Lab 9

Ben Thomas

11/20/2019

Data

In this lab, I'm going to use PCA (principle component analysis) to examine a dataset with information on 8x8 images of handwritten letters. First, I'll download the data. Note that there are also two functions below: `plot_letter` and `pc_grid`.

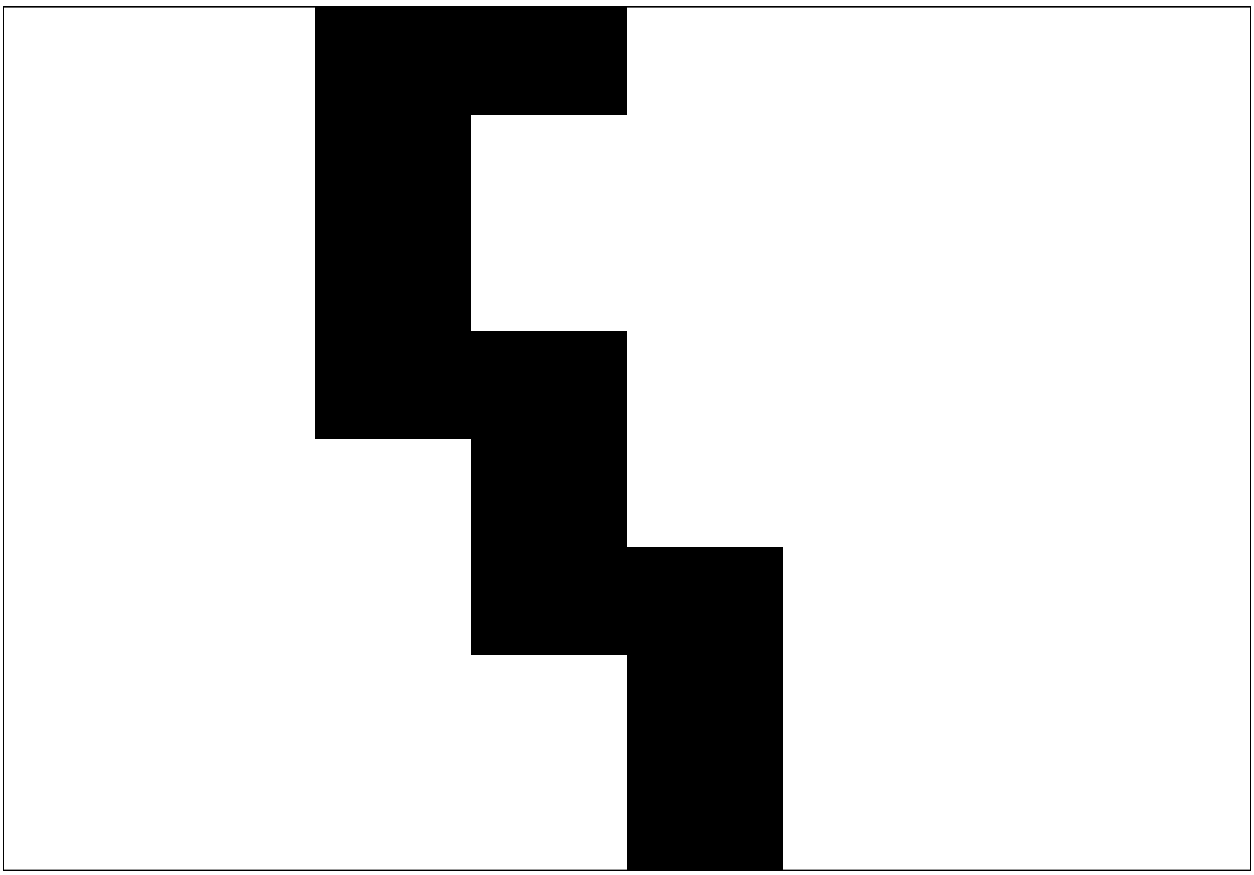
```
d <- read.csv("https://raw.githubusercontent.com/stat-learning/course-materials/master/data/handwritten

plot_letter <- function(x, hasletter = TRUE) {
  if(hasletter) {
    a <- as.numeric(x[, -1])
  }else{a <- as.numeric(x)}
  m <- matrix(a, nrow = 8, byrow = TRUE)
  m <- t(apply(m, 2, rev)) # rotate matrix
  par(mar = rep(0, 4))
  image(m, axes = FALSE, col = rev(grey(seq(0, 1, length = 256)))) #this should be a divergent palette
  box()
}

pc_grid <- function(pca, data) {
  d <- data
  grid_points <- as.matrix(expand.grid(seq(-1.5, 1.5, length.out = 5),
                                       seq(-1.5, 1.5, length.out = 5)))
  pc_points <- pca$x[, 1:2]
  nearest_ind <- rep(NA, nrow(grid_points))
  for(i in 1:nrow(grid_points)) {
    gp <- matrix(rep(grid_points[i, ], nrow(pc_points)),
                 ncol = 2, byrow = TRUE)
    nearest_ind[i] <- which.min(rowSums((pc_points - gp)^2))
  }
  nearest_grid <- data.frame(d[nearest_ind, ])
  par(mfrow = c(5, 5))
  regrid <- c(21:25, 16:20, 11:15, 6:10, 1:5)
  for(i in regrid) {
    plot_letter(nearest_grid[i, ])
  }
}
```

In this dataset, the columns represent different pixels in each image, and the rows are observations (the images of handwritten letters). A value of 1 in a column implies that the corresponding pixel is filled, and 0 implies otherwise.

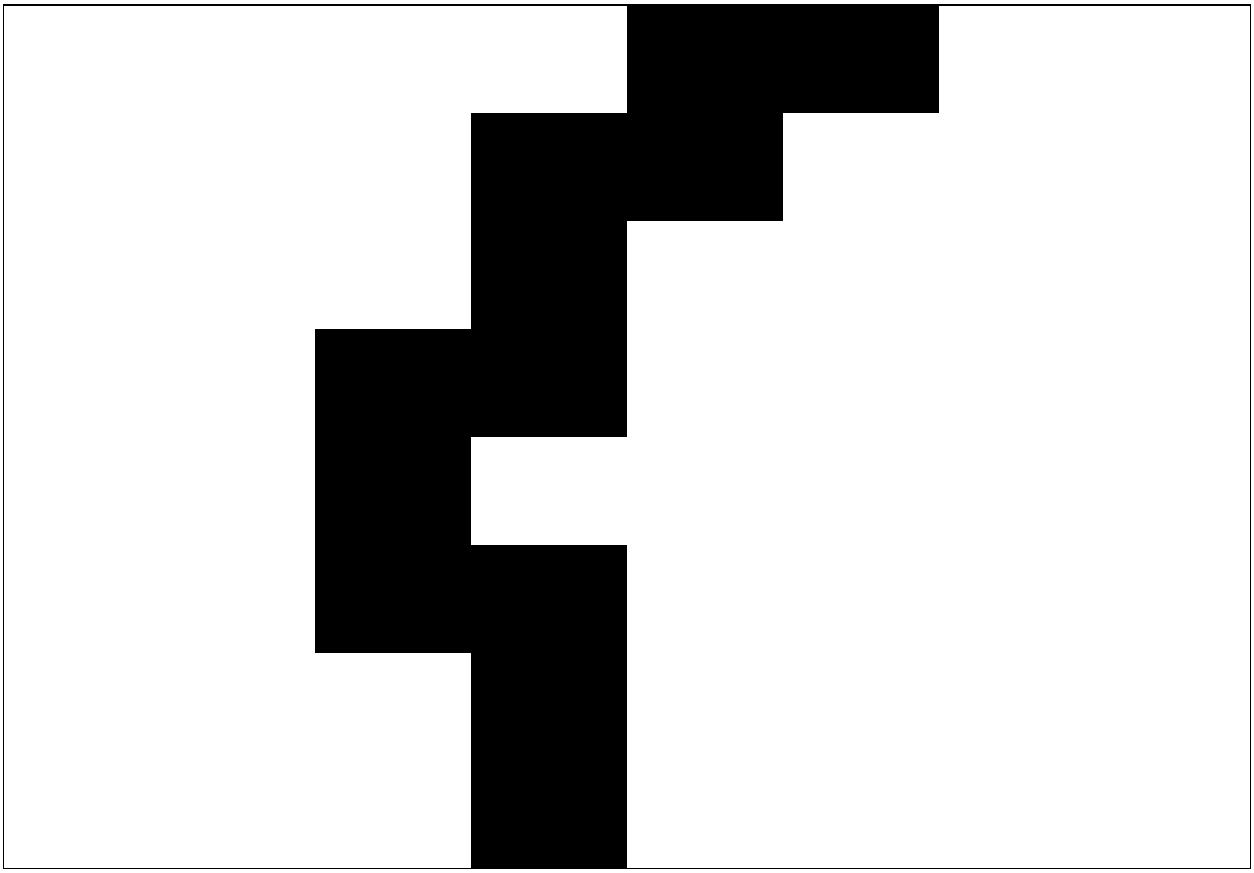
Now, I'm going to focus in more on a specific letter: L. Below, I've created a new dataset (`L.data`) which only includes observations where the letter is L and plotted a few of the observations.

```
L.data <- d[d$letter =="l",]
```

```
plot_letter(L.data[1,])
```

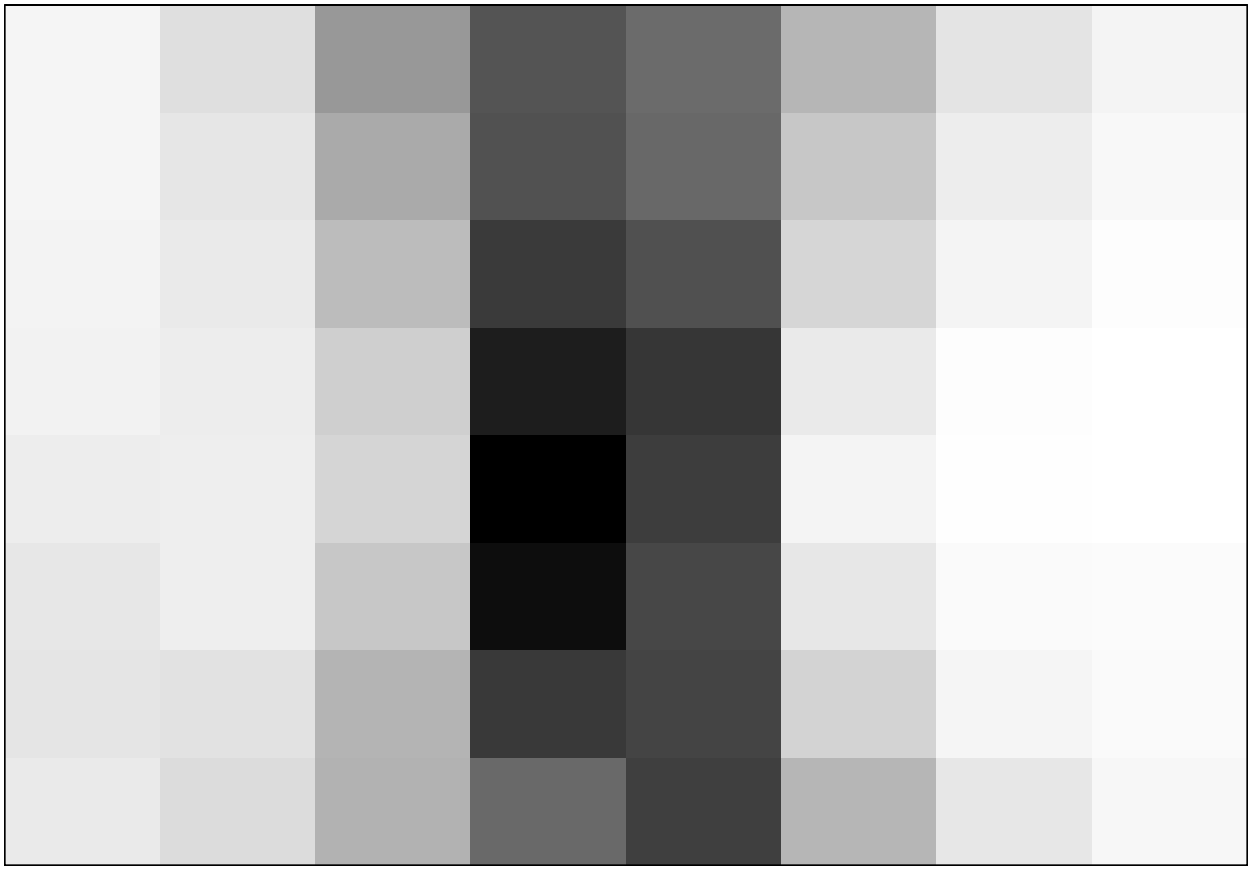


```
plot_letter(L.data[2,])
```



We can go further into this by visualizing the *mean* image of L. I've done so below. Note that in general, observations seem to be centered horizontally, and generally take the shape of a vertical line.

```
l_mean <- colSums(L.data[, -1])/nrow(L.data)
plot_letter(l_mean, hasletter = FALSE)
```



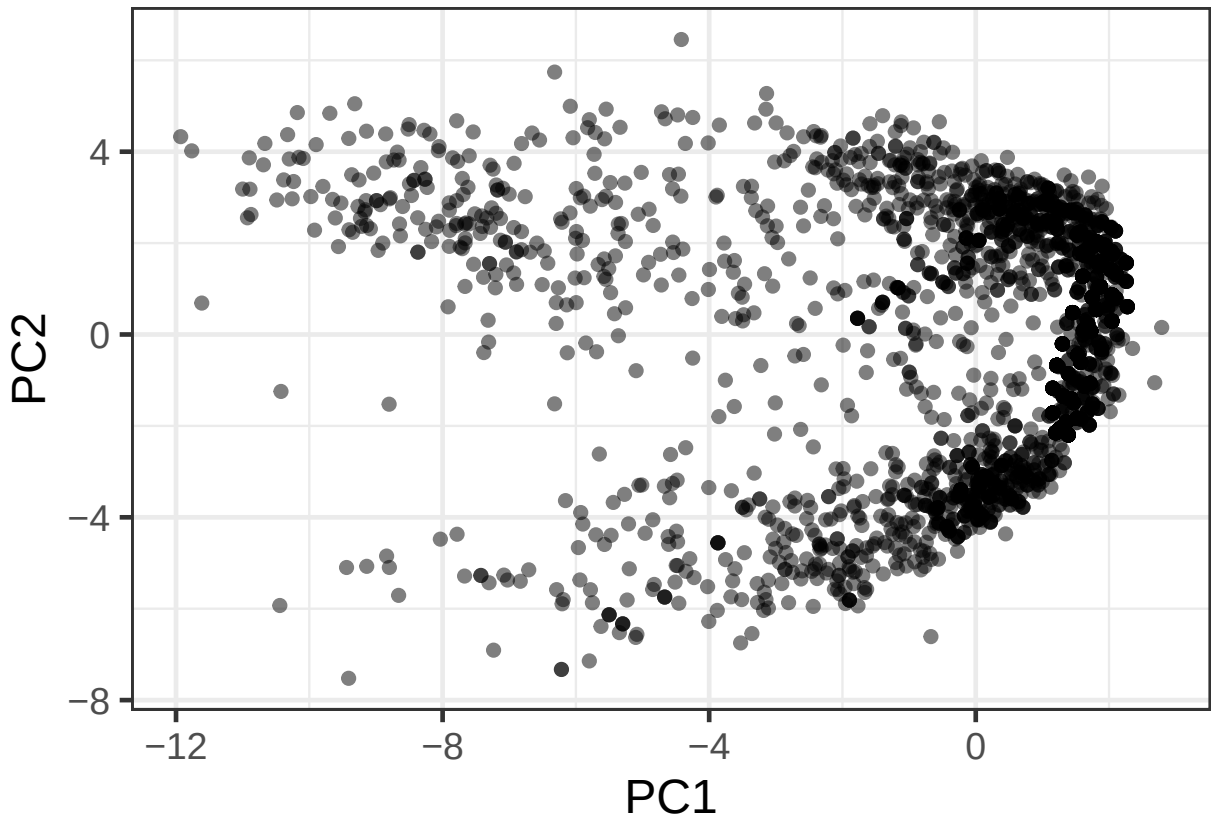
Dimension reduction

Can we capture most of the information in this dataset with fewer dimensions? To make an attempt, I've performed principle component analysis on the L dataset. Below, I've plotted the first two principle components; the shape is a little odd.

```
library(tidyverse)
```

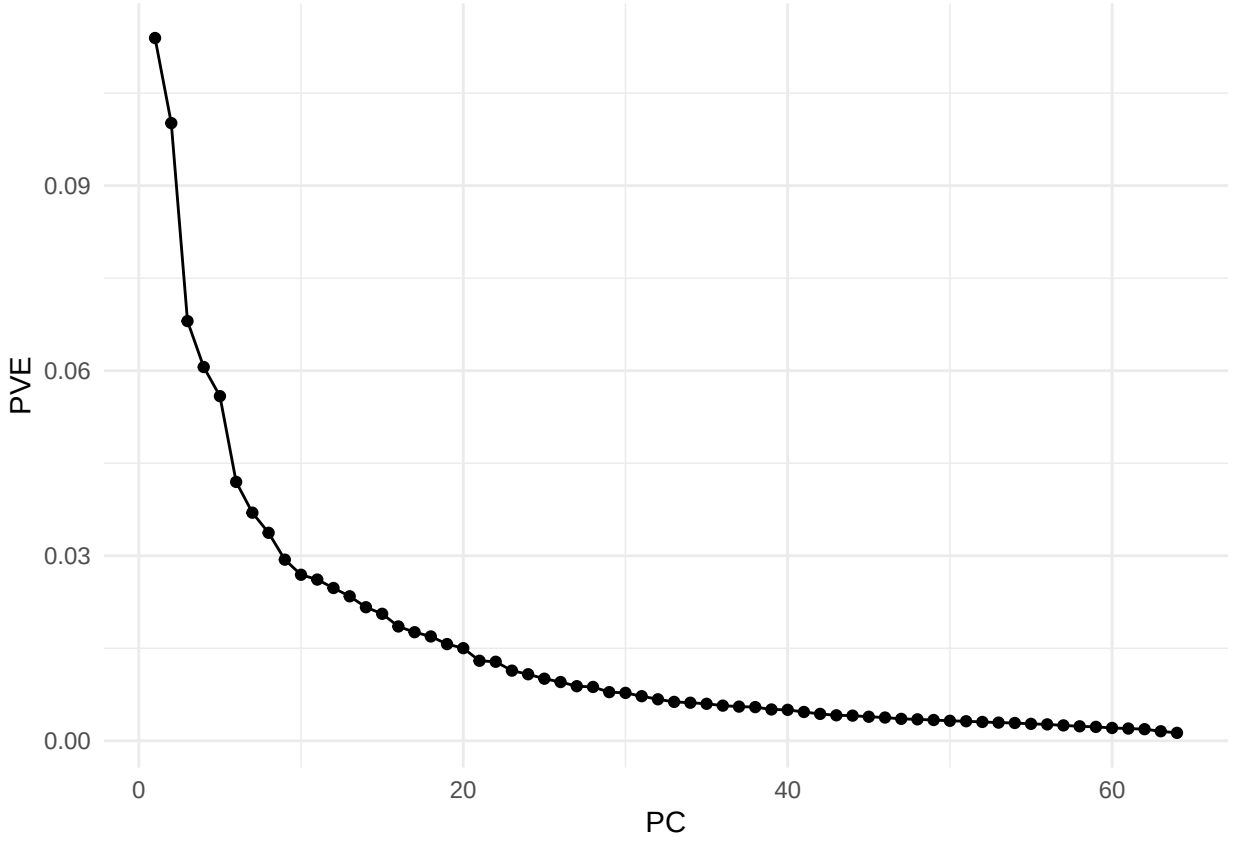
```
## -- Attaching packages ---------------------------------------------------------------------------

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.1     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ------------------------------------------------------------------------------------ t
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
pca1 <- prcomp(L.data[, -1], scale = TRUE)
pcas <- as.data.frame(pca1$x)
p1 <- ggplot(pcas, aes(x = PC1, y = PC2)) +
  geom_point(size = 2, alpha = 0.5) +
  theme_bw(base_size = 18)
p1
```



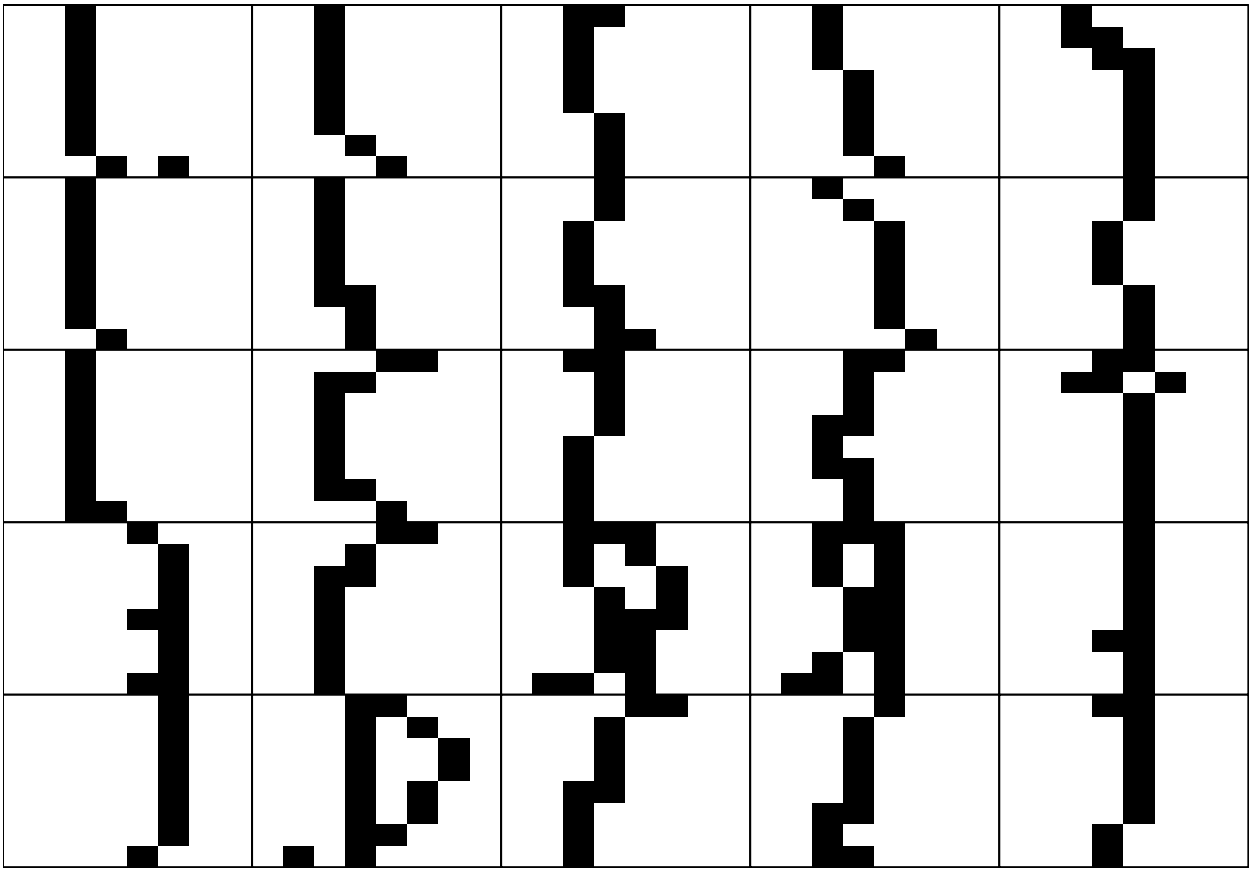
Having done this PCA on my dataset, we can construct a "Scree plot" to identify how much variation each of the principle components captures. I've done so below.

```
dnew <- data.frame(PC = 1:64,
                   PVE = pca1$sdev^2/sum(pca1$sdev^2))
ggplot(dnew, aes(x = PC, y = PVE)) +
  geom_line() +
  geom_point() +
  theme_minimal()
```



While this is all well and good, it's not initially easy to understand what these principle components are actually measuring. To gain an understanding of that, I've plotted 26 observations, at different points along the scatterplot of the first two principle components.

```
pc_grid(pca1, L.data)
```

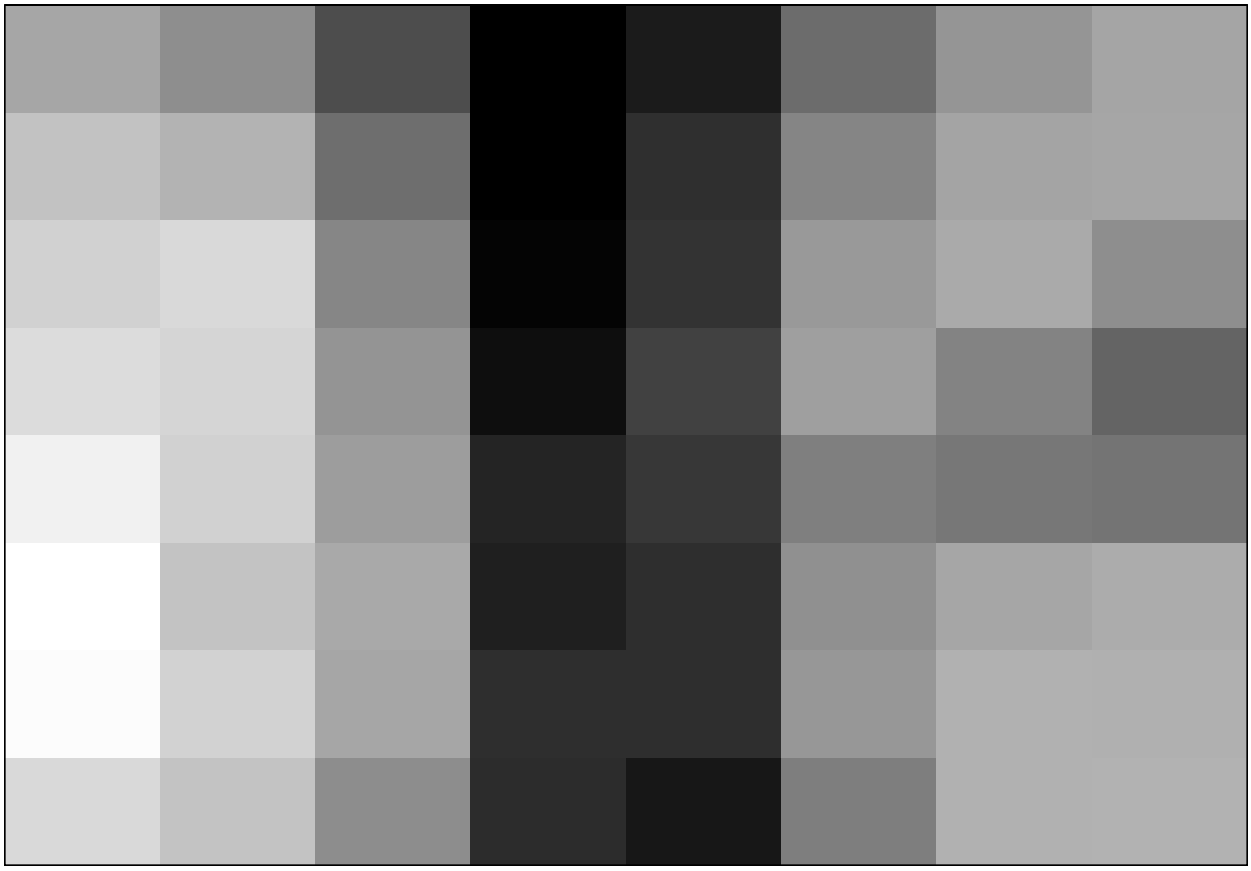


From this, it looks like the first principle component might be measuring whether or not the image is centered or not. The observations on the left side of this grid are not centered, while those on the other are definitely centered. The second principle component is a little harder to tease out, but it could be measuring the *lean* of the letter. Observations at the top of the grid seem to slope down and to the right, while those at the bottom go up and to the right.

There is another way to learn about these principle components, however, which might be more accurate than blindly guessing. We can extract the loadings of each of the components and plot them directly! I've done so below.

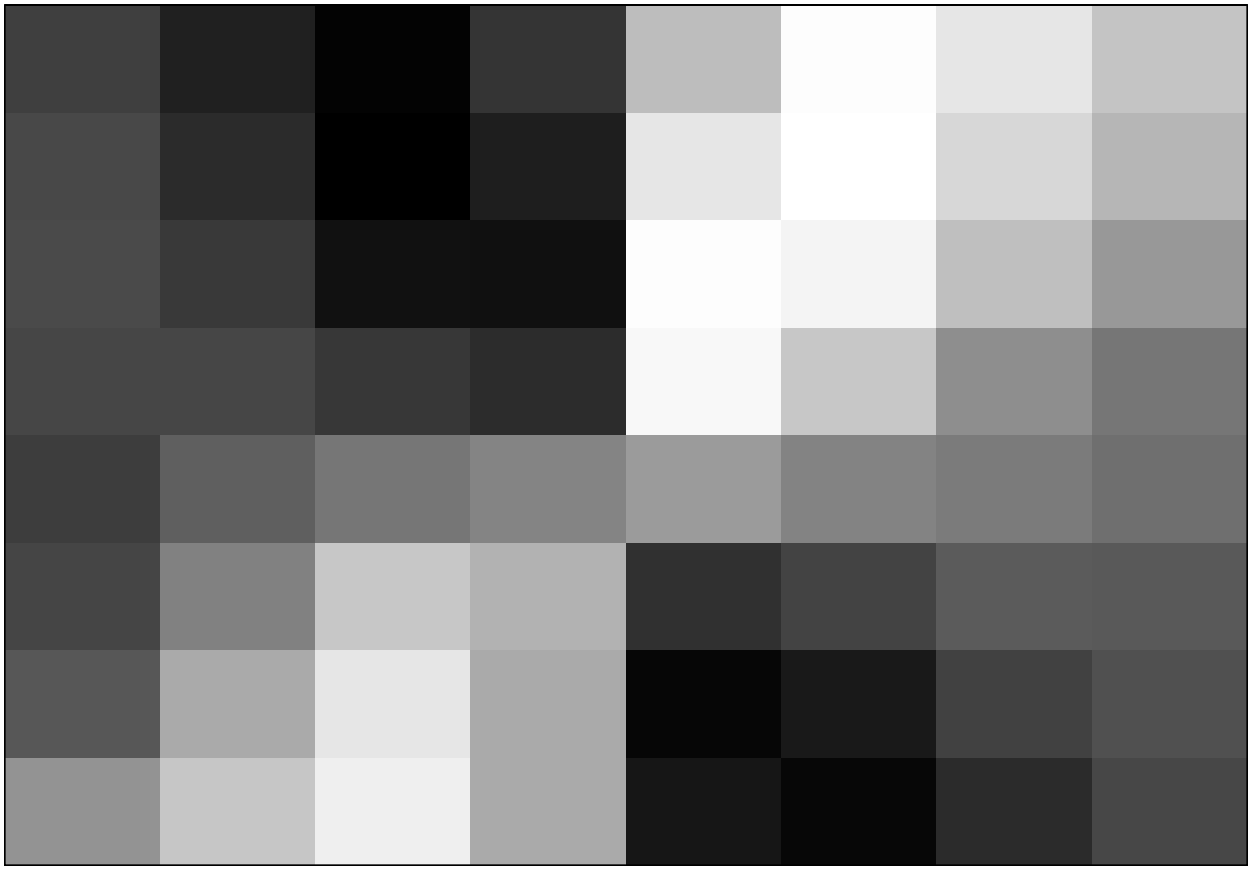
```
phi <- pca1$rotation
```

```
plot_letter(phi[,1], hasletter = FALSE)
```



Like I guessed, this first rotation seems to be capturing whether or not there is anything in the center of the image.

```
plot_letter(phi[,2], hasletter = FALSE)
```



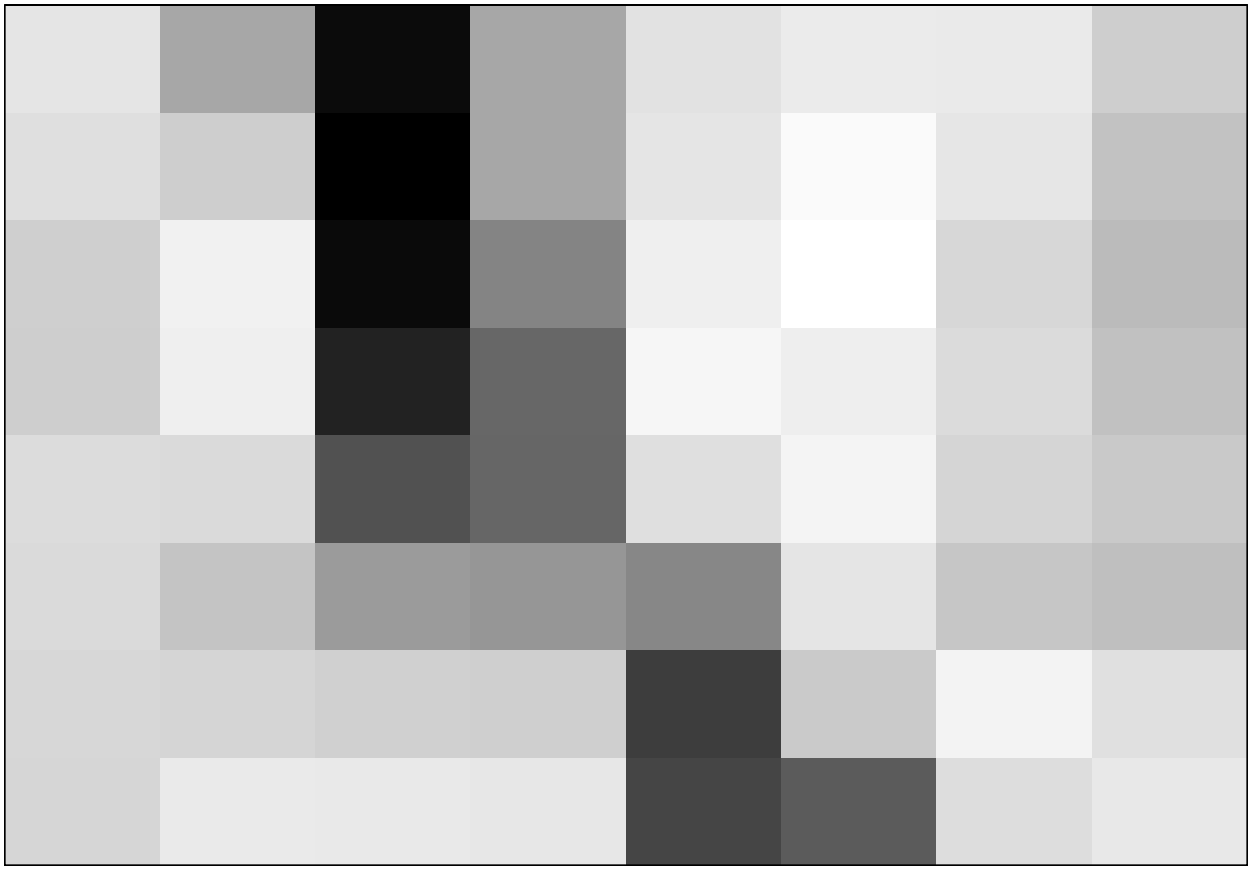
This second rotation captures whether or not there is anything in the top left or bottom right corner. Again, like mentioned in the grid, this could be thought of as capturing the lean of the letters (forward versus backward slants). ### Image reconstruction

Let's take this a step further, and think about a more intuitive way to see how much information each of these principle components captures. We can reconstruct the images of the letters using the principle components, and see how similar they are to the original images. Below, I've used the first ten principle components to reconstruct the first two observations.

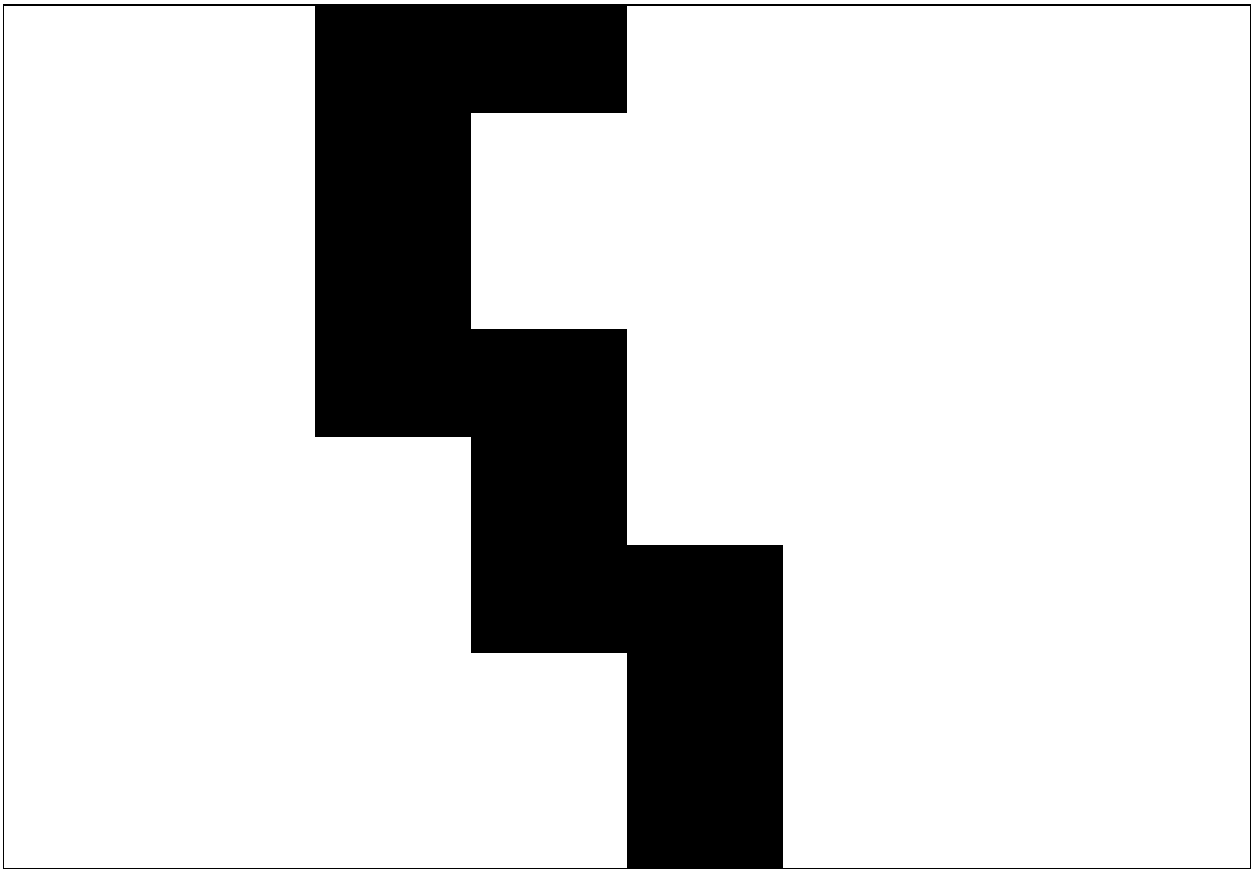
```
m <- 10
Z <- pca1$x

reconstructed.letter.1 <- l_mean + Z[1,1:m]%*%t(phi[,1:m])
reconstructed.letter.2 <- l_mean + Z[2,1:m]%*%t(phi[,1:m])
```

Below, I've plotted the reconstructed letter based on the first 10 principle components as well as the original image. While there is definitely some graininess, the overall shape seems to have been preserved.

```
plot_letter(reconstructed.letter.1, hasletter = FALSE)
```

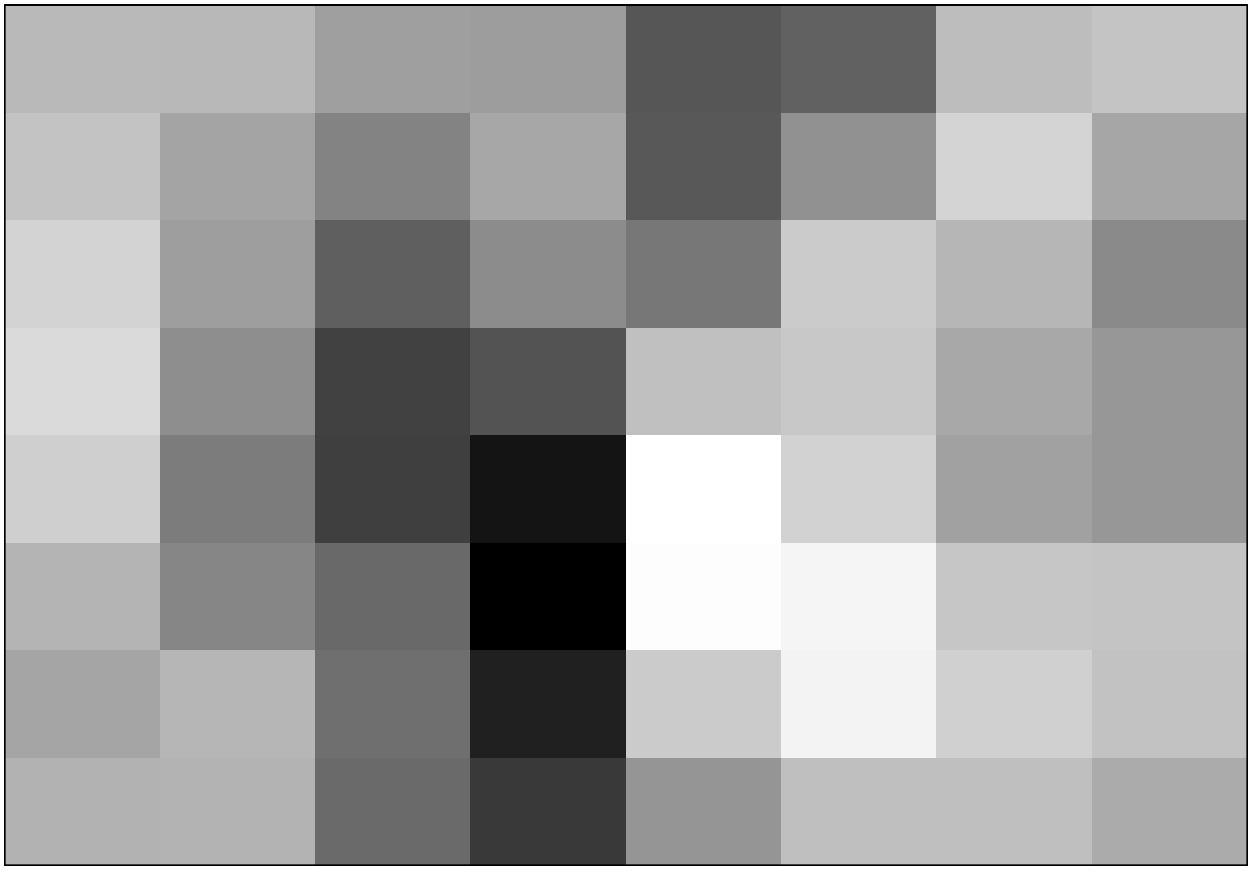


```
plot_letter(L.data[1,])
```

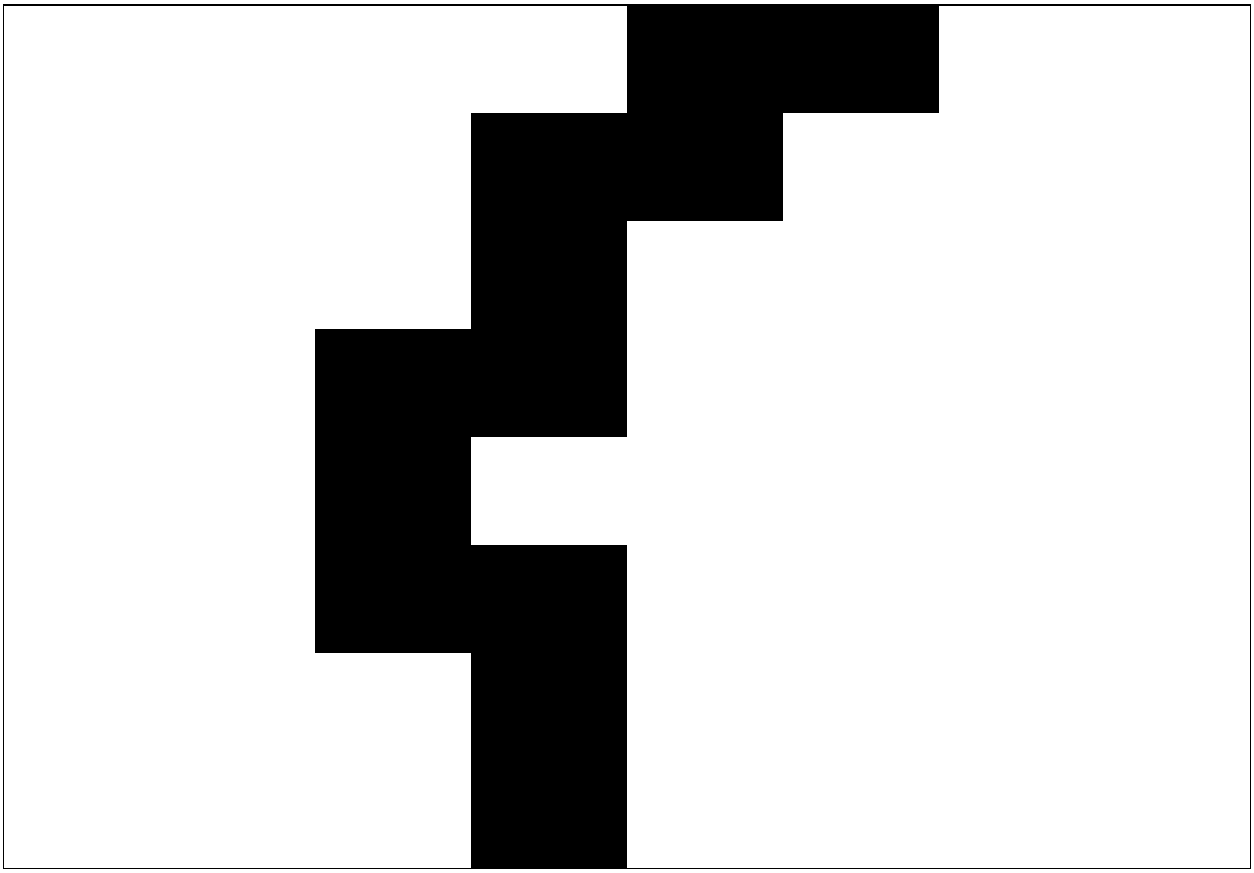


Below, I've done the same for the second observation, and the same holds true here; there's graininess, but the overall shape is preserved.

```
plot_letter(reconstructed.letter.2, hasletter = FALSE)
```



```
plot_letter(L.data[2,])
```



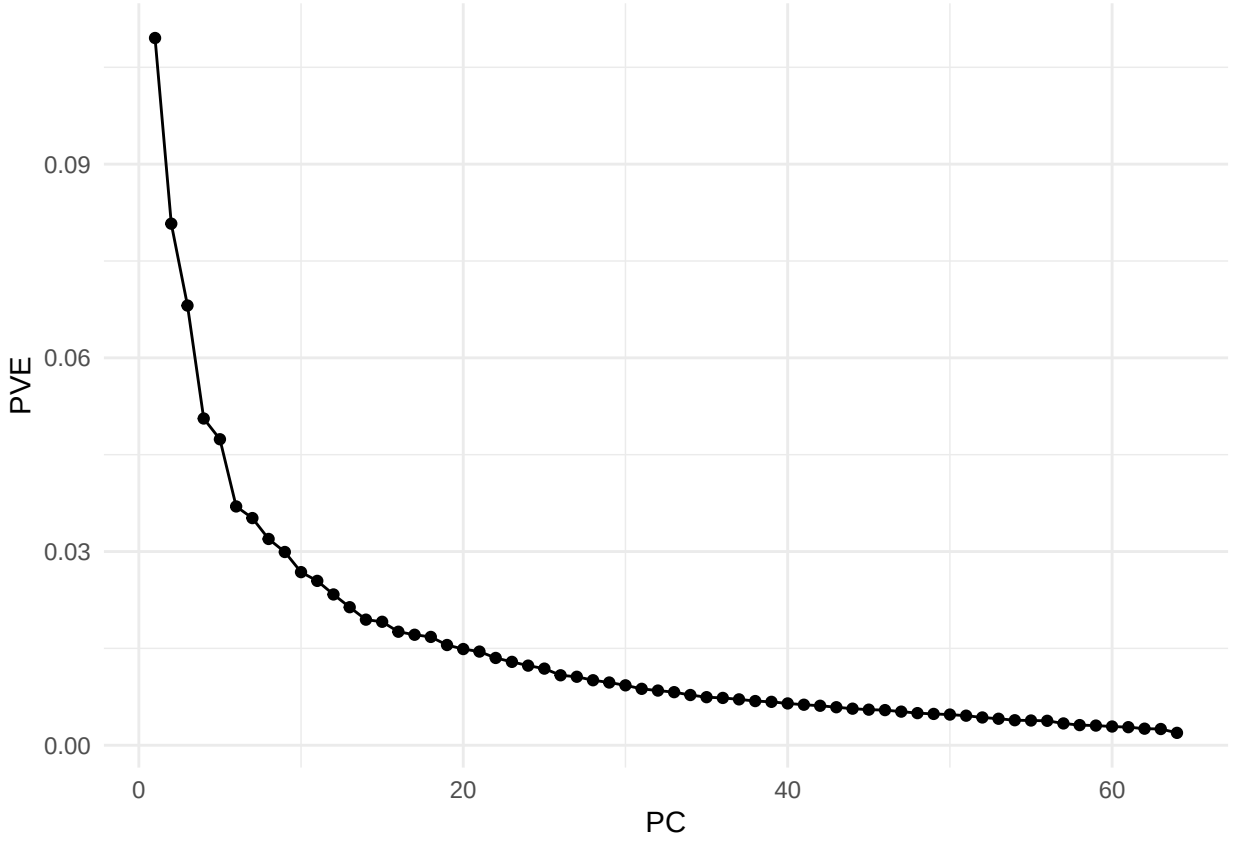
A different subset

Let's take a look at a different letter, to see how the amount of variation captured by the first 10 PCs might change. I'll use "P". Below, I've created the subset and run PCA.

```
P.data <- d[d$letter =="p",]
```

```
p.pca <- prcomp(P.data[, -1], scale = TRUE)
p.pcas <- as.data.frame(p.pca$x)
```

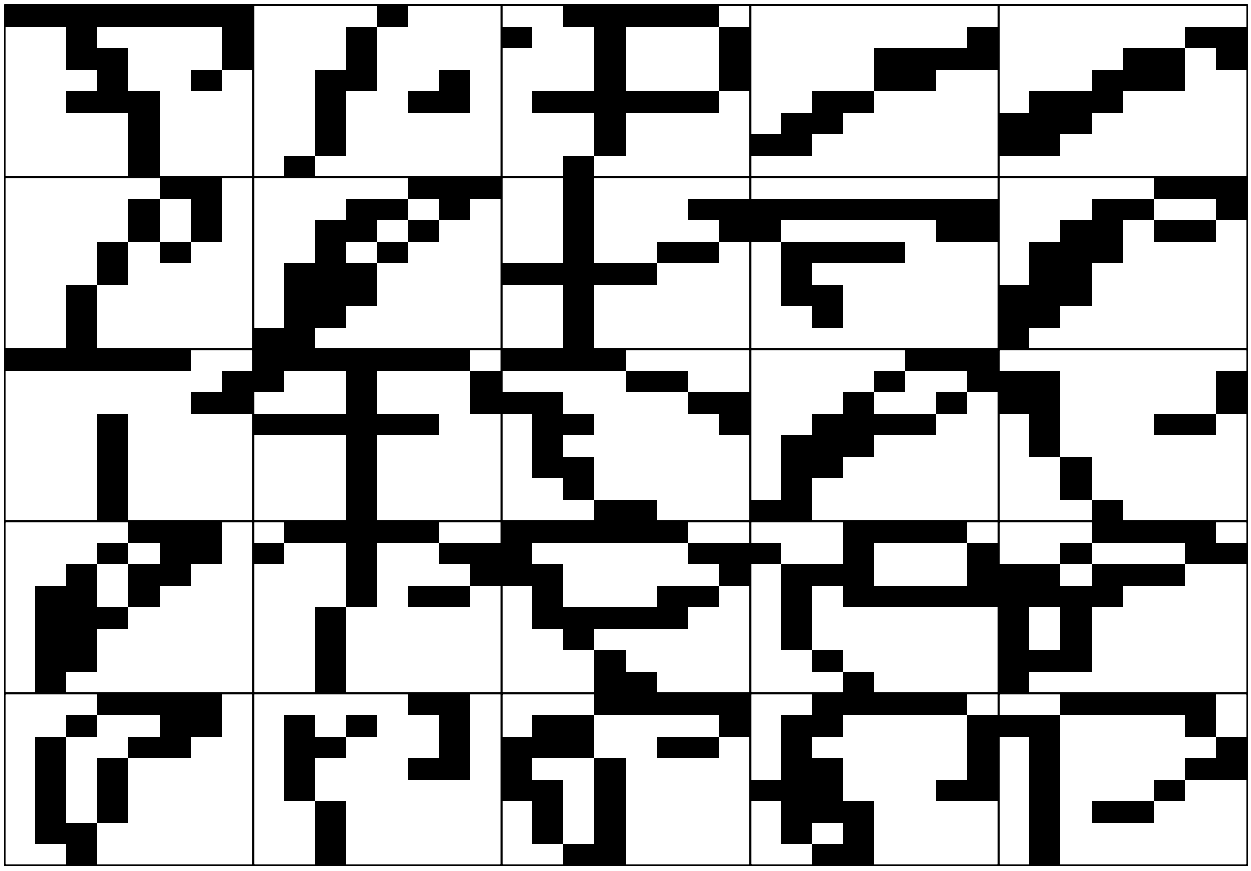
```
dnew2 <- data.frame(PC = 1:64,
                    PVE = p.pca$sdev^2/sum(p.pca$sdev^2))
ggplot(dnew2, aes(x = PC, y = PVE)) +
  geom_line() +
  geom_point() +
  theme_minimal()
```



The shape of the scree plot for P is fairly similar to that of L, implying that the PCs capture variation at a similar rate between the two.

Now, I'll create a grid of observations at different points along the first two principle components to see what they might be encoding.

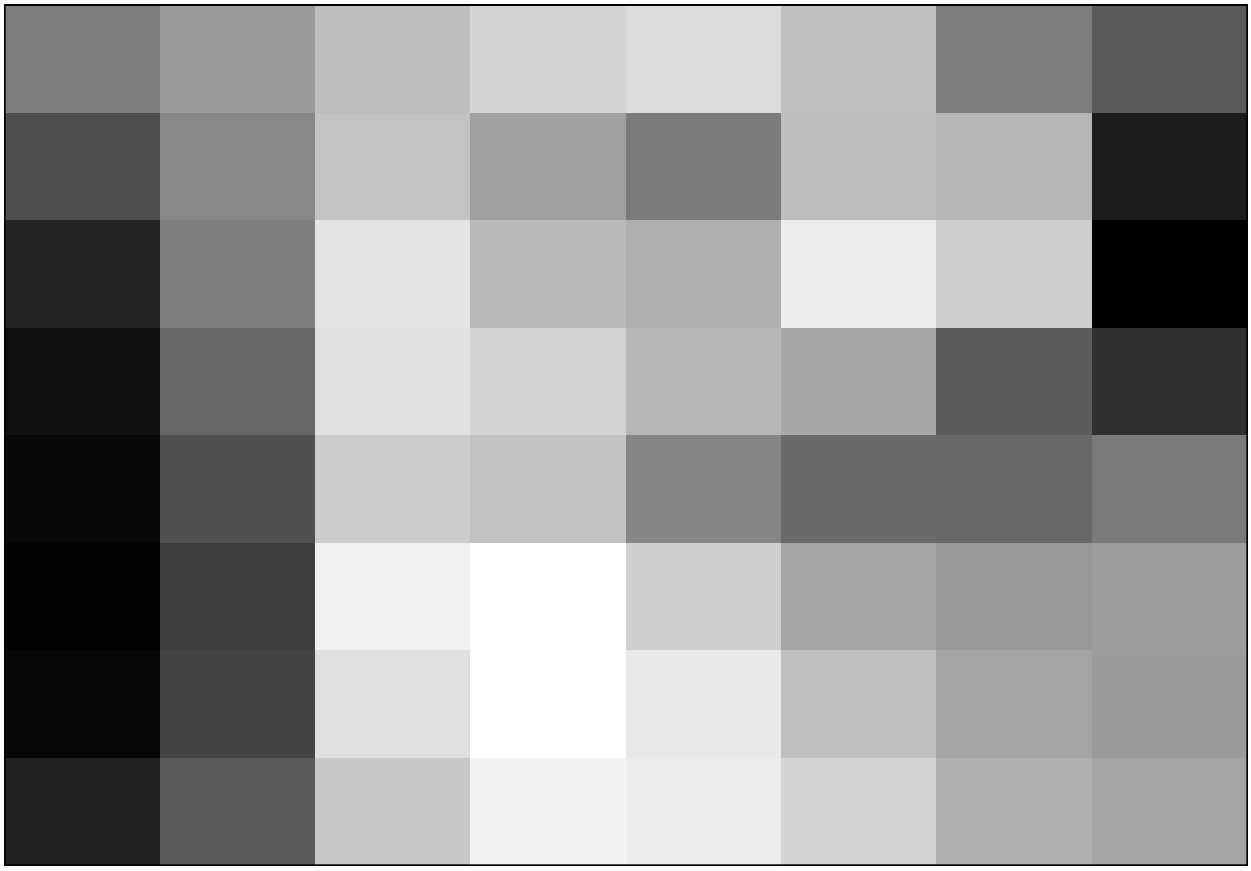
```
pc_grid(p.pca, P.data)
```



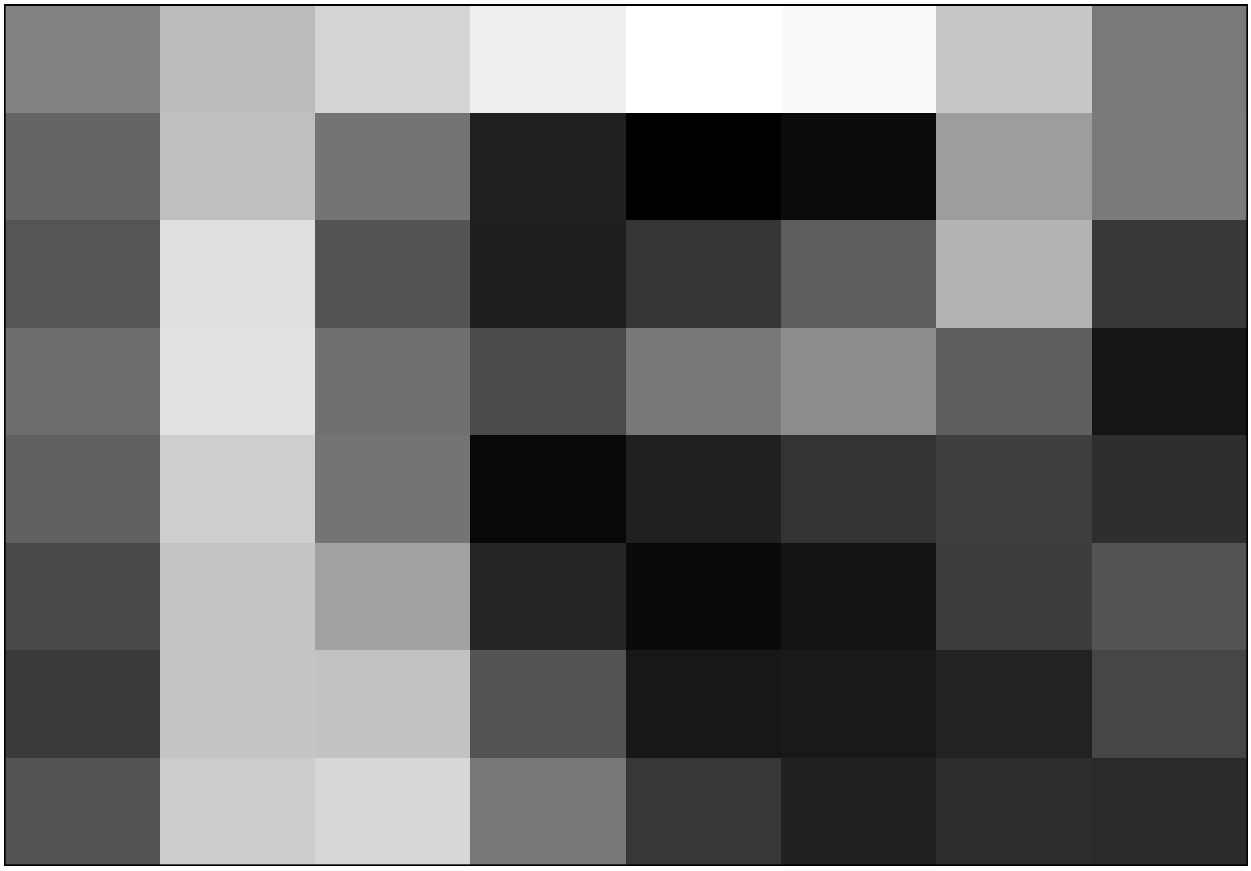
We can supplement this, again, with a plot of the actual loadings.

```
p.phi <- p.pca$rotation
```

```
plot_letter(p.phi[,1], hasletter = FALSE)
```



```
plot_letter(p.phi[,2], hasletter = FALSE)
```



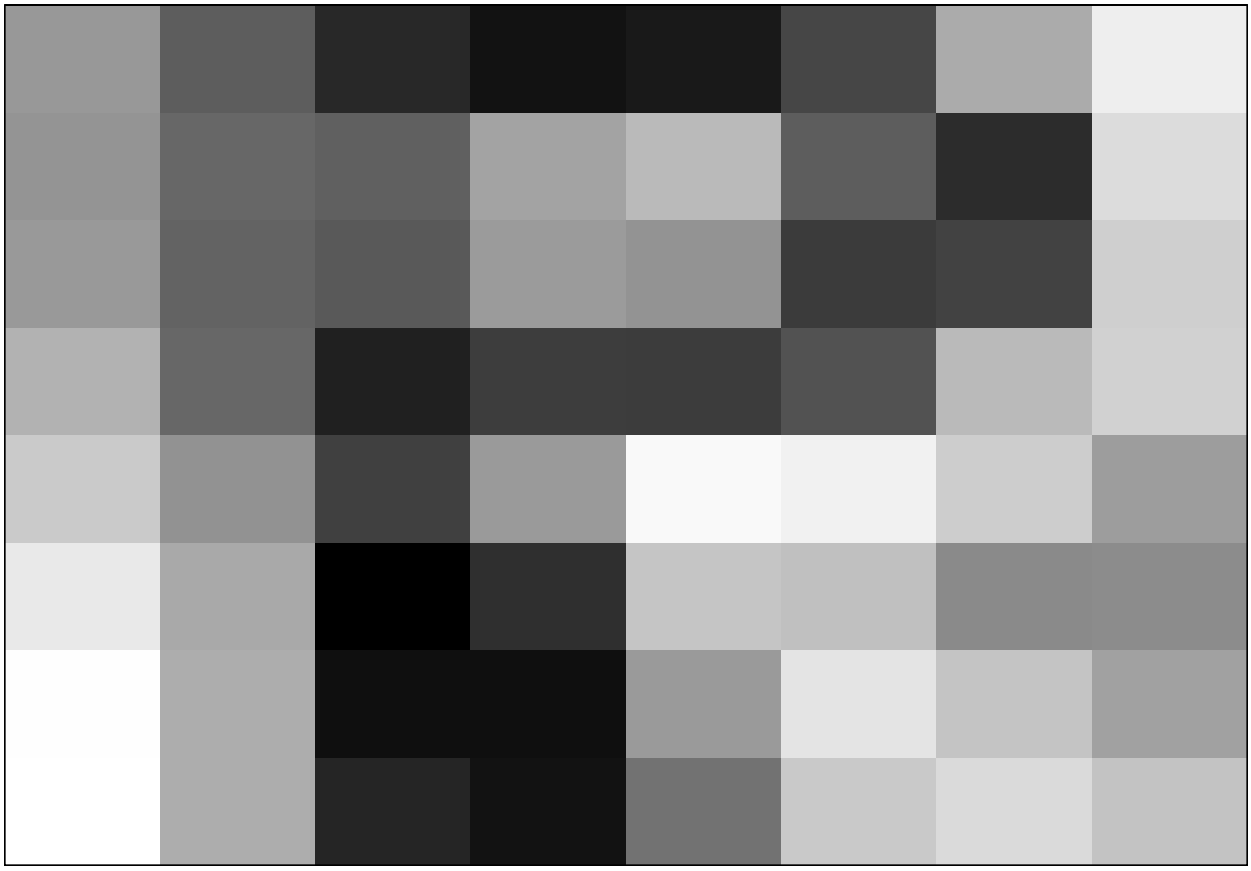
It seems here that the first PC is encoding whether or not the P is in the center or in the edges. The second one encodes whether or not the P is present in the bottom right or whether it goes further up and to the left.

Now, I'll recreate some of the letters and see how the reconstruction compares to the earlier one.

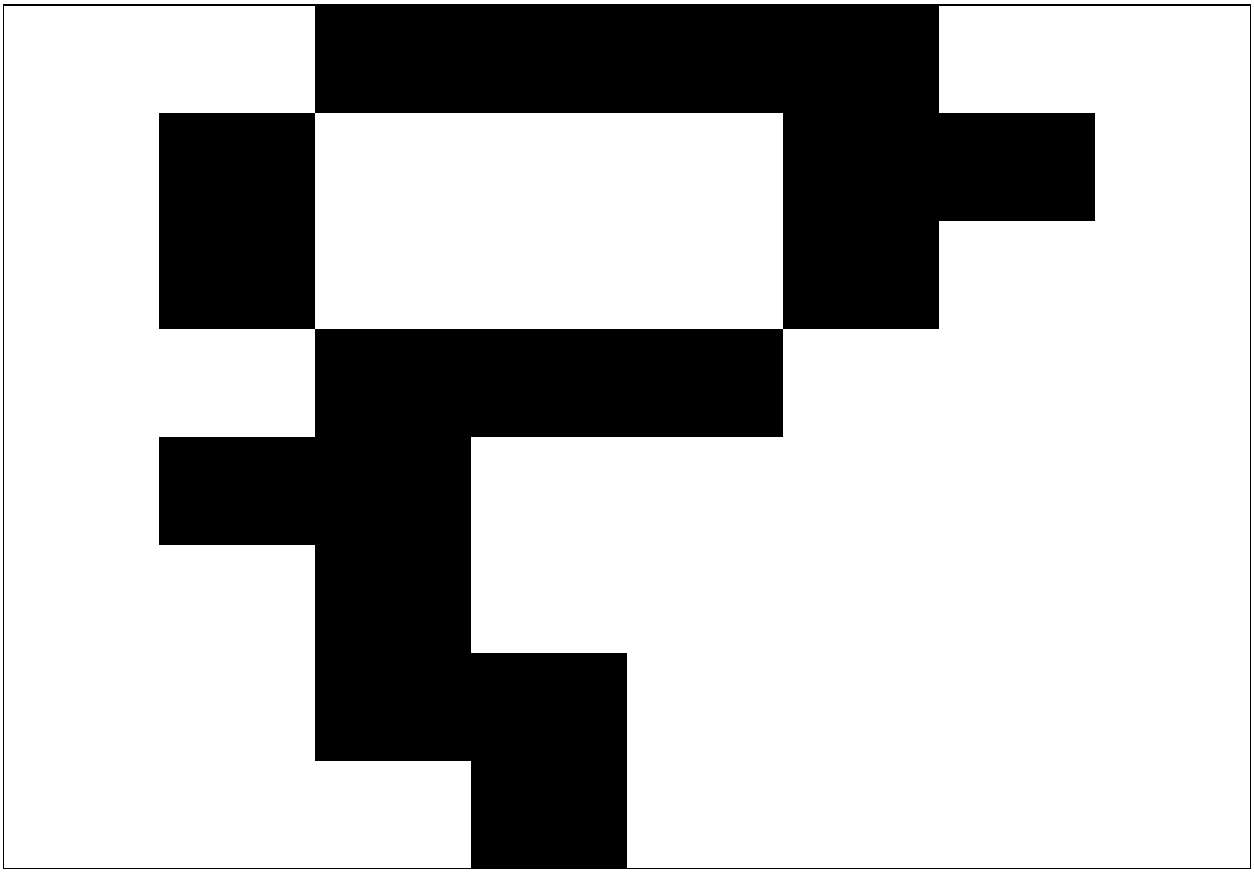
```
p_mean <- colSums(P.data[, -1])/nrow(P.data)
p.m <- 10
p.Z <- p.pca$x

p.reconstructed.letter.1 <- p_mean + p.Z[1,1:p.m]%*%t(p.phi[,1:p.m])
p.reconstructed.letter.2 <- p_mean + p.Z[2,1:p.m]%*%t(p.phi[,1:p.m])
```

Below, I've plotted the reconstructed letters. Again, while there is definitely some graininess, the overall shape seems to have been preserved, and there is a similar amount of information preserved here as in the L data. This makes sense, given the similar scree plot.

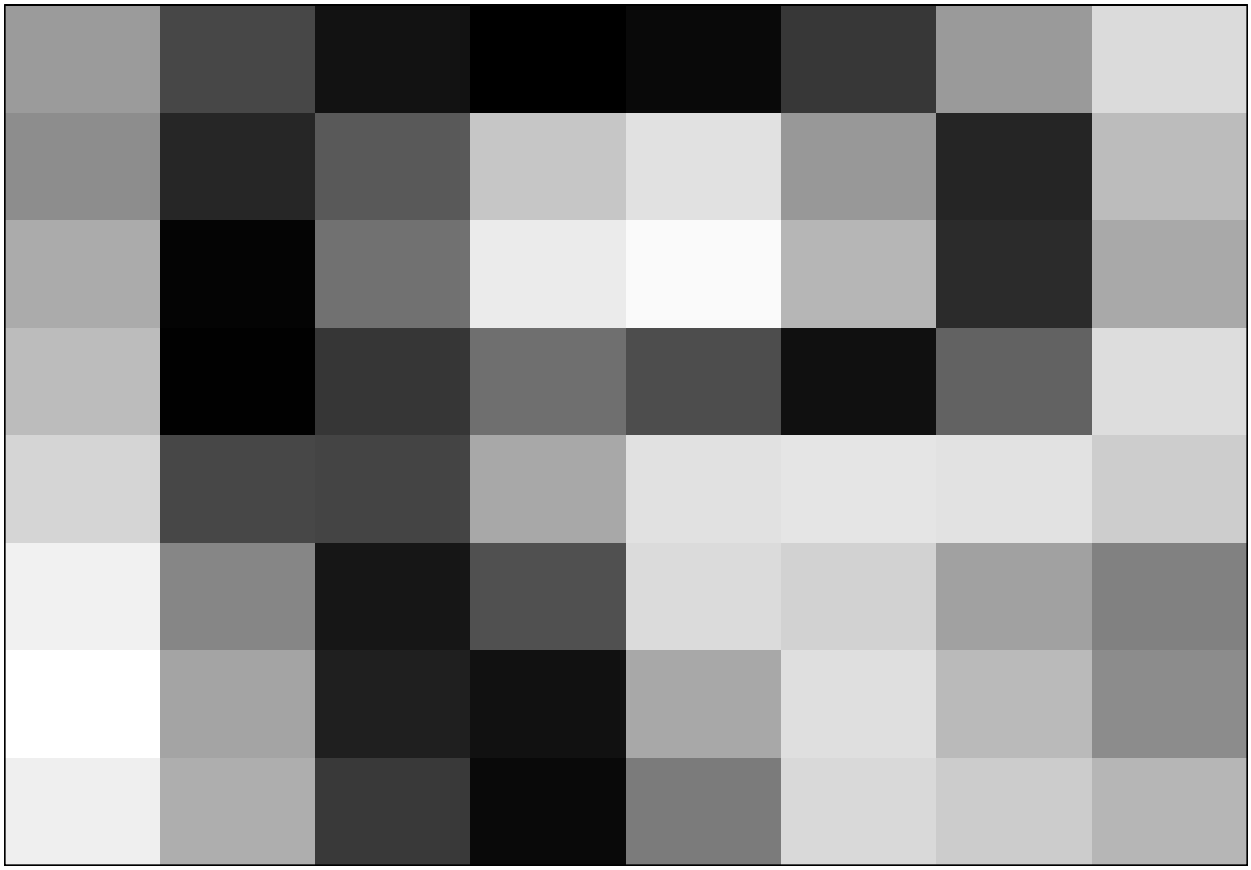
```
plot_letter(p.reconstructed.letter.1, hasletter = FALSE)
```



```
plot_letter(P.data[1,])
```



```
plot_letter(p.reconstructed.letter.2, hasletter = FALSE)
```



```
plot_letter(P.data[2,])
```