Assignment III

1. Introduction

With digitisation continuing to pervade the professional workplace, as well as the world at large, there is a growing desire for computer recognition of text in images. This process of converting text images to a machine-readable format is known as optical image recognition (OCR) (Adobe.com, no date). While text images cannot be searched for the text they contain, OCR allows the computer to recognise it, preventing the need for manual data entry and improving operational productivity. Handwriting image recognition presents a particularly challenging category of OCR as the added complication of individual writing style must be dealt with. As such, legibility becomes an important aspect of handwriting image recognition, *i.e.* how well each character can be distinguished, in the pursuit to optimise character recognition. Nevertheless, the research and methods to help answer them are lacking (Pei Y. and Ye L., 2022). In this undertaking we aim to explore a data set of handwritten digits to firstly determine how well we can recognise and distinguish groups of characters from the others, and followingly to assess the effects of any typographical changes deemed appropriate on our ability to distinguish between the characters.

The data set used in this analysis is a subset of the MNIST dataset (Deng, 2012), which was created by LeCun and others at AT&T Laboratories in 1998. It is made up of a combination of digits handwritten by high school students and US Census Bureau employees respectively. These images represent digits from 0 to 9 and were converted to 28x28 pixel greyscale images, with each pixel taking a value from 0 to 255; hence 0 indicates pure black pixels and 255 pure white, with values in between representing a shade of grey. Each image's pixels may be flattened into a vector of 784 dimensions, with each component taking an integer value in this range, and these vectors make up the feature data. Here we use a subset of 10,000 of MNIST's original 60,000 characters so as not to make the computational expense overly demanding.

In order to achieve our objective of grouping the images in the data set before and after any alterations, however, simple univariate methods are not fit owing to the abundance of features - each image is represented by 784 features described by the feature space Ω, with each feature ω taking an allowed grayscale value:

$$\Omega \subset \mathbb{R}^{784}, \ and \ \omega \in [0, 255], \ \forall \omega \in \Omega .$$

Therefore, a number of multivariate analytical methods will instead be used, namely principal component analysis for dimensionality reduction and K-means clustering to group the characters. This will allow measurement of the clustering accuracy, which can be compared before and after any typographical changes are made to quantify their impact. In this way we will be able to assess whether character legibility can be improved, which may give some suggestion as to which form of a digit is best for achieving better optical image recognition ability.

2. Methods

2.1. Dimensionality Reduction

Although a more complex multivariate analysis behoves the data set because of its large dimensionality, this property also offers the opportunity to use dimensionality reduction techniques, which can help in locating the most crucial features in the data. Such techniques reduce the number of dimensions to a more wieldy size while still preserving important patterns in the data (Tenenbaum et al., 2000). Of the dimensionality reduction techniques of principal component analysis (PCA) and multidimensional scaling (MDS), PCA presented itself as the most appropriate given that our data was not comprised of a distance matrix, which MDS would handle, but a data matrix, with each image being made up of features which describe it without relation to the other images.

PCA uses centred and scaled numerical variables of the data matrix $\boldsymbol{X}$ (which MNIST already includes) to first evaluate a covariance matrix $\boldsymbol{\Sigma} = Cov(\boldsymbol{X})$. This is followingly eigendecomposed to produce its corresponding eigenvalues λ_i, which measure the variation along each PC, and eigenvectors v_i , which define each PC axis' direction. Using the matrix of eigenvectors $\boldsymbol{V}$, we can obtain the PC scores for our original data according to the relation $\boldsymbol{Z} = \boldsymbol{XV}$, and choose to keep a certain fraction of the columns of $\boldsymbol{Z}$ for further analysis. Since PCA leverages variable correlations, it was fundamentally necessary that there be a high degree of correlation between the variables (pixels) in the images in order for the method to work well. This can be justified on the one hand by graphical inspection of a sample of example images shown by *Figure I.*

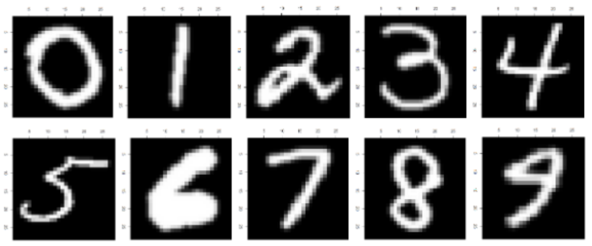


Figure I: from these example images of each digit, we see that if a certain chosen pixel is white/black, its neighbours are very likely to be of the same colour. Thus, significant positive correlation can be inferred to exist in the data between the features. (The exceptions to this are the pixels representing the boundary between the written character and the black background, which tend to have colour opposite to their neighbour on one side. These will show negative correlation)

Furthermore, we could choose to inspect the correlations between a small subset of adjacent pixels as a representative of the larger 28x28 grid, given that the high dimensionality hindered our ability to visualise all variables. The central 3x3 pixel square was selected and the correlations between the nine pixels that were obtained were suggestive of widespread correlation, as shown by *Figure II.*

On the other hand, after having conducted the PCA, we could justify the presence of correlation in our data by examining the proportions of the total variance of the data that each principal component (PC) explains. Correlated data manifests itself in a steep decline in these proportions for successive components/dimensions as a small proportion of principal components can capture a large proportion of the variance. This in turn suggests that many of the dimensions have overlap in the information they contain, and therefore that they are correlated. *Figure III* shows the graphical basis on which both our observation of variable correlation and our determination of how many principal components to retain were made.

The proportion of total variance explained by the i^{th} PC is given by $v_i = \frac{\lambda_i}{\lambda_1+\lambda_2+...+\lambda_p}$, where λ_i denotes the i^{th} eigenvalue of the covariance matrix $\boldsymbol{\Sigma}$, such that the cumulative proportion of variance explained by the first p' PCs is $V_{p'} = v_1 + v_2 +... + v_{p'} = \frac{\lambda_1+...+\lambda_{p'}}{\lambda_1+\lambda_2+...+\lambda_p}$. We can see from the scree plot above that these percentages of the variance explained by each PC dimension, v_i, decline fairly quickly relative to the total 784 dimensions in the data - this is indicative of variable correlation, justifying our

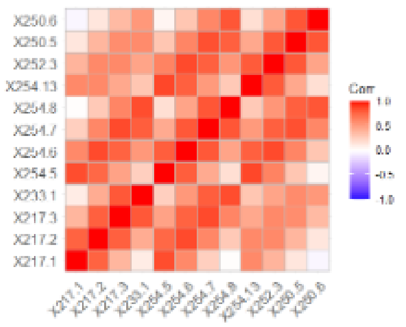


Figure II: correlation plot for the nine pixels in our chosen central 3x3 sample grid. The ubiquity of red indicates positive correlation between the overwhelming majority of pixel pairs, and thus suggests significant correlation in the data set as a whole.

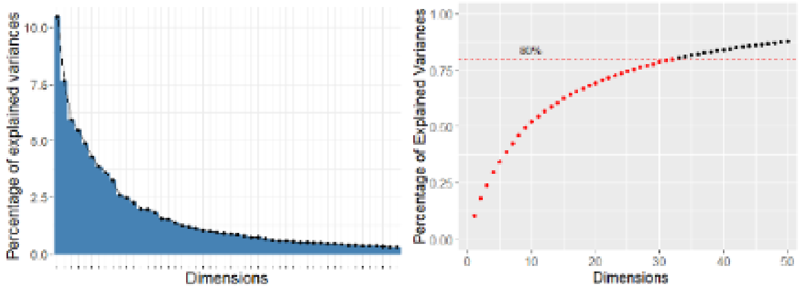


Figure III: The scree plot (left) shows the percentage of the total variance in the original data matrix that is explained by each successive principal component up to the 50th dimension. The first PC explains 10.5% of the variance and this number decreases to less than 1% after the 23rd. The right plot shows the cumulative percentages of variance explained by the PCs - 80% explanation is achieved with 33 PC dimensions.

application of PCA. The cumulative percentages plot suggests that we should retain 33 columns of the principal component score matrix $\boldsymbol{Z}$ to retain 80% of the information, i.e. $p' = 33$. This represented a significant reduction in dimensionality from 784, demonstrating the method's success here.

2.2. Clustering

With the number of principal components chosen, a clustering method could then be used to identify clusters of characters that showed similarity within the PCA-transformed data. Of the common clustering algorithms, K-means clustering was most appropriate here due to our objective of clustering rather than classification, and to the large number of variables in MNIST. This is an unsupervised learning approach in that no *a priori* information about the groups is given to the algorithm, so it must group the data based only on its features. K-means clustering partitions the data into clusters in which each data point belongs to the cluster with the nearest mean (centroid) based on squared Euclidean distances. The algorithm ends when a certain criterion is minimised; this is often the within-cluster sum of squares (WSS):

$$WSS = \sum_{k=1}^{K} \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \overline{x}_{kj})^2,$$

where the term in brackets is the distance between a data point and the centroid of the cluster to which it belongs and is summed over each point in each cluster. The WSS gives a measure of how dispersed the clusters are. When applied to MNIST, the data was clustered into a chosen ten groups, the number of unique digits in the data set, using the data's coördinates in the 33-dimensional space created through PCA with the goal of correctly clustering the characters according to the digit they represent. We then evaluated the clustering result both qualitatively, by looking at the PC transformed data plotted in their assigned clusters on the first two dimensions of the PC axis, and quantitatively, by obtaining the between-cluster sum of squares and also measuring the clustering accuracy, revealing whether legibility was improved.

3. Results

The performance of the clustering was evaluated by comparing the clustering digit assignments to the true labels. However, since the clustering was unsupervised, the algorithm produced clusters of which the identity of the number they represent was unknown. To match the clusters to digit labels, we found the most frequently appearing digit in each cluster and assigned this number to represent the cluster's target. This resulted in ten accuracy scores which are given for reference in the second column of *Table I.* It was found that the most accurately clustered digit was 6, 78% of whose images were assigned by the K-means algorithm into the cluster representing 6. The most poorly clustered digit was 7, which had around 33% accuracy.

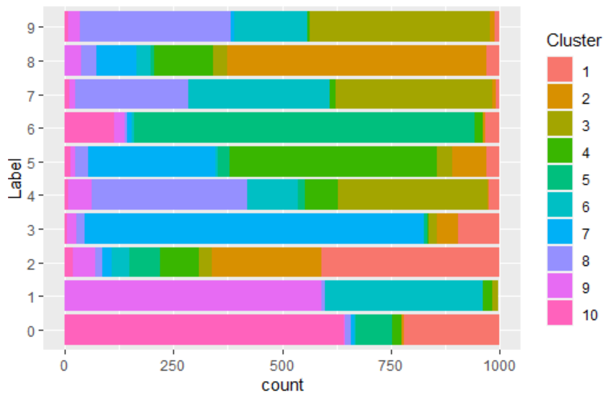
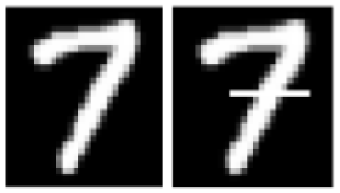


Figure IV: the proportion of each cluster that was assigned to the various digits. We can see from labels 7 and 9 that these digits were often confused as large proportions of them were assigned to clusters 8, 6, and 3, representing the digits 4, 7, and 9. We therefore consider a typographical change to one of these digits.

Although 0 is commonly seen with multiple typographical presentations, *e.g.* a central dot, it was not very frequently confused with other digits, and in fact had one of the highest accuracies, so we decided that this alteration was unnecessary. However, given the frequent confusion of 7 and 9, we may consider one of these for alteration. Since it is more commonly seen that a 7 has multiple typographical forms, we altered them with a central horizontal stroke achieved by manually altering the middle (14th) row as follows:


.

All instances of 7s in the original data set were altered to exhibit this change, and the PCA and K-means clustering were run again to see its effect. Clustering accuracies for each digit were obtained and compared to those calculated using the unaltered data.

Digit	Cluster Accuracy Before	Cluster Accuracy After
0	0.642	0.639
1	0.593	0.633
2	0.408	0.389
3	0.783	0.786
4	0.358	0.344
5	0.479	0.406
6	0.784	0.774
7	0.328	0.335
8	0.597	0.628
9	0.414	0.424
Mean	0.539	0.536

Table I: comparison of accuracies before and after alteration of the 7s. Most of the digits saw little change in accuracy with the greatest difference being $-\ 0.073$ for digit 5. 1 and 8 also showed non-negligible improvements in accuracy.

From this table of results we can see that the typographical alteration did not have any significant impact on the overall clustering accuracy, although the 7s were clustered slightly more accurately, improving from 33% to 34%. Accuracy for the digit 1 improved more markedly, perhaps because the alteration greatly reduced similarity between the 7s and 1s. However, many of the other digits suffered decreased accuracy, and overall the clustering accuracy remained around 54% after the alteration.

To quantitatively measure the effect of the typographical change on the separation between clusters, we calculated the between-cluster sum of squares (BSS), given by

$$BSS = \sum_{k=1}^{K} \sum_{j=1}^{p} |C_k| (\bar{x}_{kj} - \bar{x}_{ij})^2,$$

where the term in brackets is the Euclidean distance between a cluster centroid and the mean of the centroids and its square is summed over all the clusters (Everett B., Hothorn T., 2011). This measure increased by 3.4×10^8 as a result of the change, which represents a small improvement in the cluster separation, and thus image recognition, resulting from the K-means algorithm. This suggests a positive effect of the typographic intervention. The clustering of the transformed data is shown before and after the alteration in *Figure V* below for comparison of cluster separation.

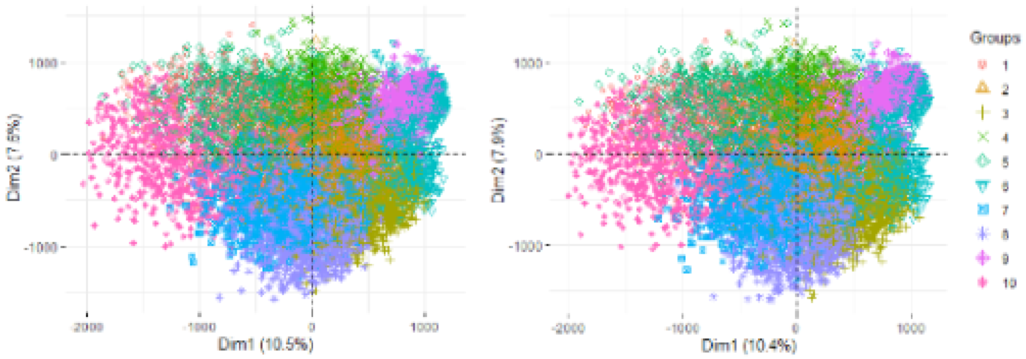


Figure V: comparison of clustering before (left) and after (right) alteration. In both plots the data are shown on the axes of the first two dimensions of the PCs. A slight increase in separation can be seen.

Conclusion

The PCA results led to our decision to keep the first 33 dimensions of the transformed data to which we applied the clustering algorithm. These methods were justified, yet also possessed certain limitations. We implemented PCA on the grounds of high data dimensionality as well as evident variable correlation. However, PCA works best for continuous variables and our data were technically discrete, with the pixels taking an integer value from 0 to 255. Despite this, we were able to approximate the features as continuous due to the large number of allowed values. The use of K-means clustering also has limitations, as it cannot easily identify heterogeneous clusters, i.e. of non-circular shape or of various densities. This may introduce some bias in that certain digits are likely to have denser clusters than others due to the complexity of the digit's form. For example, 0s are likely to exist in a more compact cluster than 3s due to the nature of the digit's relative simplicity. Consequently, the algorithm's achievable accuracy may have been impacted.

Having completed our analysis, the results obtained suggest that we did not have great success in achieving our initial objectives. The overall accuracy of the K-means algorithm's image recognition was not improved as a result of our typographical intervention, despite the proportion of 7s correctly clustered increasing marginally, indicating that legibility of the handwritten characters was not significantly improved. The choice of the explained variance cutoff being 80% may have limited the clustering performance, and greater accuracy may have been achieved by retaining a larger proportion of the PCs. Another factor limiting the clustering's success is that, although the horizontal stroke reduced similarity between the 7s and the 1s, at the same time it increased similarity to other digits such as 4 and 5, leading to the overall accuracy not changing. Thus, it can be seen that deciding on the choice of typographical change presents a challenge in not only distinguishing it from digits with similar appearance, but also in ensuring the change does not increase its similarity to others. This is particularly challenging for the case of handwritten characters due to the inconsistency and diversity arising from human action. Further investigations of this sort are needed to shed light on ways that OCR can be effectively improved in application to handwriting legibility.

References

Adobe.com (no date). What is OCR (optical character recognition)? | Adobe Acrobat. Available at: https://www.adobe.com/acrobat/guides/what-is-ocr.html. (Accessed: April 3, 2023).

Deng, L. (2012). The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, *29*(6), 141–142.

Pei Y. and Ye L. (2022). Cluster Analysis of MNIST Data Set. *J. Phys.: Conf. Ser.* 2181 012035. Available at https://iopscience.iop.org/article/10.1088/1742-6596/2181/1/012035/pdf

Tenenbaum, J.B., de Silva, V. and Langford, J.C. (2000). A Global Geometric Framework for Nonlinear Dimensionality Reduction, *Science*, 290(5500), pp. 2319–2323. Available at: https://doi.org/10.1126/science.290.5500.2319.

Everett B., Hothorn T. (2011). An Introduction to Applied Multivariate Analysis with R. *Springer New York* ISBN: 1441996508. Available at https://link-springer-com.ezproxy.st-andrews.ac.uk/book/10.1007/978-1-4419-9650-3

Multivariate Analysis: Assignment III Code

Alexander Ross

2023-04-03

1. Load libraries and data set

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(ggcorrplot)
```

```
## Loading required package: ggplot2
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
mnist <- read.csv("emnist-mnist-test.csv")
```

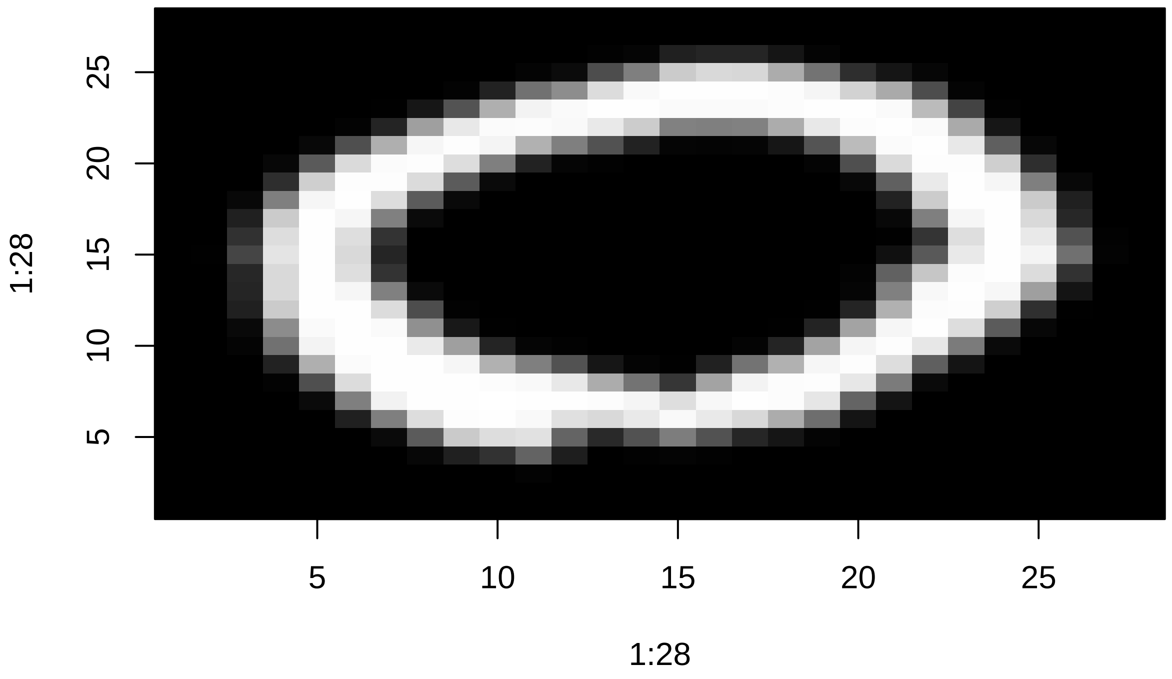
2. Exploratory Analysis

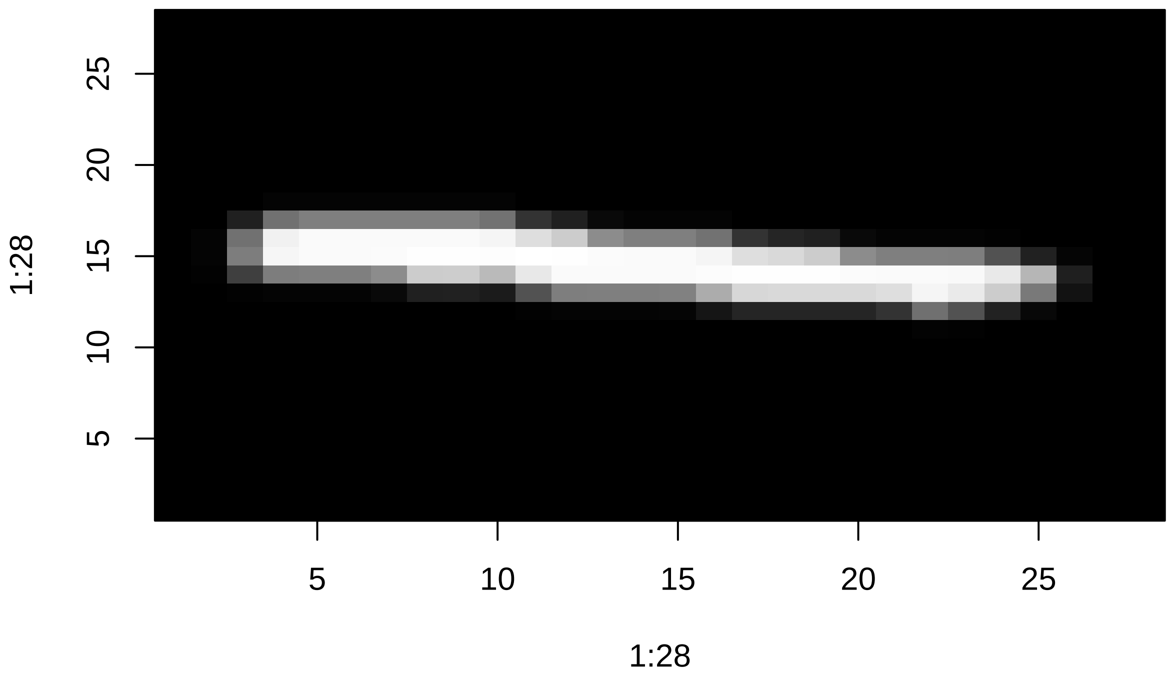
Separate into features and labels

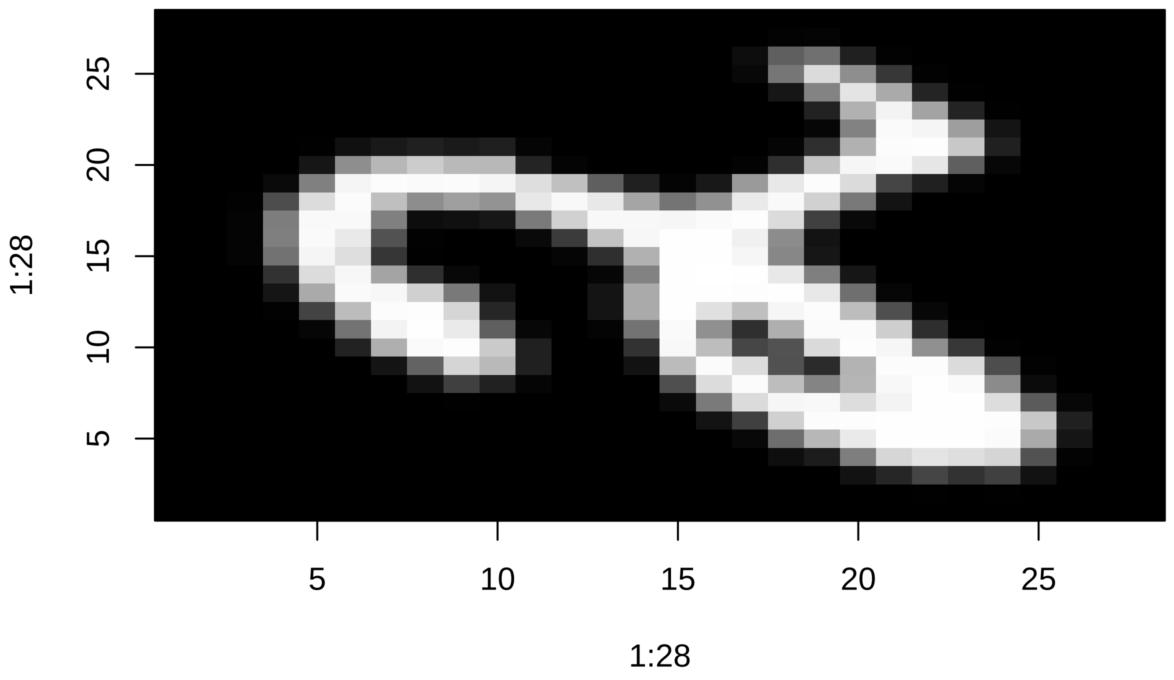
```
mnist_ft = mnist[,-1]
mnist_lb = as.factor(mnist[, 1])
ft_matrix = as.matrix(mnist_ft)
```

Visualise an example character for each digit

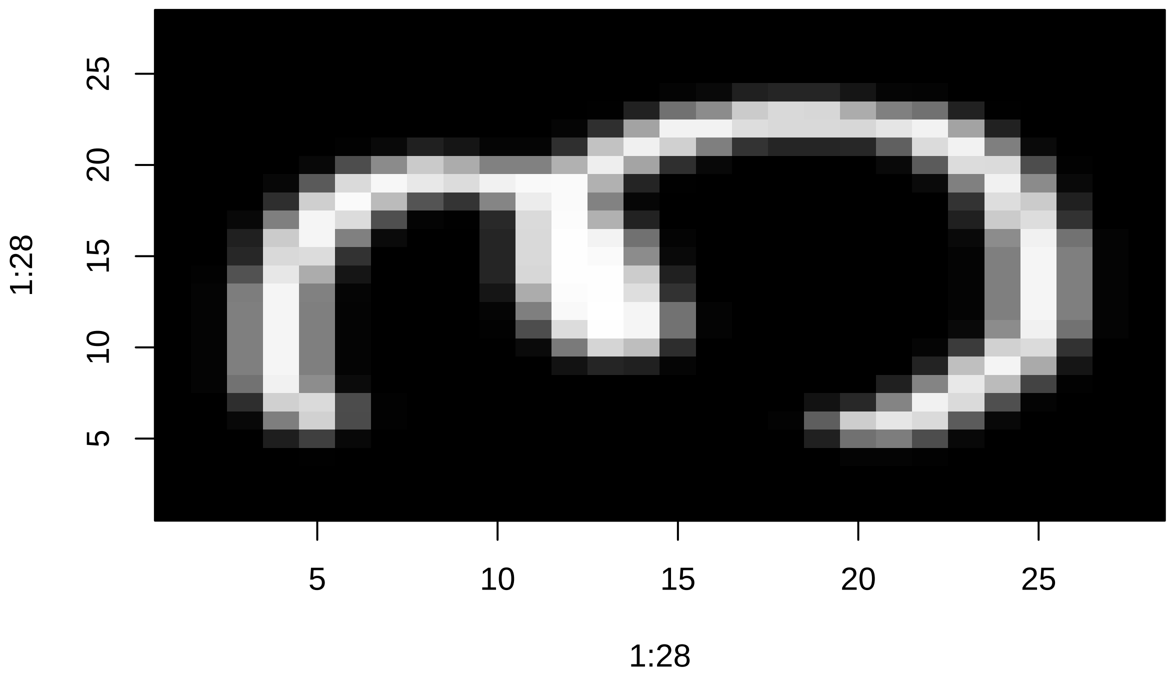
```
ex_digits <- c(5,7,21,19,15,31,14,16,13,35)
for(i in ex_digits){
  vis <- matrix((mnist_ft[i,]), nrow=28, ncol=28)
  vis_num <- apply(vis, 2, as.numeric)
  image(1:28, 1:28, vis_num, col=gray((0:255)/255))
}
```



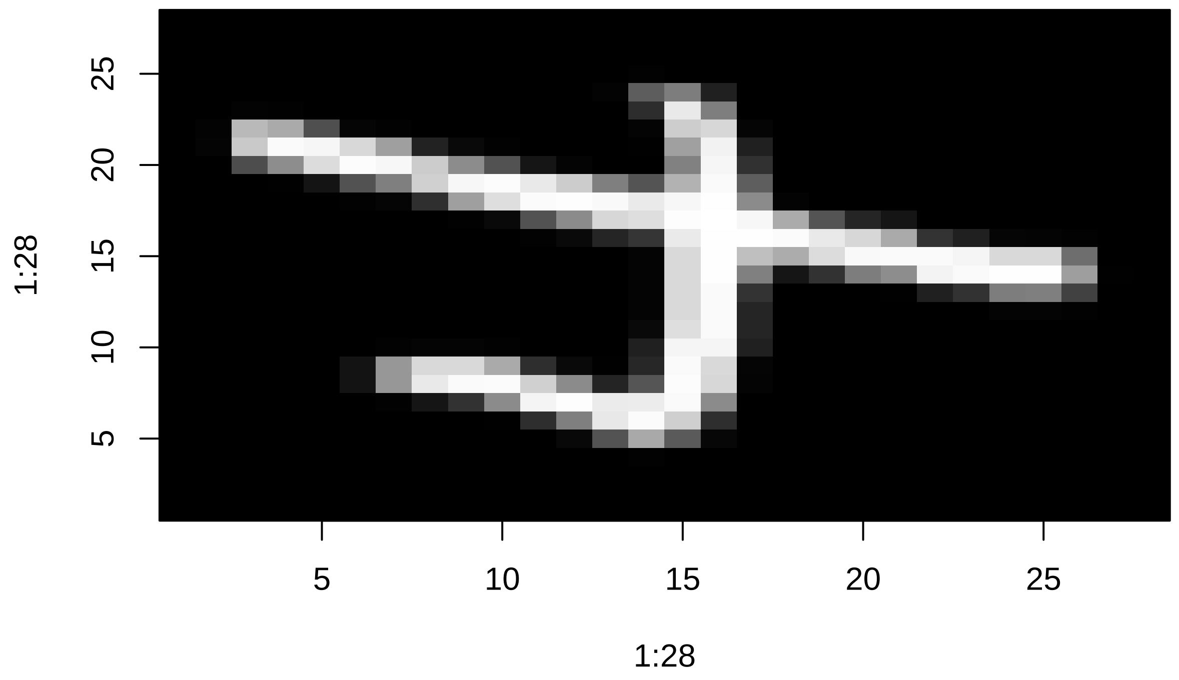

25
20
15
10
5
1:28
5
10
15
20
25
1:28

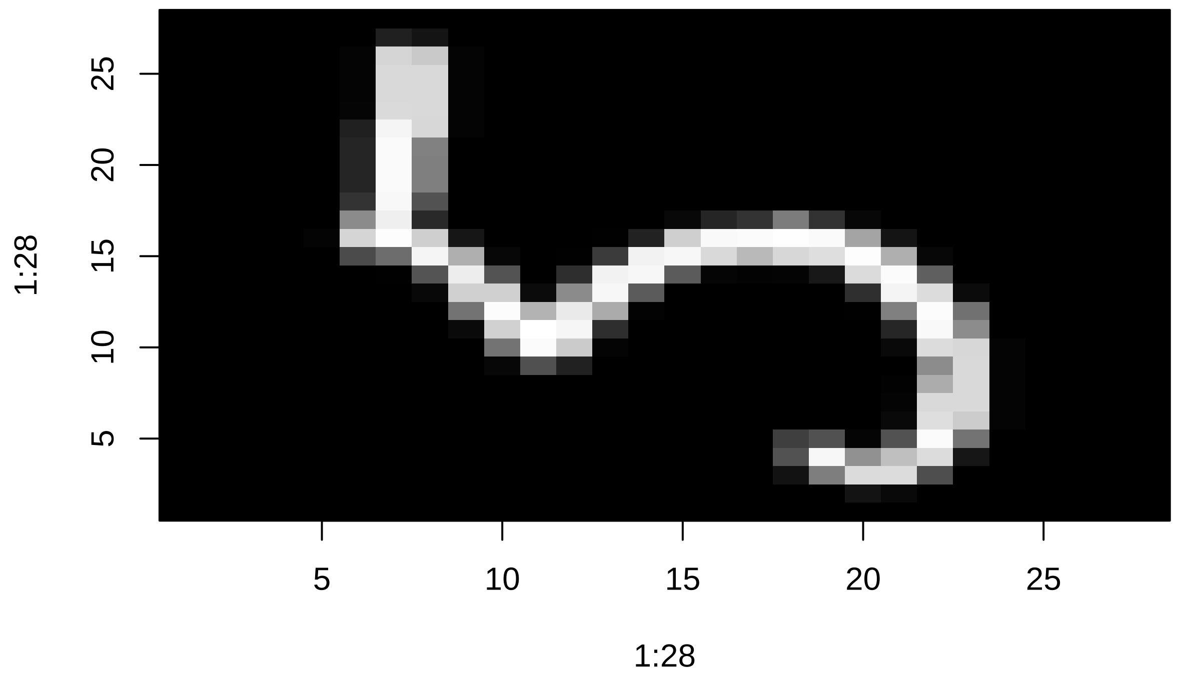


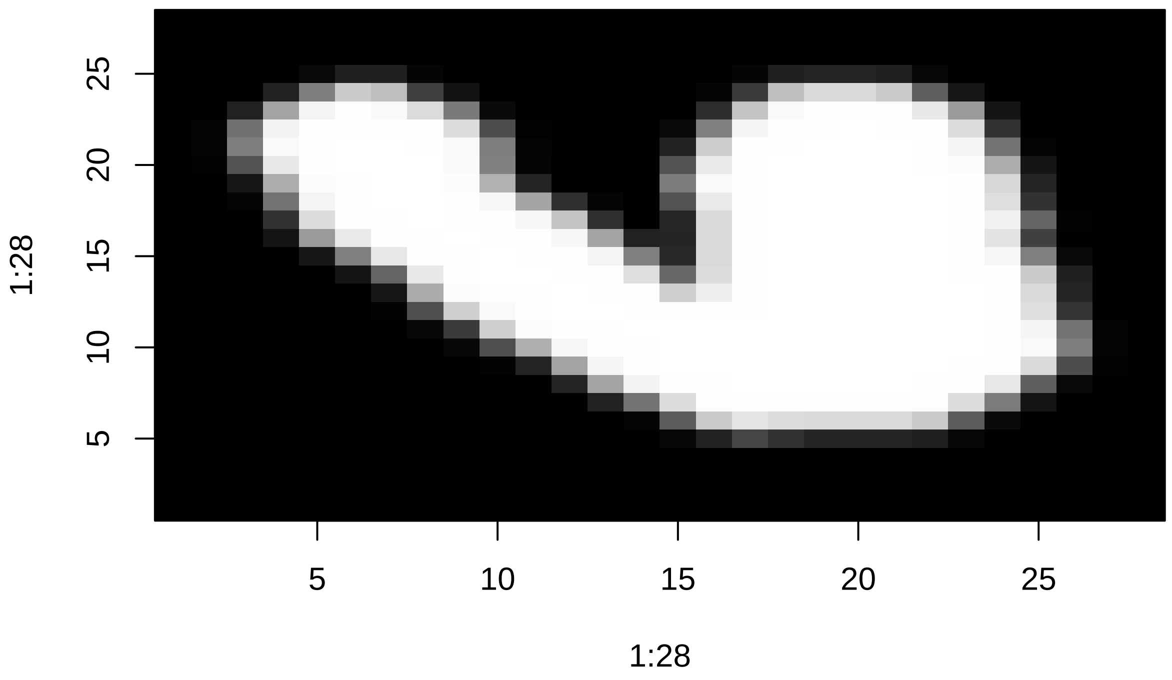
25
20
15
10
5
1:28
5
10
15
20
25
1:28

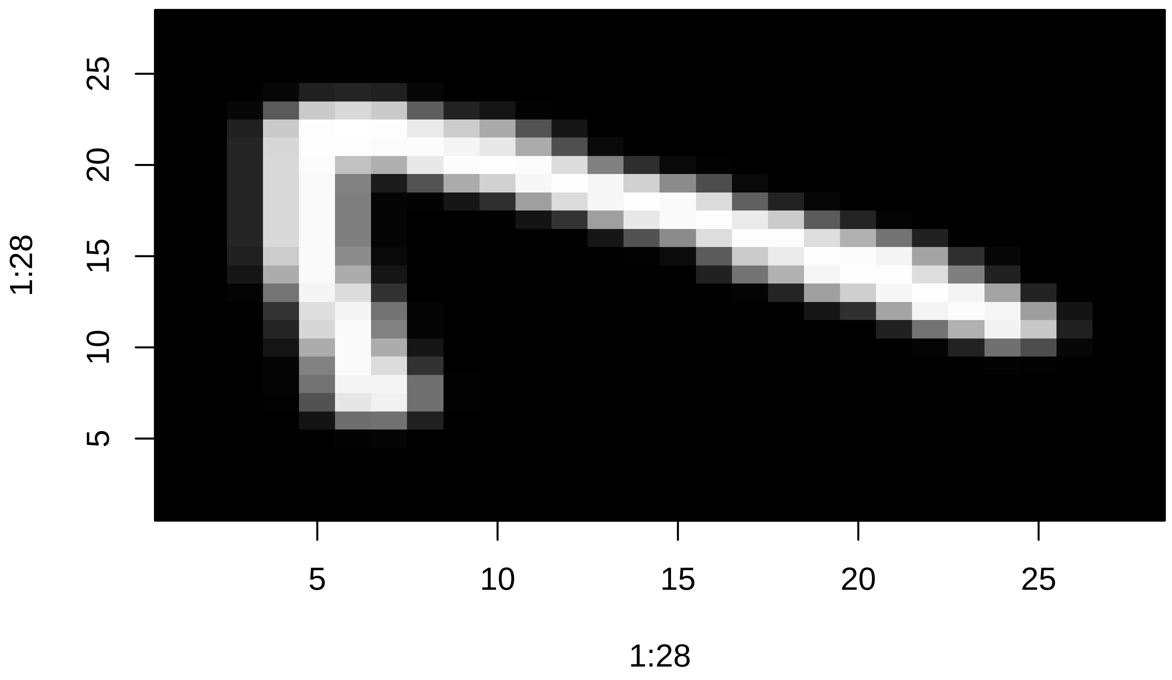


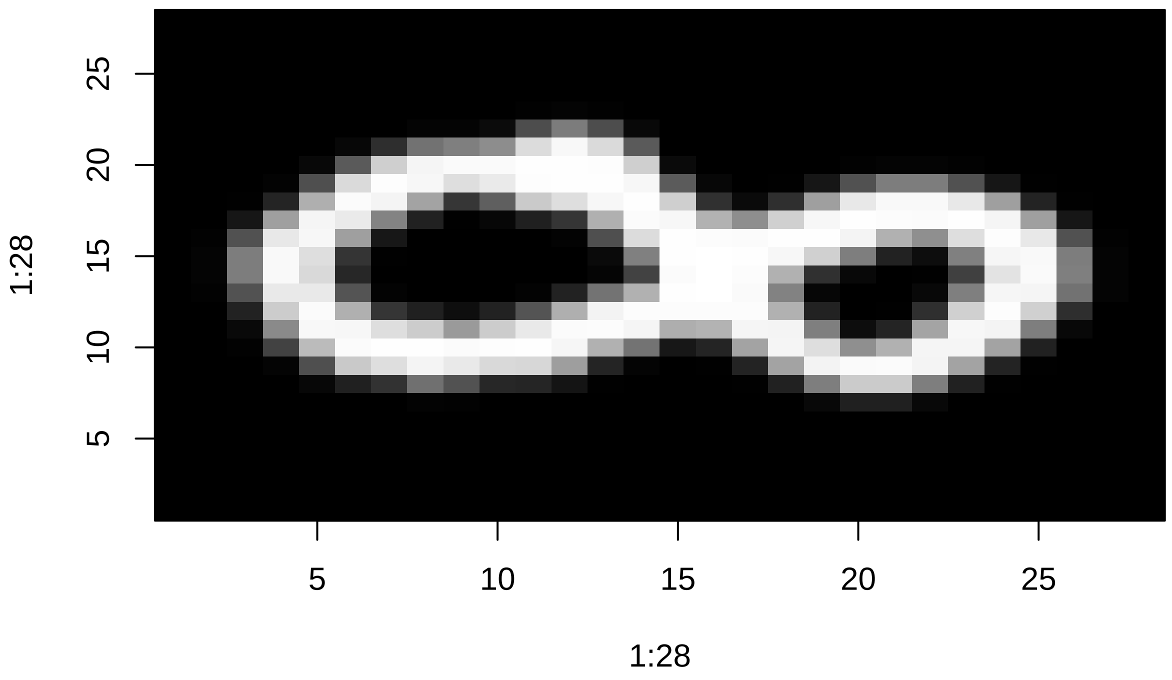
25
20
15
10
5
1:28
5
10
15
20
25
1:28


1:28
25
20
15
10
5
5
10
15
20
25
1:28

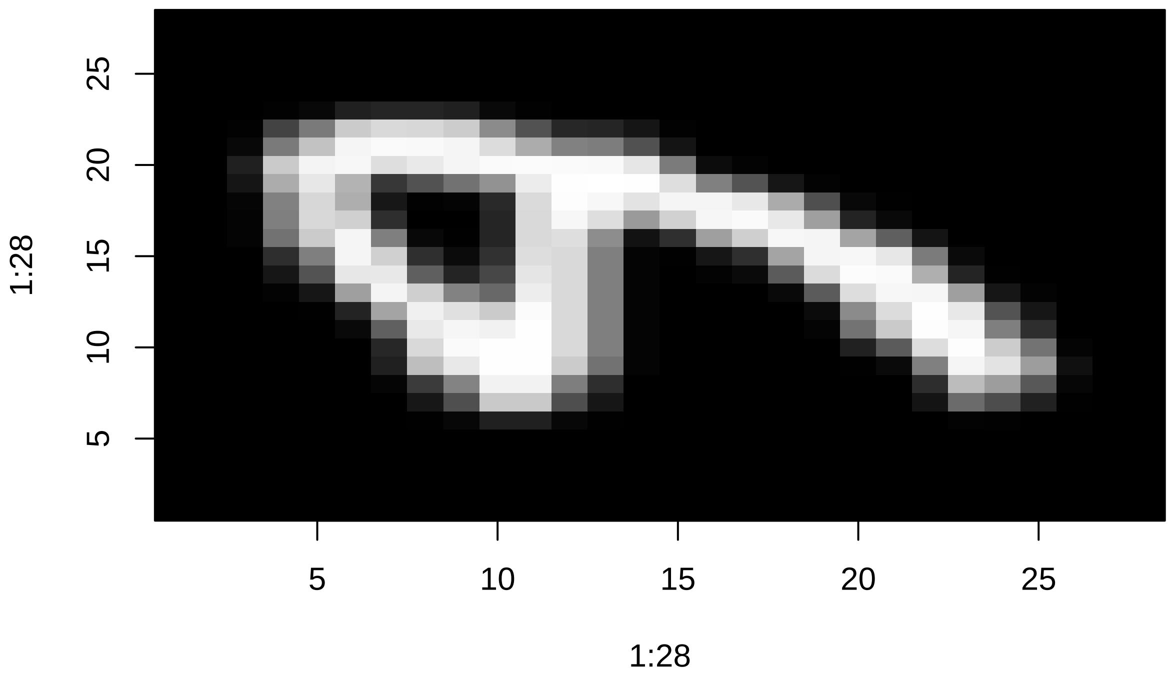

5
10
15
20
25
1:28
5
10
15
20
25
1:28


1:28
5
10
15
20
25
1:28
5
10
15
20
25


25
20
15
10
5
1:28
5
10
15
20
25
1:28

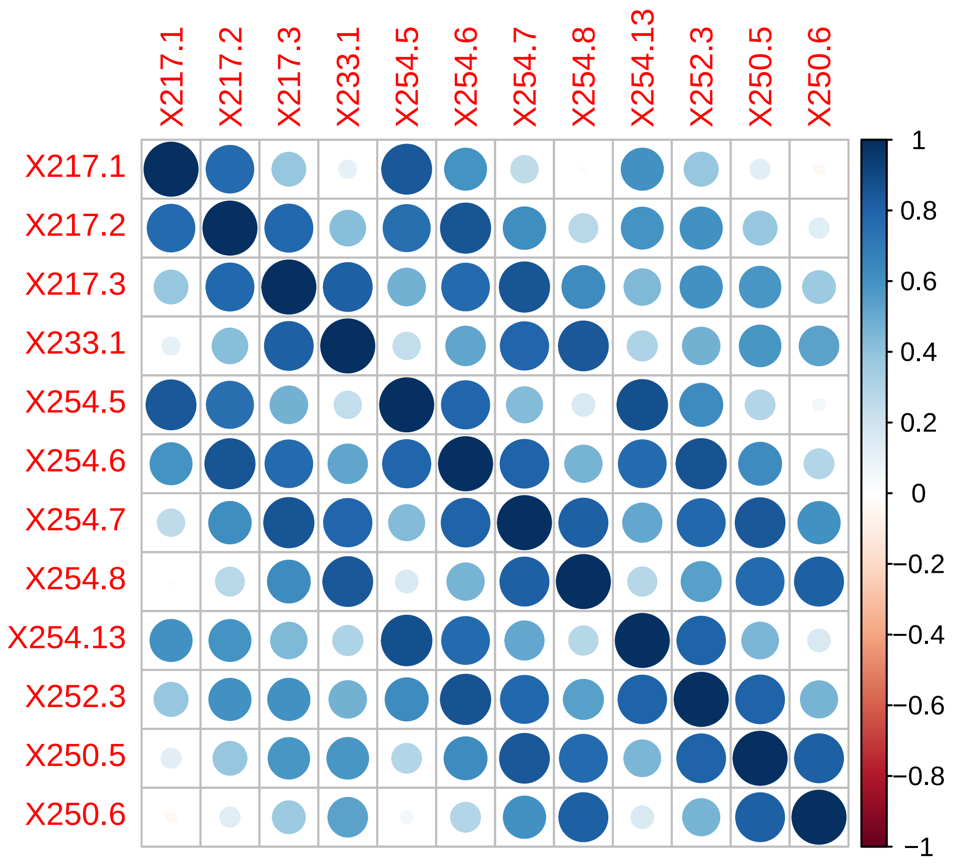


25
20
15
10
5
1:28
5
10
15
20
25
1:28

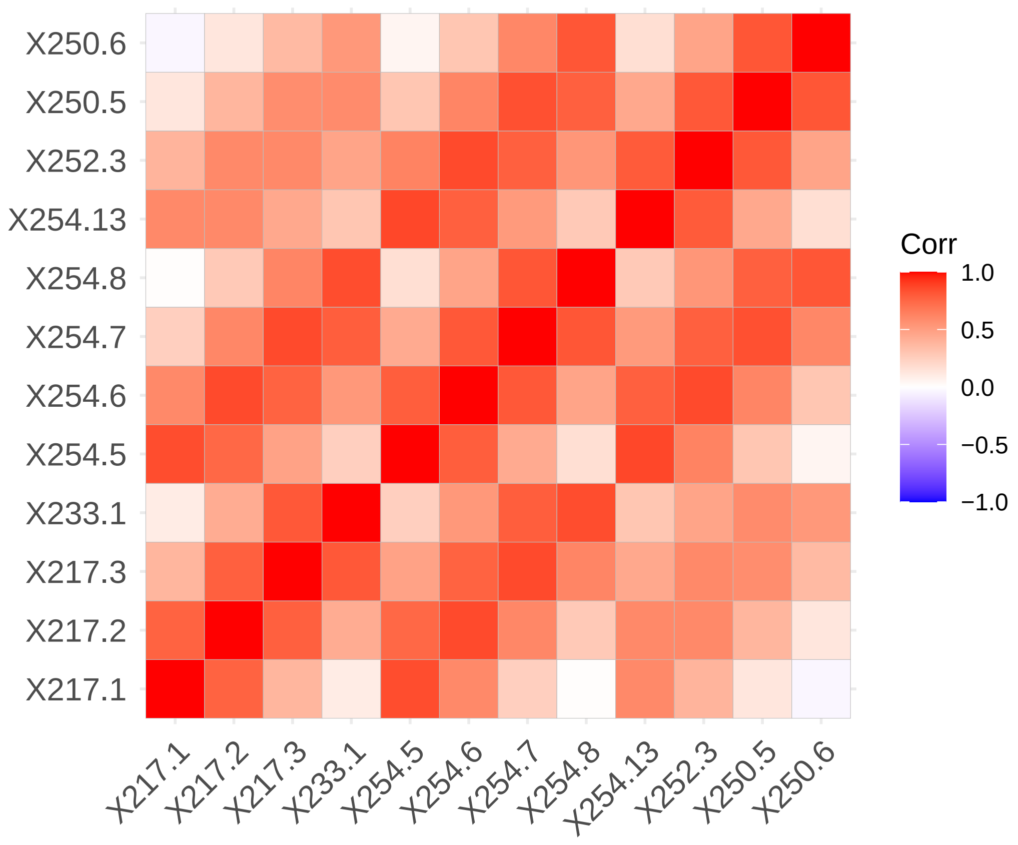


Correlations for sample of 9-pixel square in the centre

```
cormat_reduced <- cor(mnist_ft[,c(350:353, 378:381, 406:409)])
corplot_reduced <- corrplot(cormat_reduced)
```



```
scatmat_reduced <- ggcorrplot(cormat_reduced)
scatmat_reduced
```



3. Dimensionality Reduction

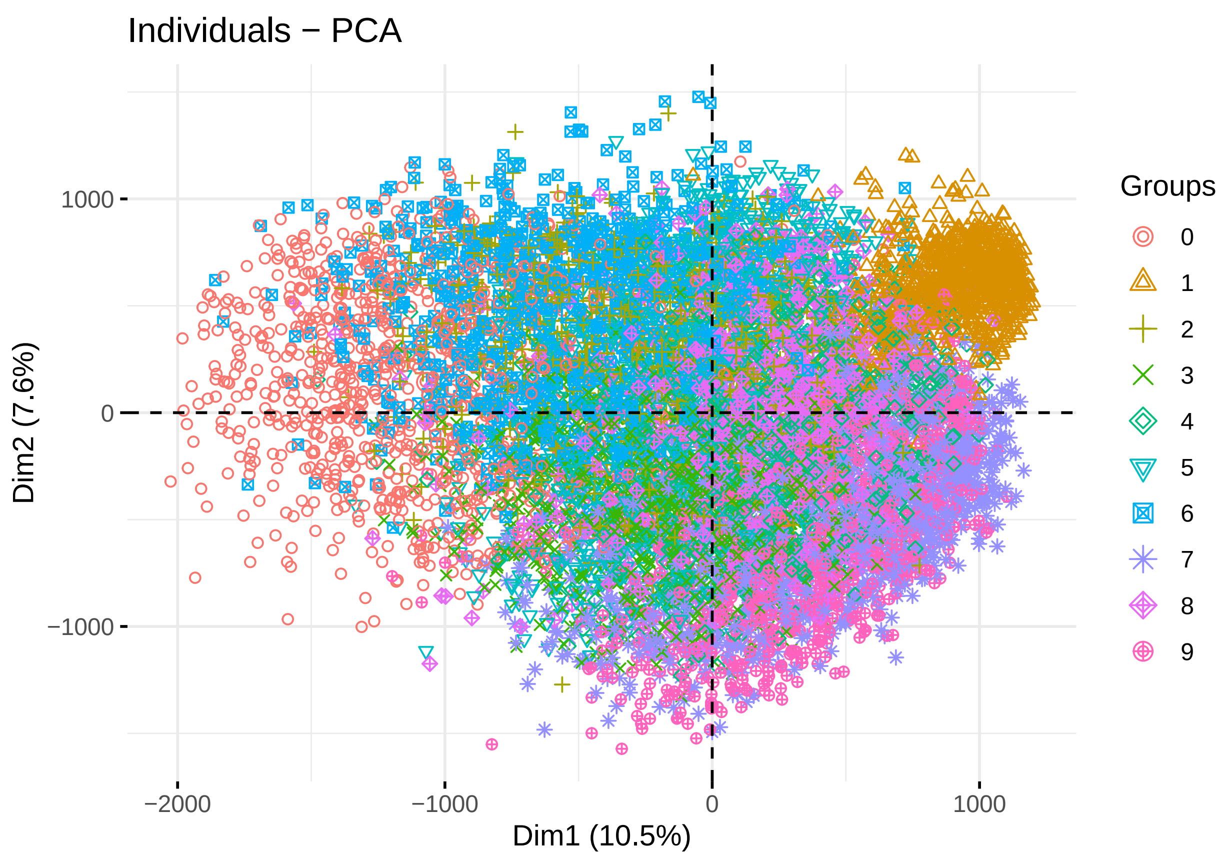
Perform PCA

```
pca = prcomp(ft_matrix, center=T)
print(paste('PCs:', ncol(pca$x)))
```

```
## [1] "PCs: 784"
```

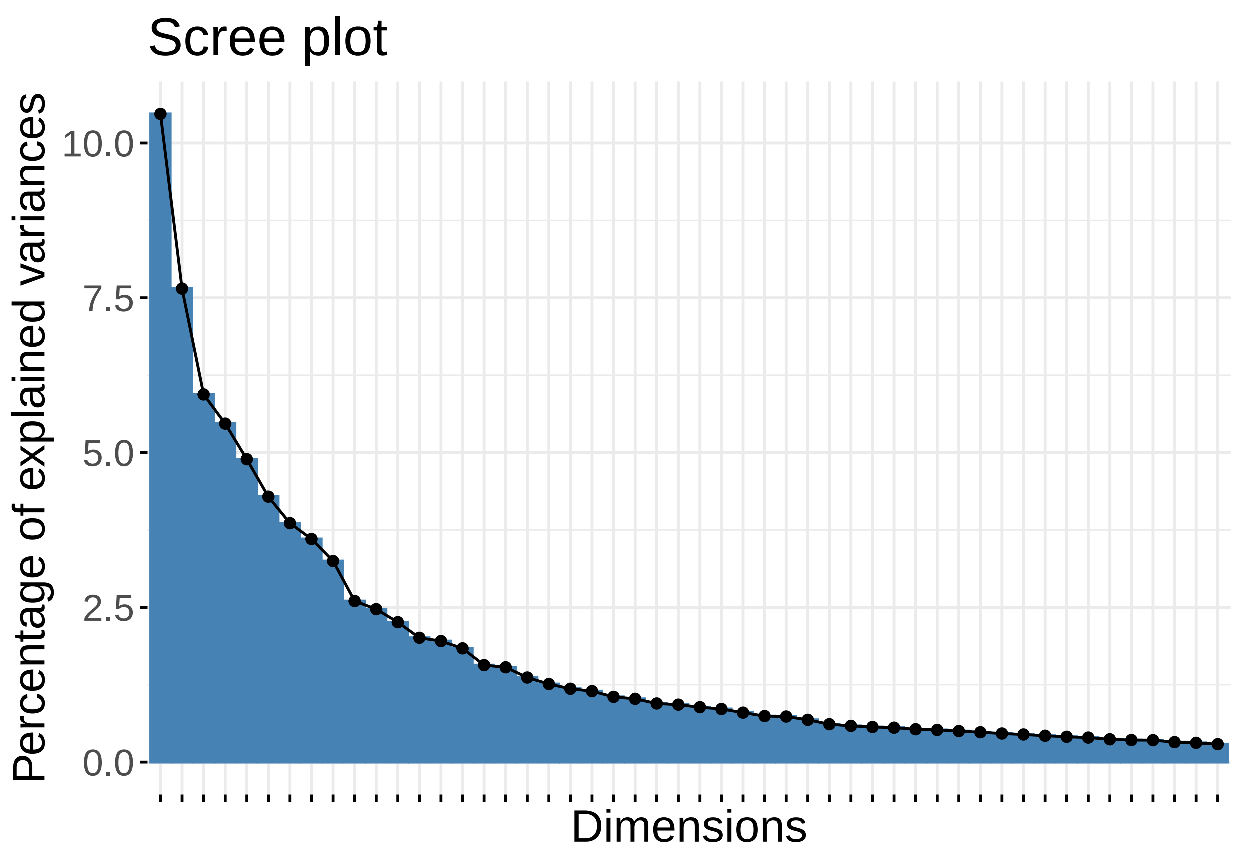
Visualise labels on PC axes

```
fviz_pca_ind(pca, label="none", habillage=as.factor(mnist_lb))
```



Proportions of variance visualisations

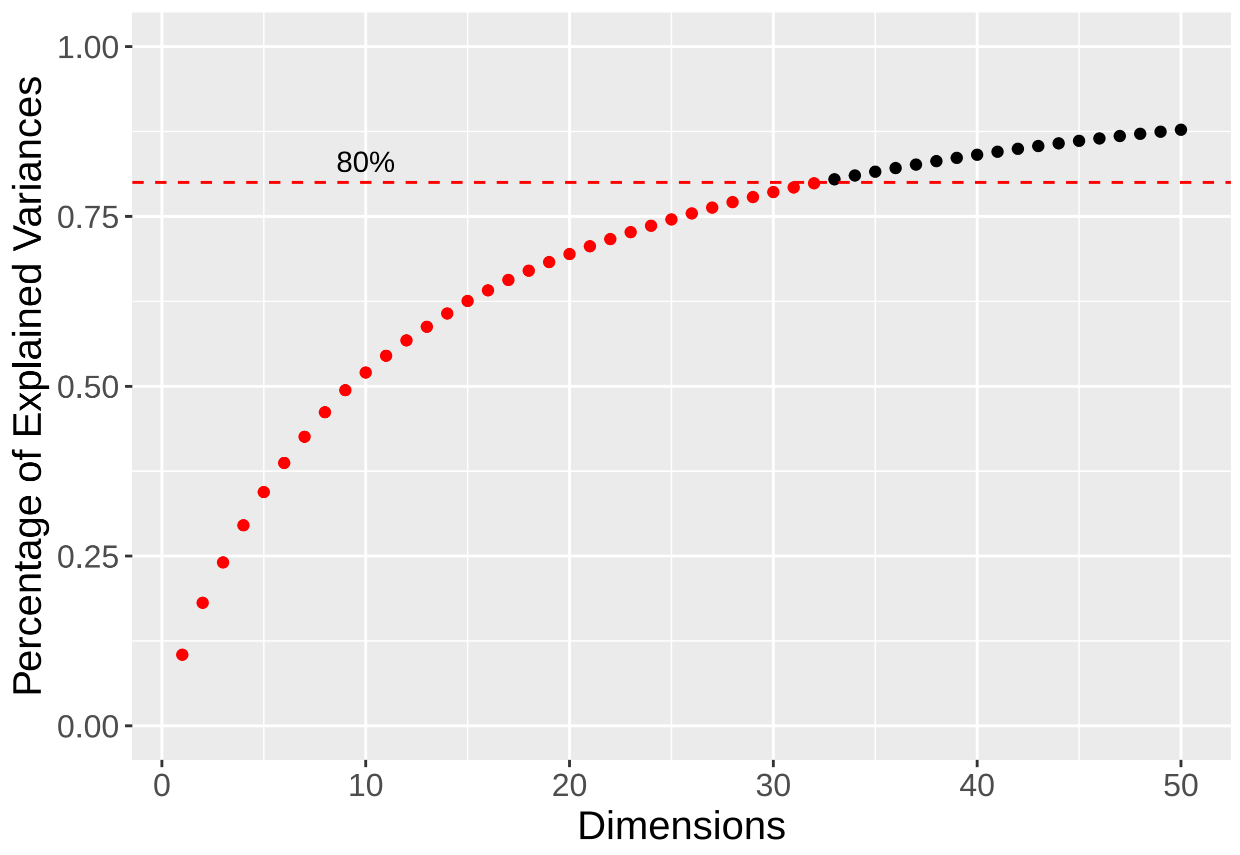
```
scree = fviz_eig(pca, ncp=50, addlabels=F, barfill="steelblue")+
  theme(axis.text.x=element_blank(), text=element_text(size=17))
scree
```



```r
Sigma <- cov(mnist_ft)
eig <- eigen(Sigma)
prop_var <- eig$values/sum(eig$values)

prop_var_df = data.frame(x=c(1:784), y=cumsum(prop_var))
prop_var_df_50 = prop_var_df[c(1:50),]
prop_var_df_50$colour <- ifelse(prop_var_df_50$y < .8, "red", "black")

ggplot(prop_var_df_50) + geom_point(aes(x, y), col=prop_var_df_50$colour) +
  geom_hline(yintercept=0.80, linetype="dashed", col="red") +
  annotate("text", x = 10, y = 0.8, label = "80%", vjust = -0.5) +
  xlab("Dimensions") +
  ylab("Percentage of Explained Variances") + ylim(0,1) +
  theme(text=element_text(size=15))
```



4. Clustering

Scores of the data in the first 33 dimensions of PCA

```
mnist_ft_scores <- pca$x[,c(1:33)]
```

Perform clustering with 10 centers

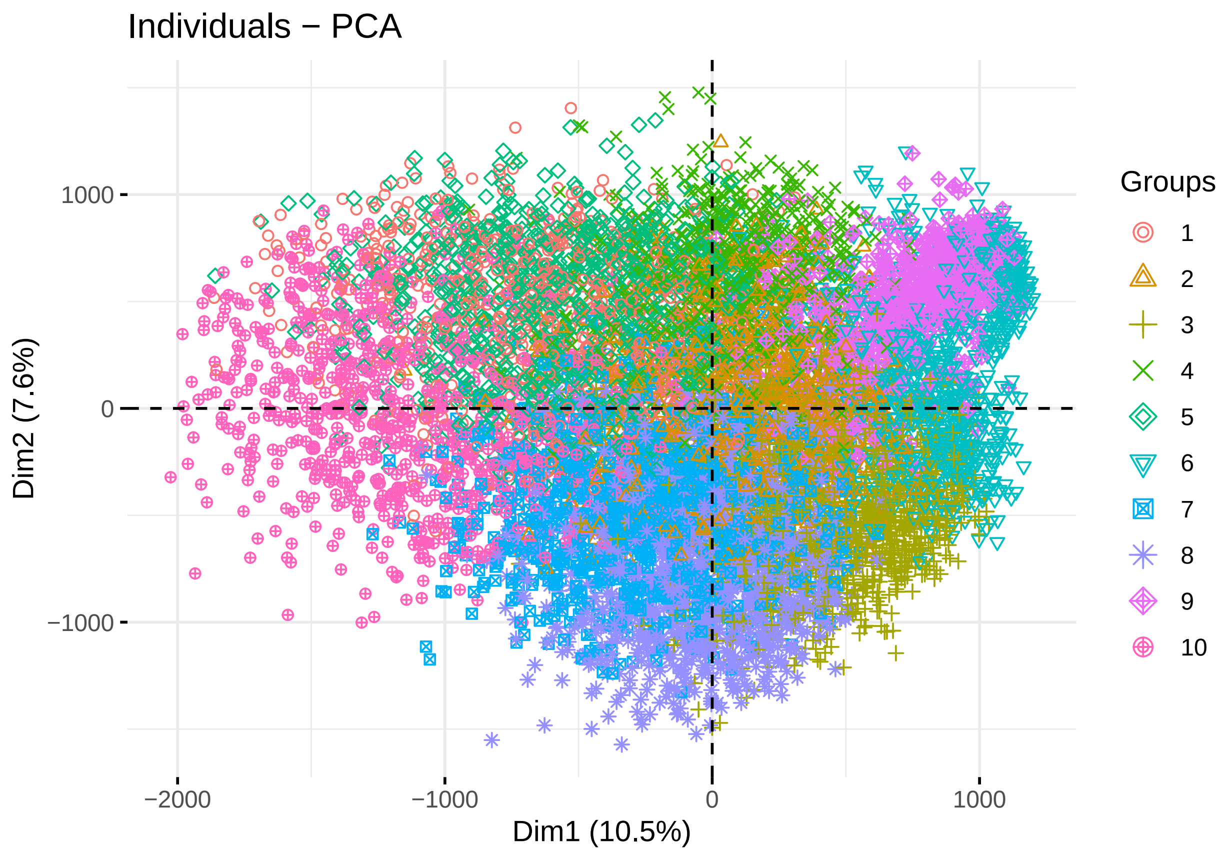
```
set.seed(50)
clustering <- kmeans(x = mnist_ft_scores, centers = 10)
```

Cluster assignments

```
k_means <- factor(clustering$cluster)
```

Plot the clusters on PC axis

```
fviz_pca_ind(pca, label = "none", habillage = k_means)
```

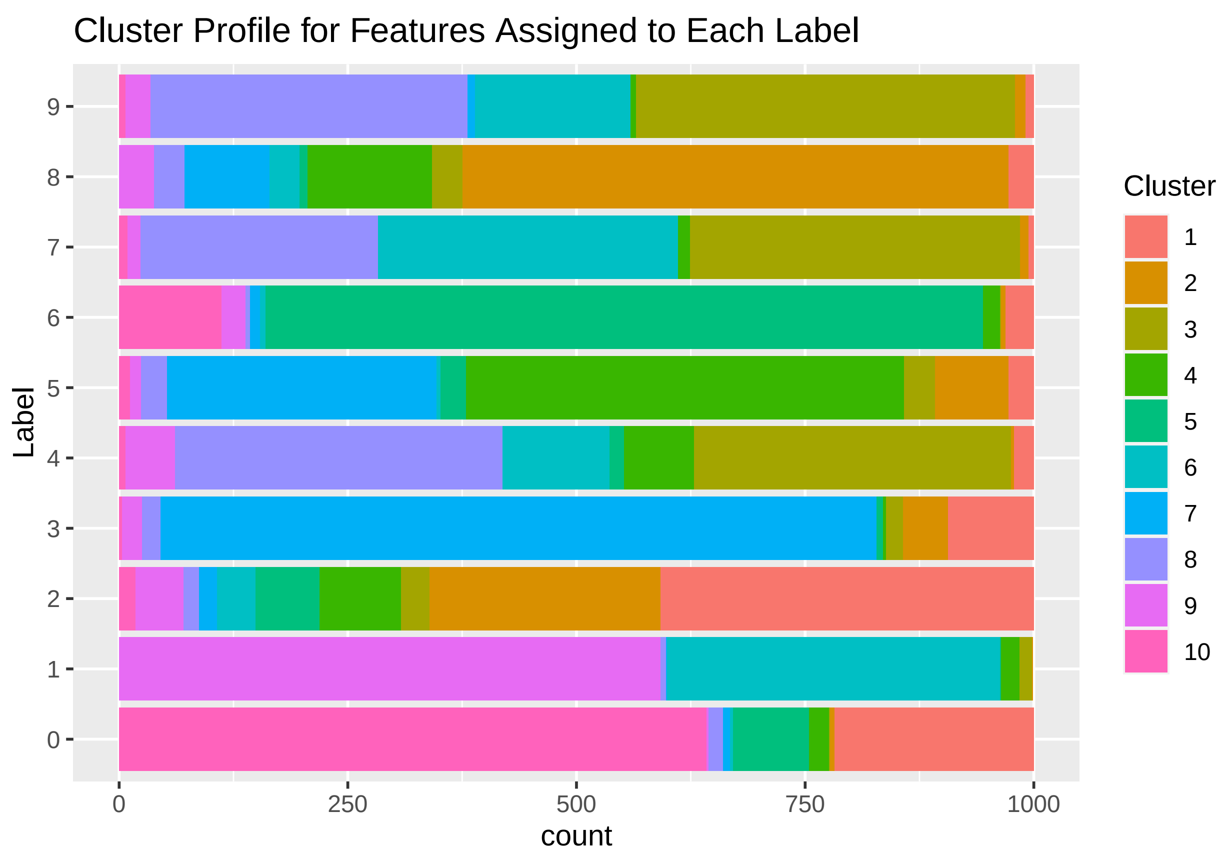


Visualise labels associated with the assignments for each cluster

```
clusters_labels <- data.frame(cbind(k_means, as.factor(mnist_lb)))

ggplot(clusters_labels) + geom_bar(aes(y=mnist_lb, fill=as.factor(k_means))) +
  ggtitle("Cluster Profile for Features Assigned to Each Label") +
  ylim(0.5,10.5) + scale_y_discrete(limits=factor(c(0:9))) + ylab("Label") +
  scale_fill_discrete(name = "Cluster")
```

```
## Scale for y is already present.
## Adding another scale for y, which will replace the existing scale.
```



Evaluate clustering performance

Find the cluster number with the highest frequency among 0s

```
args_0 <- which(mnist_lb %in% 0)
freq0 <- table(k_means[args_0])
most_frequent_0 <- names(which.max(freq0))
```

among 1s

```
args_1 <- which(mnist_lb %in% 1)
freq1 <- table(k_means[args_1])
most_frequent_1 <- names(which.max(freq1))
```

among 2s

```
args_2 <- which(mnist_lb %in% 2)
freq2 <- table(k_means[args_2])
most_frequent_2 <- names(which.max(freq2))
```

among 3s

```
args_3 <- which(mnist_lb %in% 3)
freq3 <- table(k_means[args_3])
most_frequent_3 <- names(which.max(freq3))
```

among 4s

```
args_4 <- which(mnist_lb %in% 4)
freq4 <- table(k_means[args_4])
most_frequent_4 <- names(which.max(freq4))
```

among 5s

```
args_5 <- which(mnist_lb %in% 5)
freq5 <- table(k_means[args_5])
most_frequent_5 <- names(which.max(freq5))
```

among 6s

```
args_6 <- which(mnist_lb %in% 6)
freq6 <- table(k_means[args_6])
most_frequent_6 <- names(which.max(freq6))
```

among 7s

```
args_7 <- which(mnist_lb %in% 7)
freq7 <- table(k_means[args_7])
most_frequent_7 <- names(which.max(freq7))
```

among 8s

```
args_8 <- which(mnist_lb %in% 8)
freq8 <- table(k_means[args_8])
most_frequent_8 <- names(which.max(freq8))
```

among 9s

```
args_9 <- which(mnist_lb %in% 9)
freq9 <- table(k_means[args_9])
most_frequent_9 <- names(which.max(freq9))
```

Since both labels 7 and 9 were assigned cluster 3 most often, we manually set 7 to represent cluster 6, the remaining unidientified cluster:

```
most_frequent_7 = "6"
```

Proportion of correctly clustered characters for each digit

```
accuracy_0 = sum(k_means[args_0]==most_frequent_0)/length(args_0)
accuracy_1 = sum(k_means[args_1]==most_frequent_1)/length(args_1)
accuracy_2 = sum(k_means[args_2]==most_frequent_2)/length(args_2)
accuracy_3 = sum(k_means[args_3]==most_frequent_3)/length(args_3)
accuracy_4 = sum(k_means[args_4]==most_frequent_4)/length(args_4)
accuracy_5 = sum(k_means[args_5]==most_frequent_5)/length(args_5)
accuracy_6 = sum(k_means[args_6]==most_frequent_6)/length(args_6)
accuracy_7 = sum(k_means[args_7]==most_frequent_7)/length(args_7)
accuracy_8 = sum(k_means[args_8]==most_frequent_8)/length(args_8)
accuracy_9 = sum(k_means[args_9]==most_frequent_9)/length(args_9)
accuracy_overall = mean(c(accuracy_0, accuracy_1, accuracy_2,
                          accuracy_3, accuracy_4, accuracy_5,
                          accuracy_6, accuracy_7,accuracy_8, accuracy_9))
```

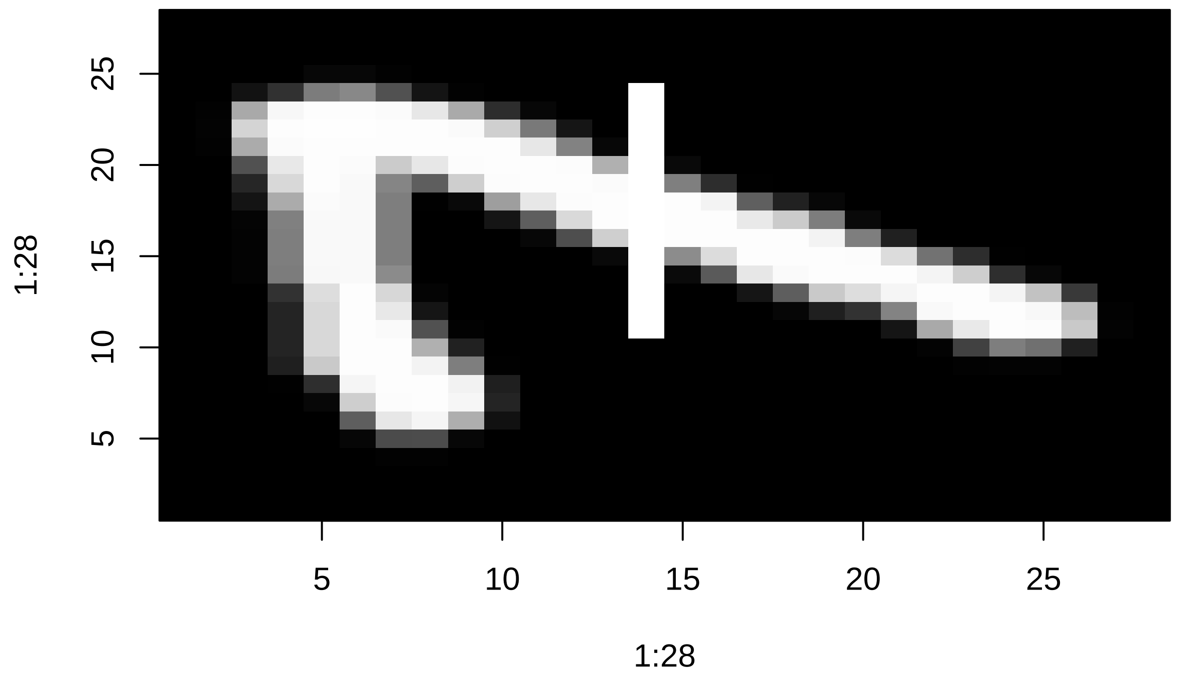
Calculate the BCSS

```
overall_mean <- colMeans(mnist_ft_scores)
bcss <- sum(clustering$size * apply(clustering$centers, 1, function(x) sum((x - overall_mean)^2)))
```

5. Typographical Alteration

Add horizontal stroke to a 7

```
first_7 <- matrix((mnist_ft[3,]), nrow=28, ncol=28)
first_7[14,] <- c(rep(0,10),rep(256,14), rep(0,4))
first_7_num <- apply(first_7, 2, as.numeric)
image(1:28, 1:28, first_7_num, col=gray((0:255)/255))
```



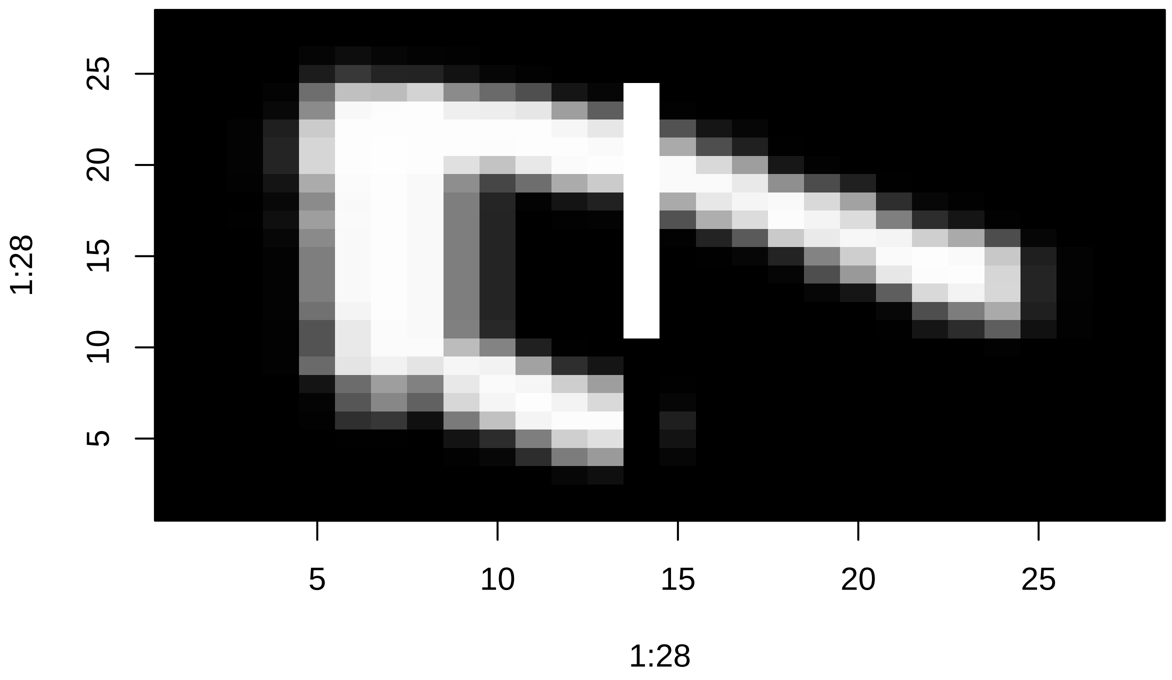
Assign the alteration to all 7s

```
mnist_ft_alt <- mnist_ft

for (i in args_7) {
  ith_7 <- matrix((mnist_ft[i,]), nrow=28, ncol=28)
  ith_7[14,] <- c(rep(0,10),rep(256,14), rep(0,4))
  ith_7_num <- as.vector(apply(ith_7, 2, as.numeric))
  mnist_ft_alt[i,] <- ith_7_num
}
```

Visualise a 7 to check alteration success

```
check_7 <- matrix((mnist_ft_alt[25,]), nrow=28, ncol=28)
check_7_num <- apply(check_7, 2, as.numeric)
image(1:28, 1:28, check_7_num, col=gray((0:255)/255))
```



6. Rerun the analysis for the altered data

```
ft_alt_matrix = as.matrix(mnist_ft_alt)
```

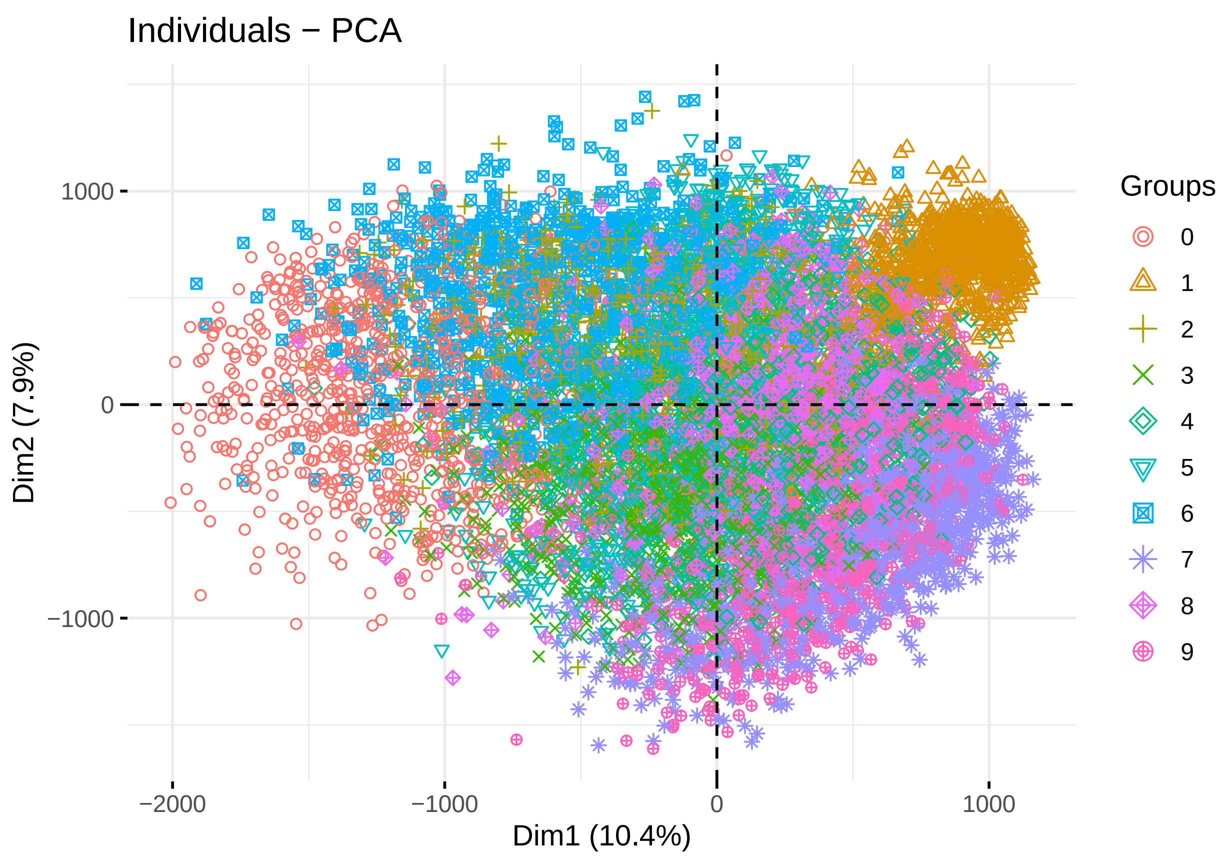
Perform PCA

```
pca_alt = prcomp(ft_alt_matrix, center=T)
print(paste('PCs:', ncol(pca_alt$x)))
```

```
## [1] "PCs: 784"
```

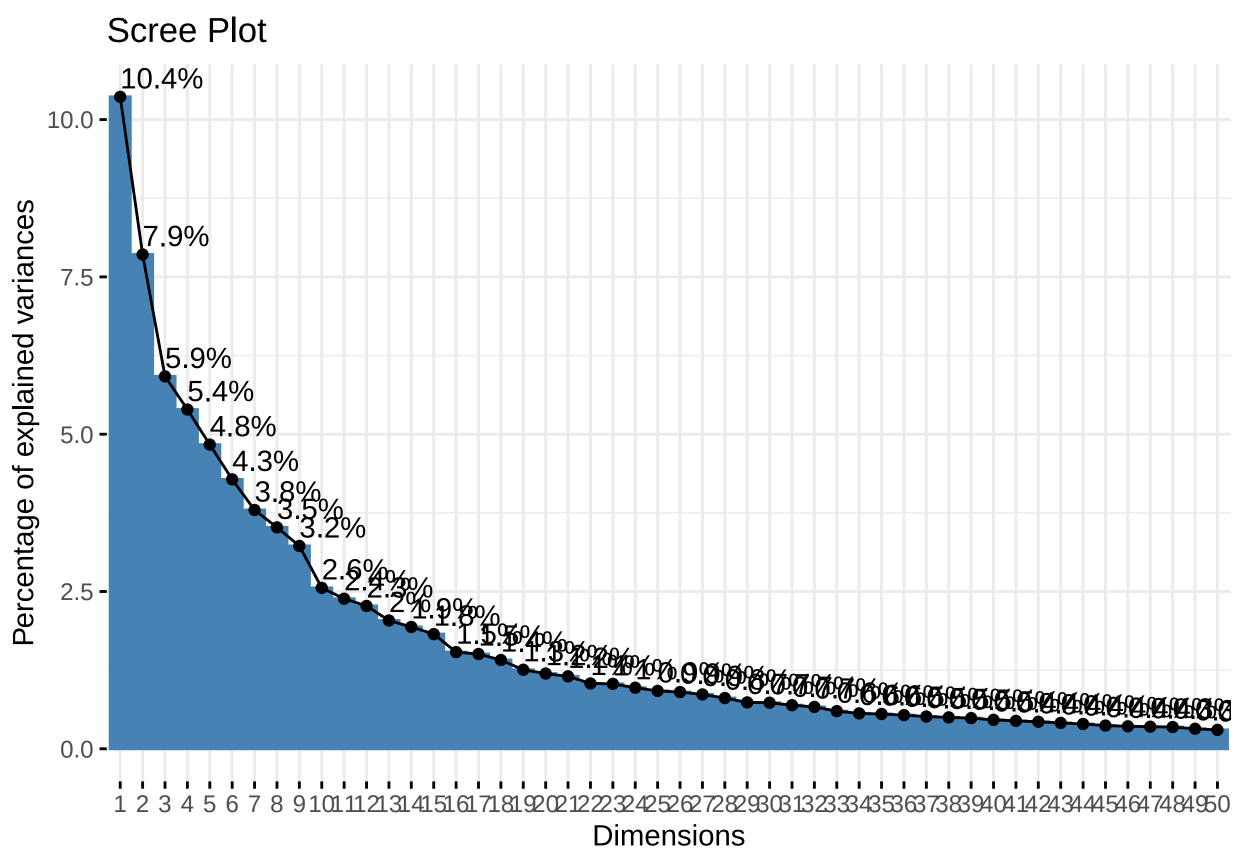
Visualise labels on PC axes

```
fviz_pca_ind(pca_alt, label="none", habillage=as.factor(mnist_lb))
```



Proportions of variance visualisations

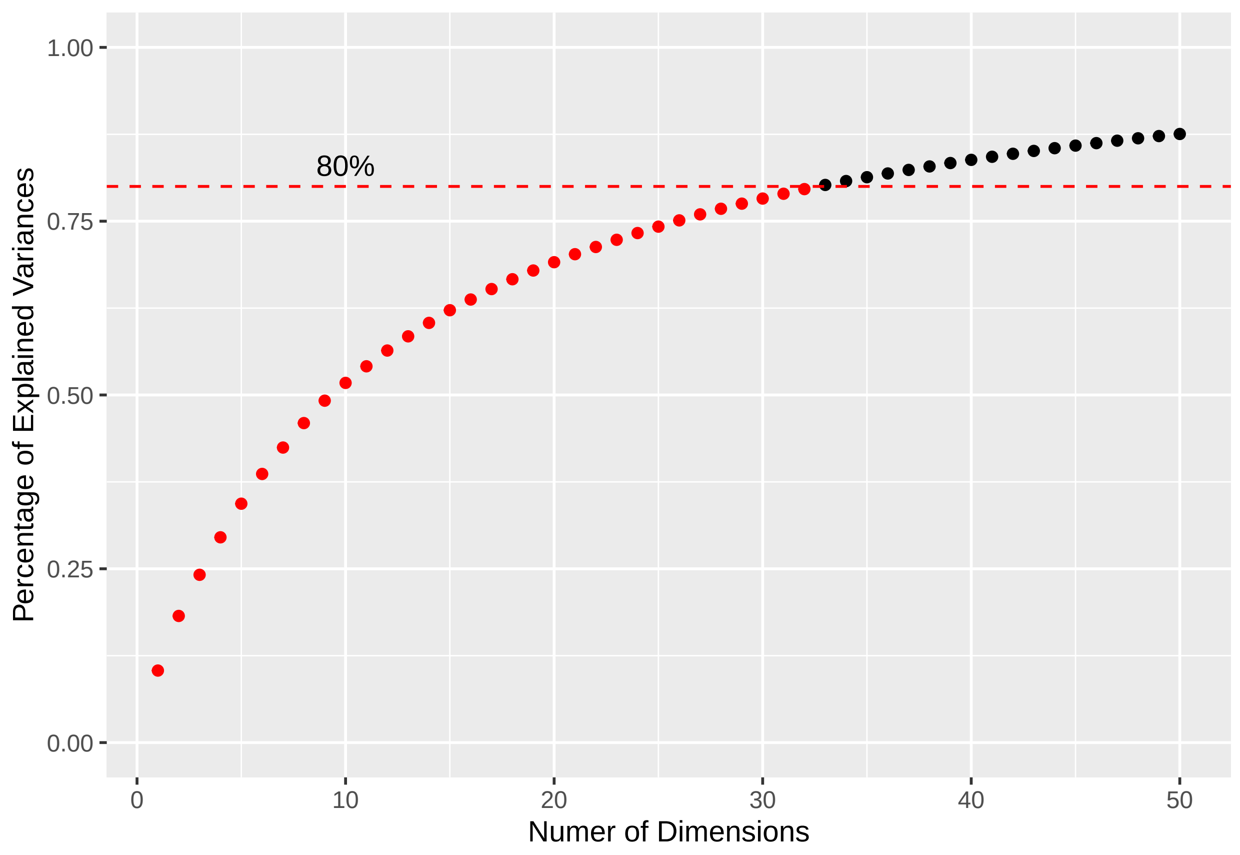
```
scree_alt = fviz_eig(pca_alt, ncp=50, addlabels=T, main='Scree Plot')
scree_alt
```



```
Sigma_alt <- cov(mnist_ft_alt)
eig_alt <- eigen(Sigma_alt)
prop_var_alt <- eig_alt$values/sum(eig_alt$values)

prop_var_alt_df = data.frame(x=c(1:784), y=cumsum(prop_var_alt))
prop_var_alt_df_50 = prop_var_alt_df[c(1:50),]
prop_var_alt_df_50$colour <- ifelse(prop_var_alt_df_50$y < .8, "red", "black")

ggplot(prop_var_alt_df_50) +
  geom_point(aes(x, y), col=prop_var_alt_df_50$colour) +
  geom_hline(yintercept=0.80, linetype="dashed", col="red") +
  annotate("text", x = 10, y = 0.8, label = "80%", vjust = -0.5) +
  xlab("Numer of Dimensions") +
  ylab("Percentage of Explained Variances") + ylim(0,1)
```



Clustering

Scores of the data in the first 33 dimensions of PCA

```
mnist_ft_alt_scores <- pca_alt$x[,c(1:33)]
```

Perform clustering on altered data with 10 centers

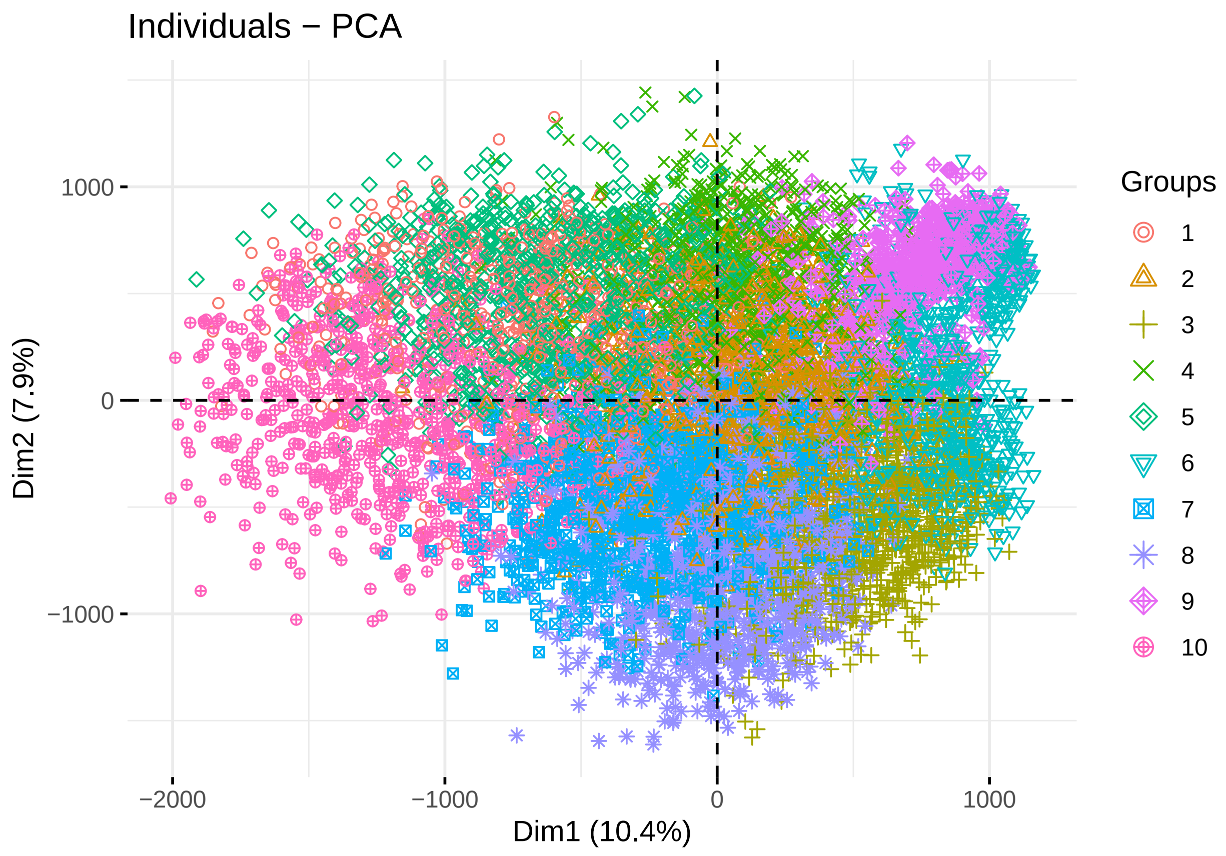
```
set.seed(50)
clustering_alt = kmeans(x = mnist_ft_alt_scores, centers = 10)
```

Cluster assignments

```
k_means_alt = clustering_alt$cluster
```

Plot the clusters

```
fviz_pca_ind(pca_alt, label = "none", habillage = k_means_alt)
```



Evaluate clustering performace on altered data

proportion of correctly clustered characters for each digit

```
accuracy_0_alt = sum(k_means_alt[args_0]==most_frequent_0)/length(args_0)
accuracy_1_alt = sum(k_means_alt[args_1]==most_frequent_1)/length(args_1)
accuracy_2_alt = sum(k_means_alt[args_2]==most_frequent_2)/length(args_2)
accuracy_3_alt = sum(k_means_alt[args_3]==most_frequent_3)/length(args_3)
accuracy_4_alt = sum(k_means_alt[args_4]==most_frequent_4)/length(args_4)
accuracy_5_alt = sum(k_means_alt[args_5]==most_frequent_5)/length(args_5)
accuracy_6_alt = sum(k_means_alt[args_6]==most_frequent_6)/length(args_6)
accuracy_7_alt = sum(k_means_alt[args_7]==most_frequent_7)/length(args_7)
accuracy_8_alt = sum(k_means_alt[args_8]==most_frequent_8)/length(args_8)
accuracy_9_alt = sum(k_means_alt[args_9]==most_frequent_9)/length(args_9)
accuracy_overall_alt = mean(c(accuracy_0_alt, accuracy_1_alt, accuracy_2_alt,
                          accuracy_3_alt, accuracy_4_alt, accuracy_5_alt,
                          accuracy_6_alt, accuracy_7_alt,accuracy_8_alt,
                          accuracy_9_alt))
```

Quantify change in cluster separation

```
overall_mean_alt <- colMeans(mnist_ft_alt_scores)
bcss_alt <- sum(clustering_alt$size * apply(clustering_alt$centers,1,function(x)
                                          sum((x - overall_mean_alt)^2)))
bcss_diff <- bcss_alt-bcss
```