0,1), (1,0), (1,1), (0,31), (0,2)
	Min s.d.	(11,15), (11,10), (11,19), (11,16), (10,15)
Blue channel	Max s.d.	(0,1), (1,0), (1,1), (0,31), (0,2)
	Min s.d.	(11,15), (11,19), (11,16), (11,20), (12,16)
Green channel	Max s.d.	(0,1), (1,1), (1,0), (0,2), (0,31)
	Min s.d.	(22,17), (21,18), (19,18), (22,18), (20,18)

From the table, it seems that the pixels in the corners of a pixel grid, like (0,1), (1,0), (0,31), etc., are much less useful since they contain too much information like backgrounds that is random and unhelpful for classification. In comparison, pixels that are mapped in the center are more helpful in differentiate images from different classes because they share some key features in common. For the whole dataset in general, the corner pixels from the green channel are the least useful, and the center pixels from the blue channel are the most useful.

Question 5

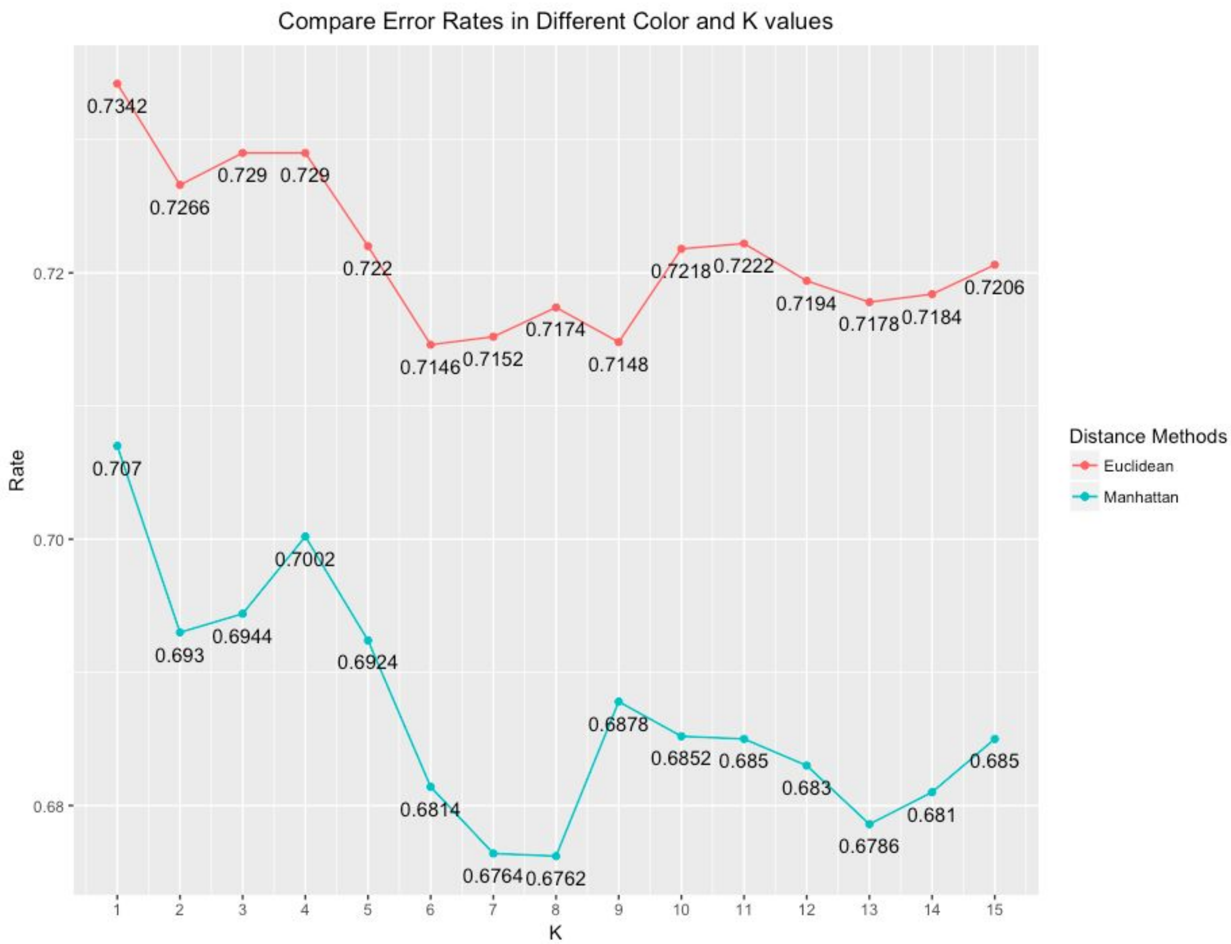
We found that, it is not necessary to calculate the distance matrix everytime we call the predict_knn().

Instead, to avoid computing the distance matrix for each of the 10 folds, in question 4, we have computed the distance matrix for all the 6000 images, including the training set and testing set. Indeed, we have computed two distance matrices using two different parameters,"Euclidean" and "Manhattan", in the “method” argument of dist(), because, in the question 6, we should at least use two different distance metrics. Then, to apply the matrices in this question, we subset each matrix to a 5000 by 5000 matrix, which only includes the distances for the training set.

Now, we can make our function more efficient, since we do not have to compute the distance matrix. Instead, we just need to find the k nearest neighbors according to the 5000 by 5000 matrices that we already have, find the predictions and compute the error rate thereafter.

According to our tests, our cv_error_knn() function only need about 9 seconds to get the error rate for one specific values of k. Therefore, we think the function is efficient enough to be used in the question 6.

Question 6



From this plot, it is clear to see that after using the 10-fold Cross-Validation, the error rate of method “Euclidean” is higher than that of the method “Manhattan” from k=1 to k=15. This means the using the “Manhattan” method is better to predict. If we check each method’s trends, we can find that both of them have a relative high error rates when the k is 1,2,3,4. After that, both trends continue to decrease, and the error rates attain their lowest around **k = 7 and 8**.

From our observation, the best k and distance metric is **Manhattan k = 8,** whose error rate is 0.6828. **The best 3 for Manhattan is k = 7, 8, 13, while the best 3 is k = 7, 8 , 9 for Euclidean.**

Then, both error rates go higher when k is larger than **13.** Therefore, it seems that adding more values of k is not meaningful because both trends of the error rate have shown an clearly increasing pattern after **k = 13.**

Question 7

Confusion Matrix when K = 7 for Manhattan

	0	1	2	3	4	5	6	7	8	9
0	199	1	71	8	26	3	29	10	149	4
1	32	57	40	21	95	17	64	17	118	39
2	41	1	187	20	139	14	47	6	39	6
3	23	4	64	64	116	52	108	25	38	6
4	27	2	100	9	250	8	53	26	22	3
5	24	3	81	39	104	112	75	17	41	4
6	17	3	107	14	124	15	200	10	7	3
7	32	2	68	14	138	25	44	115	41	21
8	81	3	22	2	33	12	10	9	314	14
9	34	22	29	14	50	12	46	32	141	120

Confusion Matrix when K = 8 for Manhattan

	0	1	2	3	4	5	6	7	8	9
0	201	1	72	6	27	3	30	6	149	5
1	31	55	39	21	101	16	61	18	117	41
2	41	1	189	21	135	9	51	8	40	5
3	24	4	74	56	117	53	103	26	36	7
4	27	1	91	9	259	8	52	25	25	3
5	23	2	81	42	102	123	68	17	37	5
6	15	3	107	21	129	14	194	8	6	3
7	28	1	72	13	150	17	45	110	46	18
8	82	3	24	2	39	11	8	10	310	11
9	37	19	36	18	48	8	41	31	140	122

Confusion Matrix when K = 13 for Manhattan

	0	1	2	3	4	5	6	7	8	9
0	230	1	54	3	31	1	28	7	142	3
1	26	50	45	13	109	15	67	10	122	43
2	51	0	189	18	140	6	46	6	39	5
3	25	3	81	56	124	47	98	21	36	9
4	28	0	100	8	263	4	45	20	30	2
5	24	2	88	34	109	112	74	14	37	6
6	13	3	128	14	129	12	178	12	8	3
7	39	0	73	14	170	18	34	93	40	19
8	85	3	21	6	45	7	7	7	309	10
9	36	18	40	14	46	4	40	24	151	127

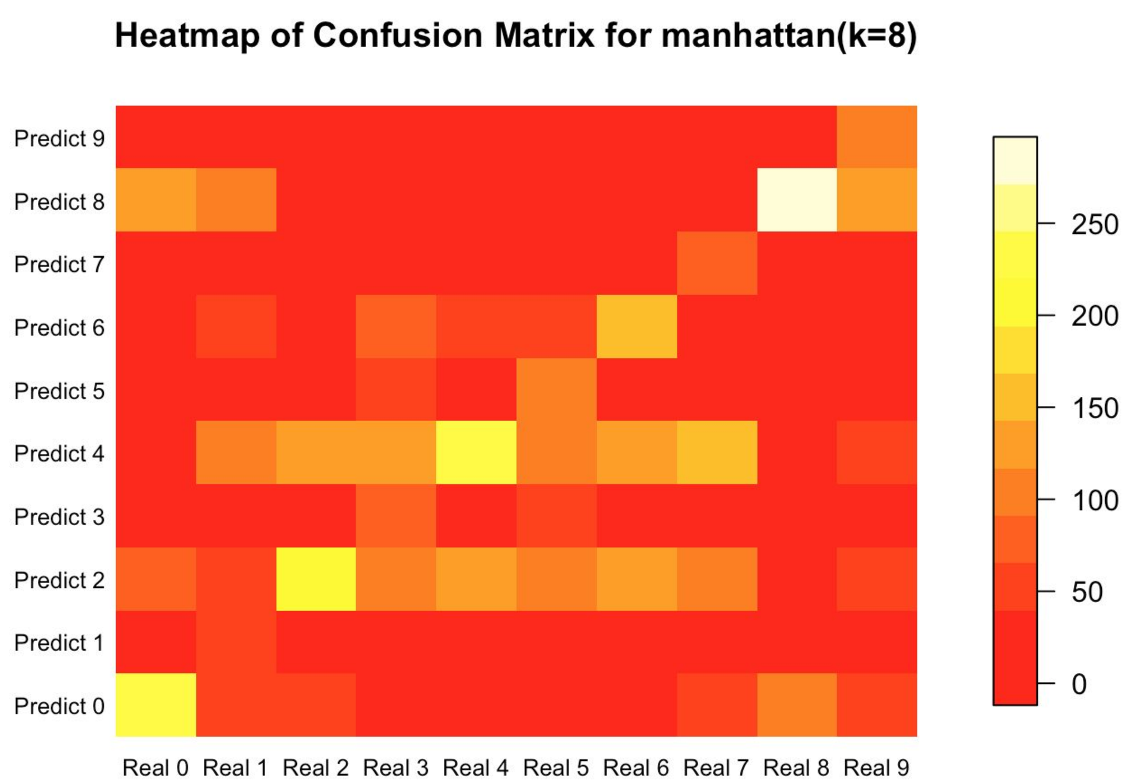
Since we use the same data set for the three best k, we would expect that the confusion matrix of the three best k have the same pattern of miss classification. If we found that one of them have extraordinary low error rate for one of the classes, we may need to change this k and consider an additional one.

Given the confusion matrices and heatmaps above, we found that the 3 best k have similar pattern of errors. The most frequent errors for each of the k all occur at class 1, while the least frequent errors all occur at class 8. Besides, for all 3 best k, it is frequent for a image to be misclassified as class 2 or 4, but a image is not frequent to be misclassified as class 1 or 9. Given the evidence above, we think the 3 best k that we found have similar patterns of misclassification. Therefore, we do not think that we need to change them.

Question 8

Confusion Matrix when K = 8

	0	1	2	3	4	5	6	7	8	9
0	201	1	72	6	27	3	30	6	149	5
1	31	55	39	21	101	16	61	18	117	41
2	41	1	189	21	135	9	51	8	40	5
3	24	4	74	56	117	53	103	26	36	7
4	27	1	91	9	259	8	52	25	25	3
5	23	2	81	42	102	123	68	17	37	5
6	15	3	107	21	129	14	194	8	6	3
7	28	1	72	13	150	17	45	110	46	18
8	82	3	24	2	39	11	8	10	310	11
9	37	19	36	18	48	8	41	31	140	122

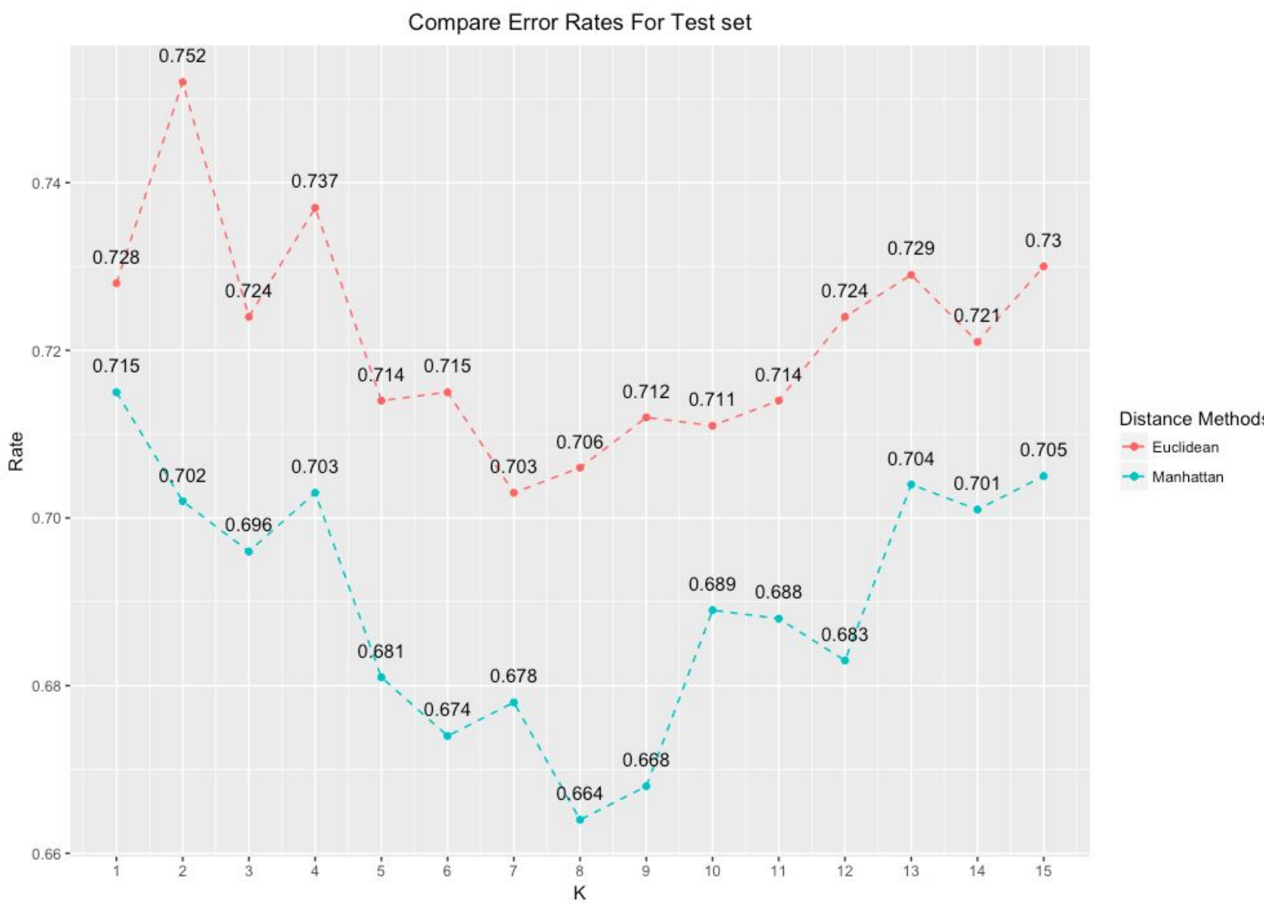


From this heatmap, we can clear see that the classification 1 will be misclassified most frequently among all 10 classifications. From the heatmap, we can find that the classification 1 will be misclassified to classification 4, 6, and 8. This means the graph of **automobile is most likely to be misclassified to deer, frog, and ship**.

Besides, images of class 0 are frequently misclassified as class 8, while images of class 8 are frequently misclassified as class 0. It implies that the graphs of **airplane and ship are hard to be distinguish by KNN**.

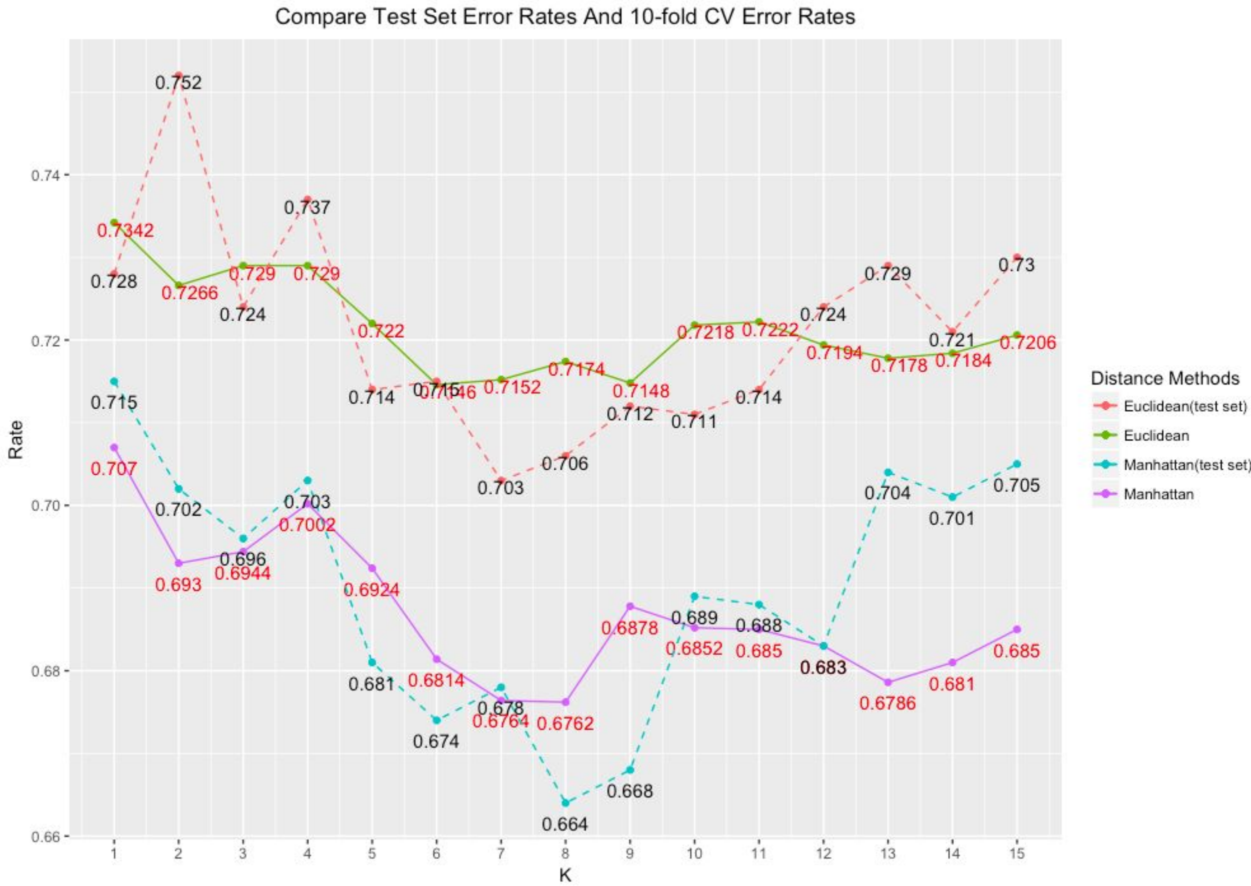
Moreover, many other classes are frequently misclassified as class 2 or 4. this evidence can explain for the high error rates of knn. Therefore, given the high error rate of KNN, **we think that it is necessary to develop a more advanced method to classify the images**.

Question 9



This plot shows the test set error rate for method “Euclidean” and “Manhattan”, we can find that the variation of the trend becomes bigger. For test set, method “Manhattan” is also better than the method “Euclidean”. In this plot, “Manhattan” attains its lowest error rate at K = 8 and “Euclidean” attains its lowest error rate at K = 7. In the following plot, we will compare these two trends with the 10-fold CV error rates.

Compare all 4 trends:



Comparing those four trends, we found that **the trend of both test set** holds a larger variance, which probably comes from the relatively small size of test. Similar with problem 6, the error rate from “Manhattan” distance is generally lower than the error rate from “Euclidean” distance. It proves that “Manhattan” distance is better in measuring **high-dimensional KNN**. The lowest error rate occurs at k = 8 in “Manhattan” distance with test set.

Question 10

It is a pleasure to have our group members help each other throughout the whole project. Therefore, our contributions to the project have some unavoidable overlaps.
Ruochen Zhong: #4, #5,#6,#7,#8,#9, for all graphs.
Jixian Fu: #4, #5,#6,#7,#8,#9, for the algorithm parts.
Chloe Liu: #1, #2, #3, including the code and report.

Zhuocheng Li: #4, #5,#6,#7,#8,#9, for the distance matrix, checking the error rates.

Appendix:

Source Citation:

Question1

#https://stackoverflow.com/questions/48218491/os-independent-way-to-select-directory-interactively-in-r?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa

#https://rstudio.github.io/rstudioapi/reference/file-dialogs.html

#https://stats.idre.ucla.edu/r/faq/how-can-i-read-binary-data-into-r/

#https://rdrr.io/r/base/readBin.html

#https://stackoverflow.com/questions/10089283/combining-different-matrices-in-a-for-loop?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa

Question 2

#http://www.rexamples.com/4/Reading%20user%20input

#https://www.rdocumentation.org/packages/grid/versions/3.5.0/topics/grid.raster

#https://stackoverflow.com/questions/11306075/how-to-create-rgb-image-from-three-matrices-in-r?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa

Question 3

https://cran.r-project.org/web/packages/gridExtra/vignettes/tableGrob.html

#https://cran.r-project.org/web/packages/gridExtra/vignettes/arrangeGrob.html

http://home.wlu.edu/~lambertk/classes/101/Images.pdf

Question 4

#https://www.youtube.com/watch?v=UqYde-LULfs

#https://www.youtube.com/watch?v=GtgJEVxl7DY&t=627s

#https://www.youtube.com/watch?v=DkLNb0CXw84

#https://stackoverflow.com/questions/26693693/get-all-the-maximum-value-indexes-in-a-r-vector

Question 6

http://ggplot2.tidyverse.org/reference/geom_text.html

Question 8

https://stackoverflow.com/questions/10770550/r-how-to-edit-elements-on-x-axis-in-image-plot

Code:

STA141A Final Project

```
## Group member:
# Ruochen Zhong
# Jixian Fu
# Chloe Liu
# Zhuocheng Li

#https://stackoverflow.com/questions/48218491/os-independent-way-to-select-directory-interactively-in-r?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa
#https://rstudio.github.io/rstudioapi/reference/file-dialogs.html
install.packages("rstudioapi")
install.packages("grid")
install.packages("gridExtra")
install.packages("ggthemes")
install.packages("reshape")
install.packages("ggplot2")
install.packages("ggrepel")
library(rstudioapi) #for selectDirectory()
library(grid) #for grid.raster()
library(gridExtra)
library(dplyr)
library(ggthemes)
library(reshape)
library(ggplot2)
library(tidyverse)
library(ggrepel)
library("fields")
library(grDevices)



#read binary data into a matrix
Into.matrix <- function(file) {
  #https://stats.idre.ucla.edu/r/faq/how-can-i-read-binary-data-into-r/
  #https://rdrr.io/r/base/readBin.html
  data_batch = file(file, "rb")
  vec = readBin(data_batch,what="integer",size=1,n=3073*10000,signed=FALSE,endian = "big")
  close(data_batch)
```

```
  mat <- matrix(vec,ncol=3073,byrow = TRUE)
  return(mat)
}

#--------------------------------------------------------------
#1.
load_training_images <- function() {
  #select a directory
  my.directory<- selectDirectory(caption = "Select Input Directory", label = "Select",
                                path = NULL)
  #Set the working directory
  setwd(my.directory)

  training.files <-
c("data_batch_1.bin","data_batch_2.bin","data_batch_3.bin","data_batch_4.bin","data_b
atch_5.bin")

  training_data <- numeric()

#https://stackoverflow.com/questions/10089283/combining-different-matrices-in-a-for-lo
op?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_q
a
  training_data <- lapply(training.files, function(x) lnto.matrix(x))
  training_data <- do.call("rbind", training_data)
  saveRDS(training_data, "training_data.rds")
  return(training_data)
}

load_testing_images <- function() {
  test_data <- lnto.matrix("test_batch.bin")
  saveRDS(test_data, "test_data.rds")
  return(test_data)
}

training_data<- load_training_images()
test_data<- load_testing_images()

data_rescale<-function(labels,k=500)sort(as.vector(sapply(unique(labels),function(i)whi
ch(labels==i))[1:k,]))
```

```
train2<-training_data[data_rescale(train[,1],k=500),]
test2<-test_data[data_rescale(test[,1],k=100),]

#----------------------------------------------------------------------
#2.
#user input
#http://www.rexamples.com/4/Reading%20user%20input
data_select <- function() {
  my.data <- readline(prompt="Enter training or testing to select a dataset:")
  my.data <- as.character(my.data)
  if (grepl("training", my.data, ignore.case = TRUE)) {
    my.data <- train2
  } else if (grepl("testing", my.data, ignore.case = TRUE)) {
    my.data <- test2
  } else{
    print("Error! You need to enter training or testing.")
    my.data <- data_select()
  }
  return(my.data)
}
my.data <- data_select()

obs_select <- function() {
  my.obs <- readline(prompt="Enter a number to choose an observation: ")
  my.obs <- as.integer(my.obs)
  if (!(my.obs <= nrow(my.data) & my.obs >=1)) {
    print(paste("Error! You need to enter an integer between 1 and ", nrow(my.data),"."))
    my.obs <- obs_select()
  }
  return(my.obs)
}
my.obs<- obs_select()


#get class names
labels<- read.table("batches.meta.txt",header=FALSE,col.names="class")

#Display one image (input is training_data or test_data)
view_images <- function(data,obs) {
```

```
  my.class <- labels$class[data[obs,1]+1]
  #Draw image
  #https://www.rdocumentation.org/packages/grid/versions/3.5.0/topics/grid.raster

#https://stackoverflow.com/questions/11306075/how-to-create-rgb-image-from-three-ma
trices-in-r?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google
_rich_qa
  r<- data[obs,2:1025]
  b<- data[obs,1026:2049]
  g<- data[obs,2050:3073]
  col <- as.raster(rgb(r,g,b,maxColorValue = 255))
  col <- matrix(col,nrow = 32,byrow=TRUE)
  plot.new()
  grid.raster(col, interpolate=FALSE)
  title(paste("Label: ",my.class))
  return(col)
}
dev.off()
view_images(train,3828)

#--------------------------------------------------------------------------------
#3.

#randomly select one observation from each class
random_select<- function(i) {
  sub<- train2[train2[,1]==i,]
  robs <- sample(c(1:500),1,replace=TRUE)
  view_images(sub,robs)
}
plots_list <- vector( mode="list" )
plots_list<- lapply(c(0:9),function(x)rasterGrob(random_select(x)))

#References: https://cran.r-project.org/web/packages/gridExtra/vignettes/tableGrob.html
#https://cran.r-project.org/web/packages/gridExtra/vignettes/arrangeGrob.html
lay <- rbind(c(1,1,1,1,1),
             c(2,3,4,5,6),
             c(7,7,7,7,7),
             c(8,9,10,11,12))
#class labels as table grobs
```

```
t1<-tableGrob(d=t(as.character(labels$class[1:5])),theme =
ttheme_minimal(base_size=16,parse=FALSE))
t1$widths <- unit(rep(1/ncol(t1), ncol(t1)), "npc")
t2<-tableGrob(d=t(as.character(labels$class[6:10])),theme =
ttheme_minimal(base_size=16,parse=FALSE))
t2$widths <- unit(rep(1/ncol(t2), ncol(t2)), "npc")

grid.arrange(t1, plots_list[[1]],plots_list[[2]],plots_list[[3]],plots_list[[4]],
             plots_list[[5]],t2, plots_list[[6]],plots_list[[7]],plots_list[[8]],
             plots_list[[9]],plots_list[[10]],ncol=5,
             top=textGrob("Sample image from each
class",gp=gpar(fontsize=20,fontface="bold")),
             layout_matrix = lay,newpage=TRUE)
dev.off()
#------------------------------
#plot RGB channels of an example image
rgb_gradients<- function(i) {
  sub<- train2[train2[,1]==i,]
  robs <- sample(c(1:500),1,replace=TRUE)
  r<- as.raster(rgb(sub[robs,2:1025],0,0,maxColorValue = 255))
  b<- as.raster(rgb(0,0,sub[robs,1026:2049],maxColorValue = 255))
  g<- as.raster(rgb(0,sub[robs,2050:3073],0,maxColorValue = 255))
  pr<- rasterGrob(matrix(r,nrow=32,byrow=TRUE), interpolate=FALSE)
  pg<- rasterGrob(matrix(g,nrow=32,byrow=TRUE), interpolate=FALSE)
  pb<- rasterGrob(matrix(b,nrow=32,byrow=TRUE), interpolate=FALSE)
  title= paste("Label: ",labels$class[i + 1])
  grid.arrange(pr,pb,pg,ncol=3,top=title,newpage = TRUE)
}

rgb_gradients(1)

#Reference: http://home.wlu.edu/~lambertk/classes/101/Images.pdf
#sd of each column (pixel)
sd_train <- apply(train2[,2:3073], 2, sd)
rgb <- c(rep("red",1024), rep("blue",1024),rep("green",1024))
df<- data.frame(sd=sd_train, channel=rgb)
df$channel <- factor(df$channel, levels=unique(df$channel))
#boxplots of sd with respect to color channels
ggplot(df, aes(x=channel,y=sd)) +
```

```
  geom_boxplot() +
  ggtitle("Standard deviations of three color channels")
#-----------------------------------------------
#Order sd of all pixels in a dataframe (input is a dataframe)
order_sd<- function(df) {
  larg_sd <- order(df$sd, decreasing=TRUE)[1:5] #most variation -- top 5 are all from
green channel
  smal_sd <- order(df$sd, decreasing=FALSE)[1:5] #lesat variation -- top 5 are all from
blue channel
  result <- cbind(larg_sd,smal_sd)
  return(result)
}

#put pixels in the 2-D 32*32 grid (input is a vector)
in_grid <- function(data) {
  data<- ifelse(data>2048, data-2048,
              ifelse(data>1024,data-1024,data))
  row <- data %/% 32
  col <- data %% 32
  pixel <- cbind(row,col)
  return(pixel)
}

#For the whole dataset:
result<- order_sd(df)
in_grid(result[,1]) #largest 5 sd
in_grid(result[,2]) #smallest 5 sd

#Separate by colors
red<- order_sd(df[1:1024,])
in_grid(red[,1]);in_grid(red[,2])

blue<- order_sd(df[1025:2048,])
in_grid(blue[,1]);in_grid(blue[,2])

green<- order_sd(df[2049:3072,])
in_grid(green[,1]);in_grid(green[,2])
```

```
#-----------------------------------------------------------------------------------
#4.

##https://www.youtube.com/watch?v=UqYde-LULfs
##https://www.youtube.com/watch?v=GtgJEVxl7DY&t=627s
##https://www.youtube.com/watch?v=DkLNb0CXw84


## find the distancd matrix for twe methods to save the running time for knn function
dist_mat1 = as.matrix(dist(as.matrix(rbind(train2, test2)[,-1]), method = "euclidean"))
dist_mat2 = as.matrix(dist(as.matrix(rbind(train2, test2)[,-1]), method = "manhattan"))


find_knn <- function (test, train, dist, k){

  # take the labels and the pixels
  train <- train[ ,1:3073]
  # only take the pixels
  test <- test[ ,2:3073]
  # an empty vector for the predict results
  classes = vector(mode = "numeric", length = nrow(test))

  for(i in 1:(nrow(test))){
    # sort the distance with the given row i and take the labels for the first k elements
    labels = train[order(dist[i,])[1:k], 1]
    ## table the frequency for each label
    predict_freq<-as.data.frame(table(labels))

#https://stackoverflow.com/questions/26693693/get-all-the-maximum-value-indexes-in-a
-r-vector

    # generate label list for the maximum frequencies
    max_freq = predict_freq[which(predict_freq$Freq==max(predict_freq$Freq)),1]

    # select a random from the maximum frequencies list to avoid tie
    set.seed(141)
    select_max_freq = sample(max_freq,1,replace = T)
```

```
    ##get the final results
    classes[i] = paste(select_max_freq)
  }

  ## output the outcome
  out= as.data.frame(as.numeric(classes))
  out
}


predict_knn = function (test, index1, train,index2, k, method, dist_mat1,dist_mat2) {

  ##select the dist method and subset the dist matrix for test set and train set
  if (method == "euclidean"){
    dist <- dist_mat1[5000+index1, index2]
  }else{
    dist <- dist_mat2[5000+index1, index2]
  }

  ##find the knn in the dist matrix and get the result
  out <- find_knn(test,train, dist,k )
  out
}


# try it on the test data
predict_knn(test2[121:130,] ,c(121:130),train2[1:2000,], c(1:2000),7,"euclidean",
dist_mat1,dist_mat2)
# correct classes
test2[121:130,1]


##Q5-------------------------------------------------------------------------------------
##subset the distance matrixs for the training set
mat1 <- dist_mat1[1:5000, 1:5000]
mat2 <- dist_mat2[1:5000,1:5000]

cv_error_knn = function(train_data,index, k,metric, dist_mat1,dist_mat2) {
```

```
## count the rows in total
rows = nrow(train_data)

# split into 10 equal folds
cut = rows/10

# create 10 empty rates
rate = rep(0,10)

#Perform 10 fold cross validation
for(i in 1:10) {
  #find the start point and end point for each time
  start = (i-1)*cut+1
  end = i*cut

  ## find the test set
  test = train_data[start:end, ]
  ##leave the rest for training set
  train = train_data[-(start:end), ]

  ##find the testing and training range in the dist matrix
  range = index[start:end]
  range2= index[-(start:end)]

  ##selecting the dist method
  if (metric == "euclidean"){
  dist <- mat1[range, range2]
  } else{
  dist <- mat2[range, range2]
  }

  #run the knn function and get the predicitons of classes
  predicts = find_knn(test,train, dist,k )

  #find the correct classes
  correct = test[,1]

  ##find the error rate
  rate[i] = length(which(predicts !=correct))/length(correct)
```

```
  }
  ##get the mean of the 10 rate as the final result
  meanRate = mean(rate)
  meanRate
}

## Put appropriate k-values and distance metric in function parameters below
## the number of rows should be a multiple of 10
cv_error = cv_error_knn(train2[1:5000,], c(1:5000),10,"euclidean", mat1, mat2)
cv_error


#Q6-------------------------------------------------------------------------------------------

#write a function to caculate the rate from k=1 to k=15 for those two method
display_rates <- function(data, index, method1, method2) {
  #create rate1 and rate2
  rate1 <- numeric(15)
  rate2 <- numeric(15)
  # assgin the caculated value to rate1 and rate2
  for (i in 1:15) {
    rate1[i] <- cv_error_knn(data, index,i,method1, mat1, mat2)
    rate2[i] <- cv_error_knn(data, index,i,method2, mat1, mat2)
  }
  # combine the result to be a matrix
  return(cbind(rate1,rate2))
}


#run all 5000 data
display_error_rate <- display_rates(train2[1:5000,],c(1:5000),"euclidean","manhattan")

## for draw the results
# create a dataframe
display_error_rate <- as.data.frame(display_error_rate)
# create a new variable k in the dataframe
display_error_rate$k <- c(1:15)
# draw a plot
```

```
#learn how to add label at http://ggplot2.tidyverse.org/reference/geom_text.html

     #draw the line and point firstly, and use colour to distinguish it
P1 <- ggplot() + geom_line(aes(x = k, y = rate1,colour = 'green'), data =
display_error_rate) +
     geom_point(aes(x = k, y = rate1, colour = 'green'), data = display_error_rate) +
     #label the exact value of the error rate for each k
     geom_text(aes(x = k, y = rate1, label = rate1, vjust = 2), data = display_error_rate) +
     #draw another method's error rate, use colour to distinguish it
     geom_line(aes(x = k, y = rate2, colour = 'red'), data = display_error_rate) +
     geom_point(aes(x = k, y = rate2, colour = 'red'), data = display_error_rate) +
     #label the exact value of the error rate
     geom_text(aes(x = k, y = rate2, label = rate2,vjust = 2), data = display_error_rate) +
     #make the x-axis to be divided to 15
     scale_x_continuous(breaks = c(1:15)) +
     #give an appropirate legend for the graph
     scale_color_discrete(name = "Distance Methods", labels =
c("Euclidean","Manhattan")) +
     #add title
     ggtitle("Compare Error Rates in Different Color and K values") +
     theme(plot.title = element_text(hjust = 0.5)) +
     labs(x = "K", y = "Rate")

print(P1)




#Q7-------------------------------------------------------------------------------------
create_errormatrix = function(train_data,index,k,metric, mat1,mat2) {
  ## count the rows in total
  rows = nrow(train_data)

  ## find the correct classes
  correct <- train_data[,1]

  ## cut the dist matrix into 10 folds
  cut = rows/10
```

```
  ## create an empty vector for the predicts
  all_Predicts = c()

  ##run the knn() for each fold
  for(i in 1:10) {
    #find the test set for each time of test
    start = (i-1)*cut+1
    end = i*cut

    ##subset the testing set
    test = train_data[start:end, ]
    ##leave the rest for training set
    train = train_data[-(start:end), ]

    ##find the testing and training range in the dist matrix
    range = index[start:end]
    range2= index[-(start:end)]

    ##selecting the dist method
    if (metric == "euclidean"){
      dist <- mat1[range, range2]
    }else{
      dist <- mat2[range, range2]
    }

    ##run the knn function and get the predicitons of classes
    Predicts = find_knn(test,train, dist,k )
    ## paste the predicts into the vector
    all_Predicts = c(all_Predicts,Predicts[,1])
  }

   ## get the confusion matrix
  table(as.factor(correct),as.factor(all_Predicts))
}

## for draw the resuts

# For the three best k, draw their heatmap
# for manhattan k = 7
```

```
A <- create_errormatrix(train2[1:5000,],c(1:5000),7,"manhattan", mat1, mat2 )
A <- as.matrix(A)
print(A)

# for manhattan k = 8
B <- create_errormatrix(train2[1:5000,],c(1:5000),8,"manhattan", mat1, mat2 )
B <- as.matrix(B)
print(B)

# for manhattan k = 13
C <- create_errormatrix(train2[1:5000,],c(1:5000),13,"manhattan", mat1, mat2 )
C <- as.matrix(C)
print(C)


#Q8------------------------------------------------------------------------------------

##print the matrix we consider as the best in question 7
#learn how to add legend and text
at:https://stackoverflow.com/questions/10770550/r-how-to-edit-elements-on-x-axis-in-im
age-plot
print(B)
#draw the image firstly without axes
image(B, axes = F)
#add legend, title, and define the color of the heatmap
image.plot(B, legend.only=F, axes = F, col = heat.colors(12), main = "Heatmap of
Confusion Matrix for manhattan(k=8)")
#give the appropriate x-axes and y-axes values
mtext(text=c(paste("Real", 0:9)), side=1, line=0.3, at=seq(0,1,0.111), las=1, cex=0.8)
mtext(text=c(paste("Predict",0:9)), side=2, line=0.3, at=seq(0,1,0.111), las=2, cex=0.8)


#Q9------------------------------------------------------------------------------------
test_set_error_knn <- function(test, index1, train,index2, k, method,
dist_mat1,dist_mat2) {
  ##use the function in #4 to generate predictions of classes
  predicts = predict_knn(test, index1, train,index2, k, method, dist_mat1,dist_mat2)

  #find the correct classes
```

```
  correct = test[,1]

  ##find the error rate
  rate= length(which(predicts !=correct))/length(correct)

  ##output the rate
  rate
}

##create two empty vector for the results
errRate1 <- numeric(15)
errRate2 <- numeric(15)

##find the error rate for each k
for (k in 1:15){
  errRate1[k] = test_set_error_knn(test2[1:1000,], c(1:1000), train2[1:5000,], c(1:5000),
k,"euclidean" , dist_mat1, dist_mat2)
  errRate2[k] = test_set_error_knn(test2[1:1000,], c(1:1000), train2[1:5000,], c(1:5000),
k,"manhattan" , dist_mat1, dist_mat2)
}
errRate1
errRate2

## for draw the resuts
#create a dataframe to store those two error rates
Test_error_data <- cbind(errRate1,errRate2)
Test_error_data<- as.data.frame(Test_error_data)
#create a new variable k
Test_error_data$k <- c(1:15)
# draw a plot
#learn how to add label at http://ggplot2.tidyverse.org/reference/geom_text.html

    #draw the line and point firstly, and use colour to distinguish it
P2 <- ggplot() + geom_line(aes(x = k, y = errRate1,colour = 'green'), data =
Test_error_data, linetype = "dashed") +
    geom_point(aes(x = k, y = errRate1, colour = 'green'), data = Test_error_data) +
    #label the exact value of the error rate for each k
    geom_text(aes(x = k, y = errRate1, label = errRate1, vjust = -1.5), Test_error_data)
+
```

```
    #draw another method's error rate, use colour to distinguish it
    geom_line(aes(x = k, y = errRate2, colour = 'red'), data = Test_error_data, linetype =
"dashed") +
    geom_point(aes(x = k, y = errRate2, colour = 'red'), data = Test_error_data) +
    #label the exact value of the error rate
    geom_text(aes(x = k, y = errRate2, label = errRate2,vjust = -1.5), Test_error_data) +
    #make the x-axis to be divided to 15
    scale_x_continuous(breaks = c(1:15)) +
    #give an appropirate legend for the graph
    scale_color_discrete(name = "Distance Methods", labels =
c("Euclidean","Manhattan")) +
    #add title
    ggtitle("Compare Error Rates For Test set") +
    theme(plot.title = element_text(hjust = 0.5)) +
    labs(x = "K", y = "Rate")

print(P2)

#Compare the Test set error rate graphs with 10-fold CV error rate graph
#the following ggplot is quite similar to the one above so we do not the comment the
code again
P3 <- ggplot() + geom_line(aes(x = k, y = rate1,colour = 'green'), data =
display_error_rate) +
    geom_point(aes(x = k, y = rate1, colour = 'green'), data = display_error_rate) +
    geom_text(aes(x = k, y = rate1, label = rate1, vjust = 1, hjust = 0.3), color =
'red',data = display_error_rate) +
    geom_line(aes(x = k, y = rate2, colour = 'red'), data = display_error_rate) +
    geom_point(aes(x = k, y = rate2, colour = 'red'), data = display_error_rate) +
    geom_text(aes(x = k, y = rate2, label = rate2,vjust = 2), color = 'red', data =
display_error_rate) +
    geom_line(aes(x = k, y = errRate1,colour = 'blue'), data = Test_error_data, linetype
= "dashed") +
    geom_point(aes(x = k, y = errRate1, colour = 'blue'), data = Test_error_data) +
    geom_text(aes(x = k, y = errRate1, label = errRate1, vjust = 1), Test_error_data) +
    geom_line(aes(x = k, y = errRate2, colour = 'purple'), data = Test_error_data,
linetype = "dashed") +
    geom_point(aes(x = k, y = errRate2, colour = 'purple'), data = Test_error_data) +
    geom_text(aes(x = k, y = errRate2, label = errRate2,vjust = 2), Test_error_data) +
    scale_x_continuous(breaks = c(1:15)) +
```

```
    scale_color_discrete(name = "Distance Methods", labels = c("Euclidean(test
set)","Euclidean","Manhattan(test set)","Manhattan")) +
    ggtitle("Compare Test Set Error Rates And 10-fold CV Error Rates") +
    theme(plot.title = element_text(hjust = 0.5)) +
    labs(x = "K", y = "Rate")
print(P3)
```