19479166 Diarmuid Brady, 19329261 Joseph Oluwasanya

CA274 Assignment 2

Introduction

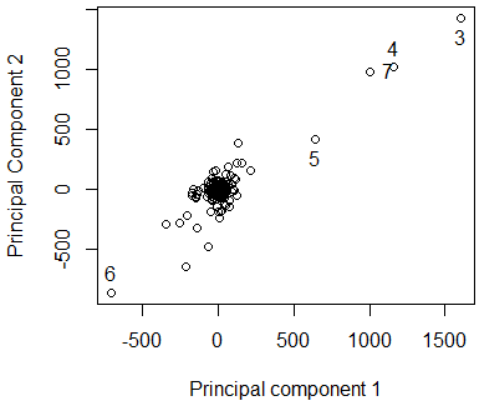
In this assignment we designed and generated random 2's and 4's with the intention of classifying a pre existing real handwritten digit set. We proceeded to optimize our models with the use of manual manipulation of parameters, analysis of parameter distribution and principal component analysis. Finally we class 4 sets of digits; 1's, 2's, 4's and 7's.

Question 1: Creating the Models

For Question 1, Diarmuid worked on the number 4's and Joseph worked on the number 2's

Generating The Number 4's (Diarmuid)

To begin it is best to identify and label each stroke of the four so discussion of each stroke will be much smoother later on. The strokes were labelled in order of how the digit four is written. The left vertical stroke will be considered stroke1, the cross stroke will be stroke2 and the right vertical stroke will be stroke3.



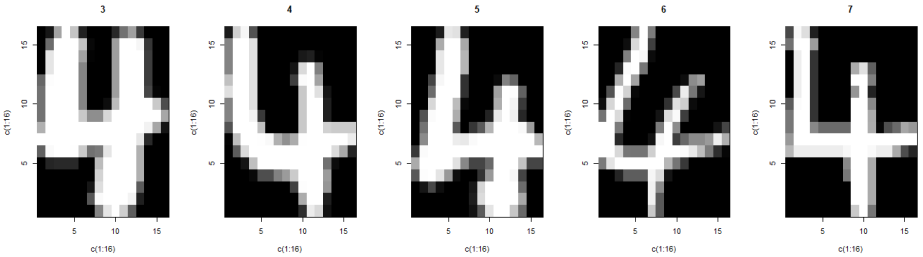
The first step that was taken in engineering the number 4's was to calculate the covariance matrix, compute the eigenvalues and eigenvectors and finally project the real 4 digits matrix onto its eigenvectors into a matrix p. P was then plotted in 3D space using the first 3 eigenvectors. Here is a plot of the first two principal components. The plots of (1, 3) and (2, 3) are not shown as they are very similar and provide no additional information.

Each point represents a digit of 4. As you can see, points have been selected from different positions in the graph (bottom left and top right corners etc.). Below some have been visualized in an image. What the 4's in the top corner have in common is that:

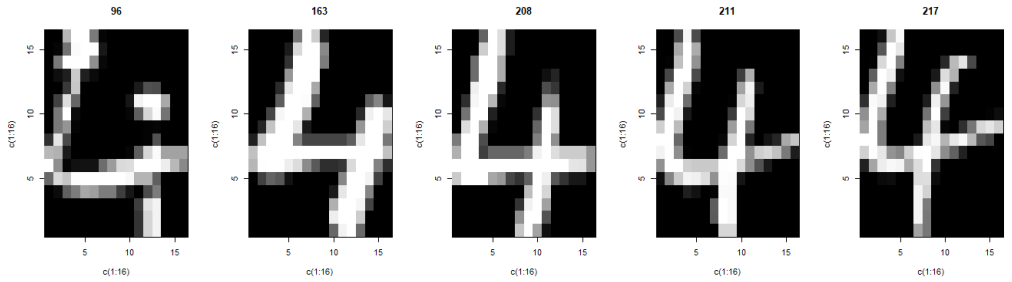
1. Their strokes are very thick
2. Stroke1 and Stroke3 and near-vertical

While digit 6 (bottom left corner) contains these attributes:

1. Thin strokes
2. Stroke1 and Stroke3 are more at an angle



By looking at the extreme values of the 4 we can get a better idea of how to parametrize the 4. Here are some of the fours in the large cluster. As you can see they all look quite similar except the first one which has gaps in stroke3.



At this point, we can begin designing each stroke. Stroke1 will move from the top-down, stroke2 will move from left to right, stroke3 will begin from the bottom up. Here is a table of the parameters that were chosen for each stroke. They are all uniformly distributed.

The images above were used to model the parameters for the generator. These are the parameters that were decided on.

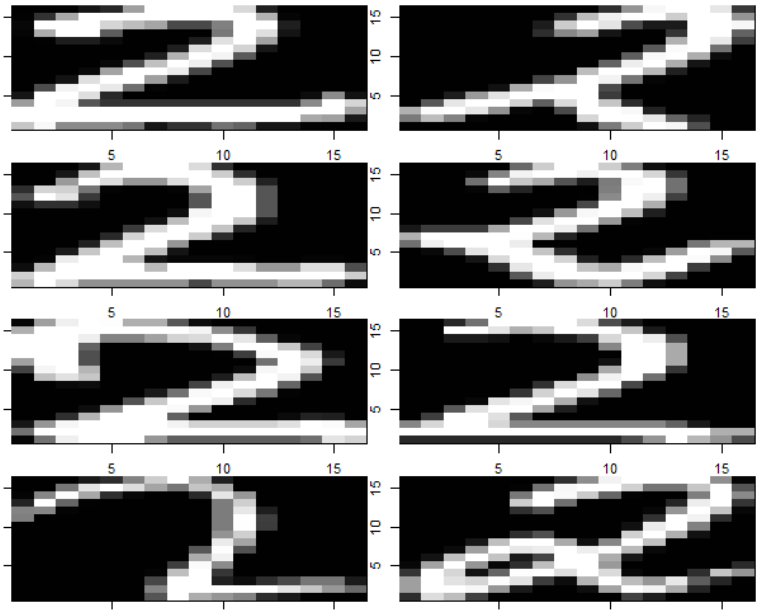
Stroke	X (Start)	Y (Start)	Length	Theta	Width
1	(1,6)	(13,16)	(10,14)	(0.8*pi,pi)	(0.5,3)
2	(xe1-1,xe1+1)	(ye1-1,ye1+1)	(14,16)	(1.5*pi,1.6*pi)	(0.5,3)
3	(10,16)	(10,16)	(10,16)	(0.85*pi,pi)	(0.5,3)

Issues began to arise particularly with stroke1. Since it had a wide range for the starting point for x1 and theta1, it often ran off the left-hand side of the page. There was also a wide range on y1 and len1, this resulted in a very low stroke. So it was necessary to put in some constraints.

Some conditions were implemented into the model using the repeat function. These conditions are based on looking at 1000 4's to look for contradictions in my hypotheses, these were the conclusions that were arrived on.

- It was ensured that the endpoint of stroke 1 begins inside the image so xe1 > 0 and ye1 > 4.5
- It appears that the bottom of stroke3 is at the very bottom so ye3 < 0.5

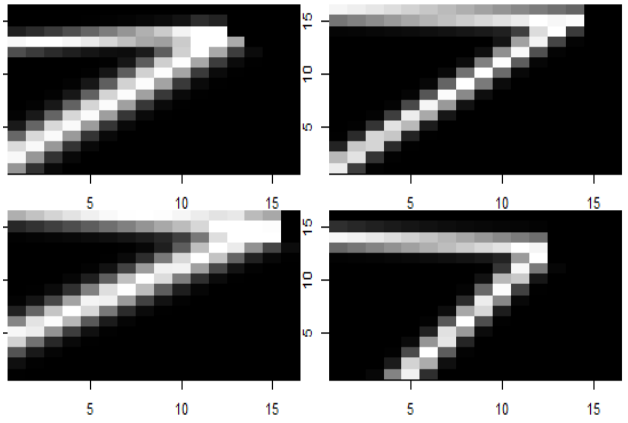
Question 1 (Joseph: creating 2's model)



We started by viewing the real 2's and observing how the different strokes connect and where. The initial model 2 we create will be like a Z shape. The first stroke (upper) starts around the top right corner and spans the left. We thought of different places for the start point of the 2 (the upper-left **or** upper-right) but concluded that using the upper right corner as we did for the 1's and 7's would be most adequate. The upper stroke is curved but we will later see if using a straight stroke positively affects the model (generalization) or negatively affects the mode (underfitting). Note also that the second stroke (middle) starts near the starting point of stroke 1 and has a diagonal angle. These are generally straight and can therefore be well fitted by our stroke function. The third stroke (lower) starts near the endpoint of the middle stroke. In some of the 2's the lower stroke has a more complex shape. The strokes for the most part seem to connect. We tried first to base the starting points of each stroke on the starting (or ending) points of the previous one and add some noise. Our first model was as follows:

All stroke lengths were fixed at 20.

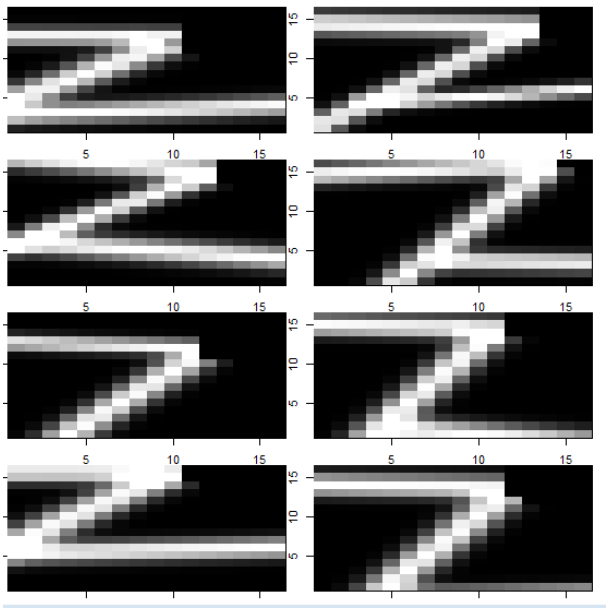
Stroke	X	Y	Length	Theta	Width
1	(12,16)	(12,16)	20	(π*0.45,π*0.55)	(0.5, 2)
2	(x1-1,x1)	(y1-1,y1)	20	(π*0.65,π*0.85)	(0.5,2)
3	(xe2, x2e+1)	(ye1, ye2+1)	20	(π*0.45,π*0.55)	(0.5,2)



My first set of working outputs had some problems:

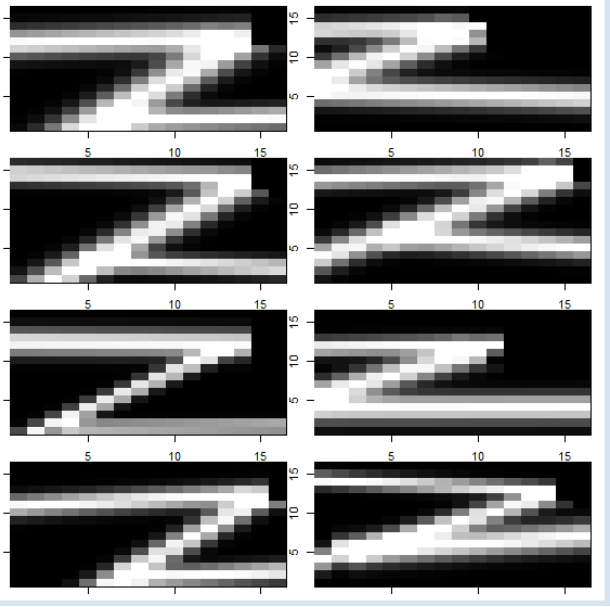
The third stroke is not visible. I first tried reducing the lengths of the strokes to 15. Maybe the third stroke is outside the range? I found that the third stroke goes in the wrong direction.

Changing the theta parameters from 0.45*π to 1.45*π and 0.55*π to 1.55*π, we obtain the first reasonable models for 2:



It seems now that fixing the middle stroke length at 20 sometimes brings the lower stroke out of the frame. I tried changing this length for the middle stroke but it affected the model negatively. (the resulting 2s were too small)

Fixing the lengths will keep the model simpler (for now). I increased the width thickness range to 0.5 to 3 for all strokes to account for thicker strokes. Finally, I fixed the (upper) y1 range to [12,15] instead of [12,16] to try and keep the 2 inframe . Here are some outputs of this last model:



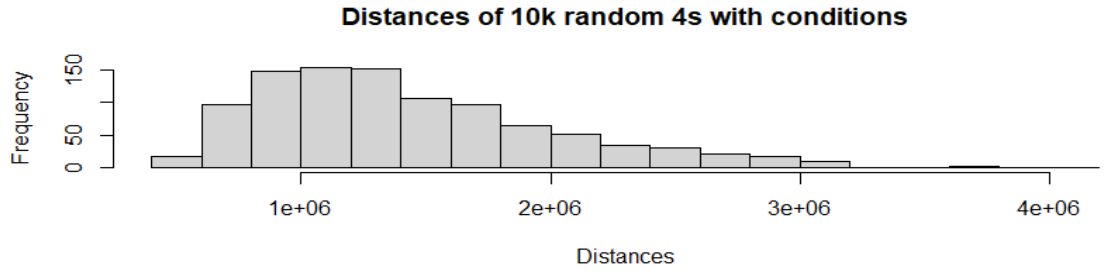
This is the best result so far. We will investigate how good the fits are in the next section.

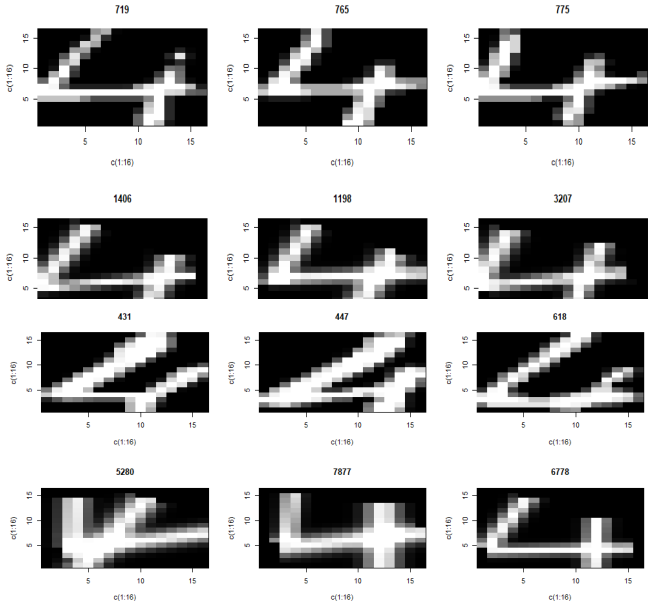
Question 2: Fitting The Models

Fitting The 4's (Diarmuid)

At first, the fours were fitted using eigenvectors and the box method with the parameters and conditions outlined above. Here are the distances between the real fours and their nearest neighbour. There appeared to be quite a wide range of values up to 4 million.

The average distance was 1439621
It is helpful to see what the model was doing right so the digits with the smallest distances were chosen (dists < 515000) and are displayed below.





As you can see these digits fit nicely into the conditions laid out. These are what we can consider 'typical' 4's.

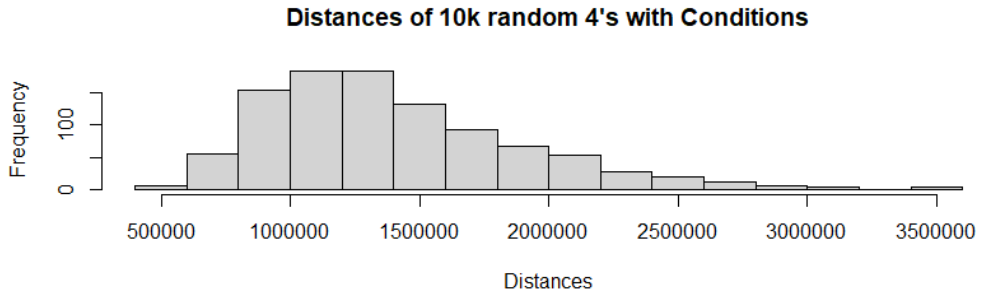
There appear to be a few very bad matches. Below are the 3 worst matches in the set alongside their nearest neighbour (dist > 3.5 million). Interestingly, the problem is that the model is too rigid and it lacks flexibility or variation. The example of 447 shows that stroke1 can cut diagonally across the image and it can end in a very low position. 618 demonstrates how low cross stroke (stroke2) can be. These images are what we consider 'challenging' for our model.

From this a few things are clear:

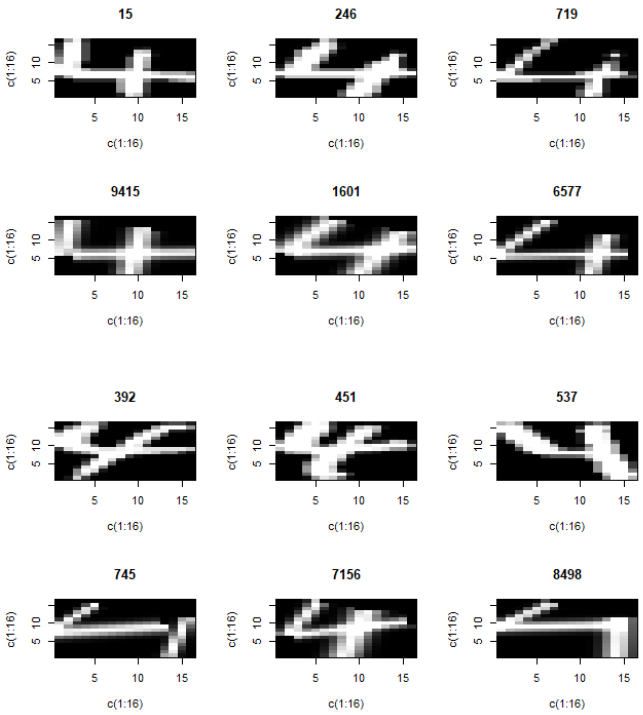
- The lower bound for theta1 should be extended to 0.75pi based on 447
- The condition for ye1 should be lowered to > 2 based on 618.
- The starting point of stroke1, x1, should be extended to 12 based on 447
- The length of stroke1, len1, should be longer extend to 17.
- The condition for xe1 will be extended to > -1 and < 5 to avoid clashing with stroke3
- The endpoint for stroke3, xe3, should be constrained to > 7 to mitigate clashing with stroke1 as seen in 5280

With the updated parameters and conditions, the model was used to generate an additional 10k random digits. Here are the distances. The mean distance was 1399306 and the standard deviation was 485815.5

There has been a slight improvement from the last model, once again, let's look at the best



matches (to see how the model has changed and what type of 4 does it best generate) and the worst matches (to see the models blindspots)



Best Matches

On top are the real digits and below are the nearest neighbour. Here we can see the model is more accommodating for diagonal strokes, this is what we were aiming for.

Worst matches

This is a very interesting revelation. The worst fits for our updated model are completely different from before. This illustrates 2 points.

1. The changes made previously have worked effectively as they are no longer showing up as the worst matches, this is a positive sign as it reflects an effective approach to solving the issues which arise.
2. There is a wide variety of digits in the data set which must be accounted for, when solving one issue, another unforeseen will arise, I will explore these further below.

Each poorly represented real digit above (392, 451, 537) will be analyzed to see what is missing or overlooked in the current model.

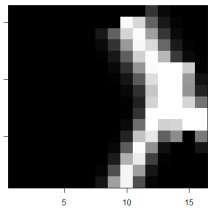
1. **392 -** Two elements stand out. Stroke1 is very high up and its length is very small, perhaps extending the lower bound for our model could be an option. Secondly, stroke2 has a very large length and angle across the image, expanding the length of stroke2 and increasing the angle may ameliorate the situation. It is to be noted that further constraints may be required to ensure that, as we add variation to the system, stroke1 and stroke3 do not collide.
2. **451 -** It is clear that the lower bound constraint for the stroke3 endpoints, xe3, is too restricted. Therefore removing the bound and replacing it with a distance constraint

with stroke1 would be more appropriate as this solve our while allowing for adequate variation in our model.

3. **537** - Stroke1 and stroke3 are backwards! This demonstrates a need for greater variation in our model for the trajectory of these strokes. Theta1 and theta3 will be adjusted to accommodate this variant.

NotClash Function

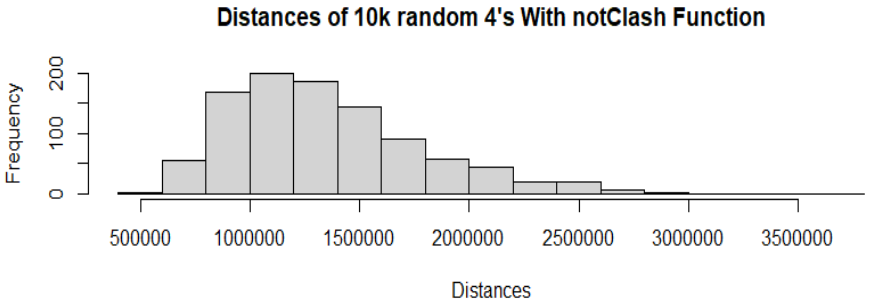
One issue which had to be addressed was the clashing of stroke1 and stroke3. As I added more variation into the model, this became increasingly problematic.



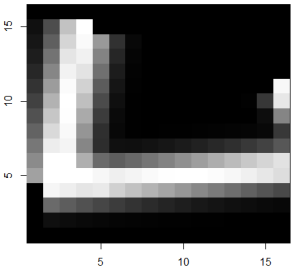
At first, constraints were set on the end and start points of stroke1 and stroke3. It turned out that there were cases where the endpoints and start points crossed over and there were no problems so it was an inadequate judging criterion. Here is one example of the two strokes clashing.
Below is the code for the function

```
notClash = function(x1,y1,x3,y3,xe1,ye1,xe3,ye3,theta3,width1,width3)
{
# This calculates the perpendicular distance between two start points
start = sqrt((x1 - x3)^2 + (y1 - x3)^2)
slope1 = (y3 - y1)/(x3 - x1)
angle1 = pi - (theta3 + atan(abs(slope1))-pi/2)
startdist = start * sin(angle1)

# This calculates the perpendicular distance between two end points
end = sqrt((xe1 - xe3)^2 + (ye1 - ye3)^2) # Full distance
slope2 = (ye3 - ye1)/(xe3 - xe1) # Slope between two start points
angle2 = pi - (theta3 + atan(abs(slope2)) - pi/2) # Angle used for projection/perpendicular
enddist = end * sin(angle2)

# Dist is scaled using the width as thicker strokes are more likely to clash
if ((width1 + width3) / 3 > 1) width = (width1 + width3) / 3 else width = 1
dist = min(startdist, enddist) / width # Find the minimum distance between the two strokes
dist > 3 # Criteria for determining if two strokes are too close together
}
```

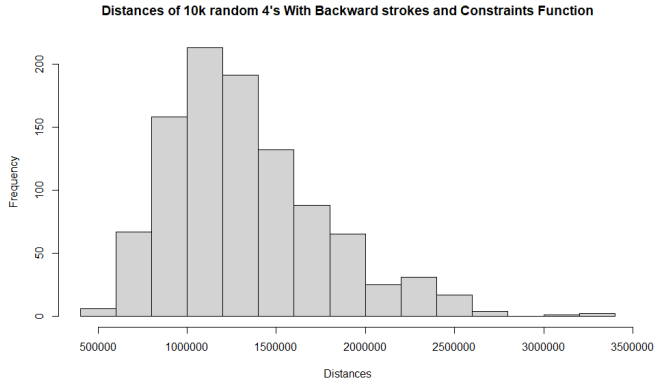
The random digits were generated with the function. Here the mean distance was 1347540. An improvement from before. Next, we will address the backwards stroke



Backwards Stroke



The model was enabled to generate backward strokes for stroke1 and stroke3. This change expressed itself as a slight adjustment to the theta parameters. As with any system, the added variation produced further problems to be tackled. To the right is an example of stroke3 running off the image. The distances for nearest neighbours were calculated and there was a dramatic increase with a mean of 1440670. It was clear the precision of the model was influenced by the issue exhibited in the image above. Therefore it was decided to apply a constraint ensuring xe3 < 16. This prohibited stroke3 from running off the image.

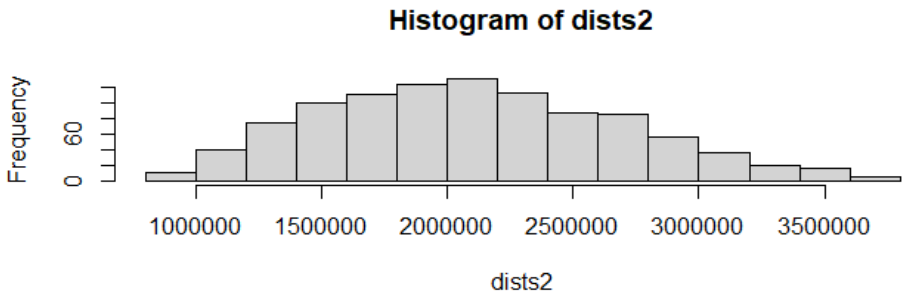


Here is the distribution of the distances with the constraints. The mean distance is 1334678. This is the most effective model so far.

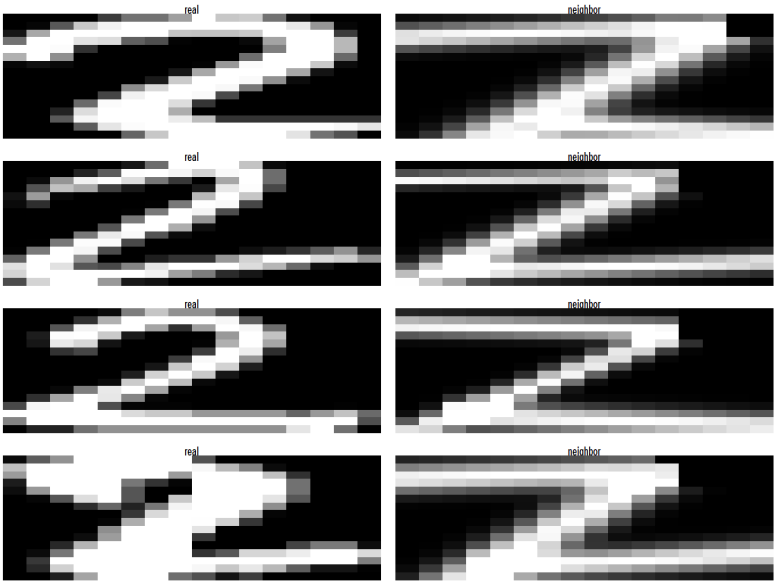
However, improvements can still be made. These will be addressed later.

Question 2 (Joseph: Fitting the 2's)

Upon fitting the synthetic 2's to the real 2's, I obtained the following histogram of distances:

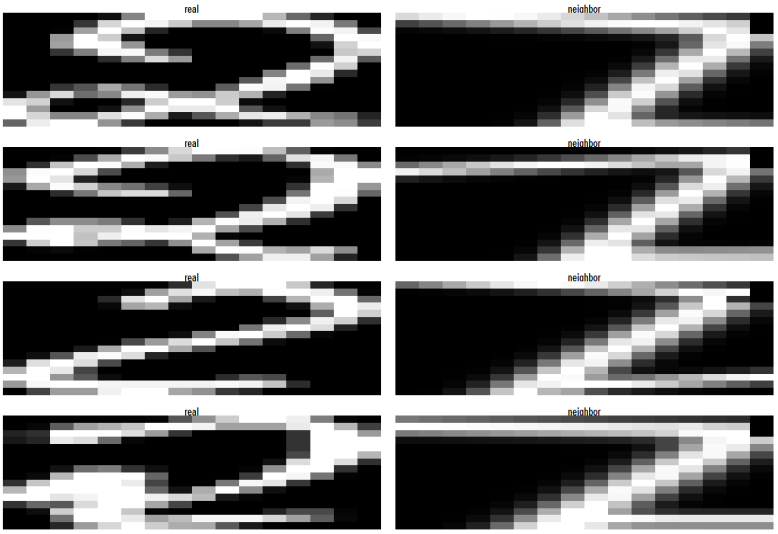


The average distance is greater than 2 million. This suggests the fits aren't great but let's look at some good and bad fits. Good fits have a distance of less than 1.5m while bad fits have a distance over 3m. Good fits:

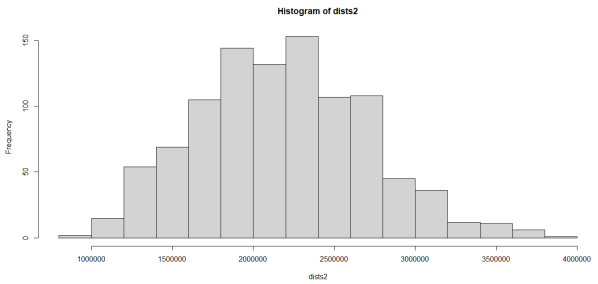


As can be seen here, in these cases the lower stroke is matched well by our model. The middle stroke is also well matched although the lengths differ in the real 2's but are fixed in the model 2's, sometimes causing the 2 to run out of the image frame (as shown in the first and last pair). For now, this is okay.

Now for the bad matches: The lower stroke for the bad matches is commonly looped or curved in shape. I will not account for this in the model. Although I decided here to try and change the starting point of the lower stroke to see the effect this has on the fits. It generally starts in the lower-left corner. The New x3 range is [1,5], y3 range remains the same. I also changed the range of theta2 to [0.65, 0.80], it didn't seem like the higher angles had matches so I tried to reduce the variance of the model. I also set the starting point of stroke 2 (x2, y2) to be fixed at the endpoint of stroke 1 instead of varying around its endpoint. The upper and middle stroke of 2's is a continued single stroke generally.



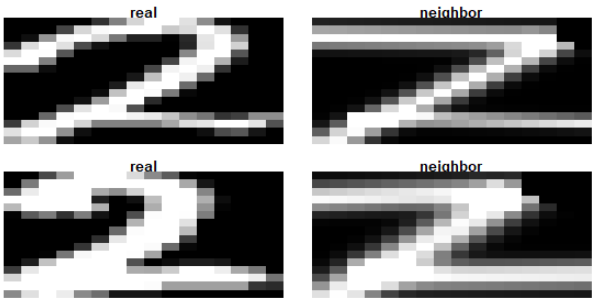
The second histogram of distances shows an improvement in the fits with the distances shifting to the left slightly:



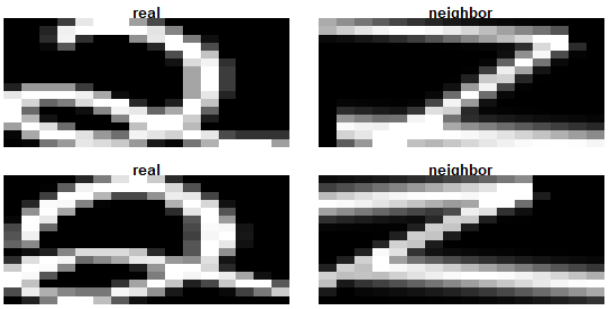
For the middle and lower stroke, the fits remain good for average 2's.

The lower stroke loop shape is better compensated for but is still badly matched. We ignore this for now. It seems the more rounded upper strokes make for poor fits also.

(1) **Good fits**

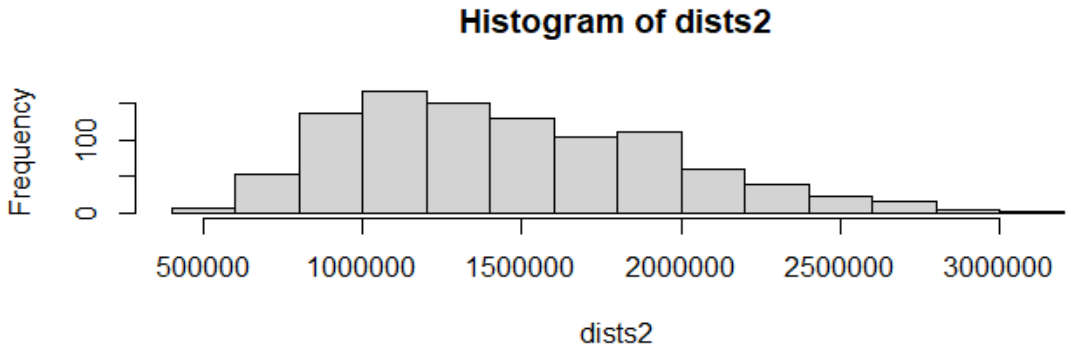


(2) bad fits

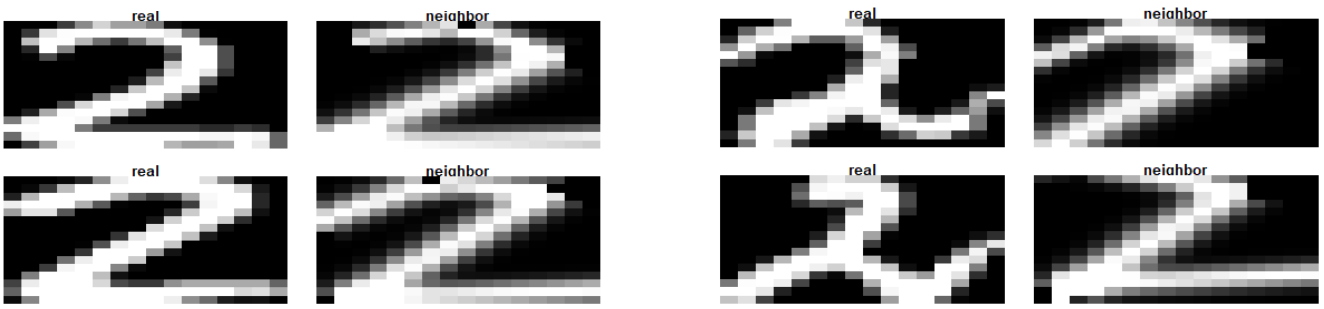


I've been ignoring the curved upper stroke up until now. I will now add a second stroke to account for the curved shape on the top and investigate its effect on the fits. We will call the stroke accounting for the curve the "hook stroke" and its parameters are x4, y4, theta4, len4, width4. The endpoint of the middle stroke hiding the bottom stroke is still a problem so I tried a new approach. The changes I made are highlighted:

Stroke	X	Y	Length	Theta	Width
1	(12,16)	(12,15)	[5,10]	(π*0.25,π*0.50)	(0.5, 3)
2	x1	y1	[16,20]	(π*0.65,π*0.80)	(0.5,3)
3	(xe2-1,xe2+1)	(ye2, ye2+1)	20	(π*1.45,π*1.55)	(0.5,3)
4	xe1	ye1	[5,10]	(π*0.50,π*0.75)	(0.5,3)



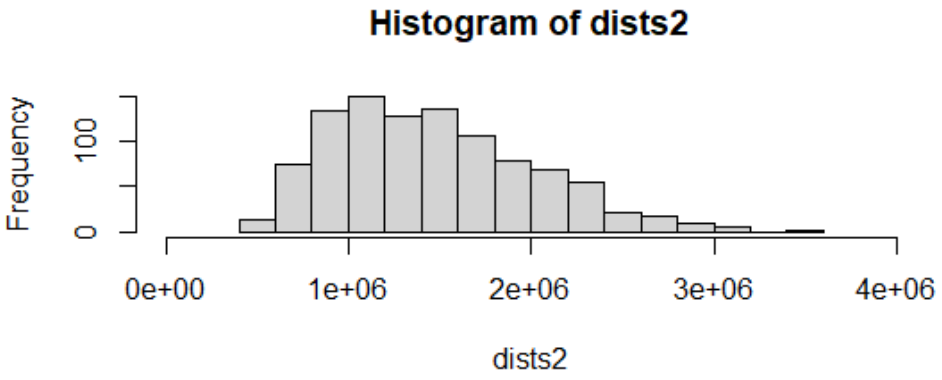
There is a noticeable improvement in the fits from adding this fourth stroke, with the histogram starting to skew to the right and the mode being just over 1m. The good fit constraint for testing changes to 1.5m and the bad fit constraint changes to > 2m. **(1) good fits (2) bad fits.**



From the bad fits, we see the weird bottom stroke misfits which are expected. We also observe the bottom stroke sometimes leaving the frame. I added randomness to the length of the middle stroke to try and improve the middle stroke (and hence the positioning of the bottom stroke) but the problem persists at times. Generating more images would further reduce the effect of this issue. A similar problem was observed with the upper stroke and hook stroke. To try and fix these, I will add in two more constraints.

1. If ye1 (y coordinate of the endpoint of stroke 1) is above 16.5, then regenerate 1
2. If ye2 < 0.5 or xe2 < 0.5, then regenerate 1.

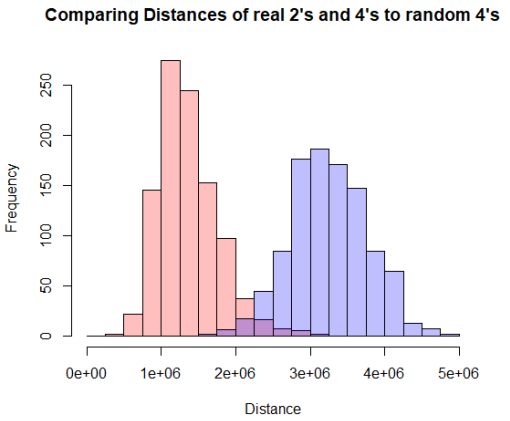
This will remove 2's whose hook stroke or lower stroke start at a coordinate outside the frame. For this histogram, the x limits were set from 0 to 4 million.



The distances are only slightly improved but the out of frame problem has been mitigated.

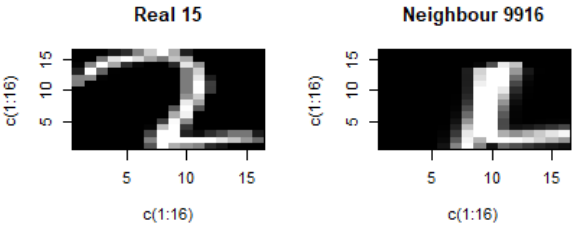
Question 3: Fitting Models To Other Digits

Fitting Real 2's Generated 4's (Diarmuid)

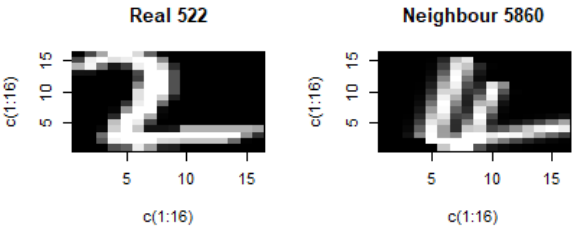


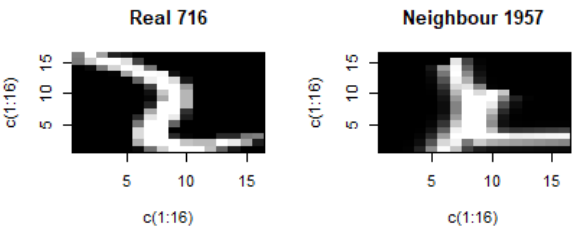
The following step involved loading in the real 2's then applying the 'findnss' K Nearest Neighbours Algorithm to fit the real 2's to the randomly generated 4's discussed above. The histogram for the distances is displayed below. The maximum distance between the 4's was 3214848. The minimum distance between the 2's and 4's was 1577750. It was discovered there were 485 instances of real 2's which were modelled more accurately than 262 real 4's themselves. To the right is the two histograms of real 4's to 4's and real 2's to 4's.

Note that the overlap between the two graphs does not indicate misclassification. It is just an overlap of distances recorded. A misclassification is when an instance of a digit is more accurately modelled by a foreign model rather than its corresponding one eg. a 2 is better modelled by a random 4 than a random 2.



To the right are the 3 real 2's which were best modelled by the 4's model. The notClash function did not work effectively for images 9916 and 1957. What can be done is to investigate both images individually and see how they were validated by the system. What appears spurious is the match between 522 and 5680. The generated 4 appears to be normal apart from perhaps the low stroke2. However, based on my previous analysis, this level of variation is necessary in our model, so that is an example where making changes or creating further constraints may disimprove the model



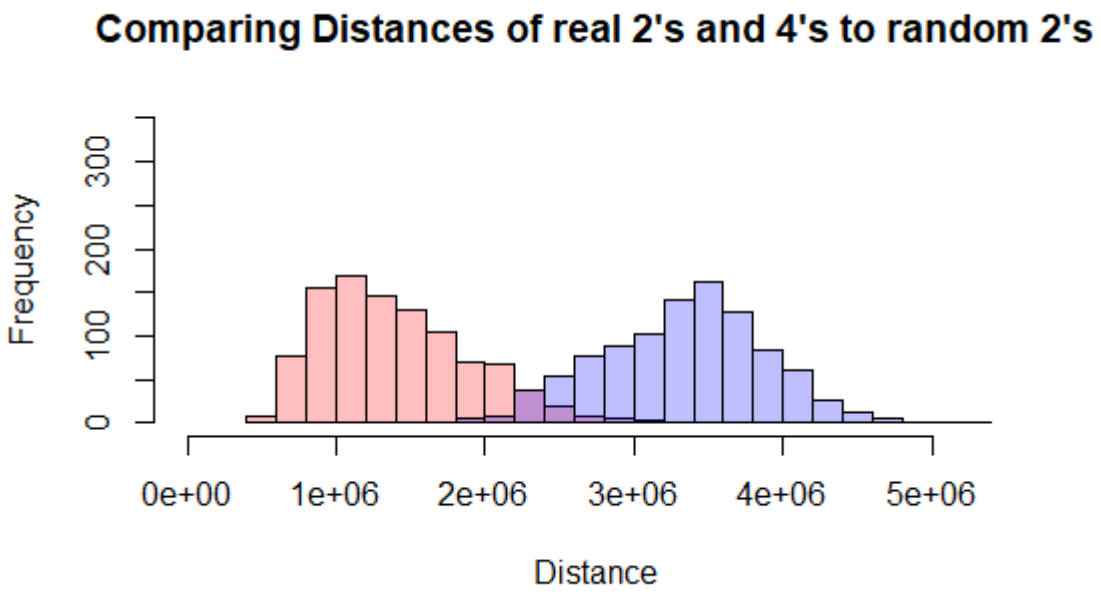


Learning Points:

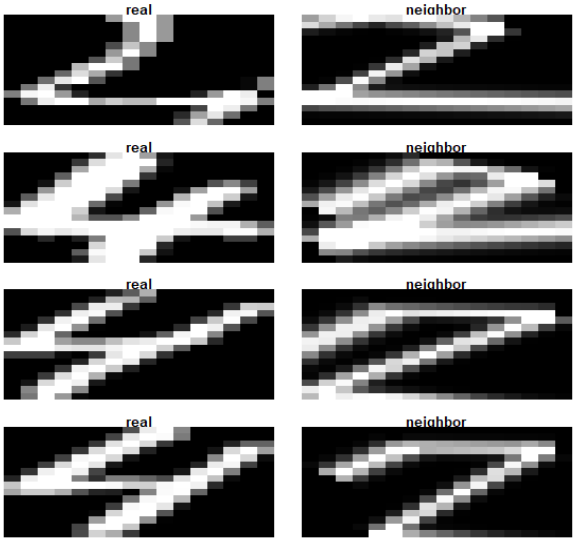
1. The crossover of stroke1 and stoke3 resembles and is mistaken for the middle stroke of the 2
2. A low stroke2 of the 4's resembles and is mistaken for the bottom stroke of the 2
3. The notClash function does not account for stokes that fully crossover and therefore their perpendicular distance

Fitting Real 4's to Generated 2's (Joseph)

KNN is run on the real 4's using model 2's, the distances between the target and nearest neighbours are calculated. A histogram of distances is generated (blue) and we compare this to the histogram of model 2's fitted to real 2's (pink).



As expected, the 4's fit the model 2's badly, with a mean distance of ~3.3 million in comparison to the mean of ~1.4 million for the model 2's to real 2's. There were few instances where the random 2 was closer to a real 4 than a real 2. Let's view some of these:

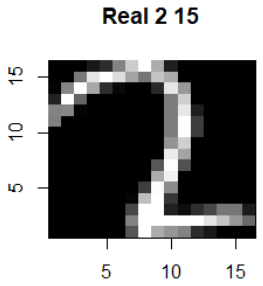


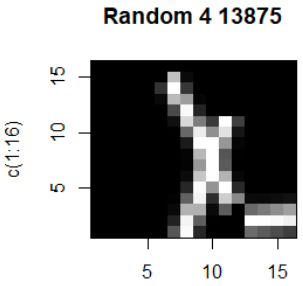
There's a pattern here, the misfit twos tend to have a parallel middle and hook stroke, giving a similarity to the first and second strokes of real 4's. We saw one match due to strangely drawn 4 (the first pair) whose second stroke is very small.

Classification of 2's and 4's (Diarmuid)

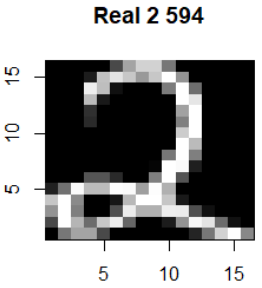
The 'findnns' Nearest Neighbours algorithm was run for 2's and 4's. A maximum accuracy rate of 98.5% was achieved. This was quite impressive for running the algorithm the first time. This was the confusion matrix:

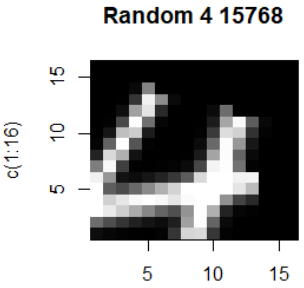
	Labels	
Results	973	3
	27	997



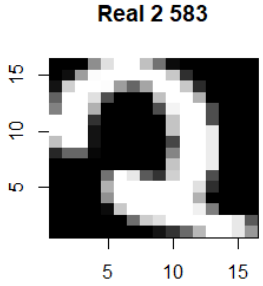


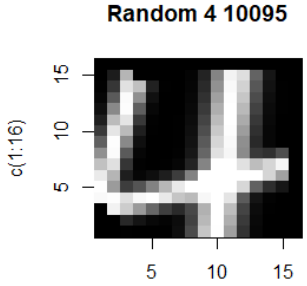
Here are examples of misclassified 2's. As foreshadowed before, the crossed strokes appear to be the culprit. 15 appears to be quite a regular 2. It is a surprise that it was misclassified. The misclassification of digit 594 is quite puzzling. The 2 and 4 look like night and day. What this illustrates is the importance of the loop in the 2's.





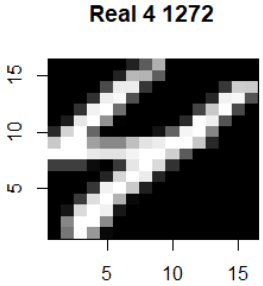
Digit 583 is a very thick and unusual form of two, it is such an outlier from the other 2's that it looks like none of the digits, even its nearest neighbour. This was tested by calculating the distance between 583 and 10095, which resulted in 3233139. The average distance between real and random 2's is 1.4 million. This is a clear outlier.

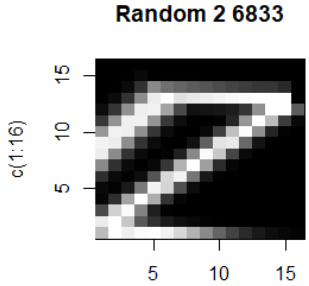




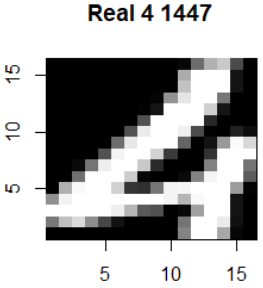
From the analysis carried out above there is a greater need for variation in the model. This is the most challenging part of the 2 by far. Attempting to incorporate a loop in the 2 would invite an immense level of variation which would be difficult to manage.

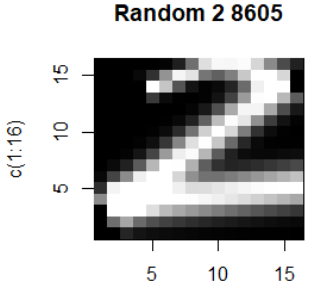
These two misclassifications of four demonstrate one issue, long diagonal strokes are not being adequately represented in our model. The "parallel hook and middle stroke problem" is seen here (discussed in the above section). One would think digit 1272 would be easily modelled using the current parameters, perhaps there is more we can do.





In the confusion matrix above there are many more misclassifications of 2's than 4's. Based on previous analysis of strokes, we have seen that real 2's are sometimes more complex to model (due to the loop of the bottom stroke and the curved shape of the digit)

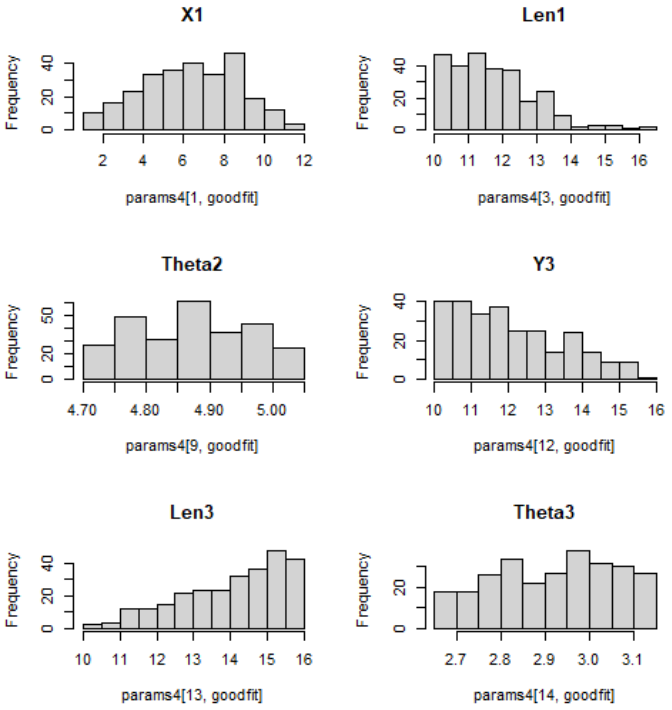




Question 5 (4): Distributions of model parameters

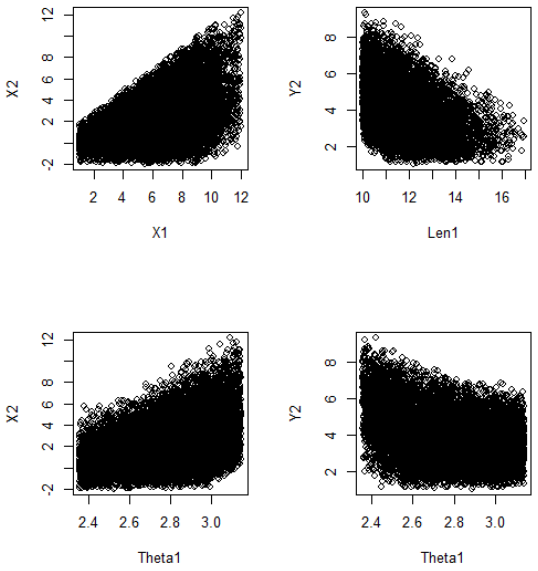
Model 4 parameters (Diarmuid)

The first step taken was to visualize each parameter in 15 separate histograms. It was clear from the histograms that the constraints had warped the uniform distribution imposed on each parameter. Next, goodfits were defined as dists < 1250000, this resulted in 441 instances. It was fascinating to discover that there was no contrast between the histograms for all values and the goodfits, their distributions were identical. This exemplified how each parameter was modelled appropriately. Although this is not to say improvements can't be made. Many of the parameter's distributions pushed their respective limits. To the right are a variety of examples that exhibit this feature. In these parameters, the following changes can be made and then tested.

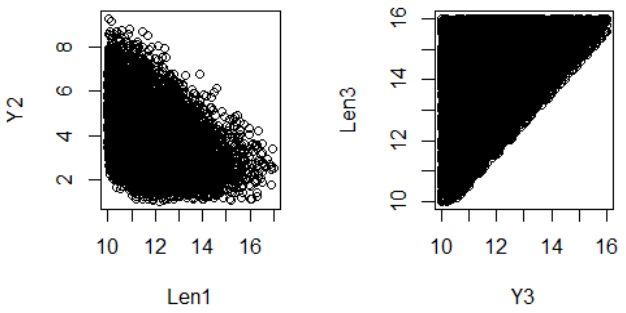


1. X1 resembles a normal distribution however it is important to note that the distribution for all values also represents a normal distribution. Testing the effect of modelling x1 as a normal distribution is something that would be worth exploring.
2. Len1 is pushing the lower bound and extending this lower bound will most likely ameliorate model performance.
3. Theta2 appears to be too restricted by its current parameters, expanding these may prove beneficial.
4. The values for Y3 are pushing the lower bound, once again these can be alleviated.
5. Increasing the upper bound for len 3 is worth testing, this may help with better representing the long diagonals
6. Extending the range for theta3 may also help with better representing the long diagonals.

Now it is time to explore the relationship between each parameter. Since there are 15 variables, there would be 105 combinations of correlation which would be impractical to display here. Therefore the correlation coefficient between each variable was calculated and from there pairs that had an absolute value of correlation greater than 0.25 were selected and investigated. There were a select few scatter plots that had a curious distribution. I chose to plot all values rather than just good fits as they are essentially the same except for frequency. To the right are four plots that possess similar patterns. These patterns occur due to the dependencies between the x and y values and the constraints in place.

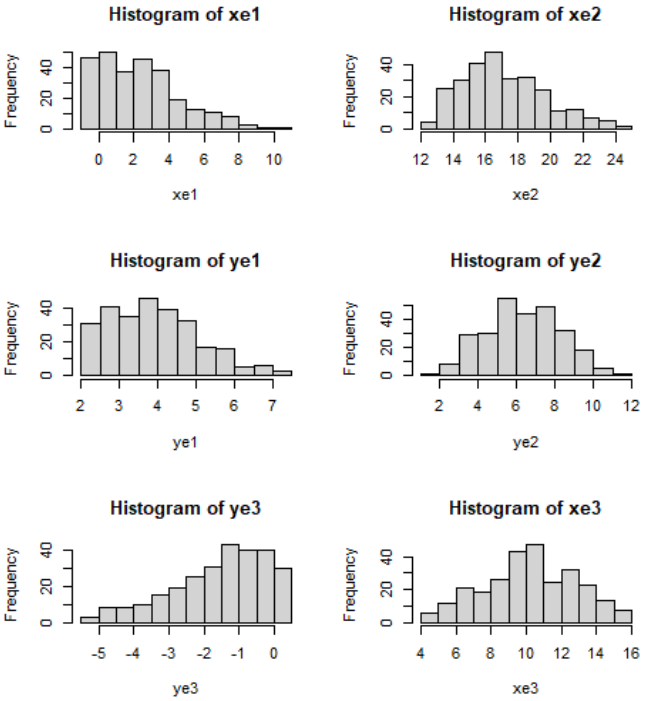


The distribution of Y3 and Len3 is incredible. One constraint I have in place is that stroke3 must touch the bottom, therefore len3 must be at least as large as Y3, the top point (starting point), thus creating this linear barrier.



End Points

Usually looking at the distribution of endpoints of our strokes would be fruitless as endpoints are by-products of pre-generated parameters and we do not generate the endpoints ourselves. However, in this case, the endpoints are being directly constrained by our model. Moreover exploring the effectiveness and practicality of these constraints is a worthwhile exercise as they play such an influential role in our model. The goodfits of endpoints are plotted to the right.



In the left column are endpoints that have been bounded and constrained, this is clear from the build-up on one side. In the right column, are the endpoints that do not have direct constraints. It is eye-opening to see that the endpoints without constraints approximate a normal distribution.

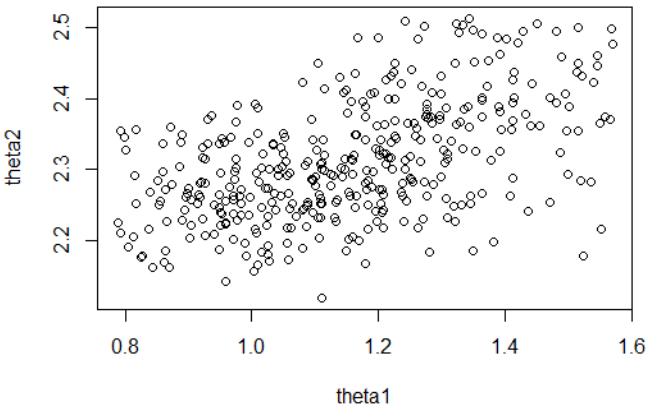
This indicates that relaxing the constraints on the parameters may improve the performance of the model.

Model 2 parameters (Joseph)

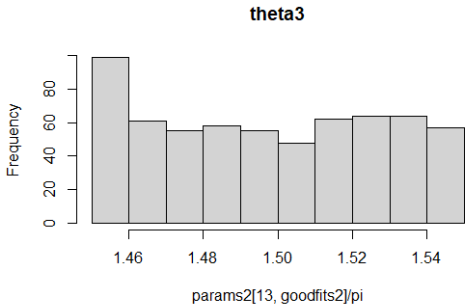
I will set goodfits criteria as distances less than 1.5 million to try and find the approximate distribution of the parameters. In this way, we can improve the process of generating 2's. Note that a number of stroke parameters are dependent on previous strokes or endpoints:

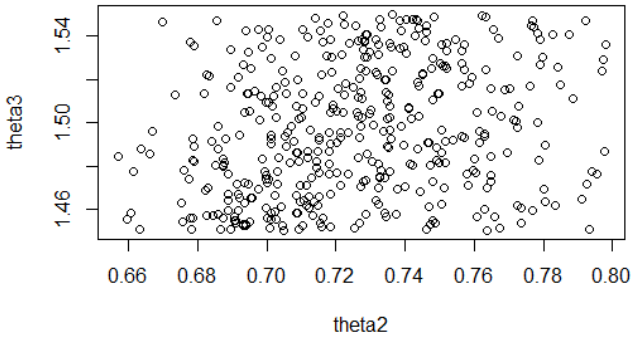
- x2 and y2 are dependent on x1 and y1
- x3 and y3 are dependent on endpoints x2e and ye2 (not tunable, will not explore this)
- x4 and y4 are dependent on the endpoints xe1 and ye1 (not tunable, will not explore this)
- len3 is fixed

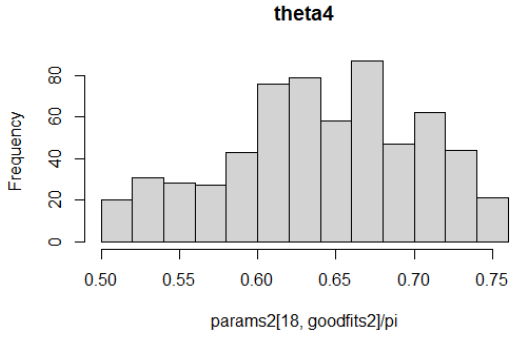
(1) Positive correlation between theta1 and theta2. They appear to follow approximately a bivariate normal distribution as the points are most dense in a region around theta1 $\in$ [1.0, 1.2], theta2 $\in$ [2.2, 2.3].

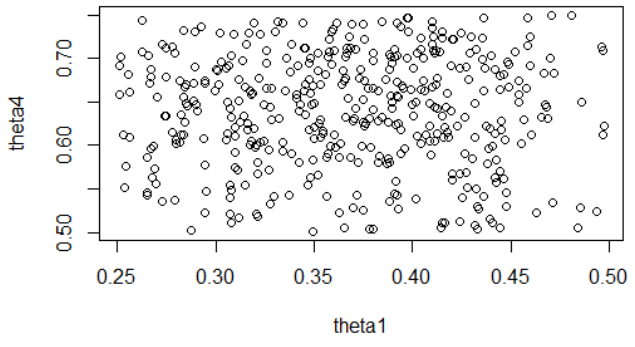


No evidence of correlation between theta2 and theta3 good fits. Theta3 is approx. uniformly distributed. There is also no notable correlation between theta1 vs theta4 (and theta3 vs theta4) but theta1 and 4 are both unimodal. (approximately normal)

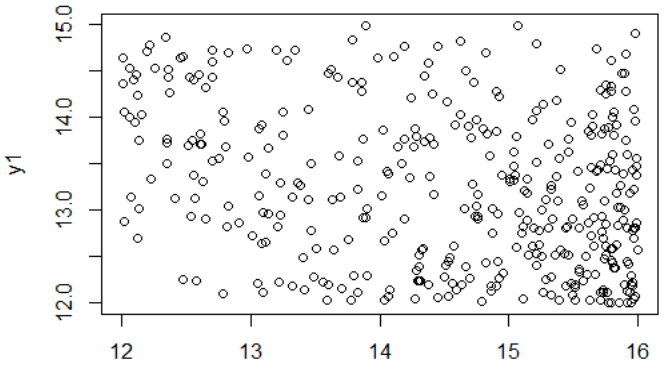




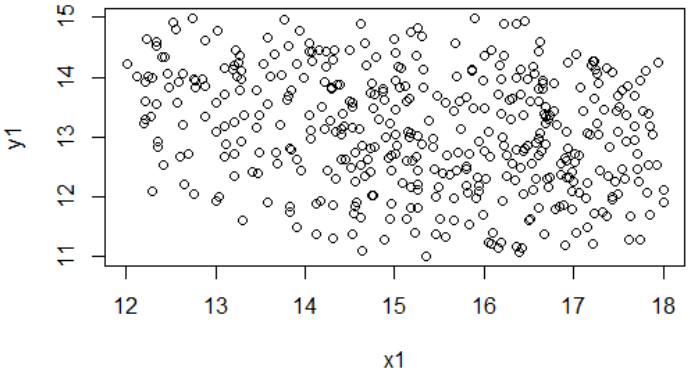


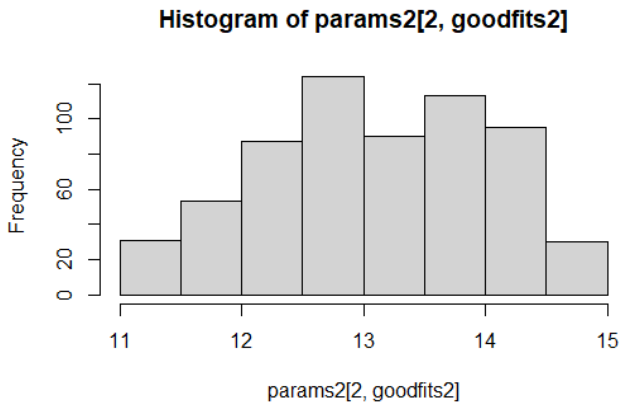


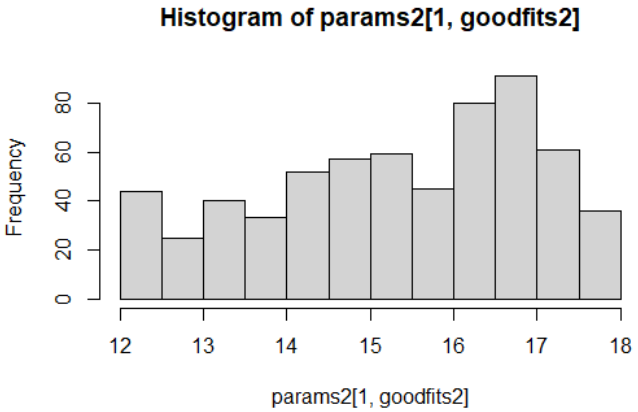
(2) The distribution of x1 is left-skewed and y1 is approx. uniform. Points seem to pile up at the lower limit of y1 and the upper limit of x1 (bottom right of the scatter plot). I will try expanding these limits to get a fuller view of the distribution. (x1 upper limit: 17, y1 lower limit: 11)



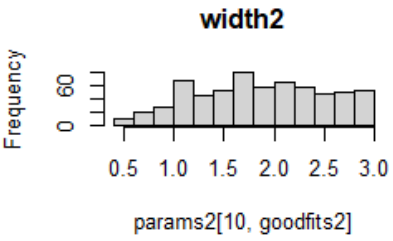
The new ranges reveal the distributions are unimodal. There is not much evidence of correlation (weak negative), the parameters could be generated from a normal distribution.

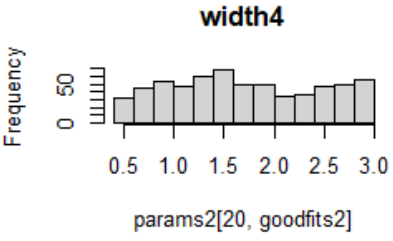




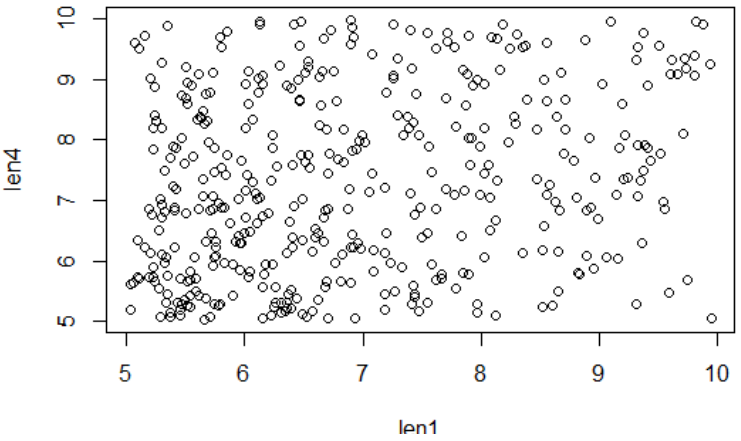


(3) there is no evidence of correlation between any of the widths and their distributions are uninformative (uniform).

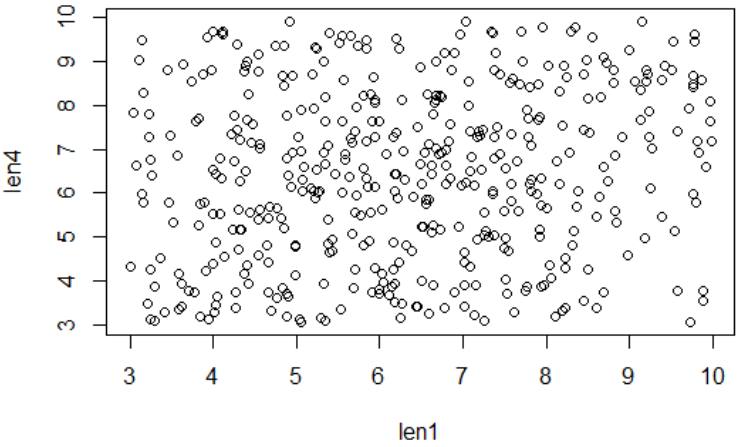




(4) there is a pile-up of data points at the lower lengths of strokes 1 and 4. These would represent 2's with even curves (top and hook stroke combined) To get a fuller view of the distribution, I will increase the range of len1 and len4 to [3, 10]



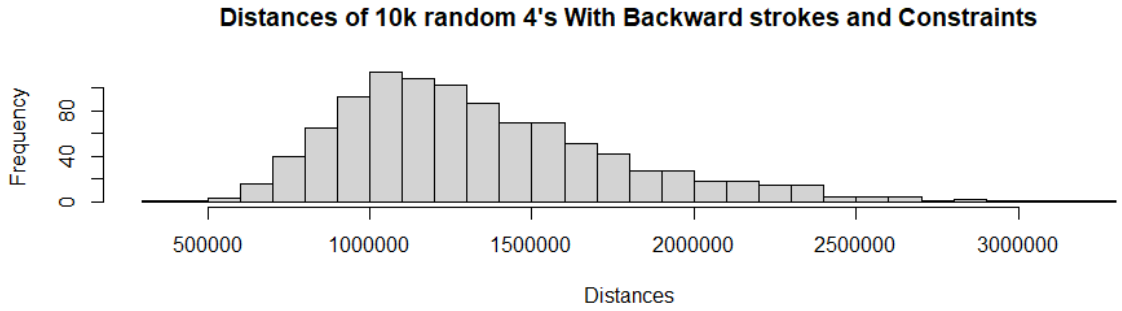
Extending the ranges reveals data points are denser towards the centre.
The rest of the lengths show little or no correlation with len1 being closest to normal. From my analysis of the parameters, there are no clear violations of normality assumptions and using PCA would be appropriate.



Question 6: Applying PCA to parameters

Applying PCA to model 4 parameters (Diarmuid)

To begin, this is the distribution of distances for our current model of 4's. The mean distance is 1336028 and the standard deviation is 431521.7.



In the last question,, a set of hypotheses were formed based on the analysis performed on the distribution of parameters. These will now be tested.

There are many changes to be made to the model, therefore a testing process should be established.

1. For each test, 1 stroke is taken and the key parameters are adjusted appropriately.
2. Then to test the changes, the Nearest Neighbours algorithm is run and the distances are plotted. The distribution of the parameters is inspected to see how the adjustments have affected the goodfits.
3. If there is a performance improvement continue making adjustments and testing their effectiveness.
4. This process is repeated until there is no significant improvement in the model.

Test 1: Changes stroke1

- X1: changed to normal distribution
- Len1: Lower bound adjusted to 7,17
- Theta1: Range expanded to 0.65pi, 1.1pi

Mean = 1310593 Standard deviation = 382897

Test2: Change stroke2

- Len2: range increased to 13,17

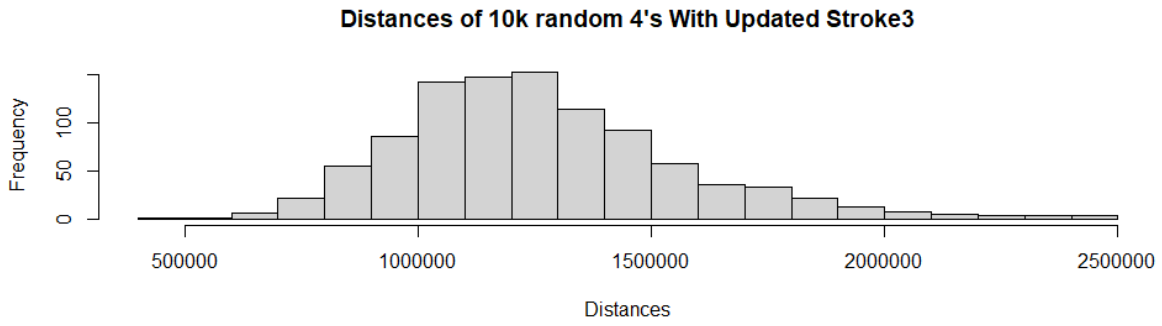
Mean = 1305001 Standard deviation = 379471

Test 3: Change stroke3

- Theta3: Expand range to 0.75pi, 1.1pi 1266825 311524.2
- Len3: Extend range up to 22

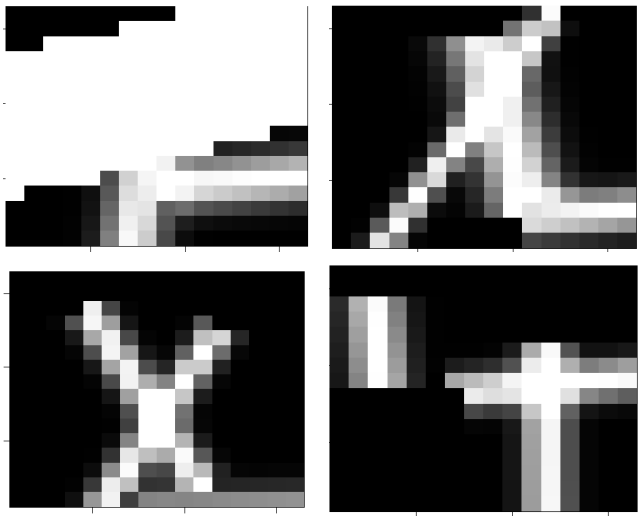
Mean = 1239983 Standard deviation = 307933.9

The results obtained from changing the parameters had a small effect, however, it was not significant. Below is the current distances for all updated changes



Applying PCA Model Parameters

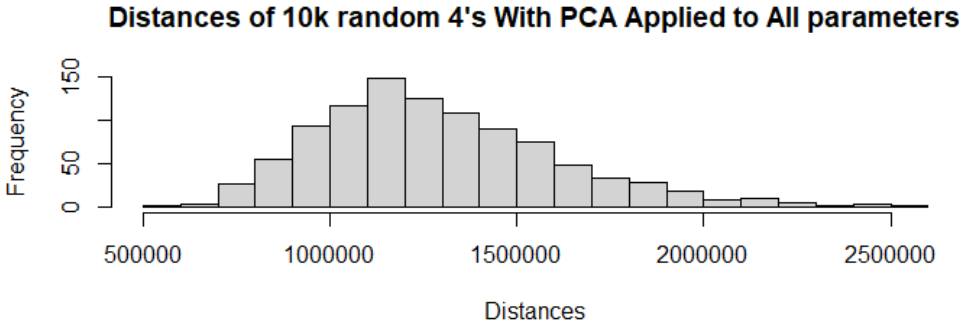
The PCA Methodology was applied to the 4 models. This meant that all constraints had been removed from the model and all parameters were normally distributed. When viewing the different examples of images that had been generated there were some defects. To the right are some of the examples

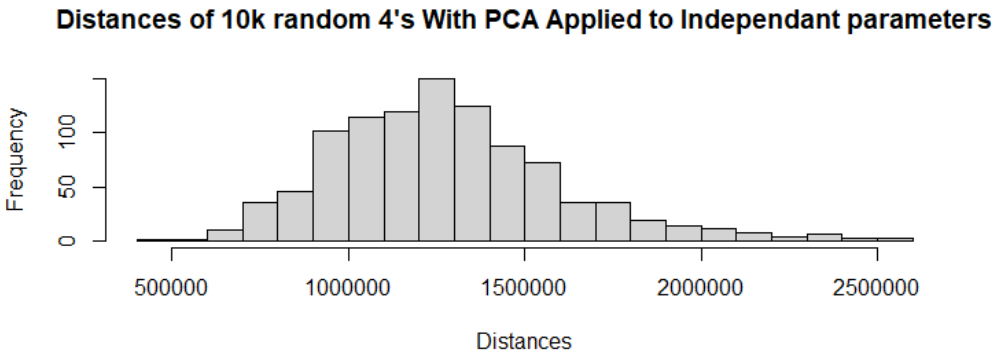


This indicated that not all parameters should be included in PCA Model and others should be modelled differently.

NN Algorithm was run for PCA on all parameters and a mean of 1288723 and a standard deviation of 318422 was obtained.

These results were not as accurate as before so changes were made to improve the system. It was identified that there were linear dependencies between the parameters (x2, y2) which affected performance. These parameters were removed and the algorithm was run again. This time a mean of 1278003 and a standard deviation of 323516 was obtained.
It was clear at this point that the model had been optimized as much as possible.





PCA is a useful tool for making improvements to the performance of a model. However, if the highest accuracy is desired, manually setting parameters, while being time and energy-intensive, yields the best results.

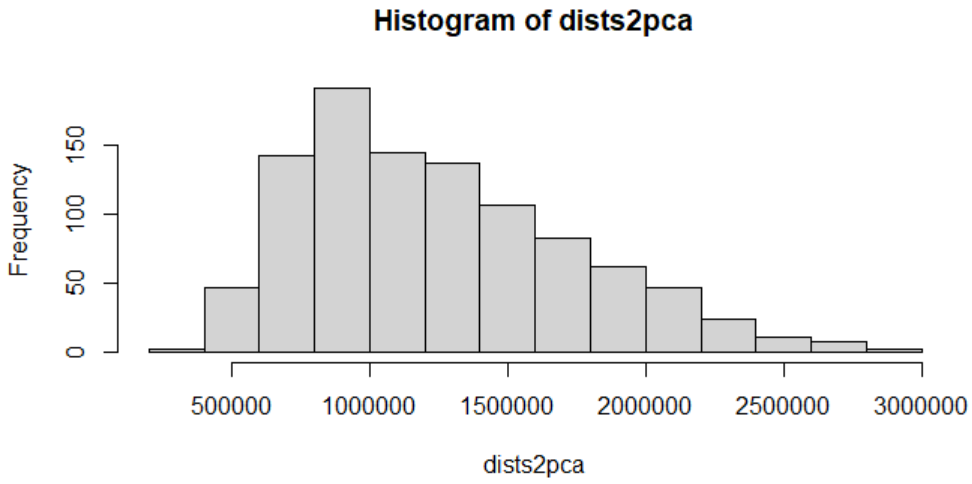
Applying PCA to model 2 parameters (Joseph)

Following the results of our previous analysis of the parameters for model 2s, I will apply PCA to the entire parameter space, generate parameter values from a multivariate normal distribution, and decompose the PCA space to return to the parameter space. With these parameters, 10,000 model 2's will be generated. I will briefly evaluate the quality of fits and then the digits will be used for the classification of real 2's and 4's.

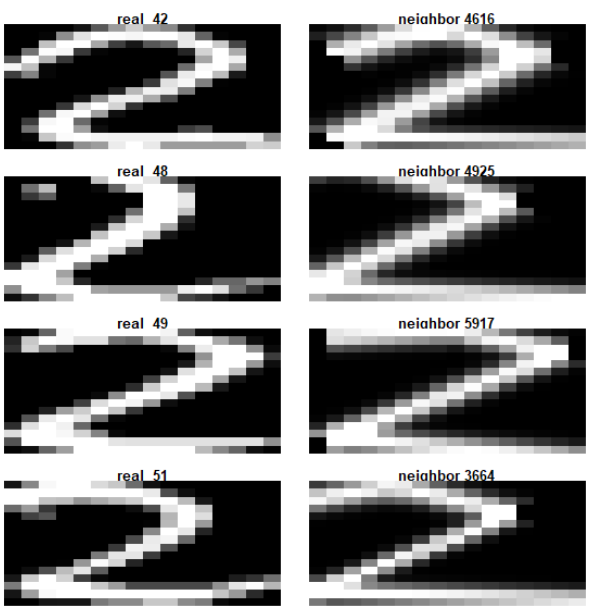
I excluded parameters x2, y2, x3, y3, len3, x4, and y4 from the PCA as they are dependent on other parameters. I also removed constraints on xe2, ye1, ye2 which I set at the end of question 2. Finally, I reverted any changes made in question 5.

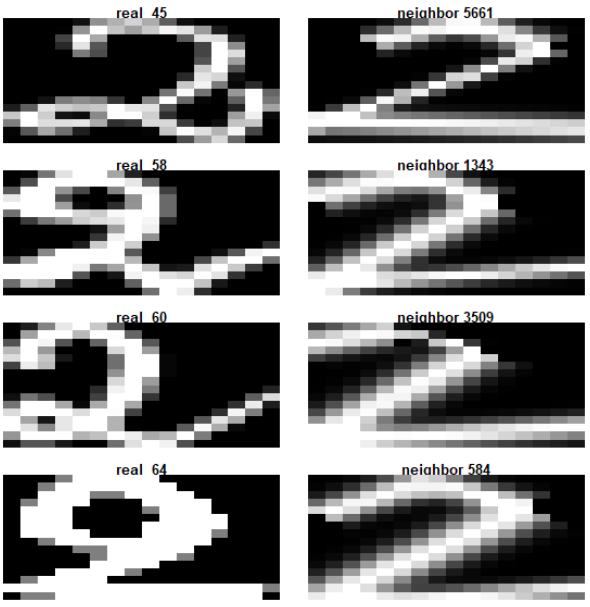
Results:

We see an improvement with a mean distance of ~1.2m in comparison to the previous ~1.4m. The distance histogram appears to have a stronger right-skew:



The goodfit criterion is changed to distances less than 1 million while the bad fits criterion is distances greater than 2 million. As can be seen in the second pair of goodfits below, the real 2's hook stroke can leave the frame so removing the question 2 constraint helped improve some of the fits. In general, using a multivariate normal model for the parameters improved the quality of model 2's. **(1) good fits, (2) bad fits.**





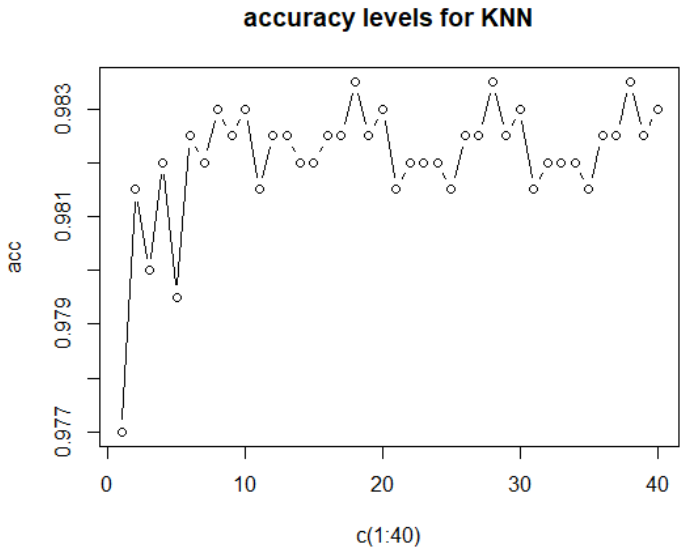
Classification of 2's and 4's (Joseph)

In this section, we classify the real 2's and 4's using our random 2's and 4's. No boxing methods were used for optimisation here. Firstly, here is the confusion matrix using k=1:

	Labels	
Results	957	3
	43	997

The accuracy is 97.7%, 0.8% less accurate than the classification of question 2. Note that only the misclassification rate for 2's has increased. I will now find the best value of k and investigate the errors for the model.

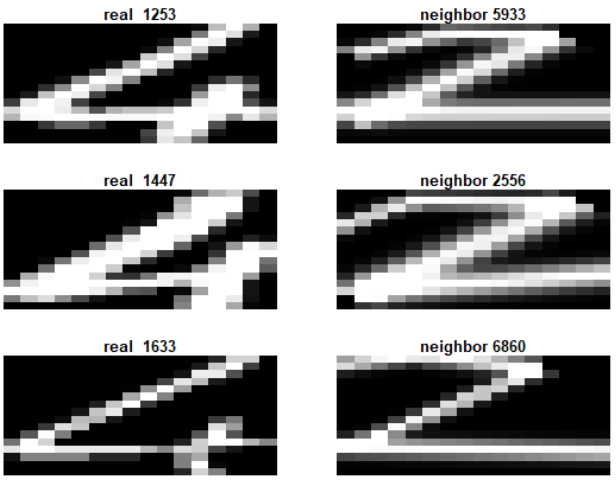
Unusually, we obtained our highest accuracy at k=18 with an accuracy of 98.4%. This is 0.1% less accurate than our initial classification in question 2. Here is the confusion matrix:




```
         labels
results    2   4
      2  970   3
      4   30 997
```

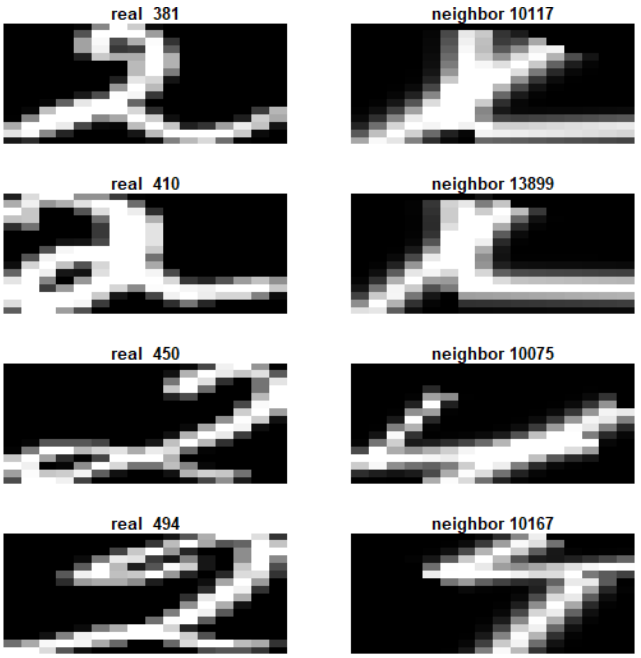
Again, the misclassifications of 2's appear to be the main problem. Let's first view the misclassified 4's:

The first and third stroke of the 4 is matched well by the middle and bottom strokes of the random 2's in these cases. The small second strokes are almost overlooked by the model, compensated for by the slightly thicker bottom stroke of the random 2. The 4's look like they could be classified correctly with more variation in theta1 range although since there are only 3 misclassifications, there may be a tradeoff.



Now onto misclassified 2's :

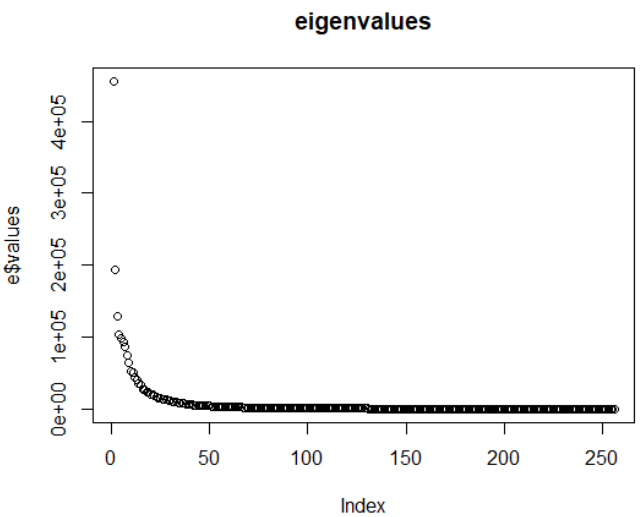
What we see here is unusual 2's matched by unusual 4's. The loop problem from earlier is prevalent in these misclassifications. Looped hook strokes also appear to be a problem here. Adding another stroke that loops on the top part of the 2 may prevent these misclassifications but adds more complexity to the mode that would affect the fits of average 2's (curse of dimensionality). Also, these unusual random 4's are the result of extensive variation in the model4 parameters. But relaxing constraints of variation was found to improve fits for 4's (in question 5) and this is evident in the high recall for real 4's.



Question 7: identifying optimum image-eigenvector number

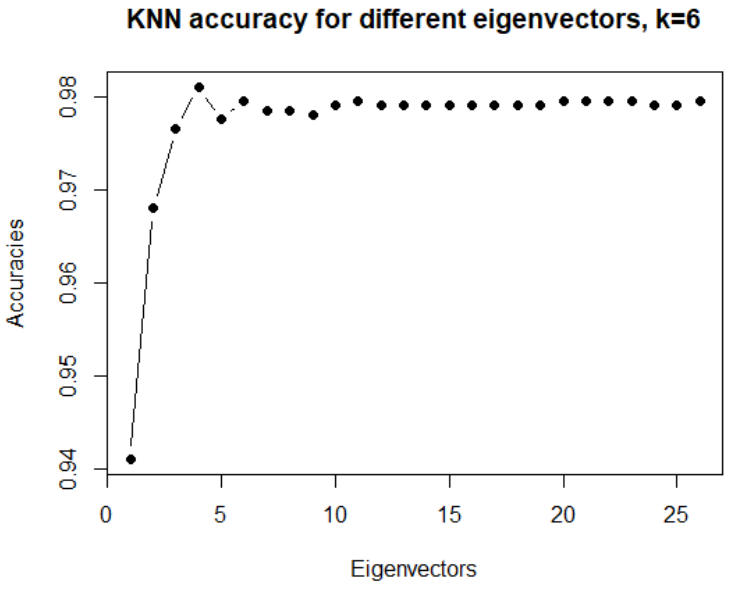
Accuracy and runtime test (Joseph)

In this section, we will run a KNN algorithm using k=6 and different numbers of eigenvectors ranging from 2 to 252 in steps of 10. That is 25 different numbers of eigenvectors. I chose k=6 based on the graph in the above question. The accuracy using k=1 may not be stable. I will use box methods for the test, setting the box sizes to equal the standard deviations of the first two principal components (675 and 440 respectively). I will record the accuracy and runtime for each number of eigenvectors.
After applying PCA, this is the plot of eigenvalues for the principal components:

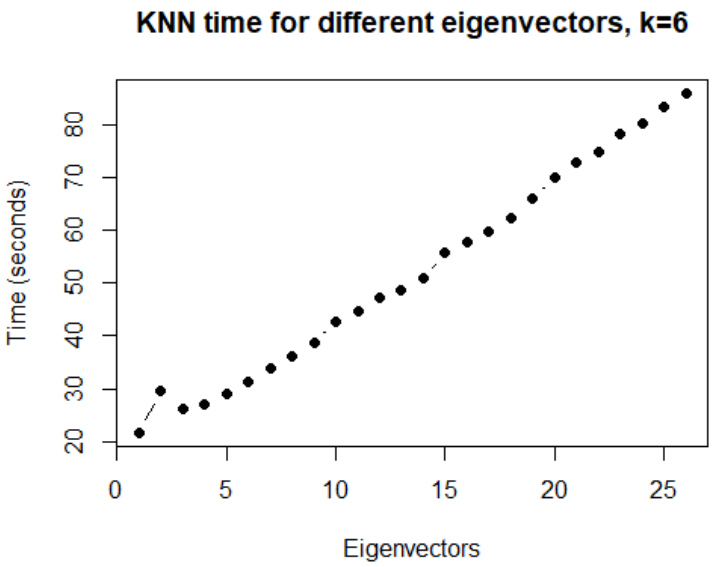


It appears that around 40 eigenvectors, almost all the variance in the data is accounted for. Note that given the way I am choosing k and the box sizes, we are not likely to achieve our maximum accuracy from this test but rather just the optimal eigenvector number.

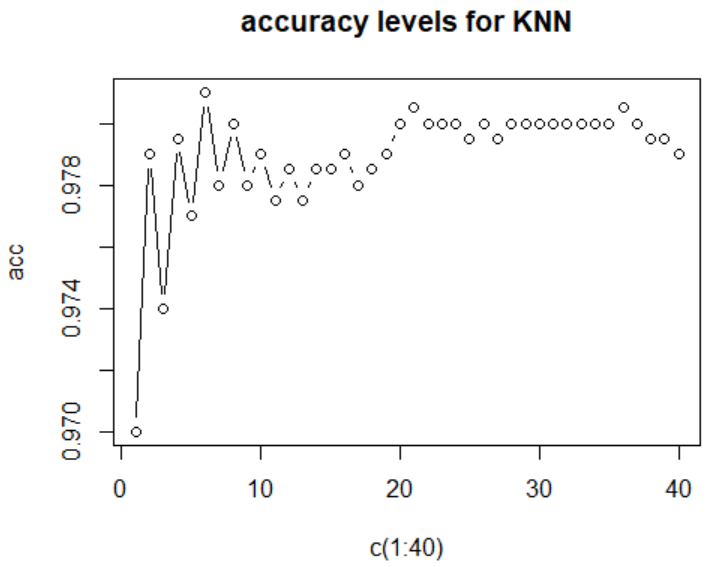
The data points are for accuracies corresponding to 2, 12, 22, and so on eigenvectors up to 252. The curve starts with a rapid increase in accuracy peaking at the eigenvector number = 32. The values then fluctuate, stabilising around 112 eigenvectors with an accuracy of 97.95%. The optimal number of eigenvectors was found to be 32. This gives us reason to believe that 40 eigenvectors are a good number to use for subsequent classifications. The accuracy at 32 eigenvectors, k=6 was 98.1%.



The runtimes increased linearly ranging from ~20s to ~86s. There was a slight fluctuation from the linear increase. This can be attributed to the machine being busy with other tasks. Now I want to find the optimal k value for the 32 eigenvector knn. Can we obtain a new optimal accuracy?



Strangely enough, 98.1% (k=6) with 32 eigenvectors was indeed the maximum using PCA with 32 eigenvectors.



Question 8: Classification of 1's, 2's, 4's and 7's (Diarmuid)

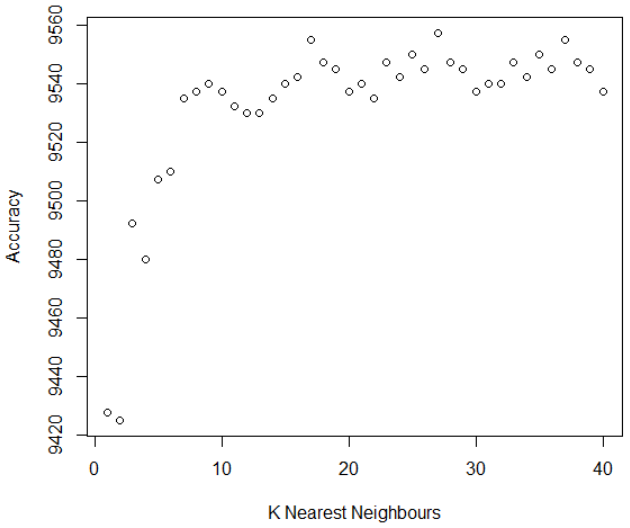
For classification of the 1's, 2's, 4's and 7's, an important step was deciding which model of the digits to use. Based on previous analysis these particular models were chosen:

1. **1's** - The ones generated using the PCA model appeared to provide the most accurate results.
2. **2's -** The twos generated using the PCA model appeared to provide the most accurate results
3. **4's** - The fours generated using manual manipulation of the parameters appeared to give marginally better results
4. **7's** - The sevens generated using manual manipulation of the parameters produced the best results

Each digit set was prepared appropriately in anticipation of classification. When running the classification algorithm we used the box method with 40 eigenvectors. The measurements for the box were set using the standard deviation of the first two principal components.

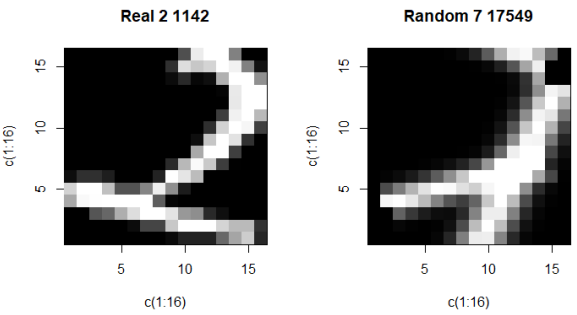
An maximum accuracy rate of 95.575% was obtained for classification using 27 Nearest Neighbours. This was lower than anticipated. Here is the confusion matrix and the graph of accuracy rates for each number of neighbours.

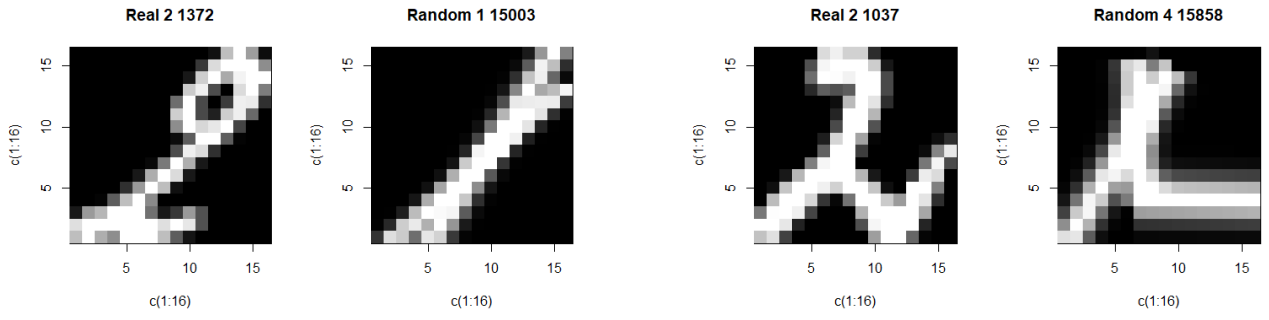
	labels			
results	1	2	4	7
1	993	15	17	12
2	0	908	0	10
4	4	30	976	32
7	3	47	7	946



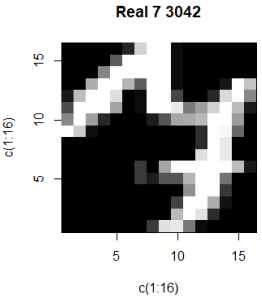
The 2's model had the most misclassifications by far. It was misclassified for all other classes. The can be attributed to the complexity of 2.

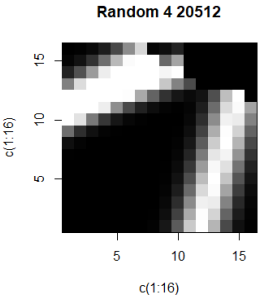
The misclassifications between 2 and other classes can be seen below. The middle stroke appears to be a huge factor when being mistaken for other digits, as can be seen particularly with the 1. The 4 cross stroke is mistaken for the 2's tail. These are issues which had been acknowledged before however due to the level of variance in the model, it was difficult to remove these errors without an overall decrease in performance. Finally, the 7 was able to create the curved effect in the 2.

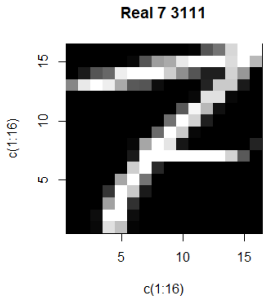


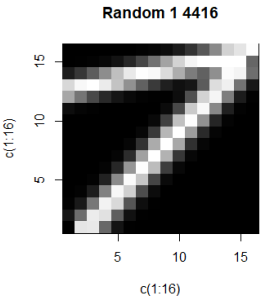


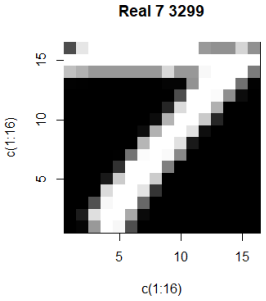
This 7 model was also misclassified with every other class. This is surprising as 7 had a very low mean distance of 1.1 million when classified with itself. This may reveal a flaw in the approach of testing the quality of the models. The example misclassifications exhibit the key issue with the 7 and that is the cross stroke, most misclassifications occur when the cross stroke is not that visible or distinct from other strokes. It appears improving on the quality of cross stroke in the model will reduce most misclassifications.

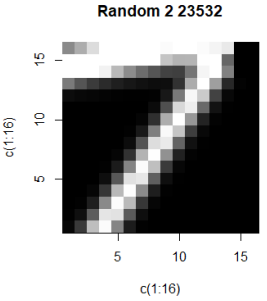




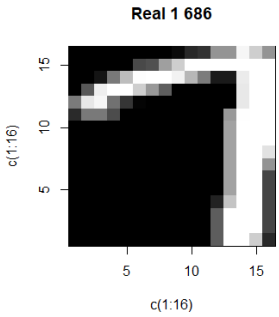


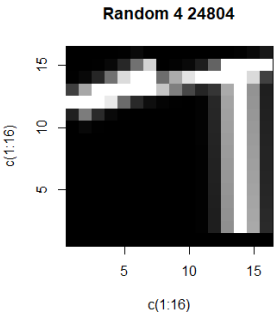


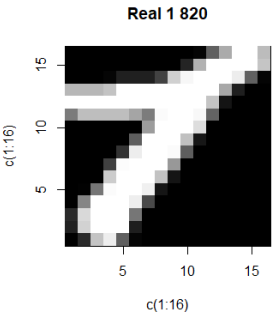


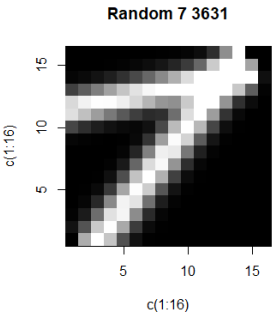


The 1 was most accurately predicted. This performance can be attributed to the simplicity of its model. Of the misclassifications that did occur they revealed flaws which had not been anticipated before. For example, with the four the crossing of stroke1 and stroke2 had not been seen before, this was a new type of error observed.









Conclusion:

We were happy with our models overall. Classifying the 2's and 4's alone yielded an impressively high maximum accuracy although including the 1's and 7's exposed further problems with our model. It is clear there are a few key elements for each digit that if addressed will bring about a significant improvement in performance.

- **1 -** Ensuring the two strokes do not clash will help with misclassifications of 2's
- **2 -** The loop problem and the curvature are two biggest issues to address
- **4 -** The clashing of both stoke1 and stoke3 for misclassification with 2's and crossing of stroke1 and stroke2 for misclassification of 1's and 7's
- **7 -** Making the cross stroke more distinct as it appears to be underrepresented.