# Lecture 6: Unsupervised Learning

Alan Lee

Department of Statistics
STATS 769 Lecture 6

August 5, 2015

# Outline

# Today's agenda

In the next two lectures we present a discussion of unsupervised learning, and look at some methods for exploring the structure of multivariate data, through clustering algorithms and through dimension reduction ideas.

Today we will cover a couple of clustering methods and one standard dimension reduction method - principal components, and explore some further methods next lecture.

We will use some netball data and some image data as running examples.

# Unsupervised learning

- The term *unsupervised learning* describes the situation where we want to understand the structure of a data set, without dividing the variables into inputs and outputs. We will look at two aspects of this, namely visualization, where we try to draw a picture of the data, and clustering, where we look for groups of similar observations.
- It is very helpful to be able to draw a picture of our data and visualize the structure directly. This is easy for two or three variables, as we can draw scatterplots, boxplots and so on.

# Unsupervised learning (cont)

- Unfortunately it is not possible to draw scatterplots directly in more than three dimensions, so we need clever ways to get around the limitations imposed by our three-dimensional world. We will look at a few ways of doing this, which involve representing a high-dimensional data set in fewer dimensions.
- Sometimes the data can be grouped into clusters, groups of similar observations. If we regard our data points a points in space (think of a high-dimensional scatterplot) we are trying to see if the points appear as clusters of close points, and identify which points belong to each cluster.

# Unsupervised learning: methods

Methods for visualization:

- Principal components - today
- Multidimensional scaling - next lecture
- Self-organizing maps - next lecture

Methods for clustering:

- $K$-means - today
- Hierarchical clustering - next lecture
- Gaussian mixtures - next lecture

# Visualization

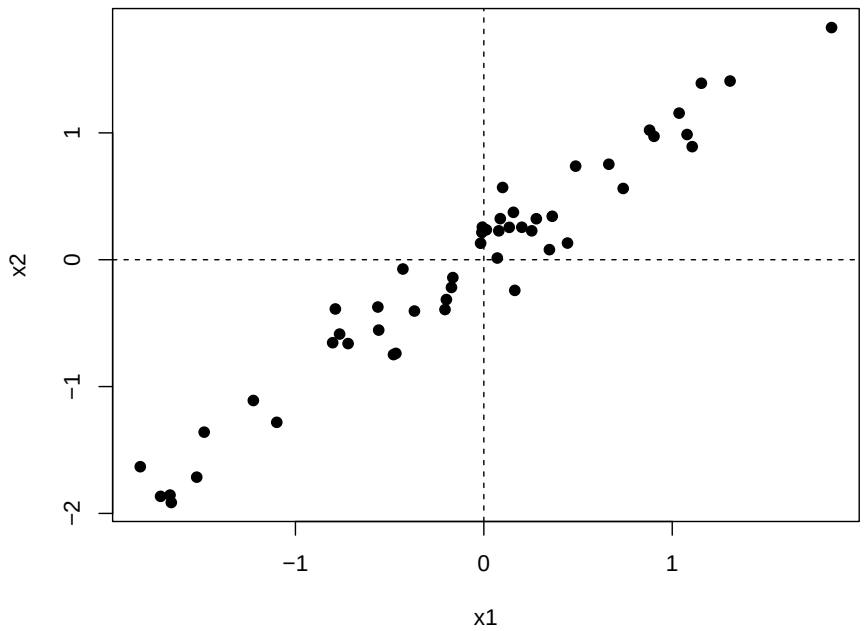
It is helpful to imagine our data geometrically as points in a high dimensional space: if $x = (x_1, \ldots, x_k)$ is our data on an individual, we can think of $x$ as a point in a k-dimensional space. (think of a scatterplot). If we have a high-dimensional data set, it can be difficult to visualize this configuration of points. With say two or three variables it is easy, but not for $k > 3$.

One strategy for dealing with high dimensional data is to construct a data configuration in a low dimensional space that captures most of the structure of the original (for example, preserves clusters).

Often data lives, at least approximately, in a low-dimensional subspace. If we can identify this, we can project the data onto the low dimensional subspace and draw the resulting picture.

# Subspaces

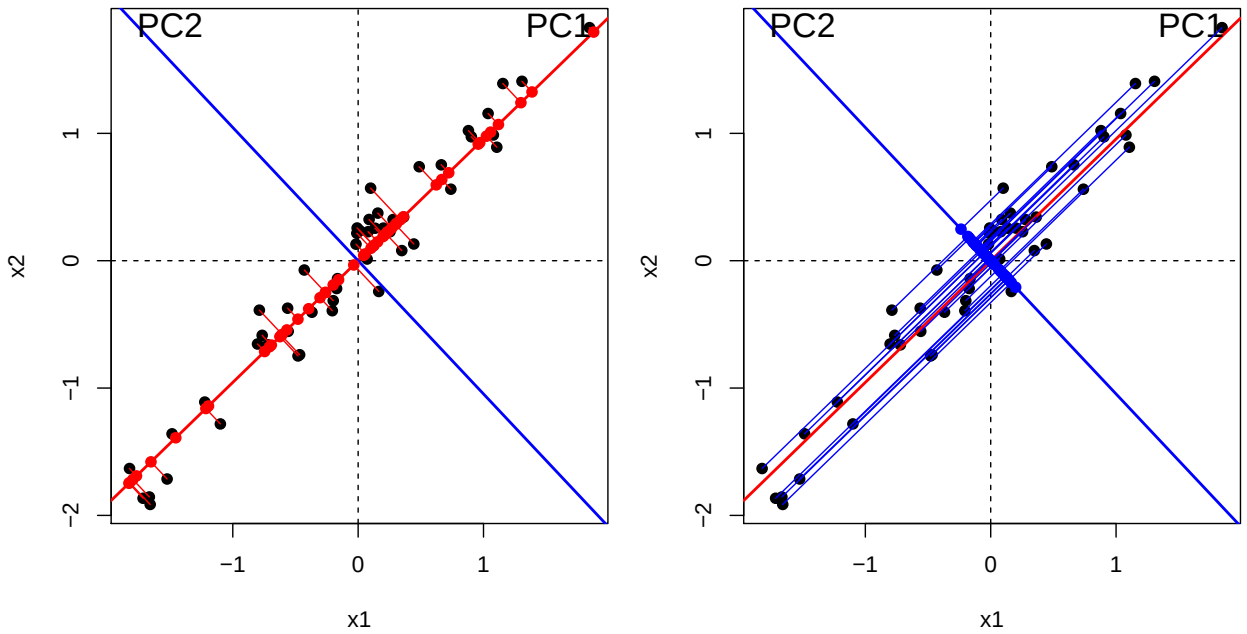
Suppose we have a data set with two variables, so the data live in two dimensions:




Here the points essentially lie in a one-dimensional subspace (a line).

# Principal components

We rotate the axes to get the most variance along PC1, and project the data onto PC1. Since the variance along PC2 is so small, we can ignore PC2 and we have reduced the dimension from 2 to 1.




In general, we can plot the first 2 or 3 principal components to visualise the higher dimensional space.

# Example: netball players

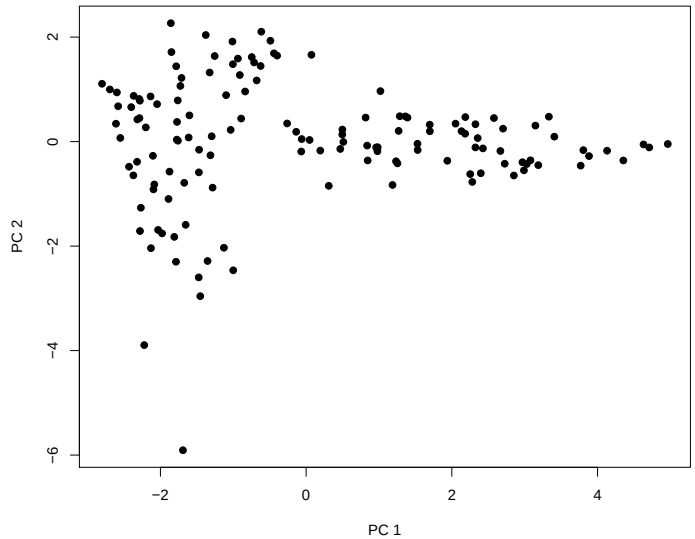
These data consist of obervations made on 137 elite netball players, and consist of various performance measures made for each player. The variables are

- **prop.made**: proportion of successful passes made
- **prop.recd**: proportion of successful passes received
- **Pens.A**: offensive penalties conceded per quarter
- **Pens.D**: defensive penalties conceded per quarter
- **Ints**: interceptions per quarter
- **Tips**: tips per quarter

# Example: netball players

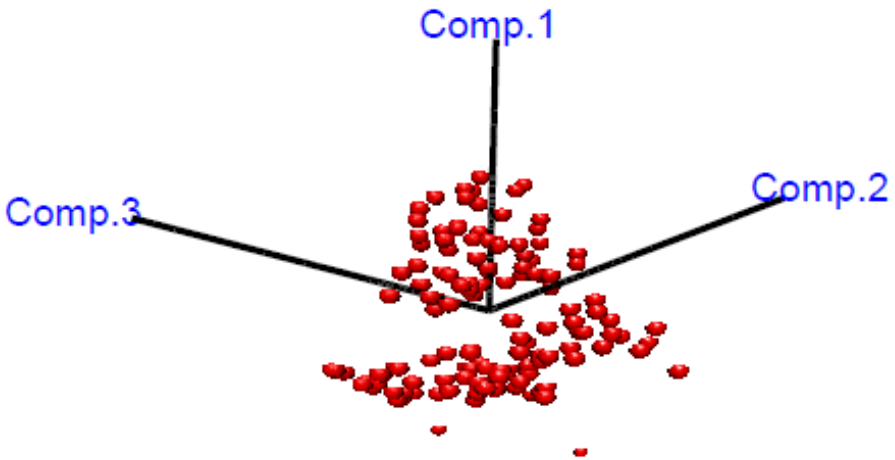
Let's calculate the first two PC's and plot them ( data have been standardized and are in a data frame `netball.standard.df`)

```
netball.pc = princomp(netball.cont.df)
plot(netball.pc$scores[,1:2], pch=19, xlab = "PC 1", ylab = "PC 2")
```



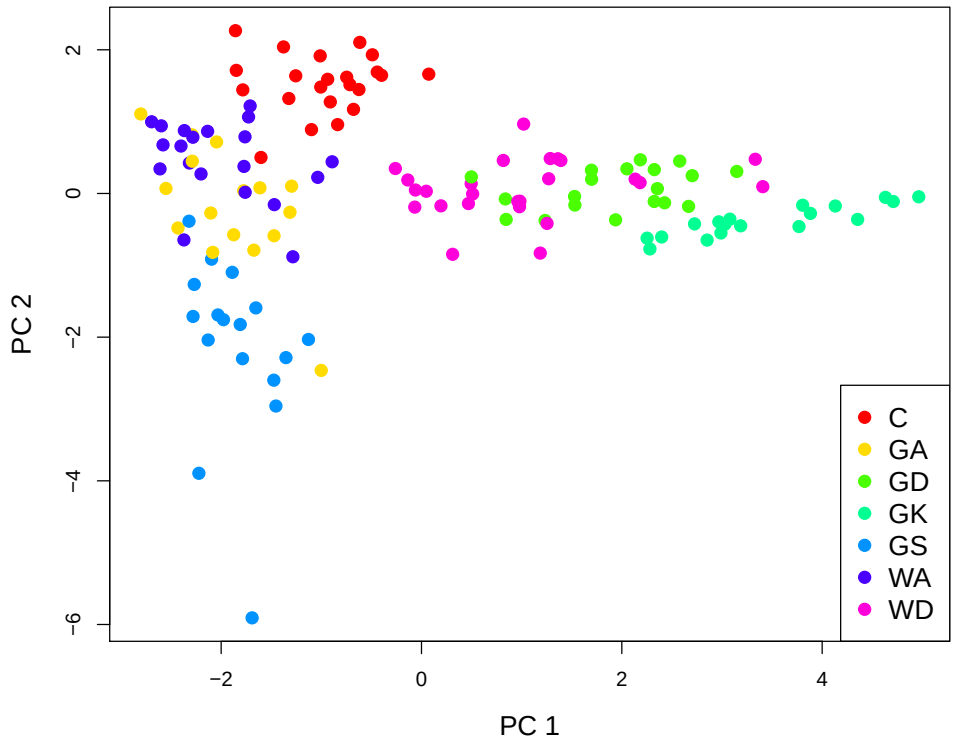
# Example: 3d plot

Alternatively a 3-d plot (see handout for code) plotting the first three principal components:



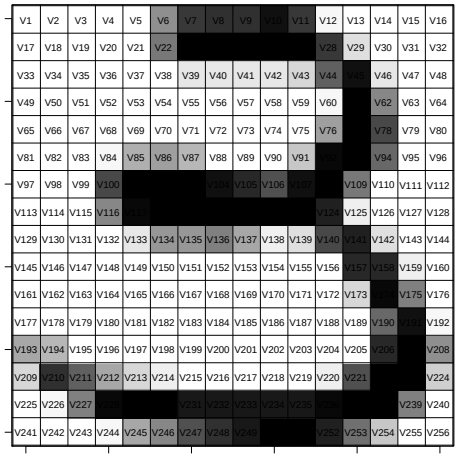
# Example: netball players, with positions added

Interpretation of the structure:



# Example: hand-written 3's

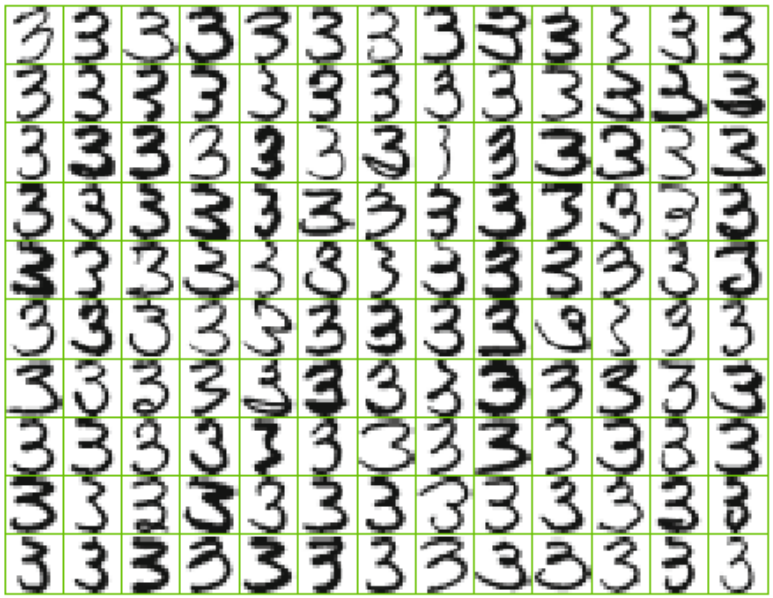
The data here consist of 658 images of hand-written 3's, represented as a 16 x 16 array of pixels, as in the picture below.



Each pixel is given a grey-scale value in the range $[-1, 1]$ with -1 representing white and 1 representing black. There are thus 16 x 16 = 256 numbers representing a particular digit, which we can take as the values of 256 variables, $V_1, \ldots, V_{256}$, say. The data are in a data frame `digits`.

# Example: hand-written 3's

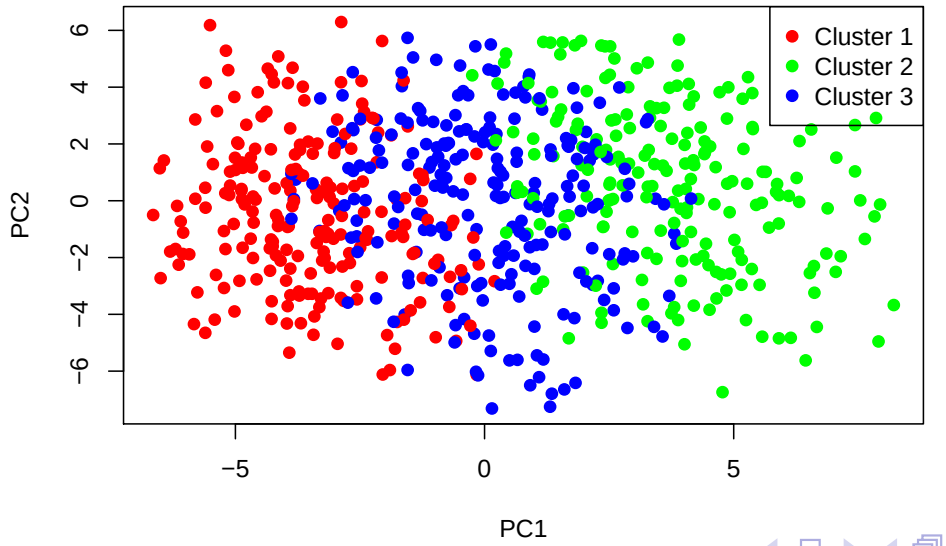
Here is a sample of 160 of these images :



# Example: hand-written 3's

Let's plot the first two PCA's:

```
digits.pca = princomp(digits)
plot(digits.pca$scores[, 1:2], type="n",xlab="PC1", ylab="PC2")
my.colours = rainbow(3)
points(digits.pca$scores[, 1:2],
  col=my.colours[clusters$cluster], pch=19)
legend("topright", paste("Cluster",1:3), col=my.colours, pch=19)
```



# Clustering

Here we consider a set of individuals with features $x = (x_1, x_2, \ldots, x_k)$ measured on each individual. Based on these features, we want to group the individuals into clusters of similar individuals.

If all the features are numeric, we can regard each individual as a point in a $k$-dimensional space. The clusters will consist of points that are close together.

The $K$-means algorithm is a method for doing this grouping.

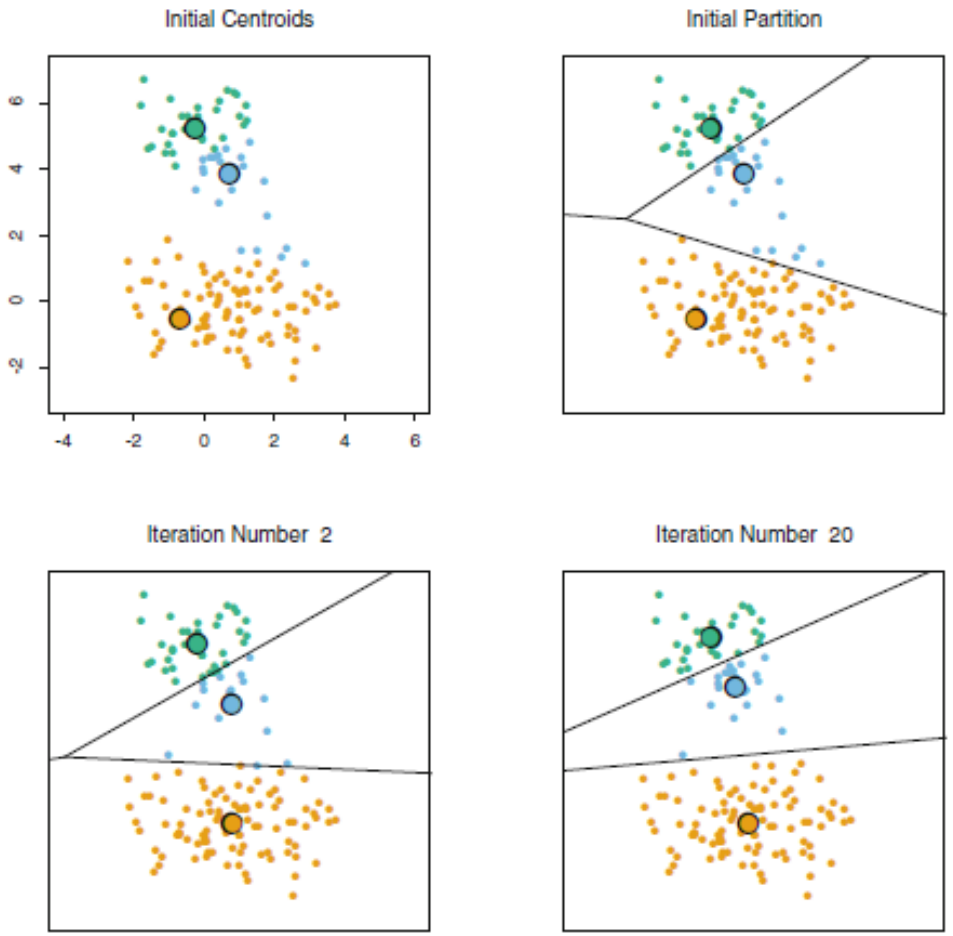
# $K$-means

Suppose we want to group the points into $K$ clusters. The $K$-means algorithm works as follows:

1. Pick $K$ arbitrary points in $k$-dimensional space, in the data cloud. These will be the cluster centres.
2. Assign points to the closest centre. These will be the clusters.
3. Recalculate the centres by averaging each feature over the cluster.
4. Repeat 2 and 3 until no further change.

The picture on the next slide illustrates this:

# $K$-means algorithm

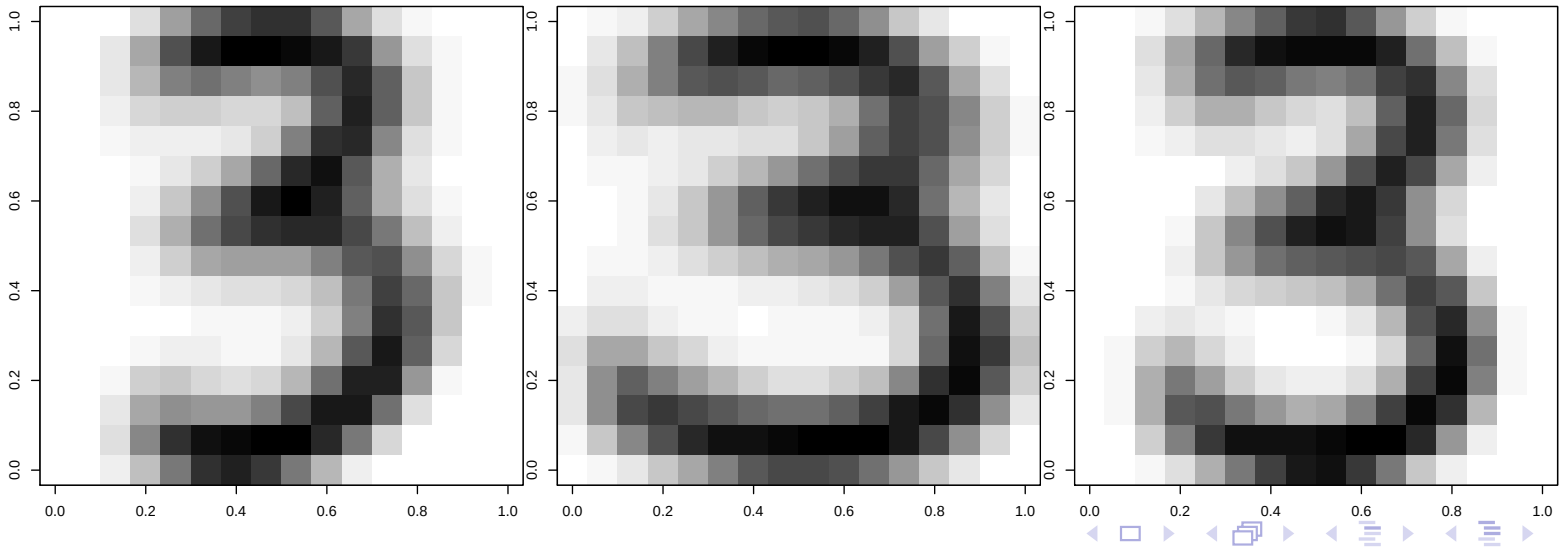


# Example: hand-written 3's

We will use the R function `kmeans` to group the digits into 3 clusters:

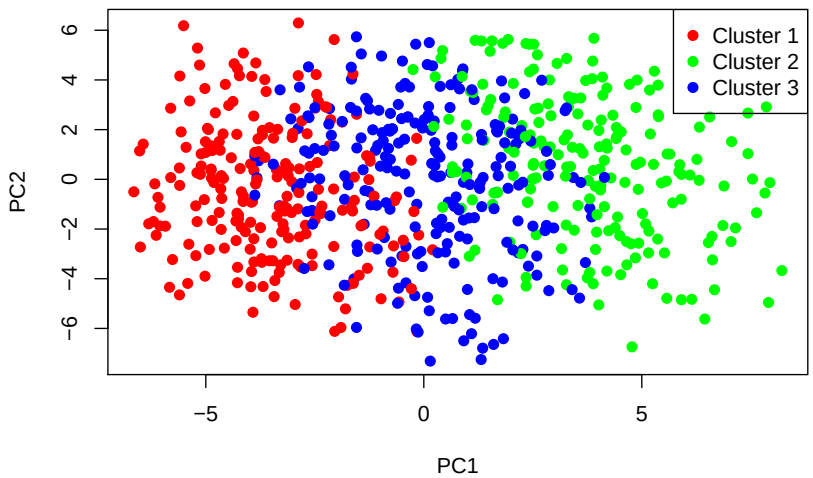
```
clusters = kmeans(digits, centers=3)
centers = clusters$centers
```

The matrix `centers` contains the coordinates of the three cluster centers in 256-dimensional space. We can draw them:



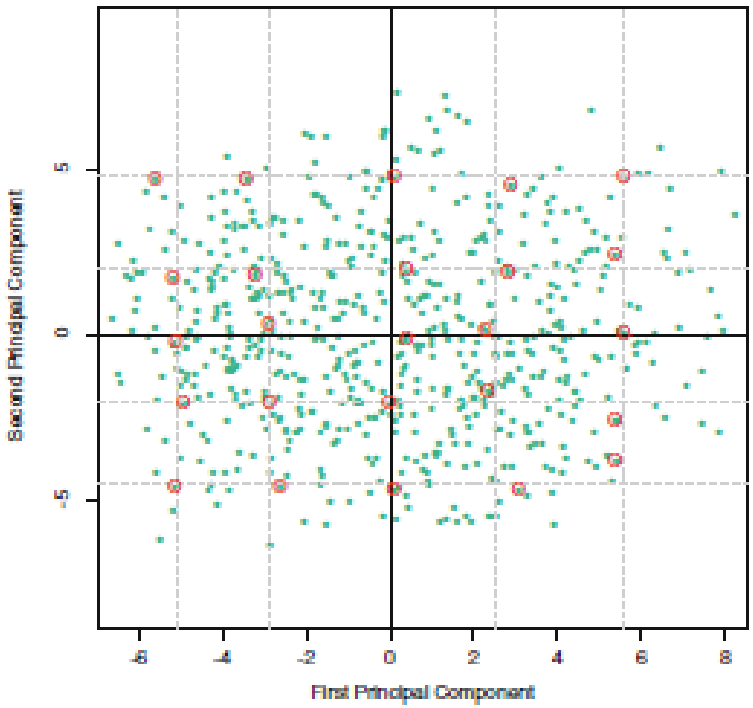
# Example: hand-written 3's

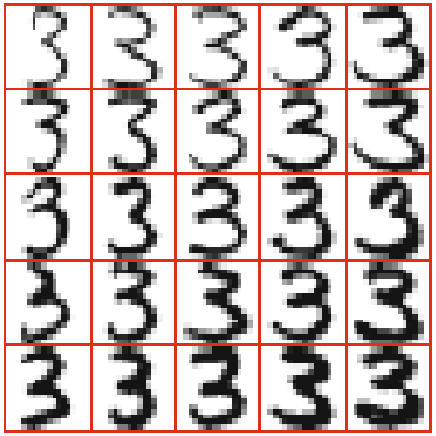
The clusters are those illustrated on slide 15: they reflect essentially the values of the first PC, which seems to be measuring the width of the digit.



# Example: hand-written 3's

The second PC represents the heaviness of the digit:



# Example: $K$-means for the netball data