Part 1

```
digits_train <- read.csv("~/ProgramZ/stat365/HW3/digits_train.csv")
digits_valid <- read.csv("~/ProgramZ/stat365/HW3/digits_valid.csv")
digits_test <- read.csv("~/ProgramZ/stat365/HW3/digits_test.csv")
```

I plotted all the images for a single digit and compared them manually using these two functions:

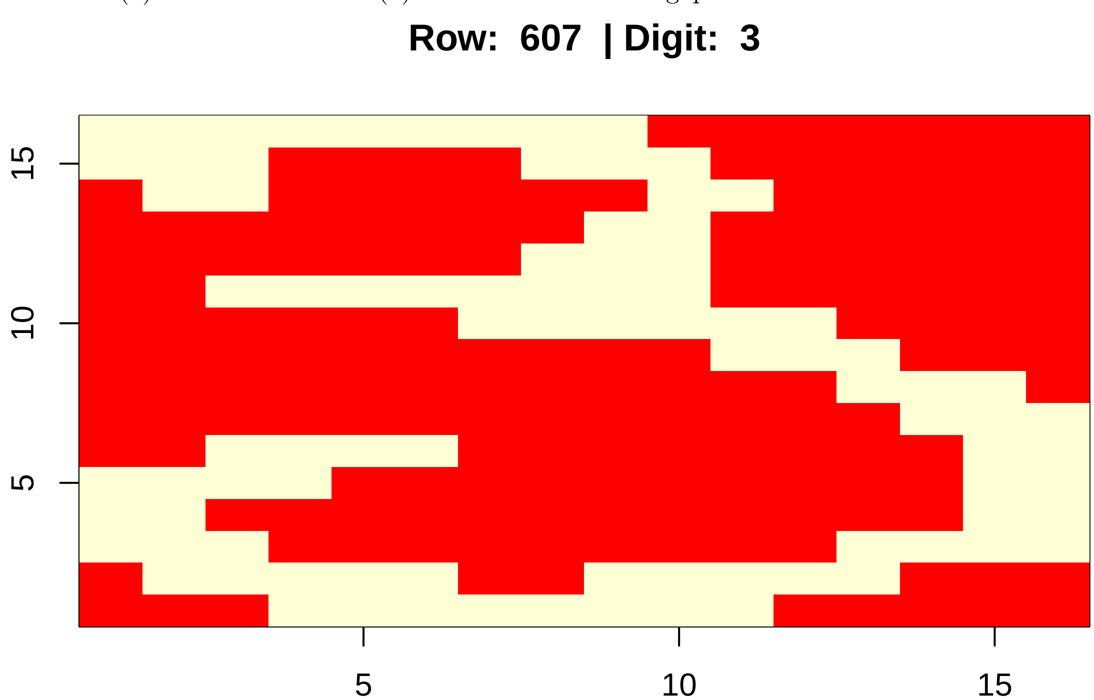
```
#Given function
plotDigit <- function(k, dat) {
  p <- matrix(as.numeric(dat[k,1:256]),16,16)
  image(x=1:16, y=1:16, p[,16:1], xlab="", ylab="",
    main=paste("Row: ", k, " | Digit: ", dat[k,257]))
}

#plot all the images for a single digit
plotAllDigit <- function (k, dat) {
  for (i in 1:nrow(dat)) {
    if (dat[i,257] == k) {plotDigit(i,dat)}
  }
}
```

Findings:

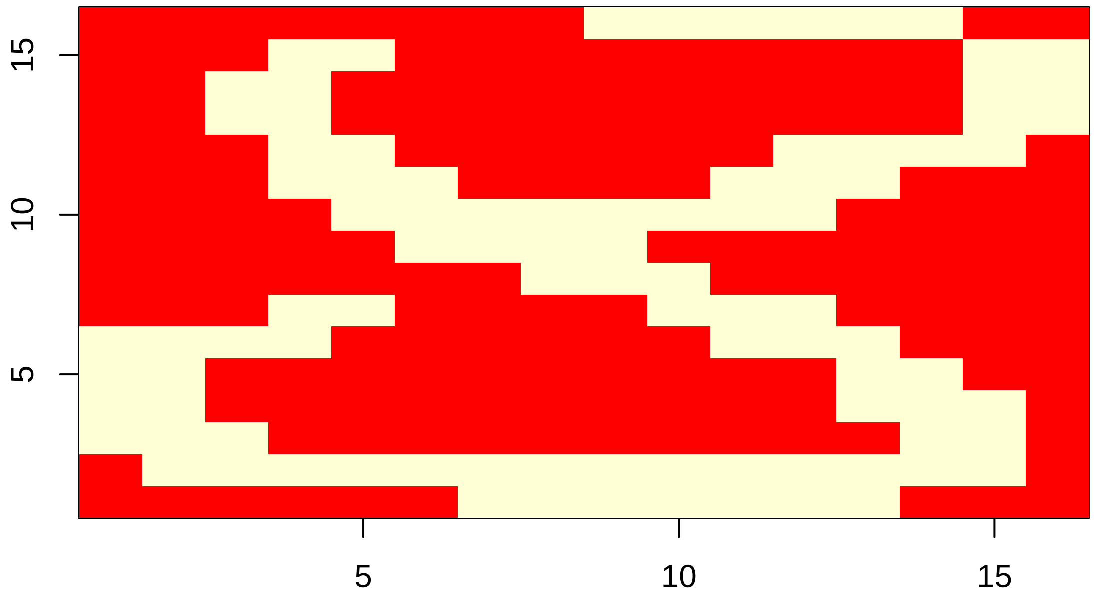
3 and 8

It is easy to confuse 3 with 8 especially if 8 has gaps on its left side

Row 607 (3) looks like row 192 (8) because row 192 has gaps on the left side:



Row: 192 | Digit: 8

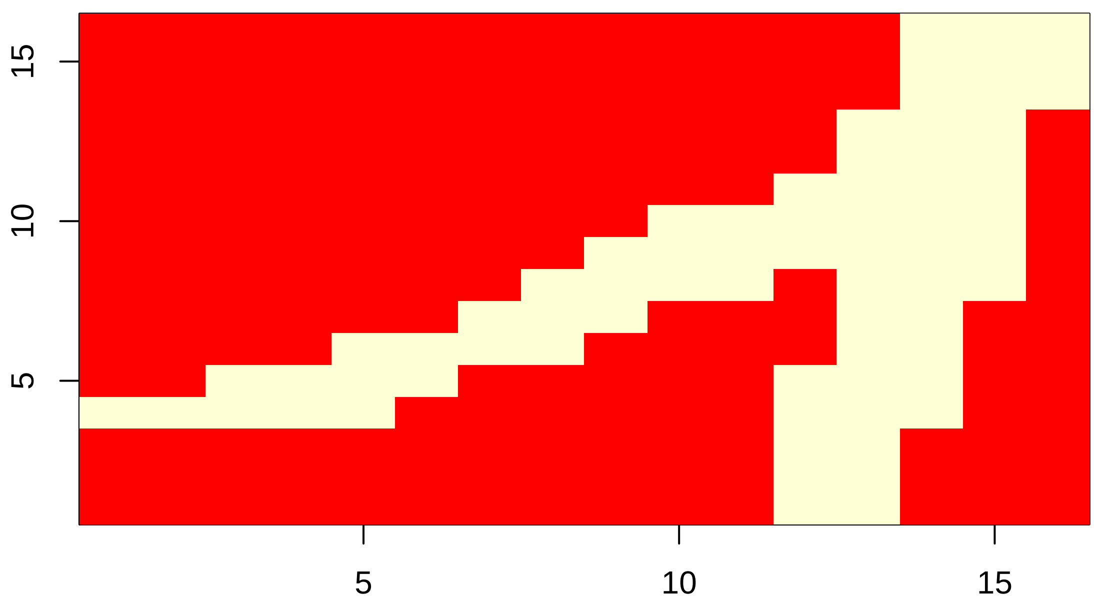


7 and 1

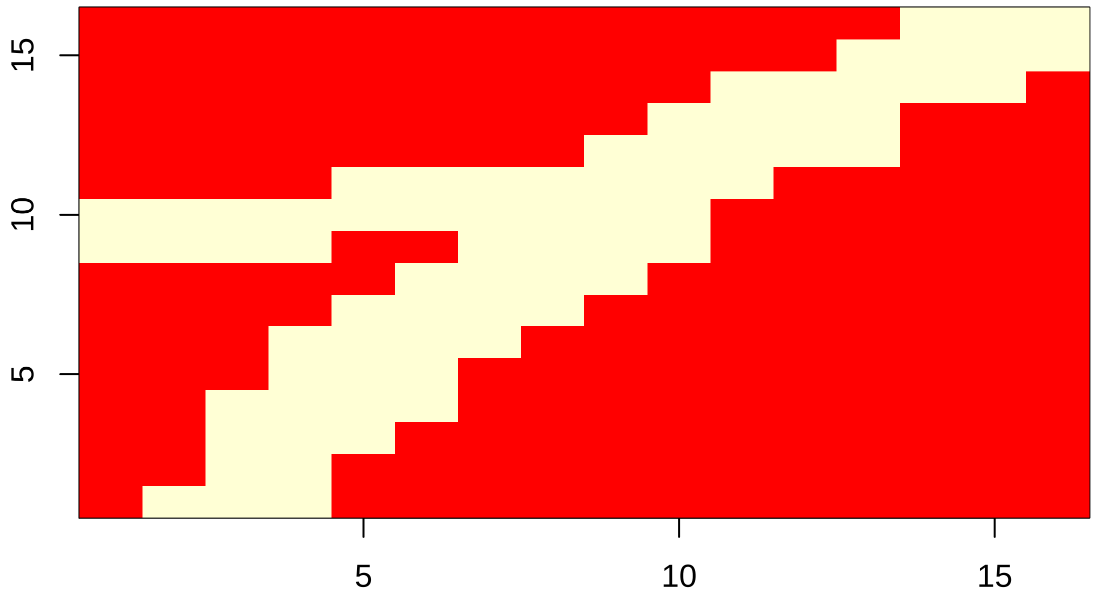
7 and 1 can look pretty similar especially if the dash through the middle is not very pronounced

Rows 359 (1), 417 (1), 560 (1) look like row 623 (7):

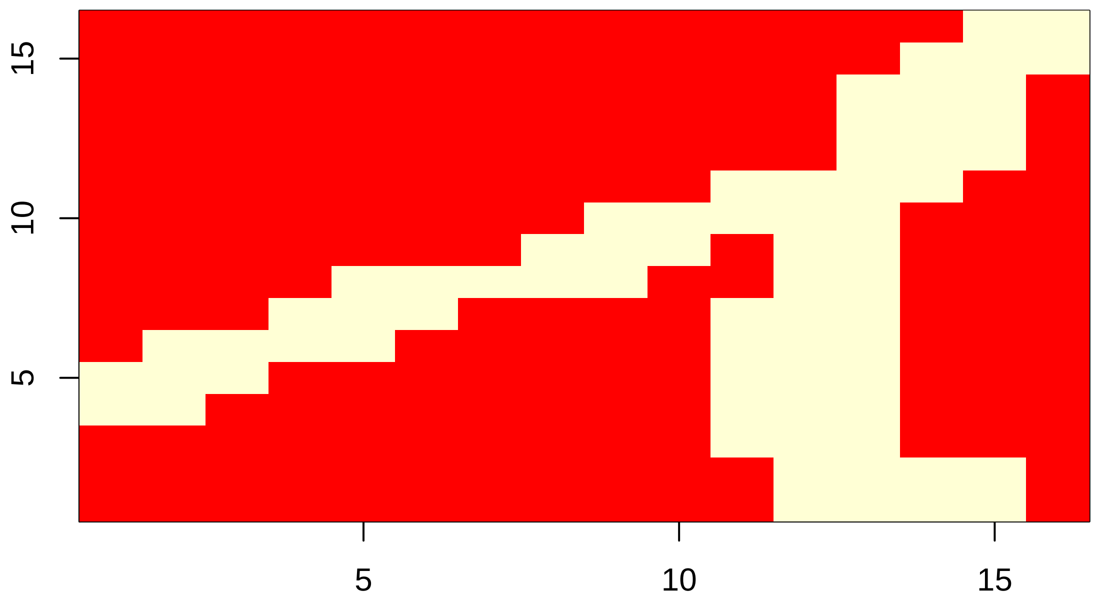
Row: 359 | Digit: 1

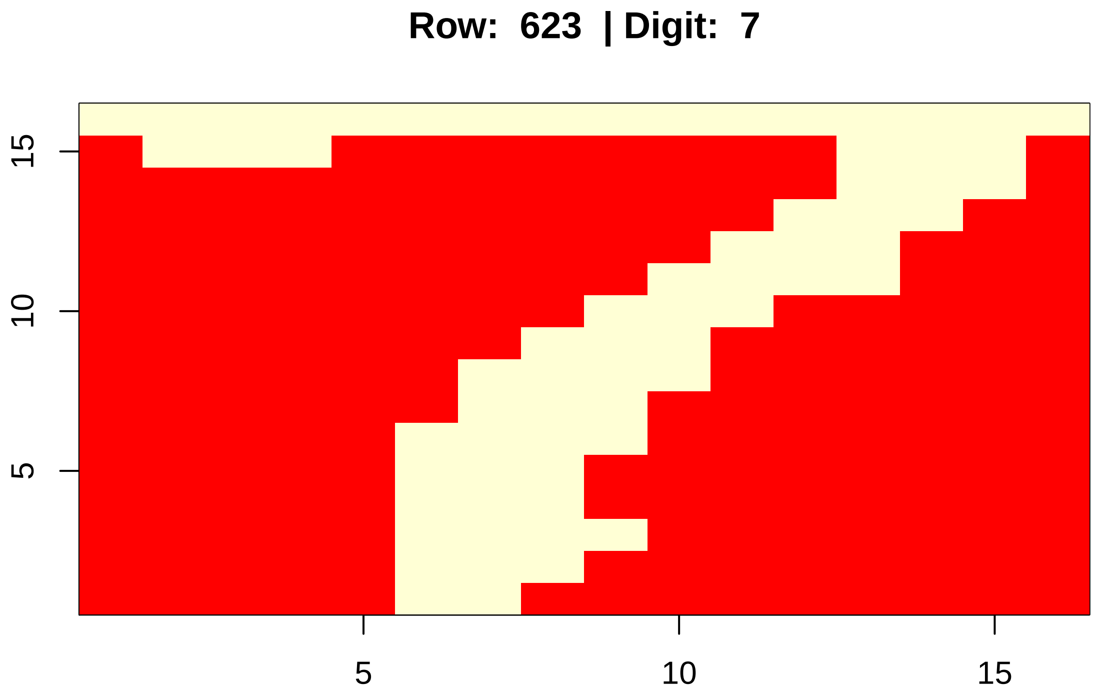


Row: 417 | Digit: 1



Row: 560 | Digit: 1

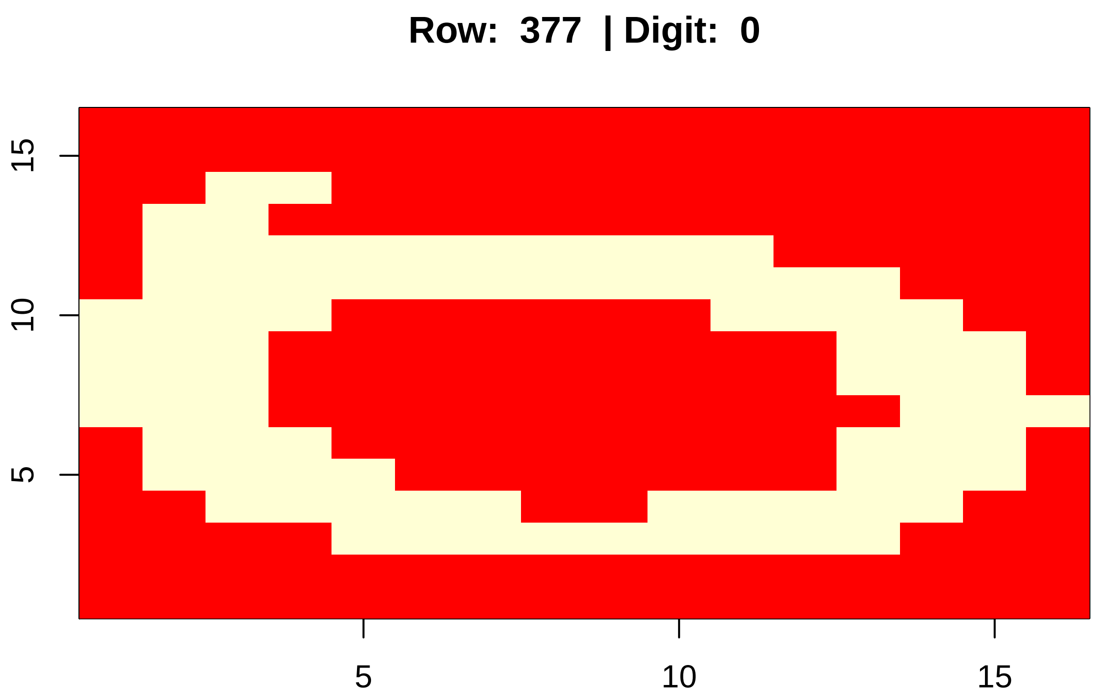




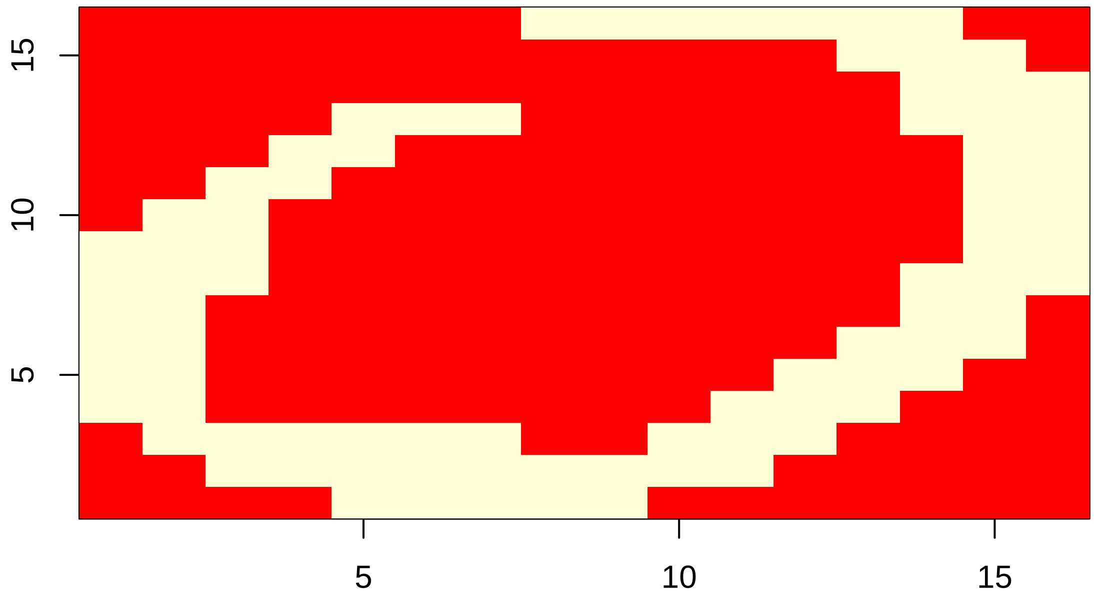
0 and 6

0 and 6 will be hard to differentiate if there are gaps at the top

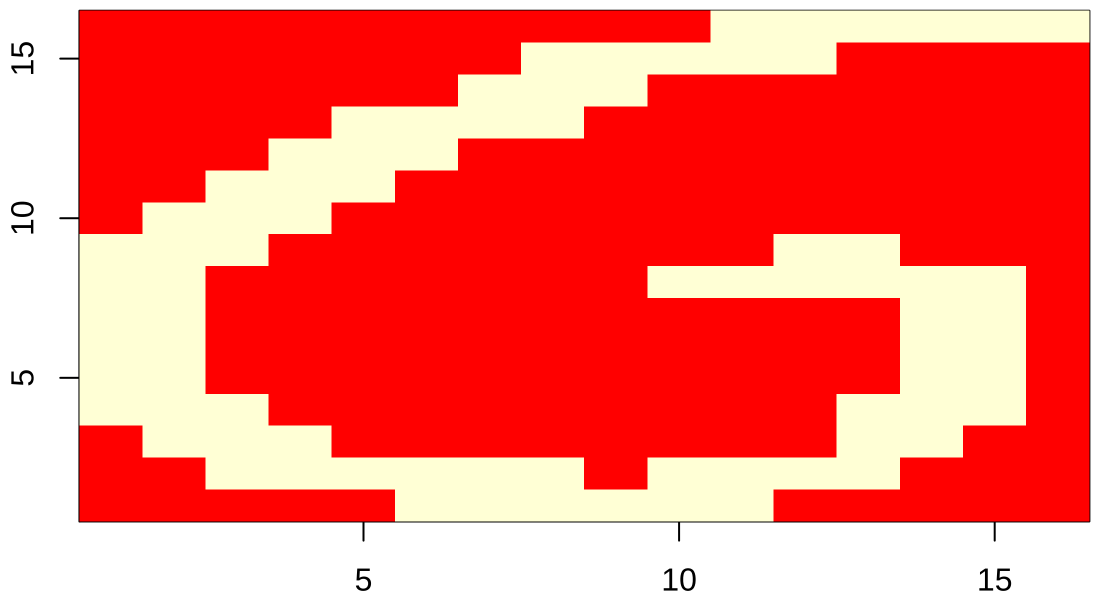
Rows 567 (0), 377 (0) and row 344 (6)



Row: 567 | Digit: 0



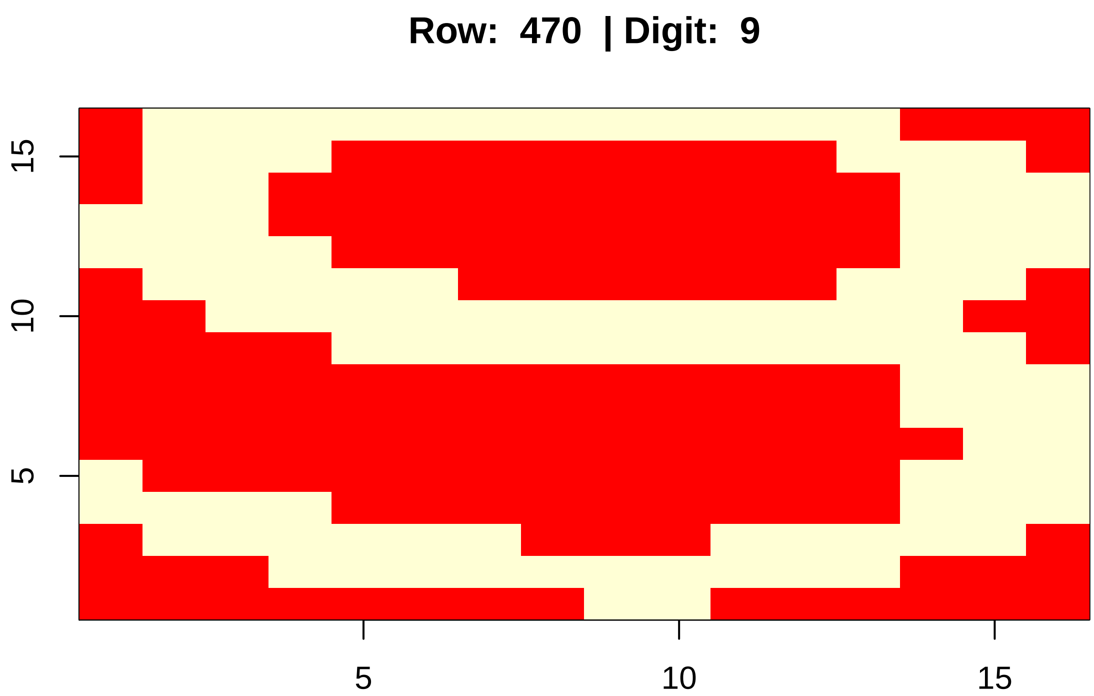
Row: 344 | Digit: 6

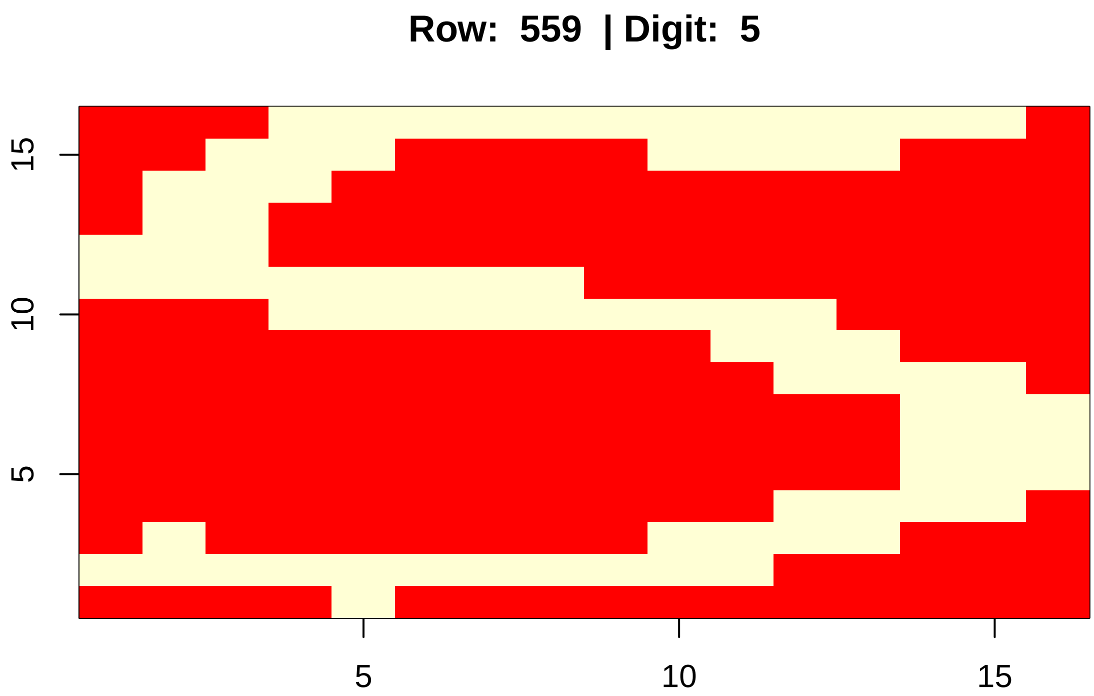


9 and 5

9 and 5 can look pretty similar as well.

Rows 470 (9) and row 559 (5)





Part 2

(1) Summarizing approach of two options

KNN model

First, we have to find optimal k with mer:

```r
require(FNN)
require(MASS)
```

```r
#misclassfication error rate
mer <- function(pred,actual) {
  matchv <- mapply(function(x,y) {ifelse ((x==y),0,1)}, pred, actual, SIMPLIFY = TRUE)
  return(sum(unlist(matchv))/length(pred))
}

#extract response variable
resp_train <- digits_train[,257]
resp_val <- digits_valid[,257]
knn_train <- digits_train[,-257]
knn_val <- digits_valid[,-257]
minK <- Inf
minMER <- Inf
for (i in 1:20) {
  pred <- as.vector(knn(knn_train, test=knn_val, cl=resp_train, k=i))
  merval <- mer(pred,resp_val)
  if (merval<minMER) {minMER = merval; minK = i; minPred<-pred}
}
print(minK)
```

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

```r
print(minMER)
```

```
## [1] 0.1132075
```

```r
head(minPred)
```

```
## [1] "7" "1" "6" "2" "0" "6"
```

LDA model

```r
ldaModel <- lda(digit~.,digits_train)
ldaPred <- predict(ldaModel,newdata=digits_valid)$class
ldaMER <- mer(ldaPred, resp_val)
print(ldaMER)
```

```
## [1] 0.1981132
```

Best model

For the KNN model we choose k = 1 because it has the lowest MER amongst all the different k values ranging from 1 to 20. The KNN model when k = 1 has a misclassification error rate of 0.113, while the LDA model has a misclassification error rate of 0.198. We choose the KNN model (k=1) because it gives us a lower MER value.

(2) Confusion matrix

```r
#minPred contains our KNN predictions
table(predicted=as.numeric(minPred), real=resp_val)
```

```
##          real
## predicted  0  1  2  3  4  5  6  7  8  9
##         0 40  0  0  0  0  0  2  0  0  1
##         1  0 24  1  0  6  0  0  4  1  0
##         2  0  0 29  0  0  0  0  0  0  2
##         3  0  0  0 26  0  1  0  0  0  3
##         4  0  1  1  0 27  0  0  0  0  0
##         5  0  0  0  0  0 36  1  0  1  2
##         6  0  0  0  0  0  1 33  0  3  0
##         7  0  0  0  1  0  0  0 26  0  0
##         8  0  0  1  0  0  0  0  0 20  0
##         9  0  1  0  0  0  0  0  0  2 21
```

- There are 4 7s that are mistakenly classified as 1 (as predicted).
- There are 2 9s that are mistakenly classified as 5 (as predicted).
- There are 6 4s that are mistakenly classified as 1 (did not predict this).
- There are 3 8s that are mistakenly classified as 6 (did not predict this).
- There are 3 9s that are mistakenly classified as 3 (did not predict this).

Overall, it seems that numbers 1, 9 and 8 are very problematic numbers to classify– it has the most misclassifications.

(3) Why is using multinomial logistic regression not advised?

Logistic regression attempts to model a linear relationship between the log(odds) and the predictors. Because digits is a multiclass response variable, we would have to do multinomial logistic regression. This would yield 9 models each of 256 predictors. This is a huge number and is computationally expensive to perform.

(4) Get CSV file

```r
#merge train and val for LDA
lda_train_val <-rbind(digits_train,digits_valid)
lda_model <- lda(digit~.,lda_train_val)
lda_pred <- predict(lda_model,newdata=digits_test)$class

#merge train and val for knn
knn_train_val <- rbind(knn_train,knn_val)
knn_resp <- c(resp_train,resp_val)
knn_pred <- as.vector(knn(knn_train_val, test=digits_test, cl=knn_resp, k=1))

HW3_tk553 <- data.frame(knn_pred,lda_pred)
colnames(HW3_tk553) <- c("knn_pred2","lda_pred2")

#Make CSV
write.csv(HW3_tk553,file = "HW3_tk553.csv",row.names=FALSE)
```

Part 3

For this part we will go back to using our variable ldaModel. ldaModel is the model object when we ran lda on only the training data set.

```
#predict first time
LDs <- predict(ldaModel)$x
digits_train2 <- data.frame(digit=digits_train$digit, LD1 = LDs[,1], LD2 = LDs[,2], LD3 = LDs[,3], LD4 =

#predict second time
valid.LDs <- predict(ldaModel, newdata=digits_valid)$x
digits_valid2 <- data.frame(digit=digits_valid$digit, LD1 = valid.LDs[,1], LD2 = valid.LDs[,2], LD3 = va

#multinomial logistic regression on 1,1-2,1-3, etc
require (nnet)
```

```
## Loading required package: nnet
```

```
m <- list()
m[[1]] <- multinom(digit~LD1,data=digits_train2,trace=FALSE)
m[[2]] <- multinom(digit~LD1+LD2,data=digits_train2,trace=FALSE)
m[[3]] <- multinom(digit~LD1+LD2+LD3,data=digits_train2,trace=FALSE)
m[[4]] <- multinom(digit~LD1+LD2+LD3+LD4,data=digits_train2,trace=FALSE)
m[[5]] <- multinom(digit~LD1+LD2+LD3+LD4+LD5,data=digits_train2,trace=FALSE)
m[[6]] <- multinom(digit~LD1+LD2+LD3+LD4+LD5+LD6,data=digits_train2,trace=FALSE)
m[[7]] <- multinom(digit~LD1+LD2+LD3+LD4+LD5+LD6+LD7,data=digits_train2,trace=FALSE)
m[[8]] <- multinom(digit~LD1+LD2+LD3+LD4+LD5+LD6+LD7+LD8,data=digits_train2,trace=FALSE)
m[[9]] <- multinom(digit~.,data=digits_train2,trace=FALSE)

#predict
df <- data.frame()
for (i in 1:length(m)) {
  df[i,1] = i
  df[i,2] = mean(predict(m[[i]],newdata=digits_valid2) != digits_valid2$digit)
}
colnames(df) <- c("Model","MER")
df <- df[order(df$MER),]
```

Let's look at the table:

```
##   Model       MER
## 9     9 0.2264151
## 8     8 0.2452830
## 4     4 0.2893082
## 7     7 0.2893082
## 5     5 0.2924528
## 6     6 0.3238994
```

It seems that model 9 (LD1-9) still yields the smallest MER. Including all 9 LDs is still the best way to go.