Clustering and Dimensionality Reduction in R

Ilke Kas

March 24, 2025

Dataset: dat1.csv

(i) Find an appropriate dimension reduction for the data and justify your choice of reduction technique. [Hint: PCA is useful only when the data follows an elliptical distribution]

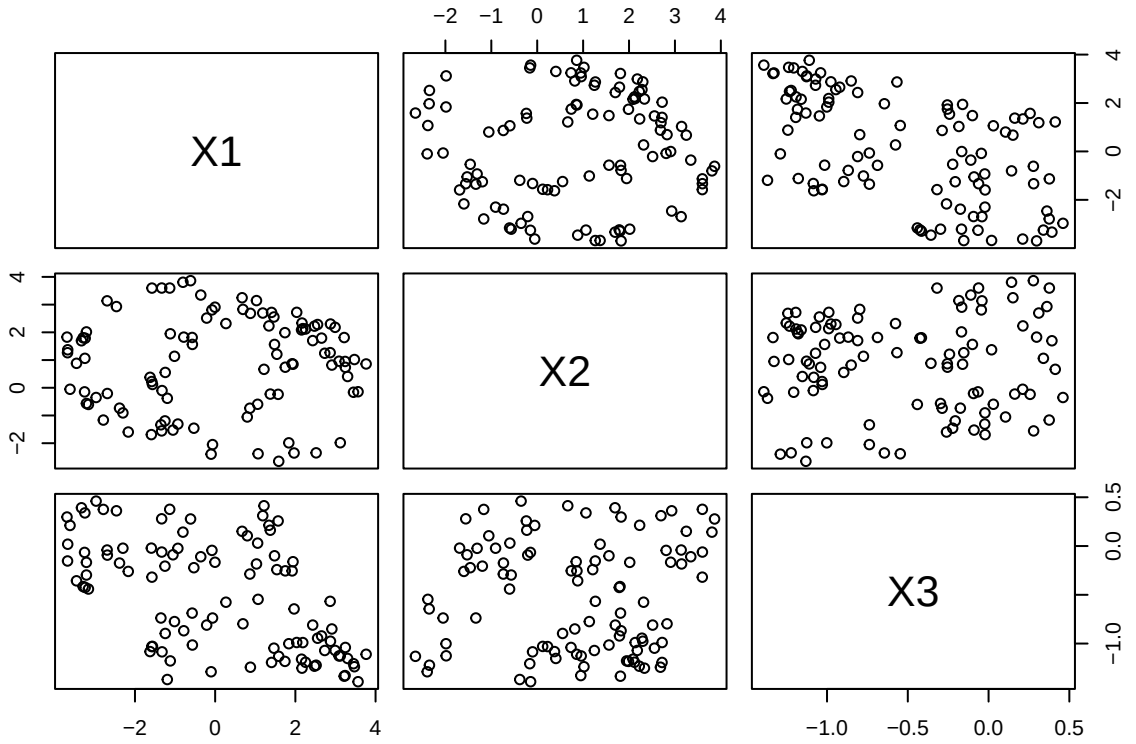
Solution:

We know from our lectures that PCA is only suitable when the data follows an elliptical distribution. In order to check the ellipticity of the data, we used pairs() function which creates a matrix of scatter plots to visualize pairwise relationships between numeric variables in a dataset.

```
# Load the dataset
dat1 <- read.csv("dat1.csv")

# Check structure
head(dat1)
```

```
##          X1         X2          X3
## 1 -2.300681 -0.9067704 -0.02238038
## 2 -2.462008  2.9295473  0.36120948
## 3 -1.332618  3.5987751 -0.06190289
## 4  1.919419  0.8657073 -0.25520272
## 5  1.030310  3.1450451 -0.18372829
## 6 -3.677086  1.3704053  0.01863426
```

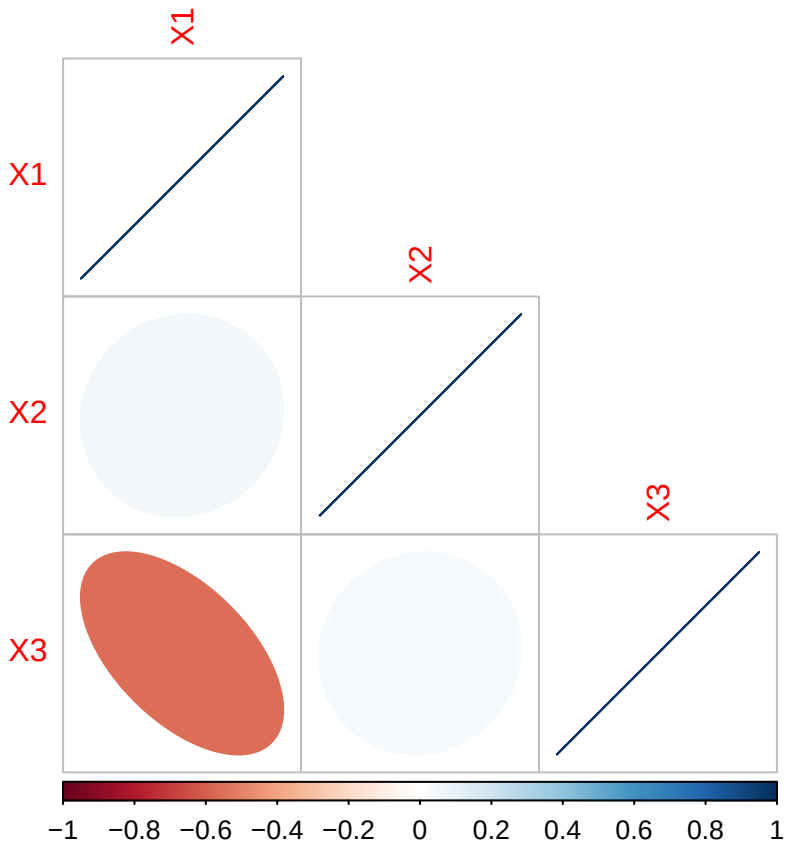
```
# Check for ellipticity and examine the correlaton matrix
pairs(dat1)
```




```
## examine the correlation matrix
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```
corrplot(cor(dat1), type="lower", method="ellipse")
```

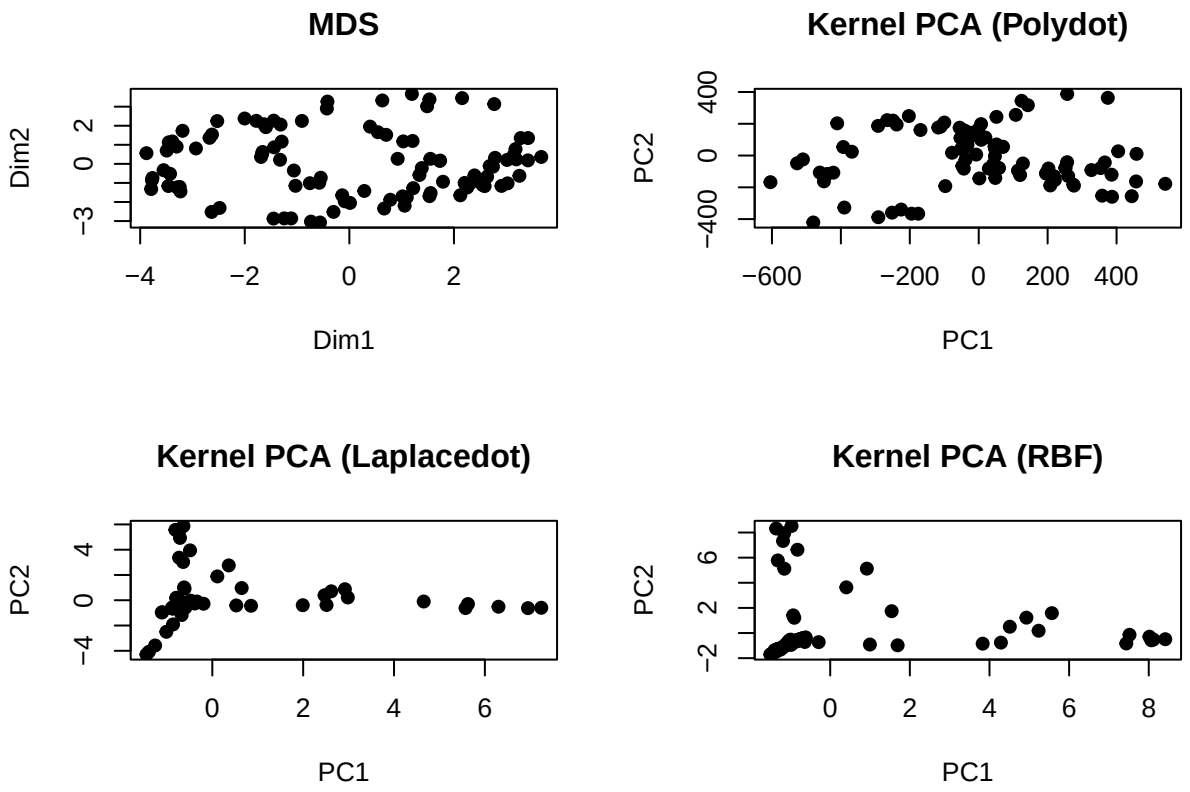


As seen in the pairwise scatter plots and the correlation plot, we can see that the variables are not linearly related to each other. Therefore, applying PCA to this dataset is not a good idea since PCA assumes a linear relationship between inputs, which is not the case in this dataset. Therefore, applying nonlinear dimension techniques will be better. I will try both MDS and kernel PCA to see which one has better result for this dataset.

(ii) Based on the scatter plot of the first two principal components/coordinates, is there clustering in the data? Explain!

```
library(kernlab)

# MDS
mds <- cmdscale(dist(dat1), k = 2)

# Kernel PCA with median-based sigma
euclidean_dists <- dist(scale(dat1), method = "euclidean")
median_sigma <- median(as.vector(euclidean_dists))

manhattan_dists <- dist(scale(dat1), method = "manhattan")
median_sigma_l1 <- median(as.vector(manhattan_dists))

kpc_poly <- kpca(~., data = dat1, kernel = "polydot", kpar = list(degree = 3), features = 2)
kpc_laplace <- kpca(~., data = dat1, kernel = "laplacedot", kpar = list(sigma = median_sigma_l1), featu
kpc_rbf <- kpca(~., data = dat1, kernel = "rbfdot", kpar = list(sigma = median_sigma), features = 2)

# Plot all
par(mfrow = c(2, 2))
```

```
plot(mds, main = "MDS", xlab = "Dim1", ylab = "Dim2", pch = 19)
plot(rotated(kpc_poly), main = "Kernel PCA (Polydot)", xlab = "PC1", ylab = "PC2", pch = 19)
plot(rotated(kpc_laplace), main = "Kernel PCA (Laplacedot)", xlab = "PC1", ylab = "PC2", pch = 19)
plot(rotated(kpc_rbf), main = "Kernel PCA (RBF)", xlab = "PC1", ylab = "PC2", pch = 19)
```



As seen in the plots of first and the second prinicipal components of applied nonlinear dimension reduction techniques, I think the kernel PCA with polydot kernel best captures the cluster structure of the dataset. When I looked at the scatter plot of PC1 and PC2 of Polydot kernel PCA, I see clustering of the data. Other kernels also has better clustering compared to MDS. Therefore, I would choose kernel PCA for this dataset.

(iii) Determine the appropriate number of clusters in the dataset based on the original data.

```
# Use elbow method or silhouette
## elbow plot of Within cluster sum of squares
library(factoextra)
```

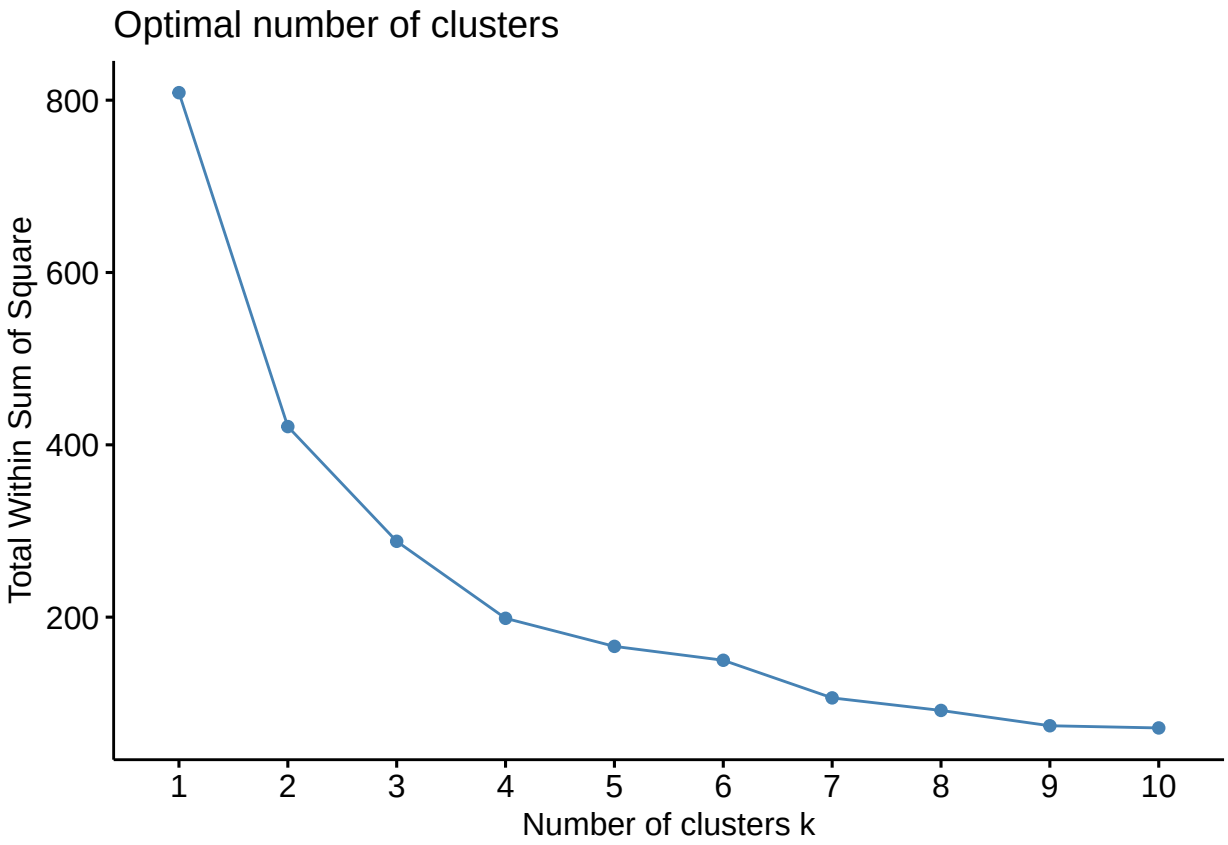
```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

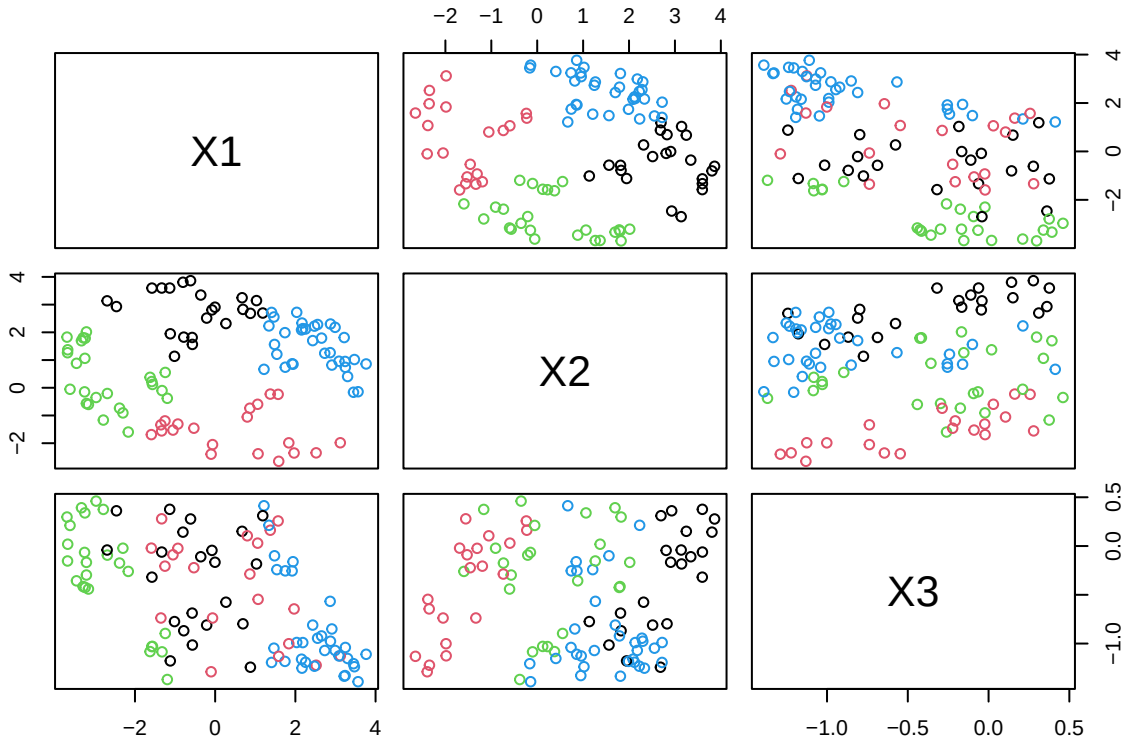
```
## The following object is masked from 'package:kernlab':
##
##     alpha
```

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

```
library(ggplot2)
fviz_nbclust(dat1, kmeans, method="wss")
```



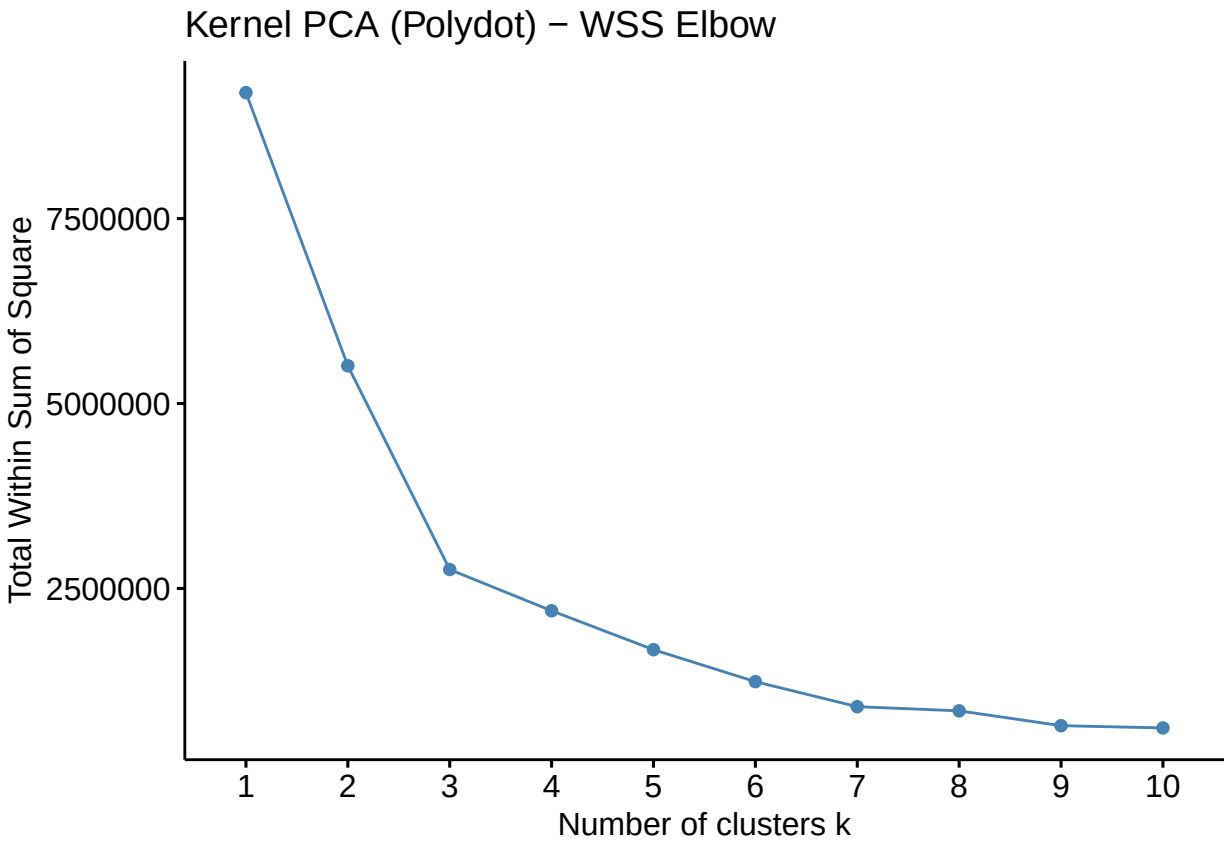

```
# apply kmeans to original data with appropriate number of clusters in the dataset
km_dat1 <- kmeans(dat1,center=4)
pairs(dat1, col=km_dat1$cluster)
```



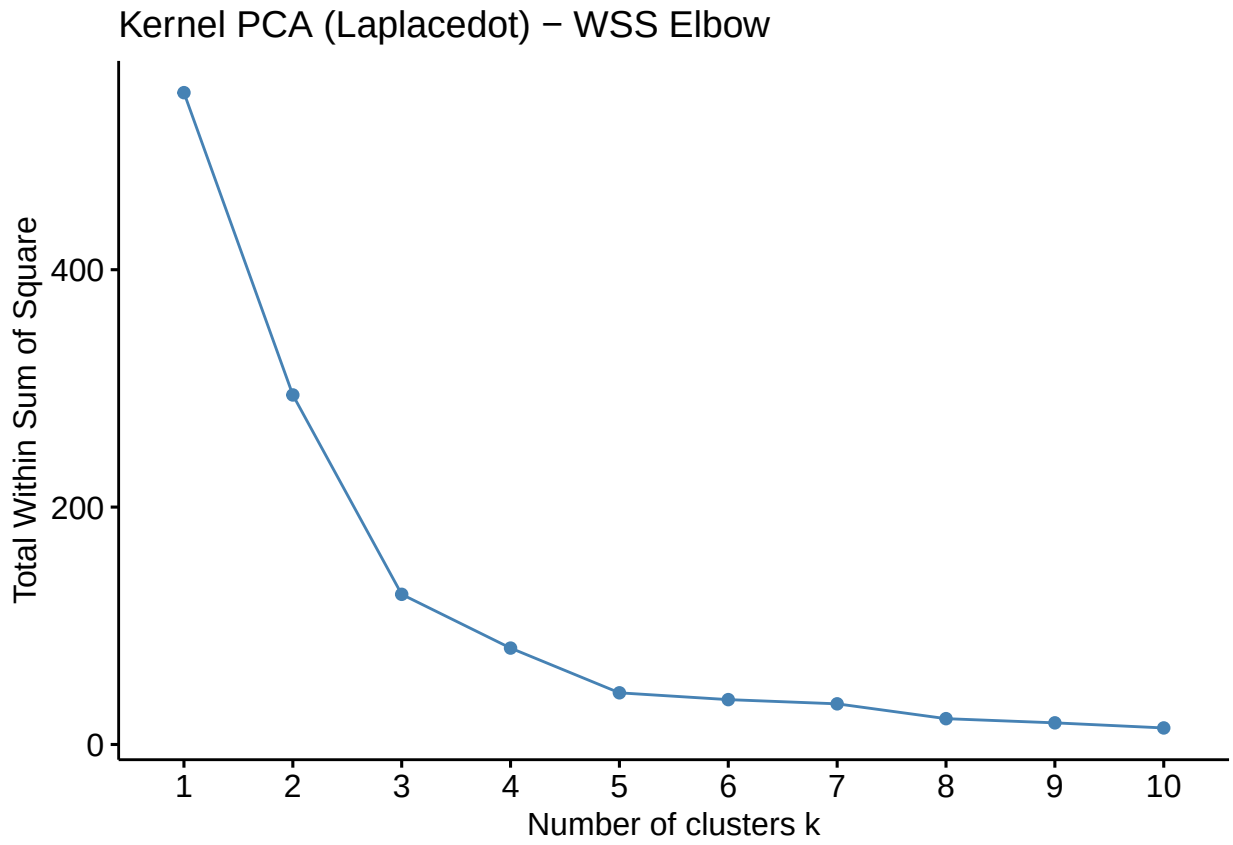
Based on the elbow method, the optimal number of clusters should be 4 when within-cluster sum of squares are used for clustering of the **original data**. I also plot the scatter plots of the pair of variables with this clustering. In these plots, the original data shows four visually distinguishable clusters, primarily separated along variables X1 and X2, while X3 offers minimal contribution to clustering.

(iv) Determine the appropriate number of clusters in the dataset based on the first two principal components/coordinates.

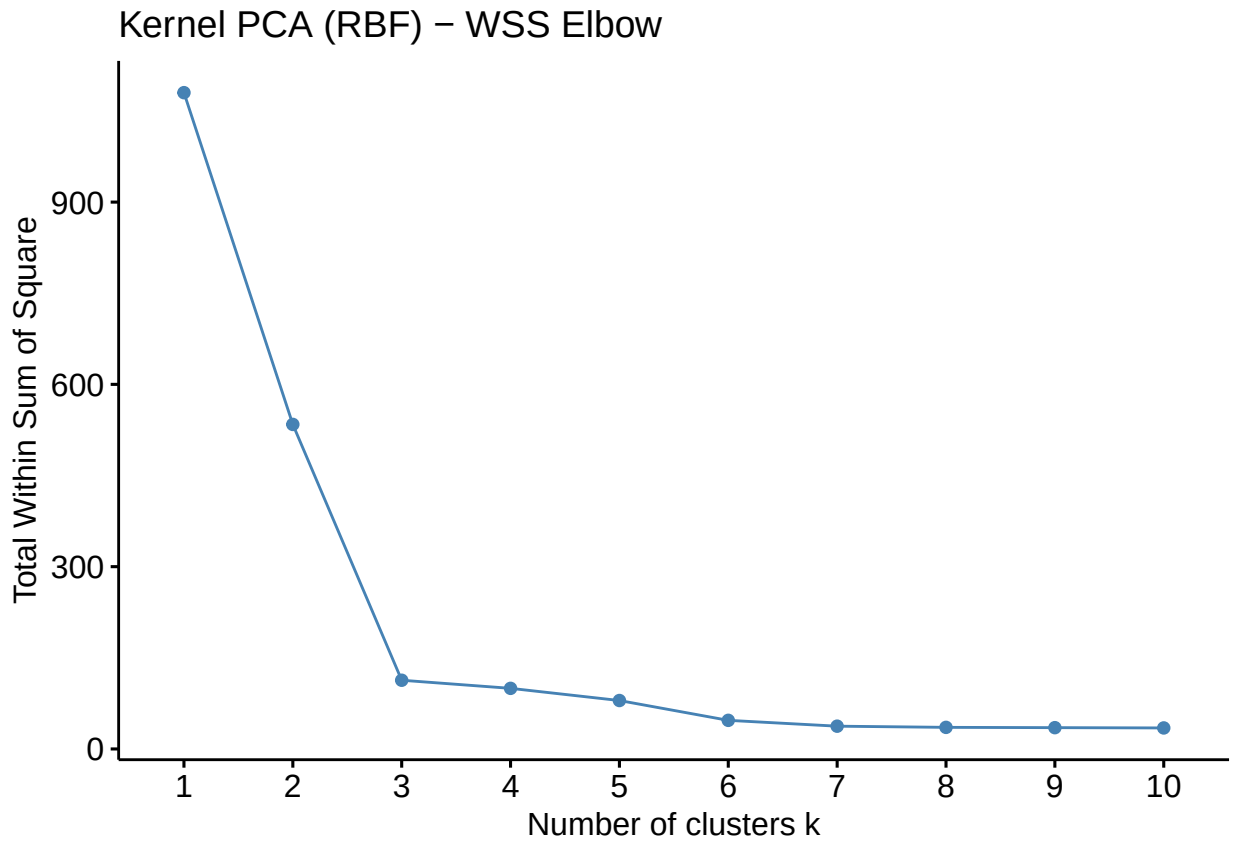
```
library(kernlab)

# Kernel PCA with median-based sigma
euclidean_dists <- dist(scale(dat1), method = "euclidean")
median_sigma <- median(as.vector(euclidean_dists))

manhattan_dists <- dist(scale(dat1), method = "manhattan")
median_sigma_l1 <- median(as.vector(manhattan_dists))

kpc_poly <- kpca(~., data = dat1, kernel = "polydot", kpar = list(degree = 3), features = 2)
kpc_laplace <- kpca(~., data = dat1, kernel = "laplacedot", kpar = list(sigma = median_sigma_l1), featu
kpc_rbf <- kpca(~., data = dat1, kernel = "rbfdot", kpar = list(sigma = median_sigma), features = 2)

# WSS plots with titles
fviz_nbclust(rotated(kpc_poly), kmeans, method = "wss") + ggtitle("Kernel PCA (Polydot) - WSS Elbow")
```




```
fviz_nbclust(rotated(kpc_laplace), kmeans, method = "wss") + ggtitle("Kernel PCA (Laplacedot) - WSS Elb
```



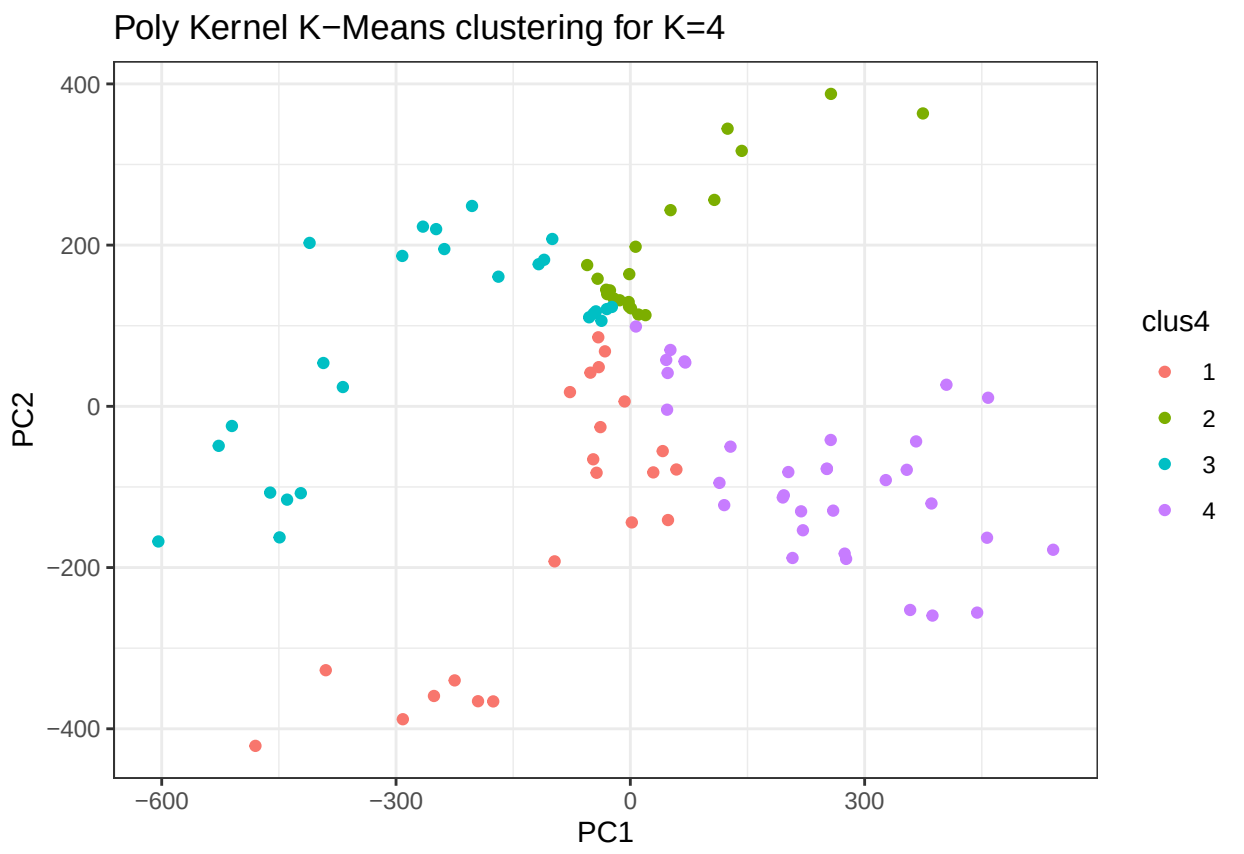

```
fviz_nbclust(rotated(kpc_rbf), kmeans, method = "wss") + ggtitle("Kernel PCA (RBF) - WSS Elbow")
```



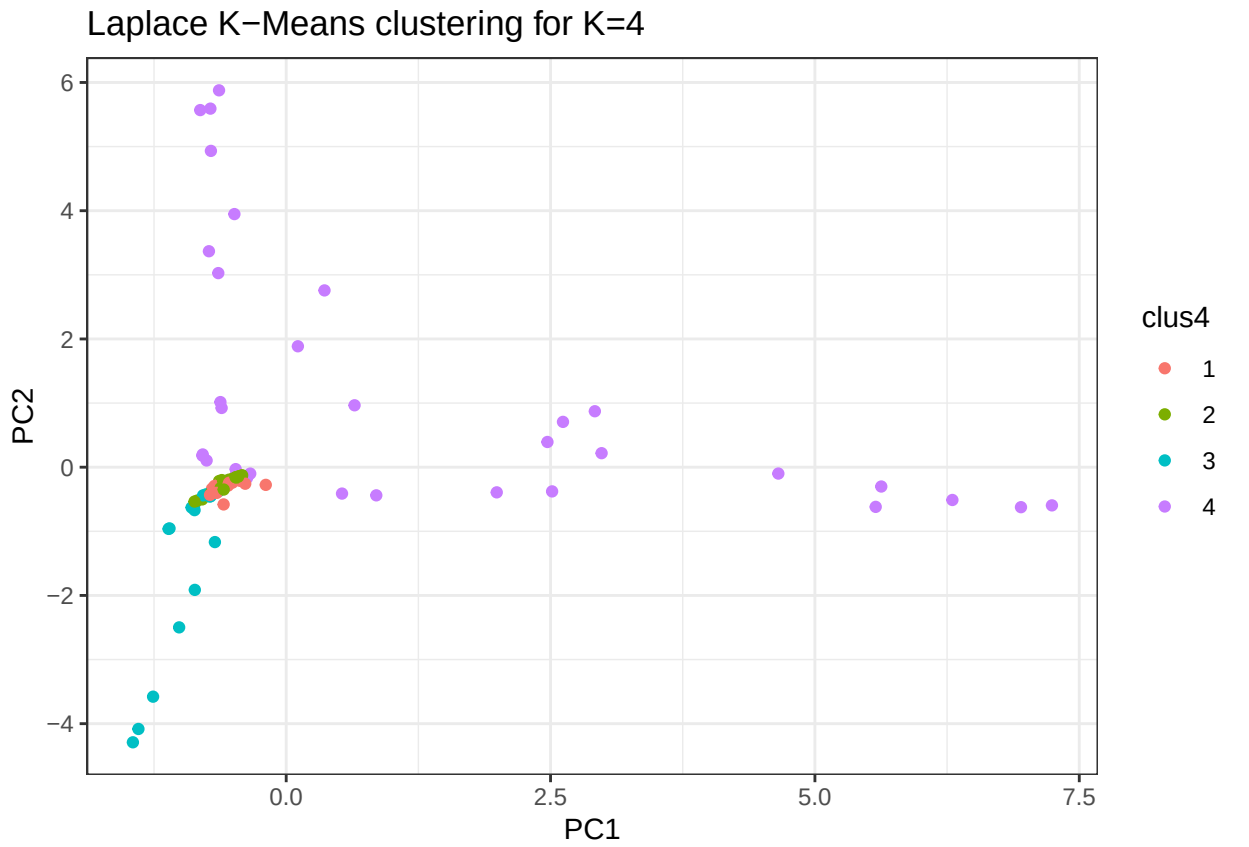
As seen I applied kernel PCA with three different kernel types: polydot, laplacedot, and rbfdot. Since the RBF kernel uses the squared Euclidean distance in its formula, I used euclidean distance when calculating the sigma value of it. I took the median of it as a heuristic since I thought it may be a good scale for how close most points are to each other. For the Laplace kernel's sigma value, I used Manhattan distance to calculate the sigma value since it uses L1 Norm in its formula and again take the median of it. According to both polydot and laplacian kernel, the optimal cluster value is 4 when we used the elbow method. However, rbf kernel PCA results with 3 as the optimal cluster value. Since the majority of the reduction methods says 4, I will say that the optimal number of clusters will be 4 this dataset.

```
library(kernlab)
library(ggplot2)

# Kernel PCA with median-based sigma
euclidean_dists <- dist(scale(dat1), method = "euclidean")
median_sigma <- median(as.vector(euclidean_dists))

km_dat1<- kmeans(dat1, center=4)

kpc_poly <- kpca(~., data = dat1, kernel = "polydot", kpar = list(degree = 3), features = 2)
kpc_laplace <- kpca(~., data = dat1, kernel = "laplacedot", kpar = list(sigma =median_sigma_l1), featur

kpc_poly_pc <- rotated(kpc_poly)
kpc_laplace_pc <- rotated(kpc_laplace)

df_dat1_pc_poly <- data.frame(PC1=kpc_poly_pc[,1],PC2=kpc_poly_pc[,2],
                          clus4=as.factor(km_dat1$cluster) )
df_dat1_pc_laplace <- data.frame(PC1=kpc_laplace_pc[,1],PC2=kpc_laplace_pc[,2],
```

```
                        clus4=as.factor(km_dat1$cluster) )

g4_poly <- ggplot(df_dat1_pc_poly, aes(x=PC1, y=PC2, color=clus4)) + geom_point() +
  theme_bw() + labs(title="Poly Kernel K-Means clustering for K=4")

g4_laplace <- ggplot(df_dat1_pc_laplace, aes(x=PC1, y=PC2, color=clus4)) + geom_point() +
  theme_bw() + labs(title="Laplace K-Means clustering for K=4")

print(g4_poly)
```




```
print(g4_laplace)
```



As seen both clustering, the Poly Kernel with degree 3 has the best dimension reduction compared the other kernels in this dataset. Therefore, by using all of this analysis, I am saying that applying Polydot Kernel with degree 3 and applying the clustering with k=4 has the best results and should be used for this dataset.

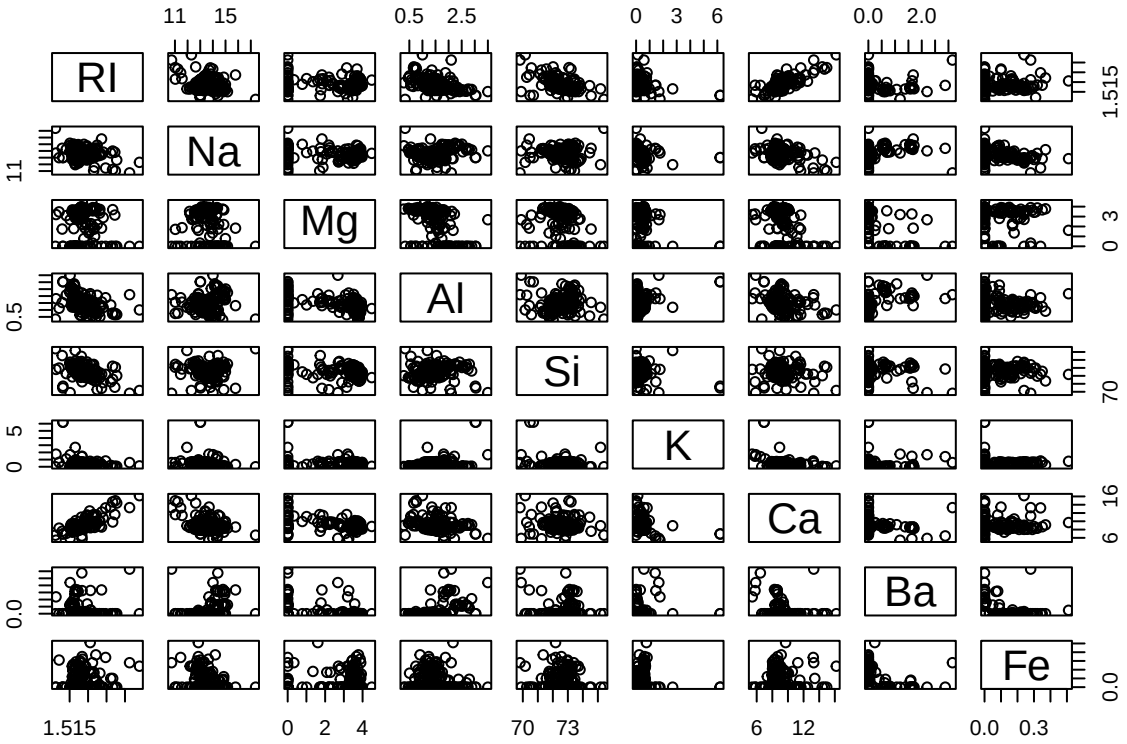
Dataset: glass.csv

(i) Find an appropriate dimension reduction for the data and justify your choice of reduction technique. [Hint: PCA is useful only when the data follows an elliptical distribution]

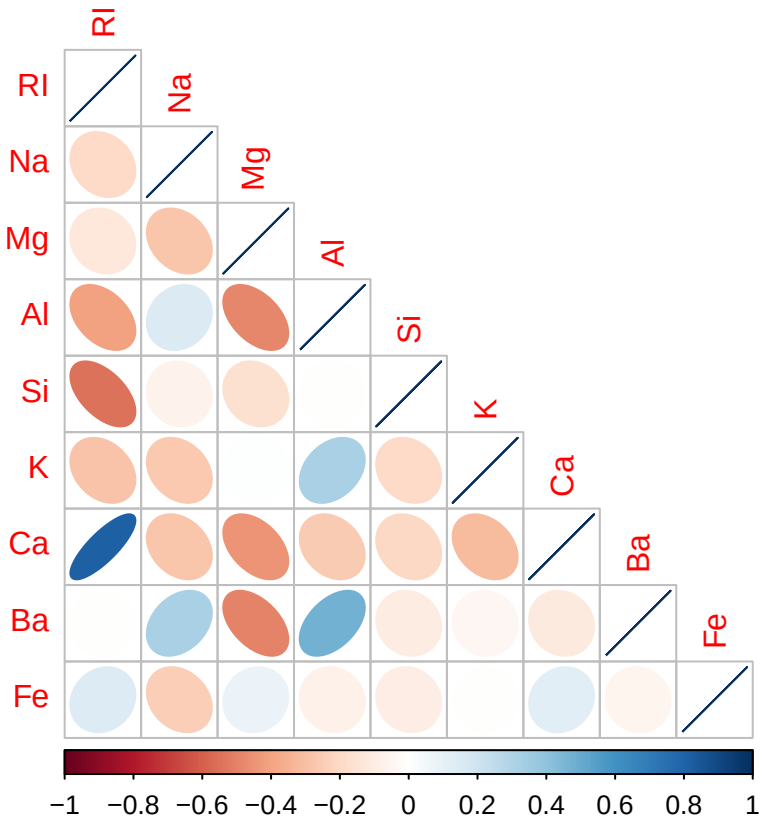
```
# Load the dataset
glass <- read.csv("glass.csv")

# Check structure
head(glass)
```

```
##        RI    Na   Mg   Al    Si    K   Ca Ba   Fe
## 1 1.52101 13.64 4.49 1.10 71.78 0.06 8.75  0 0.00
## 2 1.51761 13.89 3.60 1.36 72.73 0.48 7.83  0 0.00
## 3 1.51618 13.53 3.55 1.54 72.99 0.39 7.78  0 0.00
## 4 1.51766 13.21 3.69 1.29 72.61 0.57 8.22  0 0.00
## 5 1.51742 13.27 3.62 1.24 73.08 0.55 8.07  0 0.00
## 6 1.51596 12.79 3.61 1.62 72.97 0.64 8.07  0 0.26
```

```
# Check for ellipticity and examine the correlaton matrix
pairs(glass)
```



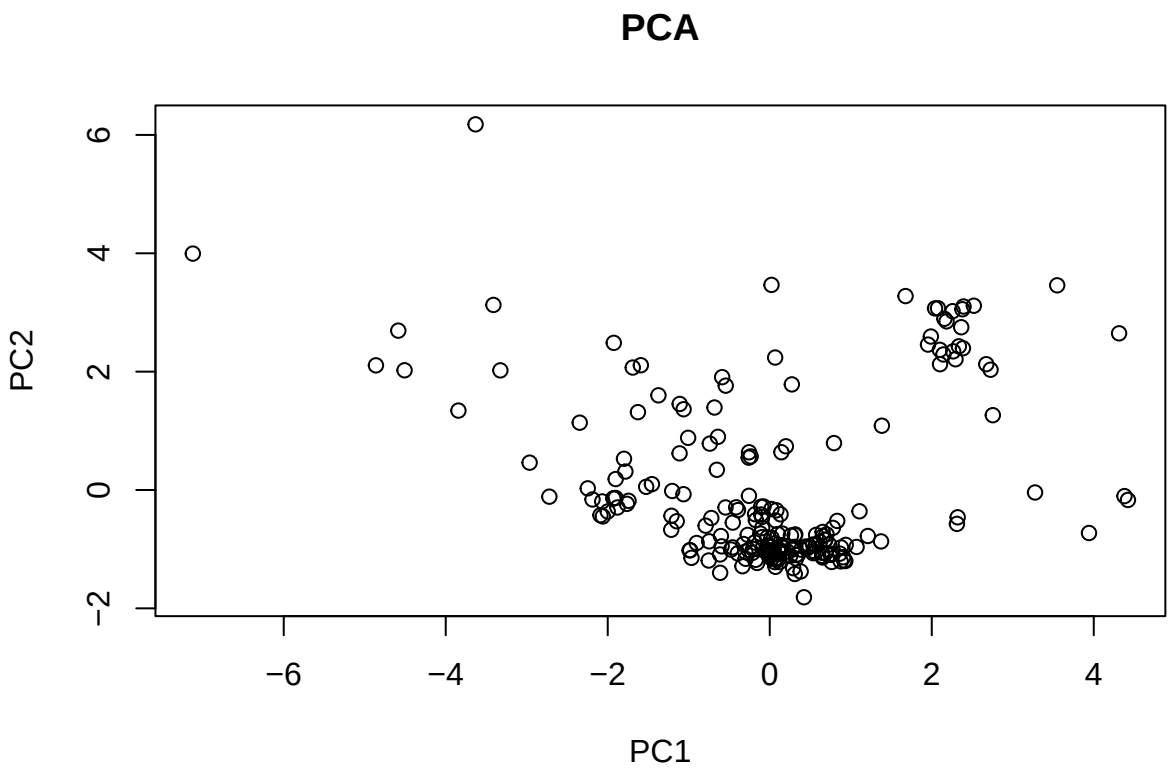

```
## examine the correlation matrix
library(corrplot)

corrplot(cor(glass), type="lower", method="ellipse")
```



As seen in the pairwise scatter plots and the correlation plot, we can see that the many variables seems like linearly related to each other such as RI and CA, FE and nearly all others, etc. Therefore, applying PCA to this dataset is a good idea since PCA assumes a linear relationship between inputs, which is the case in this dataset in my opinion.

(ii) Based on the scatter plot of the first two principal components/coordinates, is there clustering in the data? Explain!

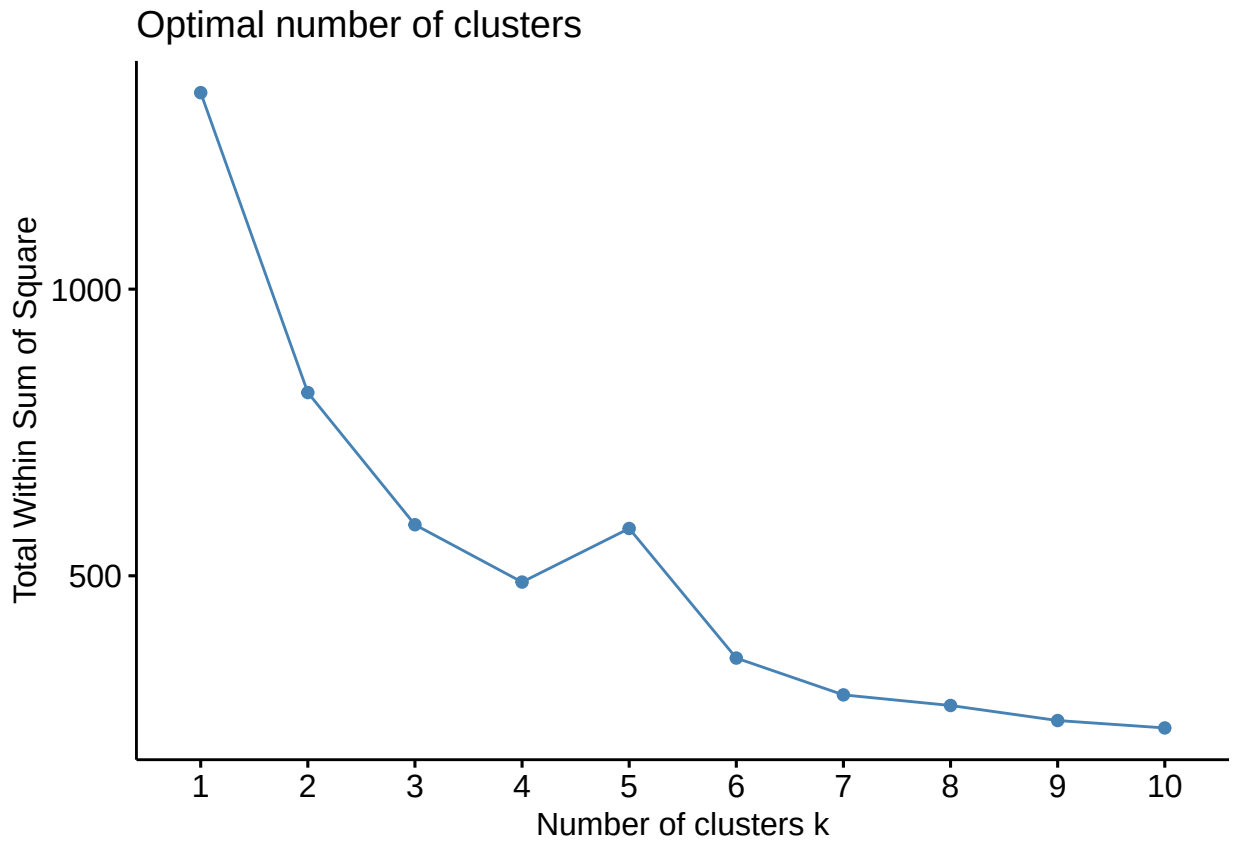
```
# Scatter plot of first two PCs
pca <- prcomp(glass, center = TRUE, scale = TRUE)$x[,1:2]
plot(pca, main="PCA", xlab="PC1",ylab="PC2")
```



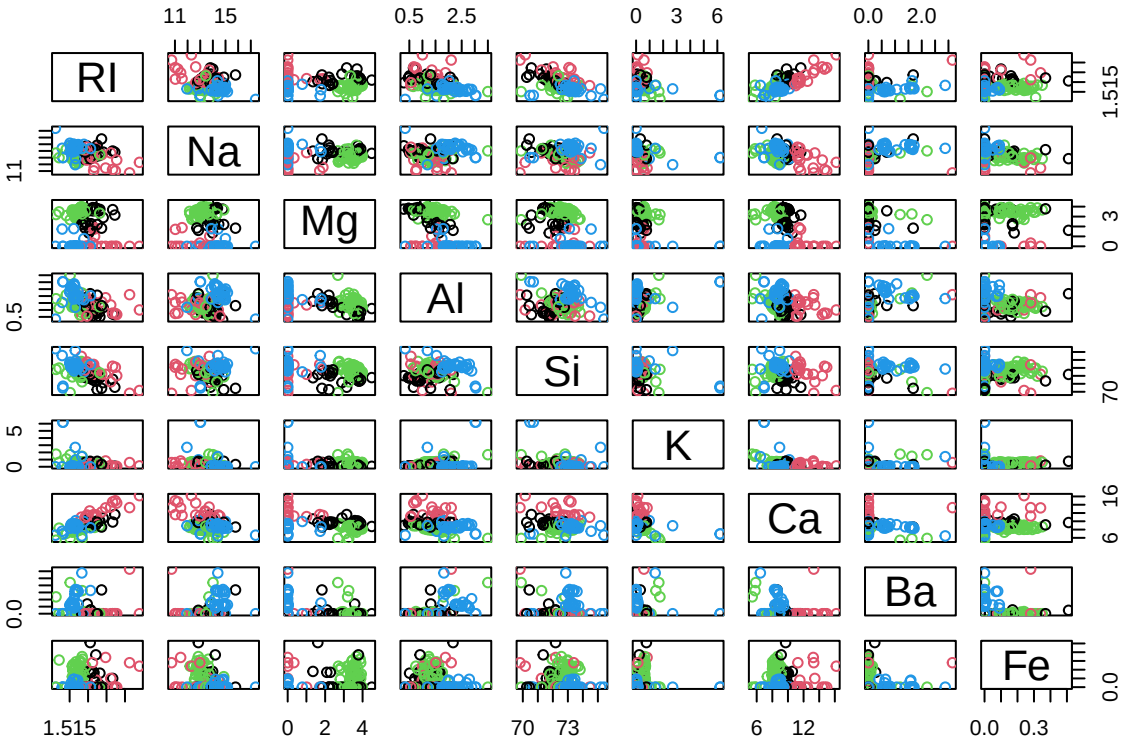
We can talk about clustering according to the graph above since there are several dense groups of points appear in distinct regions of the plot. However, the separation between clusters is not very strong, indicating that PCA captures only part of the underlying cluster structure.

(iii) Determine the appropriate number of clusters in the dataset based on the original data.

```
# Use elbow method or silhouette
# Use elbow method or silhouette
## elbow plot of Within cluster sum of squares
library(factoextra)
library(ggplot2)
fviz_nbclust(glass, kmeans, method="wss")
```



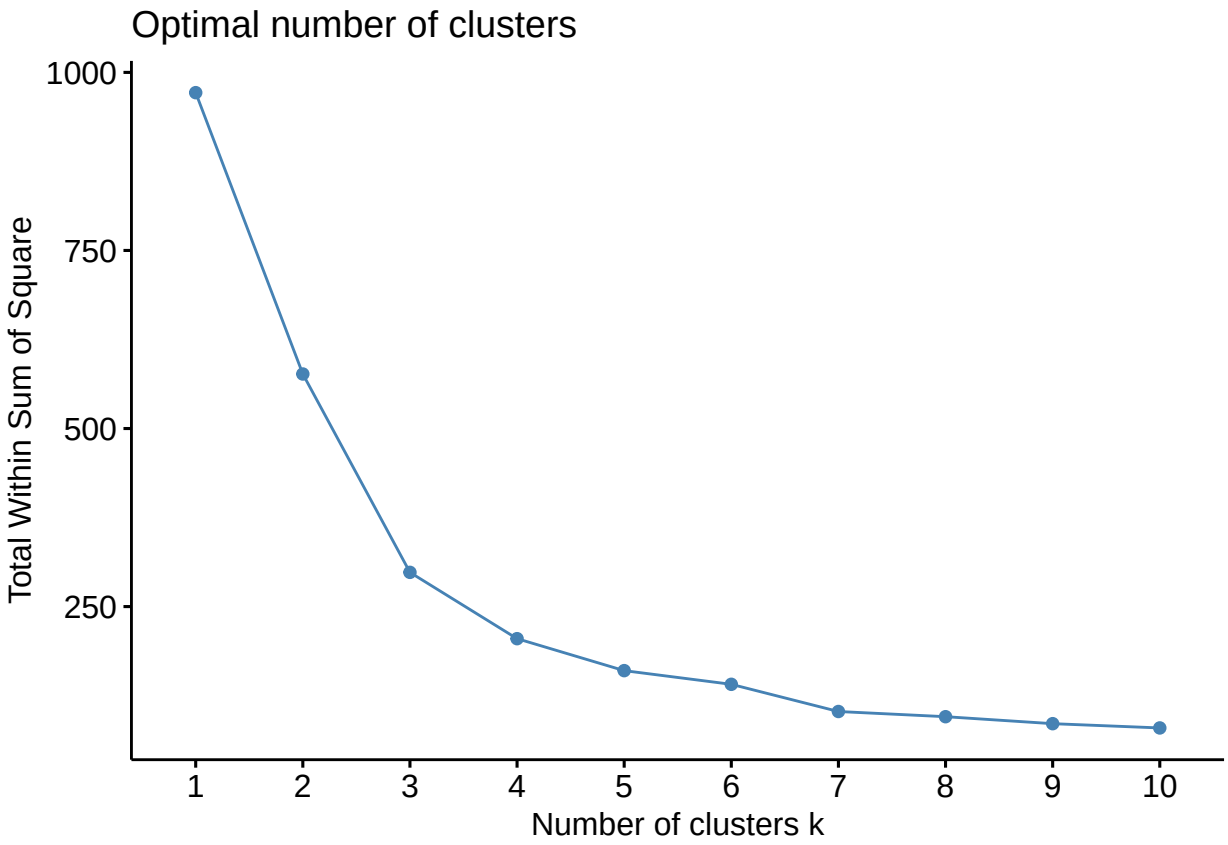

```
# apply kmeans to original data with appropriate number of clusters in the dataset
km_glass <- kmeans(glass,center=4)
pairs(glass, col=km_glass$cluster)
```



According to the plot above, the optimal number of clusters is 4. It is visible that there is a drop in within-cluster sum of squares (WSS). There is an increase in WSS when k=5. In theory, there should not be increase in WSS when k increase. However, due to randomness of the k-means algorithm these this can be happen. Anyway, according to the elbow rule, k value should be 4 optimally.

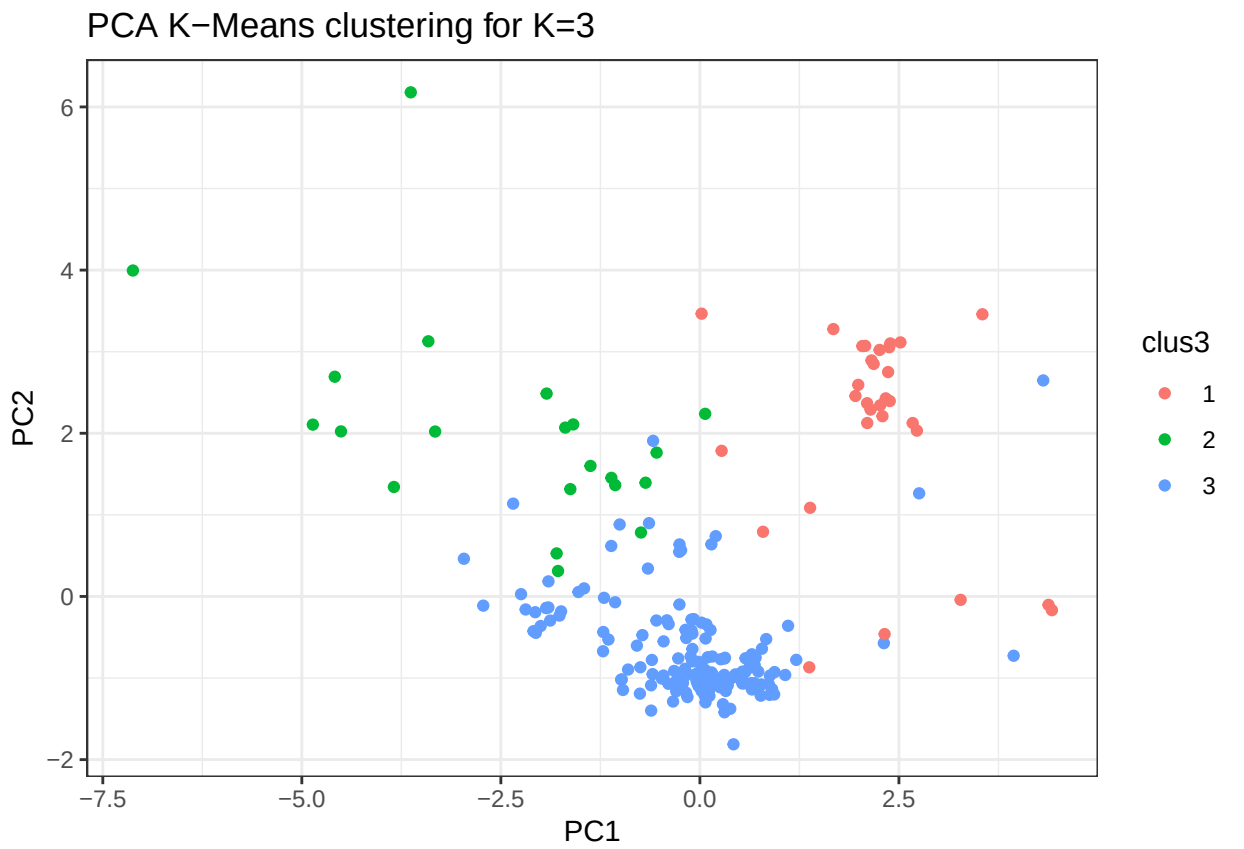
(iv) Determine the appropriate number of clusters in the dataset based on the first two principal components/coordinates.

```
glass_pca <- prcomp(glass, center = TRUE, scale = TRUE)$x[,1:2]
fviz_nbclust(glass_pca, kmeans, method="wss")
```



Based on the elbow plot using the first two principal components, the optimal number of clusters is 3. The total within-cluster sum of squares shows a sharp drop up to k=3. After that point, the decrease becomes more gradual. So, we can say that 3 clusters capture the main structure of this dataset when it's dimension is reduced by PCA.

```
glass_pca <- prcomp(glass, center = TRUE, scale = TRUE)
km_glass <- kmeans(glass,center=3)
df_glass_pc <- data.frame(PC1=glass_pca$x[,1], PC2=glass_pca$x[,2],
                          clus3=as.factor(km_glass$cluster))
glass_g3 <-  ggplot(df_glass_pc, aes(x=PC1, y=PC2, color=clus3)) + geom_point() +
  theme_bw() + labs(title="PCA K-Means clustering for K=3")
print(glass_g3)
```



The PCA plot with K-means clustering for k=3 shows that the data separates pretty well into three groups, especially with one tight cluster near the center. The other two clusters are more spread out, but there's still a clear distinction between them in the PCA space.