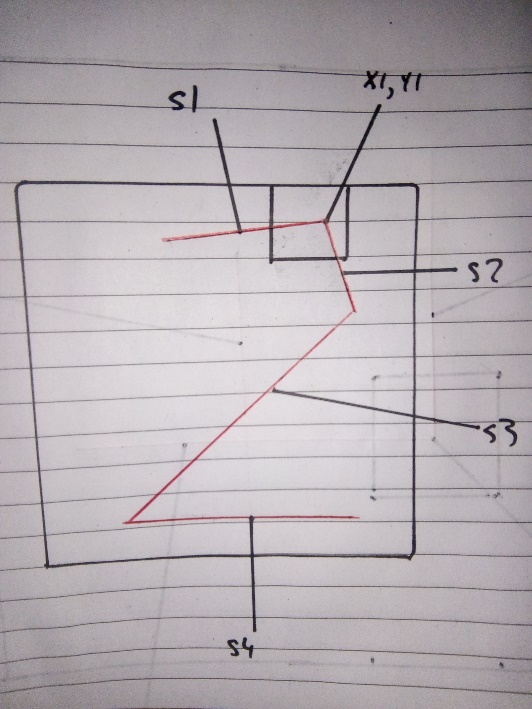
Mark Priestley

Student ID: 19375096

# Question 1

I created my model 2s with 4 strokes. The code is on the next page.

This is a diagram showing the different strokes of my model.



The position of x1,y1 is randomly selected within the box.

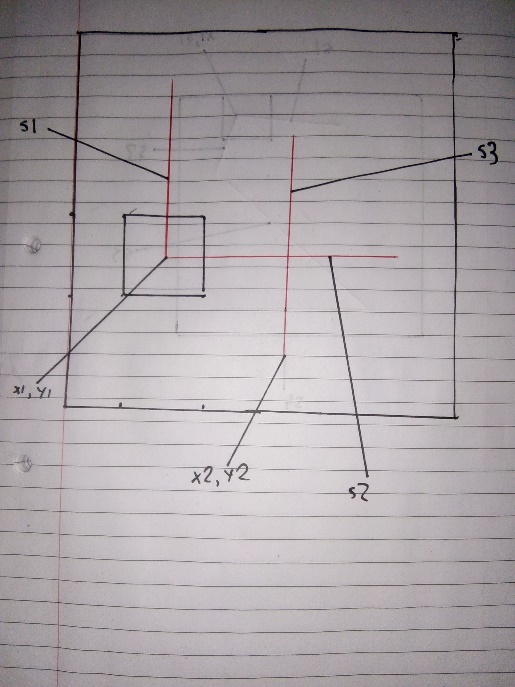
S1 and s2 are supposed to resemble the curve of the 2.

All the points are chosen in relation to x1,y1, ie x2,y2 is the endpoint of s2, and x3,y3 is the endpoint of s3.

One problem I had testing the model is that s3 would sometimes be too long, and go off the image. So after creating s3, I found the strokes endpoint. If that endpoint is off the screen, I tried to change the values of len3 and theta3.





I created my 4s with 3 strokes.

X1 and y1 are selected within a box.

Y2 needed to be less than y1, and the endpoint of s3 should be about halfway on the line s1.



# Question 2

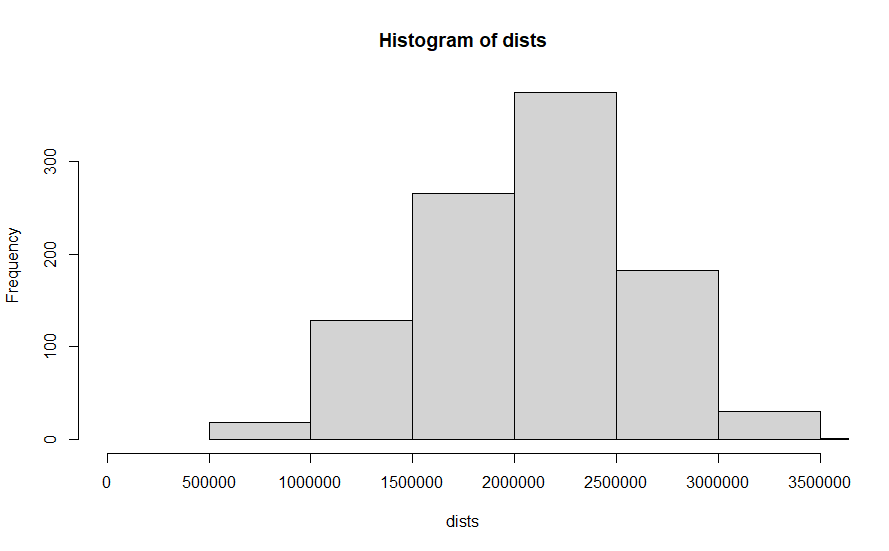
## Fitting random 2 to real 2s

I firstly ran the code from Q1 in a for loop 10000 times, and added it to a matrix b2. I also took note of the parameters I used, and stored those in the matrix params.



I then loaded the real dataset, and fit the random 2s to the real 2s.



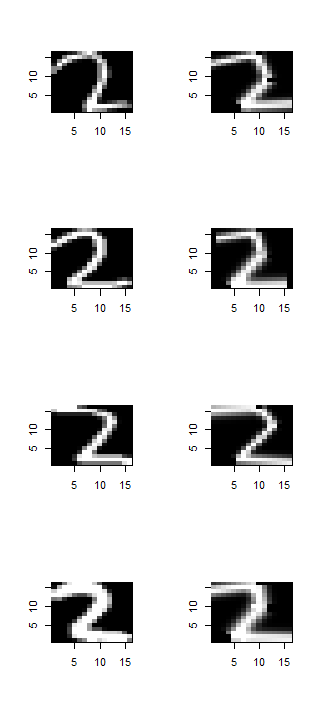
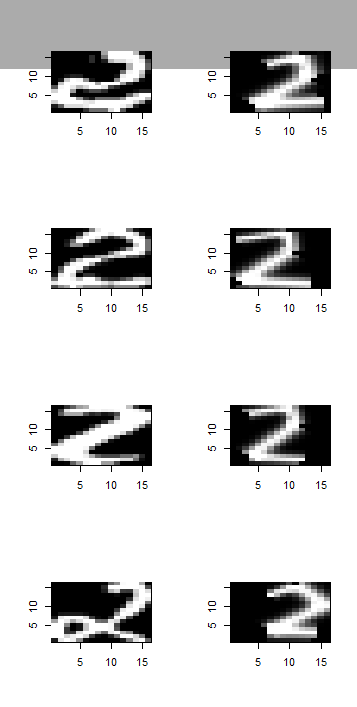


Looking at the best matches (< 800,000)



Looking at the worst matches(> 3,200,000)



 **Best Matches Worst Matches**

The best matches width of strokes are accurate, and the angle and length of the long diagonal stroke (s3 in my model) are also accurate.

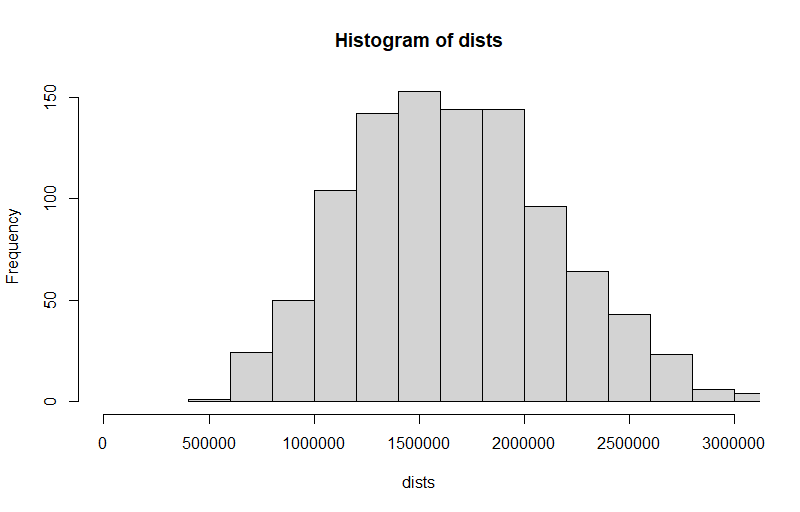
The worst matches have a combination of different problems, the position of x2 and y2 (the top of the diagonal stroke s3) is a problem in some. I believe the most common problem is some of the digits have too many “curves”, for example the real digit from the digits above has a curve transitioning from the diagonal stroke to the bottom stroke. This is quite difficult to adapt my model to, I could add in a few more strokes but I feel it is unique enough the I shouldn’t change my model for it.

## Fitting random 4 to real 4s



Checking the fits



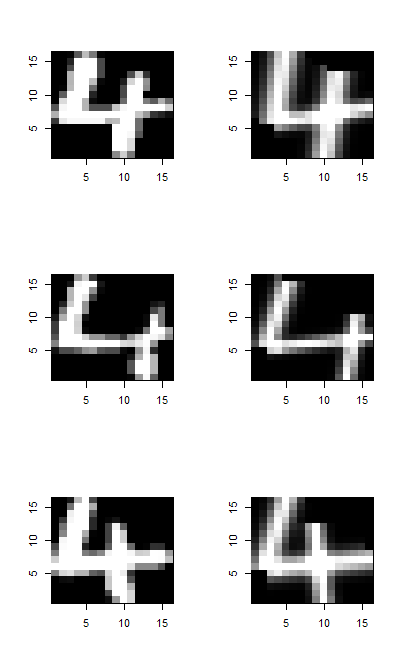
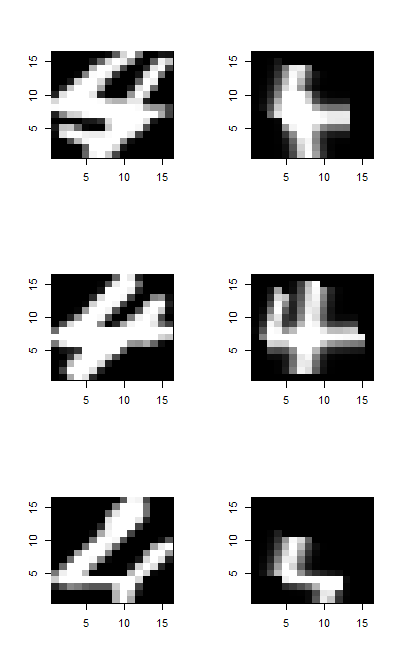


On average, the 4s are more accurate than the 2s, probably because it is mainly straight lines. However, it is still not great.

Looking at best and worst matches

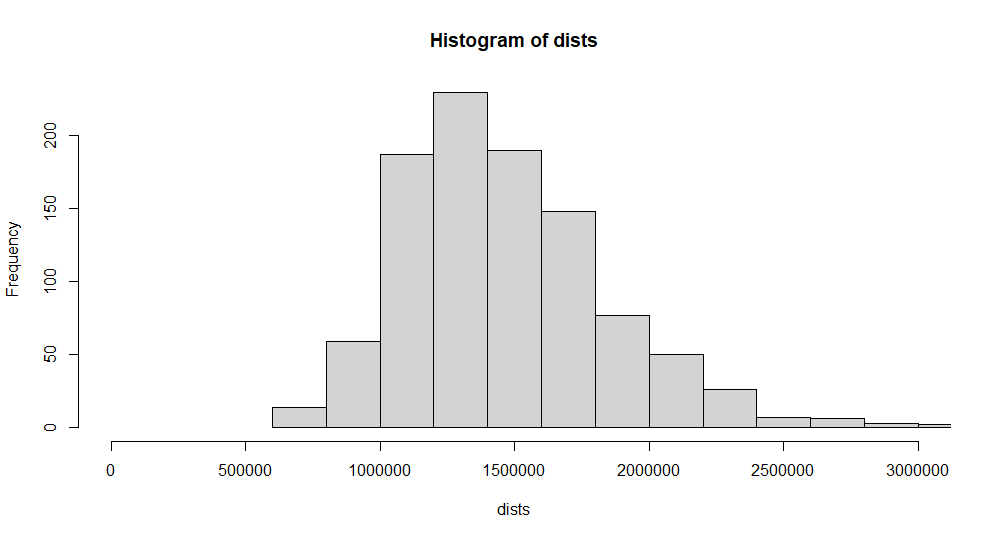


**Best Matches Worst Matches**



The best matches are quite accurate, the angle and length of all the strokes are good.

The problem with the worst digits seem to be the angle of the strokes, specifically s1 and s3. I decided to expand the range of angles available on the s1 and s2, to see if that improves the accuracy. (set the lower limit of theta 1 and theta2 to -0.3\*pi). Repeating all the steps above, I got a new histogram.



This has slightly increased the accuracy, the largest bin is around 1,300,000, instead of 1,500,000 as before. Looking again at the worst matches



The problem with these matches might be the unusually high value for y1. I don’t think these problems could be fixed with an extra stroke or a change in parameters, as I believe the issue is the unlikely combination of stroke angle and y position. Changing these parameters may bring the overall accuracy down, even if it improves those outliers accuracy.

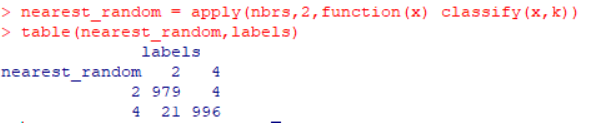
I saved the parameter matrices from the random 2s and new random 4s, and the b2 and b4 matrices, for the next few questions.

# Question 3

I loaded in the b2 and b4 matrices, and column banded them to make the matrix b. I did the same for the real digits. I also created the classify function.



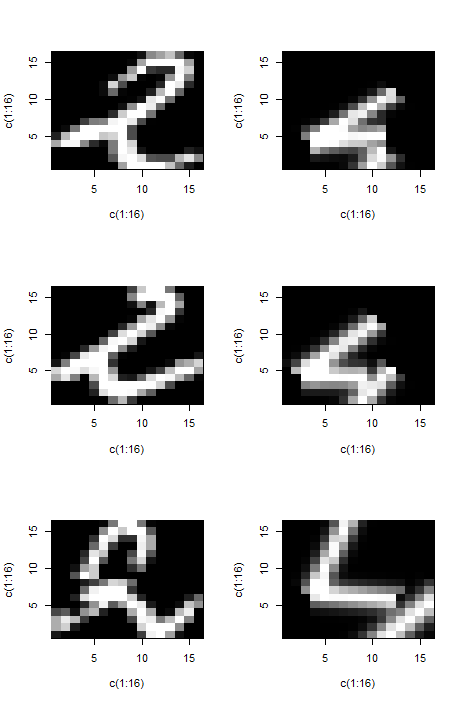
Using this, I found the confusion matrix



The real 4s are quite accurately matched, but the real 2s are matched to random 4s 21 times.

Looking at wrongly matched real 2s.

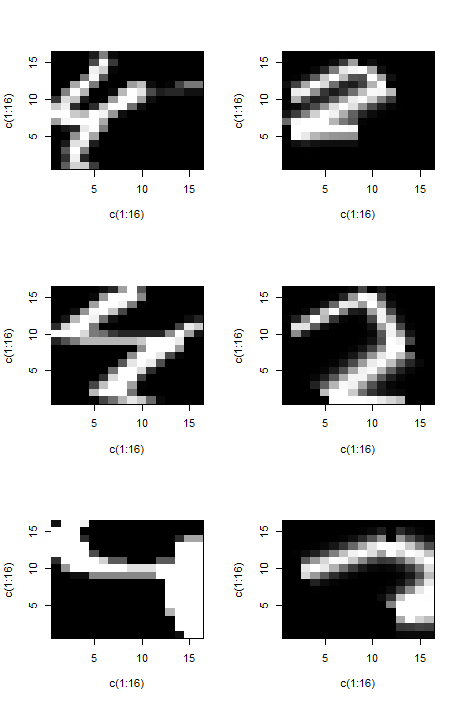




The upper stroke of the 4 matches the long diagonal stroke of the 2, the upper cross stroke of the 4 (s3 in my model) is poorly positioned in the examples, which may also be causing problems.

There is a lot of “curves” on the real 2s, which may explain why my random 2s did not match

Doing the same above but with real 4s and random 2s.

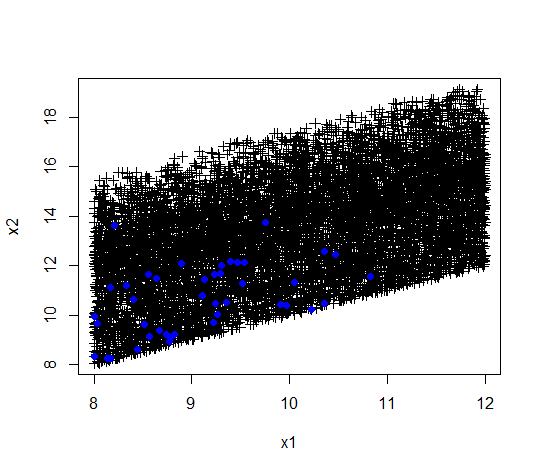


The 2 upper strokes of the real 4s are similar to s1 and s3 of the random 2s.

# Question 5

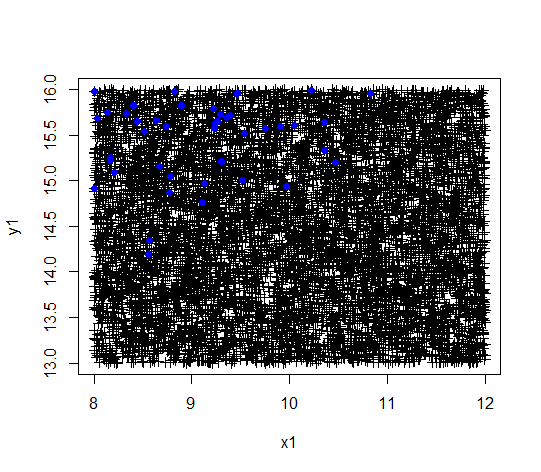
I reran the code from q2 for the 2s. I set any best fit value to anything less than 1,200,000. This gave about 60 points. The params matrix for 2s is a 14 x 10000 matrix, with the 14 parameter values being x1, x2, x3, y1, y2, y3, theta1, theta2, theta3, theta4, len1, len2, len3 and len4. The line ‘nbrs[1,best\_fits]’ will select the 60 or so best matches to the real digits. ‘params[1,nbrs4[1,best\_fits]’ will then select the first parameter (x1) for the best matches.

There are a lot of different combinations of variables, so I selected some which look interesting.



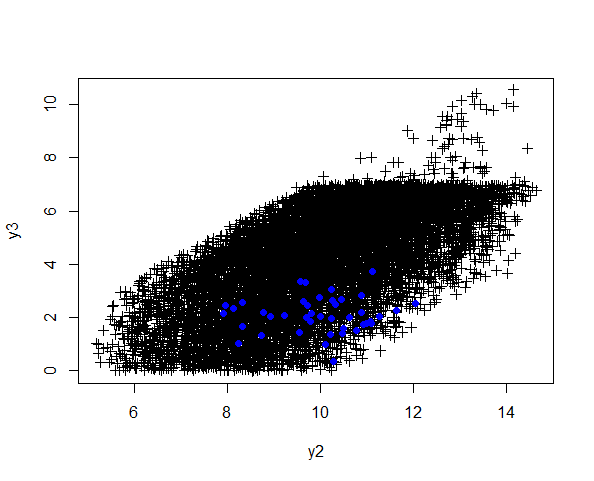
The black plus signs are the entire 10,000 dataset for x1 and x2, while the blue points are the best fits parameters for x1 and x2.

They seem to cluster in the bottom left of the plot, so maybe lower the limit of the lower parameter might give better fits.



The best fits build up in the top left of the image. This suggest a high y value generally gives better points (16 is the highest it goes).

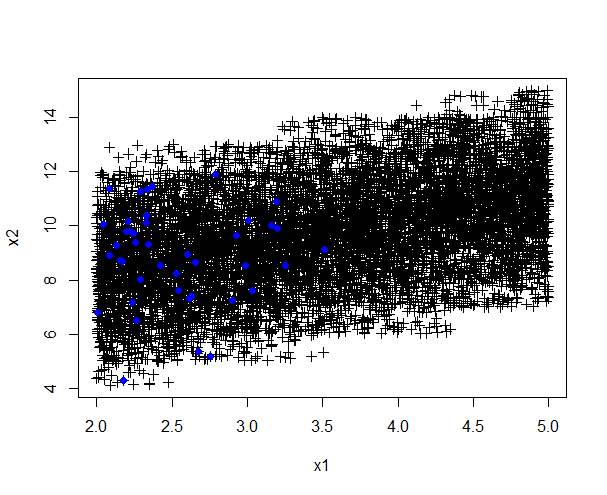




There seems to be a build up around a y2 value of 10 and a y3 value of 2.

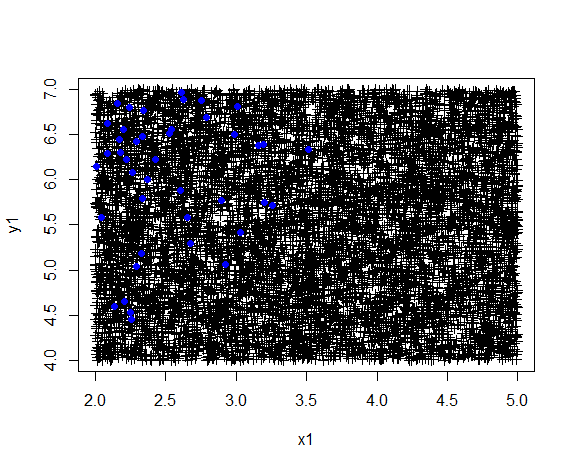
The few examples above show there is a definite correlation between the different values, they are not independent.

I did the same for the 4s. The parameters for the 4 are x1, x2, y1, y2, theta1, theta2, theta3, len1, len2 and len3. I set the best fit limit as anything less than 960,000 (about 60 values).

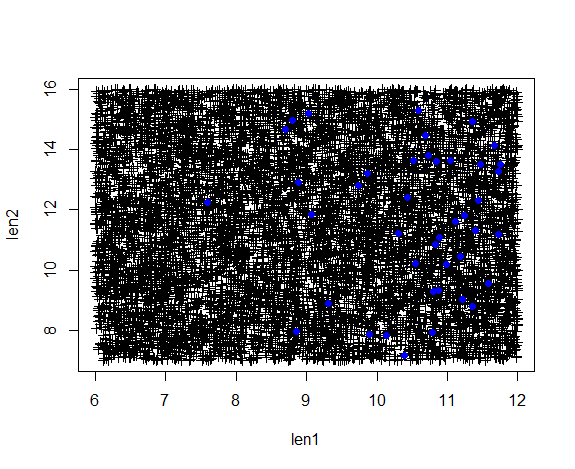


There does seem to be a build up on the left side of the plot. The x2 value is not that significant, however, the best fit parameter for x1 does build up on the lower values.





There is a build up of best points on the top left of the plot.



There is a build up of points on the right side of the plot.

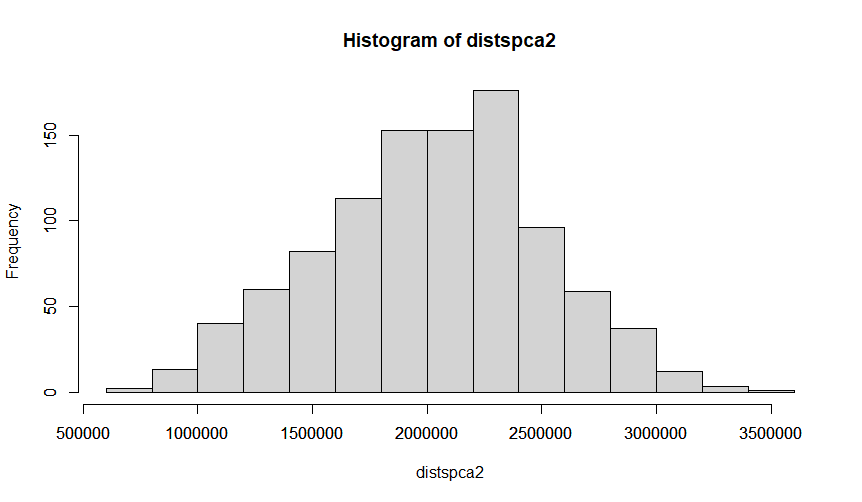
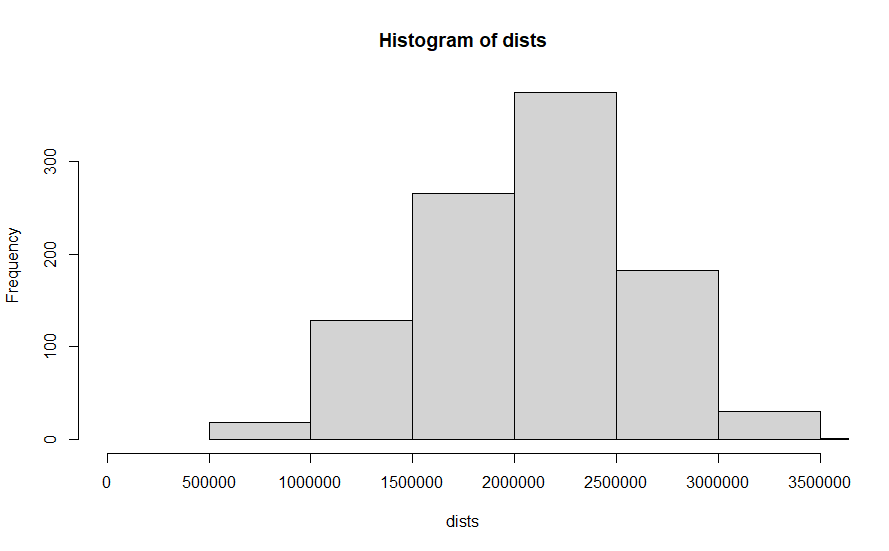
There does seem to be a correlation between the different parameters of the random 4s.

# Question 6

Using PCA to create random 2s.



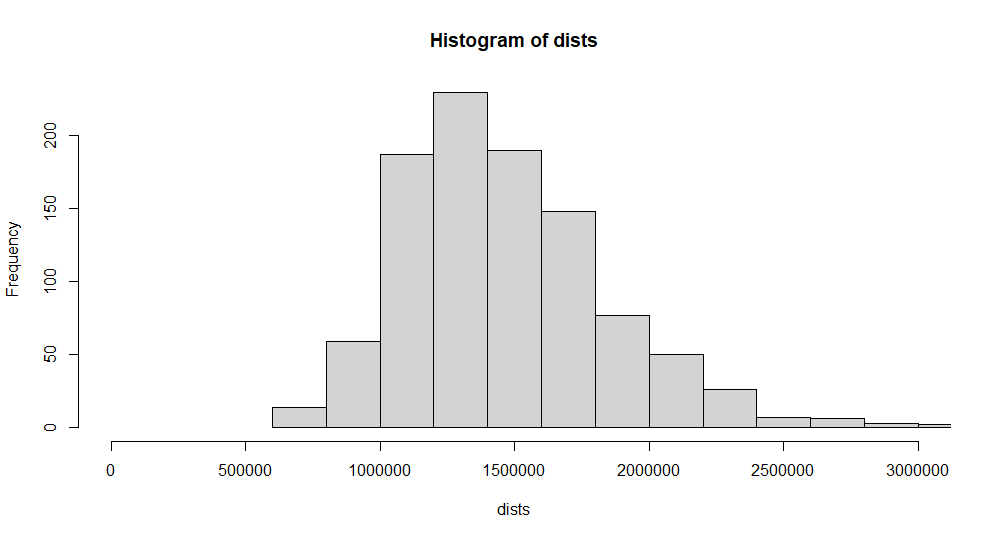
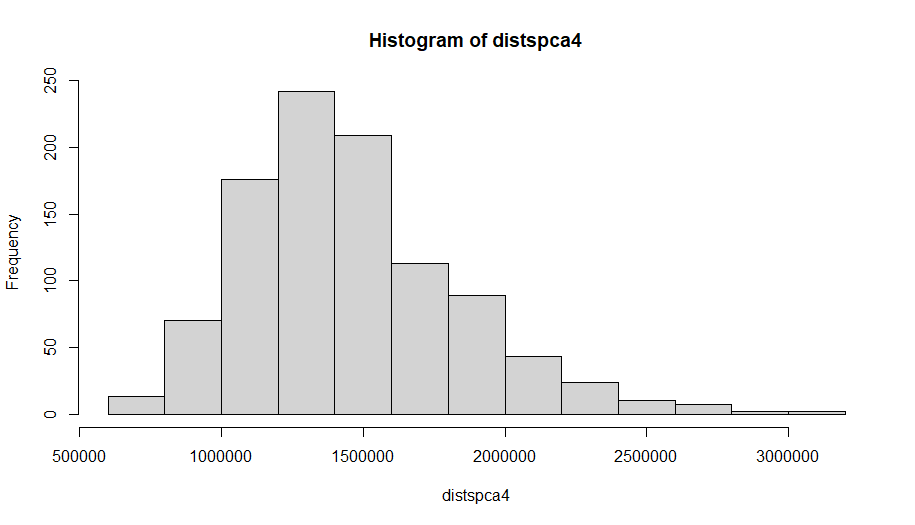
I then compared the real 4s to the nearest random 4, to get a histogram of the error.



Comparing the new distances to the histogram earlier, there does seem to be a slight improvement.

I did the same for 4s.

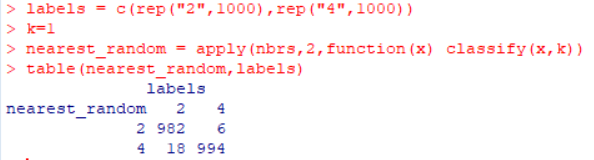




The histograms are very similar, there might be a very marginal improvement with the PCA model 4s.

I found the confusion matrix for the model 2s and 4s.





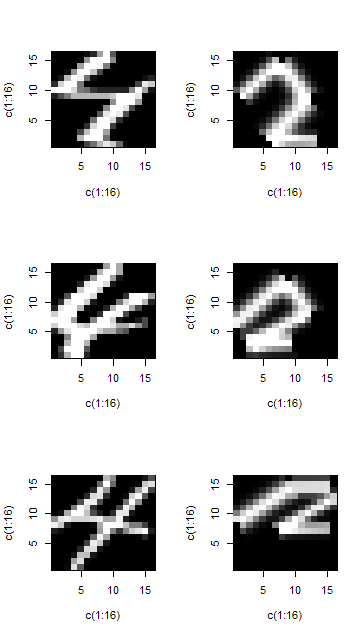
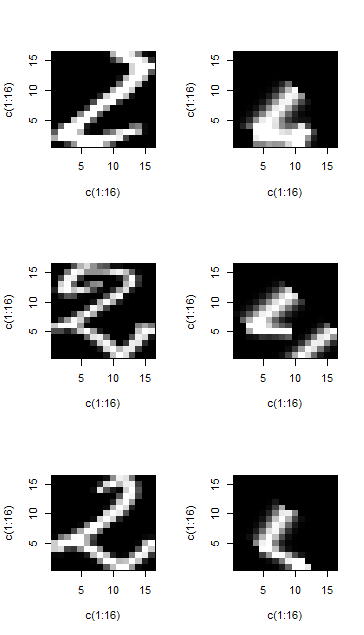
Compared to the models in Q3, which had a confusion matrix of

|  |  |  |
| --- | --- | --- |
|  | Real 2 | Real 4 |
| Random 2 | 979 | 4 |
| Random 4 | 21 | 996 |

98.75% accuracy for the old model, 98.8% for the PCA model, so no significant increase in accuracy.

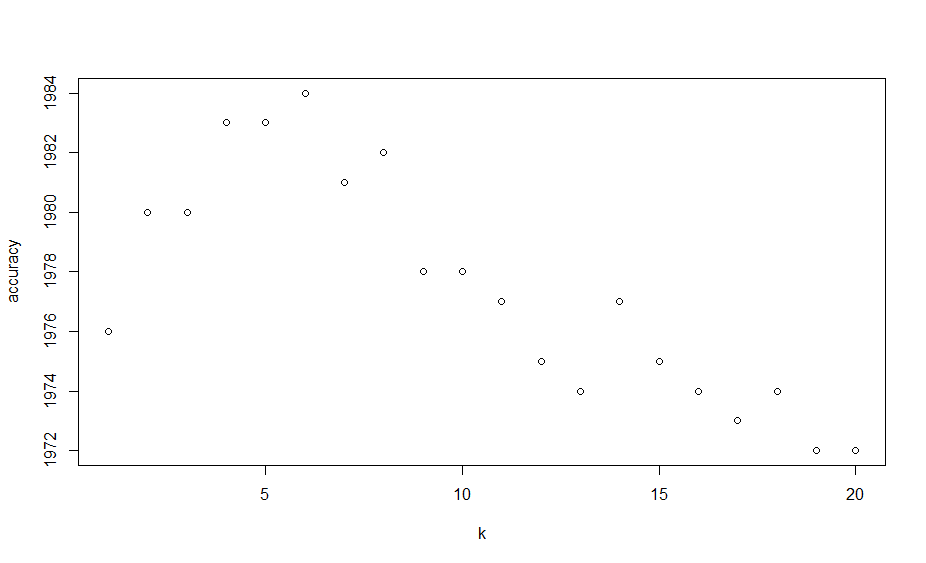
Finding the misclassified digits.





Finding the accuracy with different numbers of k



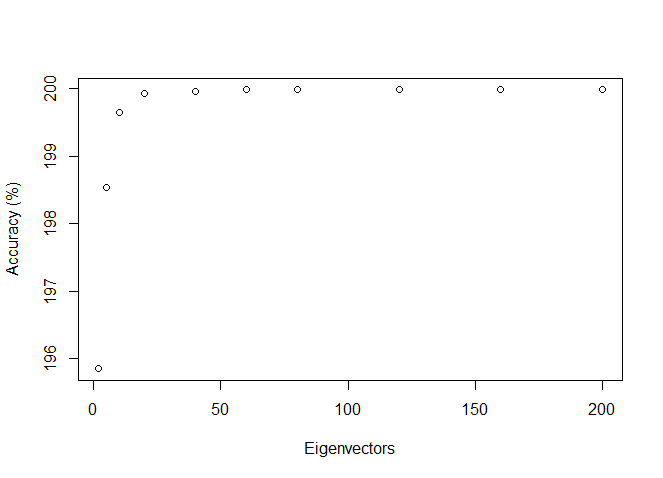


6 seems to be best value for k.

# Question 7

I used the box method to classify the random digits, to speed up execution time. I used k = 6, as it was the most accurate when classifying the real digits.





The accuracy rises very quickly and levels off with about 20 eigenvectors. The more eigenvectors you include, the longer execution time is going to be, so between 20 and 40 eigenvectors seems to be the ideal number.

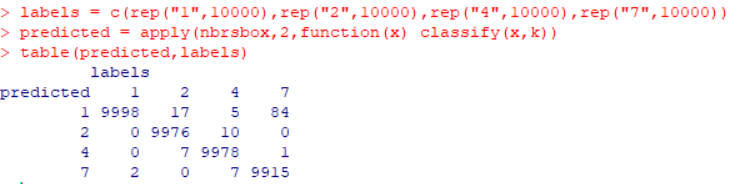
# Question 8

I loaded in the matrices for the random 1s, 2s, 4s and 7s.

I used the box method again to find the nearest neighbours, with k set to 6.

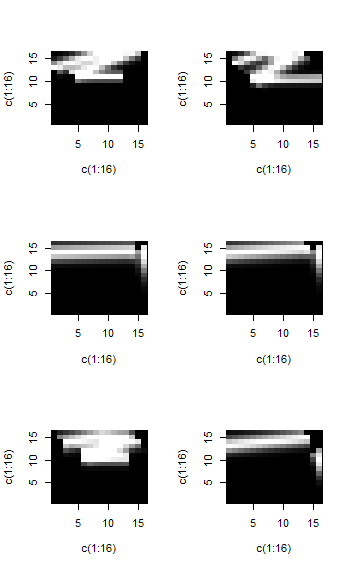
I then found the confusion matrix





I used k = 6 instead of k = 1, so I looked at one example when a 2 was misclassified as a 1.





The top left digit is the misclassified digit in this case, and the other 5 digits are its nearest neighbours. There a few 1s, whose upper stroke matches the unusually high position of the 2.

Using different values for k might improve the accuracy.