# STAT 557 Project Report: K means, K center and KNN

## Problem Statement

Typically, the wine could be classified according to various methods, including place of origin, appellation, vilification method, vantage, and sweetness[[1]](#footnote-1). The development of analytical chemistry has enabled a close examination of chemical components in the wine, which could be as much as more than 20 kinds. The precisely detected chemical components together with their quantity in the wine provide useful dataset for classifying the wine through a data mining approach.

This project extended the classification of wines based on the chemical components and their quantity in the wine using the clustering method. The data set used, data preparation, classification algorithm selected, classification performance and result interpretation are included in the following report.

## Data Set Description

The data set used is the “Wine Data Set”, which comes from the UCI Machine Learning Repository[[2]](#footnote-2). These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. The attributes name, data type and description is given in the Table 1.

Table 1 Variables Description

|  |  |  |
| --- | --- | --- |
| Attribute Name | Data Type | Description |
| Identifier | Numeric | Labels for three kinds of wine |
| Alcohol | Numeric | 13 attributes used for training the classification model, the attributes are not scaled in the dataset and will be scaled accordingly based on the needs. |
| Malic acid | Numeric |
| Ash | Numeric |
| Alcalinity of ash | Numeric |
| Magnesium | Integer |
| Total phenols | Numeric |
| Flavanoids | Numeric |
| Nonflavanoid phenols | Numeric |
| Proanthocyanins | Numeric |
| Color intensity | Numeric |
| Hue | Numeric |
| OD280/OD315 of diluted wines | Numeric |
| Proline | Integer |

All the attributes are continuous and no missing value is found for each attribute. The wine data set include 178 records in total, and the records distributed evenly in all the three classes as 59 in class one, 71 in class two, and 48 in class three.

## Data Preparation

Before proceeding to the classification, this study first inspected general distribution and potential outliers with respect to each attribute with boxplot and IQR. To compare the distribution of the 13 attributes in the same boxplot, the attributes data are scaled to the center as shown in the following Figure 1.

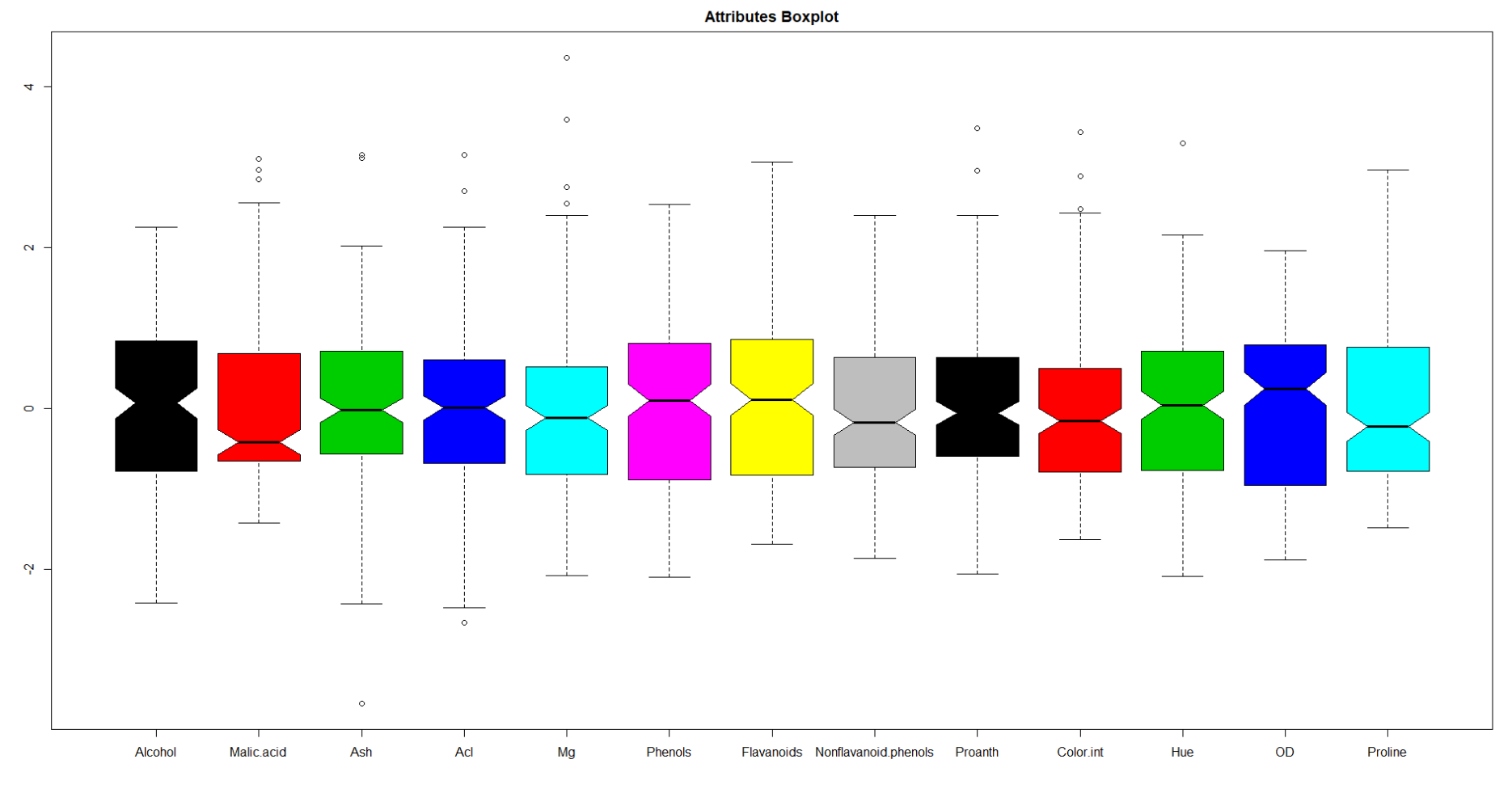


Figure 1 Comparison of Attributes Distribution

Based on the boxplot output, 7 of the 13 attributes have data points fall out of IQR boundary, which could be identified as potential outliers, while the proportion of such points is relative small (at most 4 points for each variable) comparing to the number of records (178). Therefore, this study did not process the identified outliers before conducting the classification.

In order to conduct the linear discriminant analysis, this study also checked the multicollinearity between the 13 attributes by checking the Variance Inflation Factor (VIF) as shown in the following table.

Table 2 Multicollinearity Check

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | VIF | Attributes | VIF |
| Alcohol | 2.460372 | Flavanoids | 7.029350 |
| Malic acid | 1.656647 | Nonflavanoid phenols | 1.796380 |
| Ash | 2.185448 | Proanthocyanins | 1.975683 |
| Alcalinity of ash | 2.238732 | Color intensity | 3.026304 |
| Magnesium | 1.417855 | Hue | 2.551447 |
| Total phenols | 4.334519 | OD280/OD315 of diluted wines | 3.785473 |
| Proline | 2.823849 |  |  |

In this study, the general rule of thumb of 10 as VIF (when the VIF for one variable is more than 10, it has potential multicollinearity problem) is applied([O’brien, 2007](#_ENREF_1)). By checking each attribute in the dataset, none of the attributes has a VIF greater than 10. While the “Flavanoids” has a relatively high VIF compared with other attributes, which would be removed when training the model and used as a feature selection approach.

## K Means Clustering

We apply the K means algorithm now to our data set. We are going to test different K values and find the optimal one. Also, we will use validation to see how the error rates are varying within the training and testing sets.

**A theoretical approach to determine the optimal k:[1]**

Elbow method is a theoretical approach to find the optimal K. We plot the SSE against the number of clusters and look for a sharp drop in errors. The plot suggests that k=3 or 4 are going to be the optimal values.

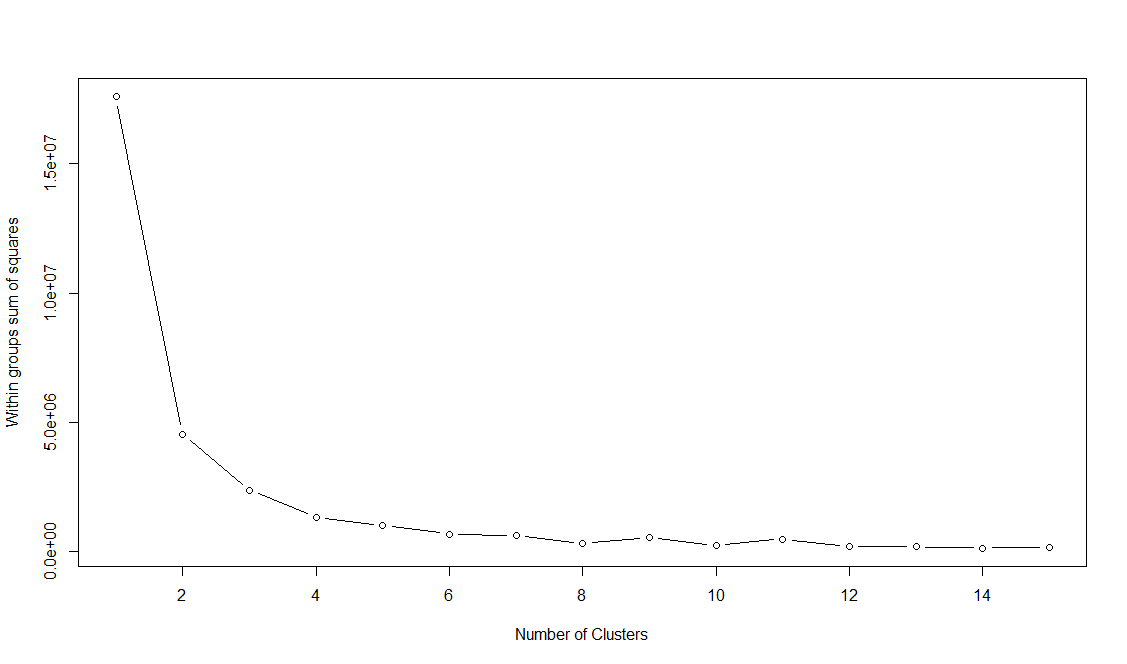


Figure 2 Give a figure name here

**Applying K means to the complete data set**

This is a way to determine how the K means will perform without any validation method. We applied K means to the complete data set.

Table 3 give a table name

|  |  |
| --- | --- |
| **K value** | **Error rate (in %)** |
| 2 | 34 |
| 3 | 52 |
| 4 | 94 |
| 5 | 82 |

Figure 3 give a figure name

## Using K-fold Cross Validation to choose number of Prototypes and studying the change of error rates:

Since the data set is small, we will choose a 5-fold cross validation. Since we have 178 data, each fold will have different number of data samples.

We will try out different K values and see how the error rates change within training and testing sets for different number of prototypes.

**5-fold cross validation: [2]**

Based on the previous elbow diagram, it only makes sense to test for k value 2-5.

5-fold means the data set will be split into 5 folds and each one of them will be used for training while the rest for testing.

We used nstart=100, meaning it will choose 100 different initial starting points and will return the results for the best one. The table below shows the results and errors on training and testing sets.

Table 4 give a table name

|  |  |  |
| --- | --- | --- |
| **K value** | **Training error rate (%)** | **Testing error rate (%)** |
| 2 | 58.5 | 68.86 |
| 3 | 82.5 | 63 |
| 4 | 73.8 | 76.2 |
| 5 | 85 | 93.4 |

Figure 4 give a figure name

The results indicate lowest training error rate for k=2 and lowest testing error rate for k=3.

So a k value of 2 or 3 is probably optimal.

**Another method to determine number of prototypes: Validation set approach: [4]**

We tried to find if we can use the simple validation set approach to support our findings in cross validation. We have divided the data set into training and testing data in 80:20 ratios. The table below shows the results:

Table 5 give a table name

| **K** | **Training error rate (%)** | **Testing error rate (%)** |
| --- | --- | --- |
| 2 | 33 | 34 |
| 3 | 62 | 68 |
| 4 | 77 | 92 |
| 5 | 97 | 81 |

Figure 5 give a figure name

Once again, K value of 2 shows minimum error rates in classification while as K value of 3 also seems to be reasonable.

**Conclusion from cross validation and error rates:**

A K value of 2 or 3 is optimal. Even though we have 3 different wine classes, 2 clusters seem to be superior in terms of performance. Also, since the data set is only containing 178 observations, our k means algorithm performs poor in terms of ability to classify, which is why we are getting high error rates.

## Can Dimension reduction improve classification? Applying PCA :

We will use Principal Component Analysis to see if we can improve classification by reducing the dimension of our dataset.

**What is Principal component analysis?**

There are 13 attributes could be used to conduct the classification, in order to examine if the dimension reduction approach could improve the wine classification performance, this study applied the PCA to reduce the raw data into a lower feature space.

Through the PCA, the most contributing attribution for each principle components could be identified. The performance of how principle components could represent the data set variation could be visualized as following Figure 6.

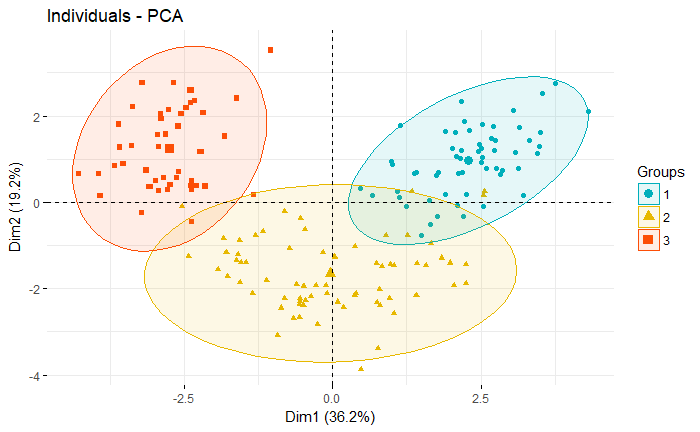
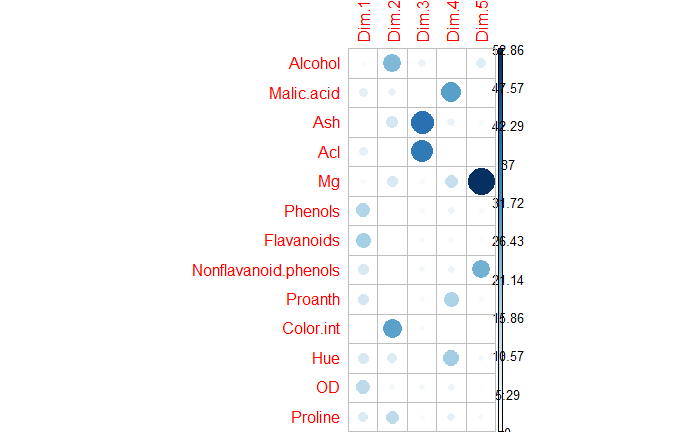


Figure 6 PCA Result

Based on Figure 6, the attributes Phenols and Flavanoids contribute most for first principle component, and it explained 36.2% of the variation of the data. The first principle component also well separated group 1 and group 3.

In the PCA, the aim is to represent the greatest variation of data with lower number of identified principle components. As the number of principle components increases, the explained variation of data also increases until the component number reach to the maximum, which is the number of variables (13 in our case). In order to determine the ideal number of principle components which could represent a relative high proportion of variation in the data, we ranked the identified principle components by proportion of variation in the data they could explained, from high to low. Then plot the variation explained by each variable and the accumulative variation explained in the flowing figures.

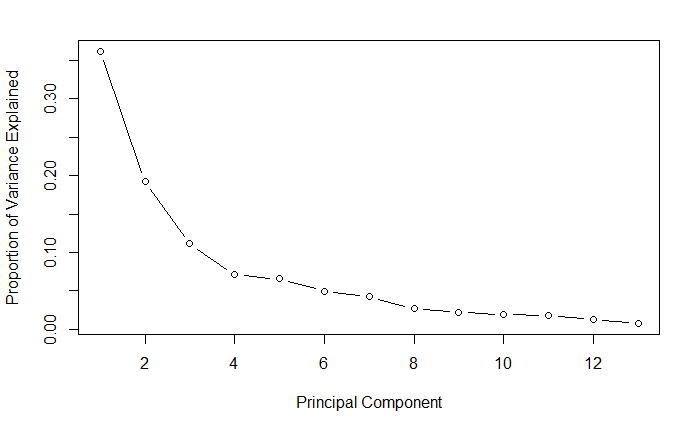


Figure 7 Proportion of Variation Explained by Each Attributes

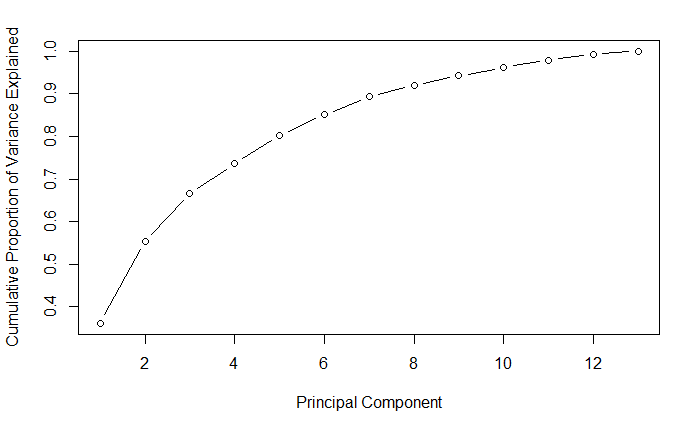


Figure 8 Accumulative Proportion of Variation Explained

Based on the result in Figure 6, the explained variation increases significantly (from 0.35 to above 0.9) when we consider the first 7 principle components, then the accumulative proportion increases slowly.

**Applying PCA with K means algorithm: [3]**

Now with the previous concept, we can apply PCA to our K mean algorithm. We will only choose k values 2 and 3 because these two are optimal. Then we will vary the number of principal components to see for improvements in classification.

The table below summarizes our findings:

Table 6 give a table name

|  |  |  |
| --- | --- | --- |
| **PCA (Number of components)** | **Error rate in % for K=2** | **Error rate in % for K=3** |
| 2 | 42 | 61 |
| 3 | 42 | 70 |
| 4 | 82 | 41 |
| 5 | 82 | 71 |
| 6 | 70 | 63 |
| 7 | 70 | 63 |
| 8 | 70 | 65 |
| 9 | 57 | 63 |
| 10 | 84 | 63 |
| 11 | 84 | 98 |
| 12 | 84 | 71 |
| 13 | 84 | 98 |

Figure 9 name here

**Conclusion on implementation of PCA:**

The results indicate that PCA is able to improve our classification by k means. Applying k means with 4 principal components and using a k value of 3 produces the lowest error rate, 41%. So only the first 4 principal components are able to improve our classification.

## K-Nearest Neighbor (KNN) Classification

This study split the entire data set into 80% training set and 20% testing set, then trained the KNN classifier with all the 13 wine attributes in the training dataset. Particularly, a random seed (9) is fixed to make sure the test is repeatable, and the wine in each of the three classes is distributed in the testing and training set with similar proportion.

To compare the classifier’s performance on both the training and testing data set, a 10-fold cross validation was conducted in the training dataset; a classification test of the trained model is also conducted on the testing data set (which is not used for training the model). To find the optimal K value for the best classifier, different K value was tested in test on both training and testing dataset. To avoid the tie vote in the result, all the K value are odd number. The corresponding classifiers’ performances with different K were listed and plotted in the following table and figure.

Table 7 Classifier Performance V.S. Number of Neighbors

| **k** | **Training Set** | | **Testing Set** | |
| --- | --- | --- | --- | --- |
| **Accuracy** | **Error Rate** | **Accuracy** | **Error Rate** |
| 1 | 0.809988 | 0.1900122 | 0.8235 | 0.1765 |
| 3 | 0.761209 | 0.2387912 | 0.7647 | 0.2353 |
| 5 | 0.759988 | 0.2400122 | 0.7059 | 0.2941 |
| 7 | 0.752479 | 0.2475214 | 0.7059 | 0.2941 |
| 9 | 0.757106 | 0.2428938 | 0.7059 | 0.2941 |
| 11 | 0.764432 | 0.2355678 | 0.6176 | 0.3824 |
| 13 | 0.769536 | 0.230464 | 0.6765 | 0.3235 |

Based on the classification performance in the table, for both the training and testing dataset, the best K value for classification is 1 for both the testing and training data set. The classification performance in the testing dataset is slightly higher than that in training set. To find the optimum value of the K, this study also tested all the possible K value in the test dataset with a cross validation approach, the result is plotted as following figure.

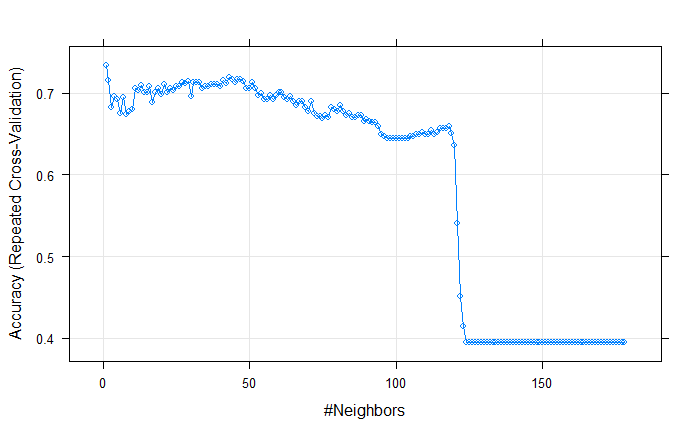


Figure 10 Classifier Performance V.S. Number of Neighbors

From the result, the optimum value of K is also 1, therefore, we may conclude that for the KNN classifier in wine training data, the best K value is 1, which means using the nearest 1 neighbor to predict new observation.

## Unsupervised Clustering

To conduct Principle Component Analysis, the attribute data were first scaled and centered to avoid the dominance by certain attributes with large variance. Then the first two principle components were selected from the 13 attributes as two dimensional datasets.

By setting the K value as 2-5, the corresponding unsupervised clustering result for K-means is listed as following.

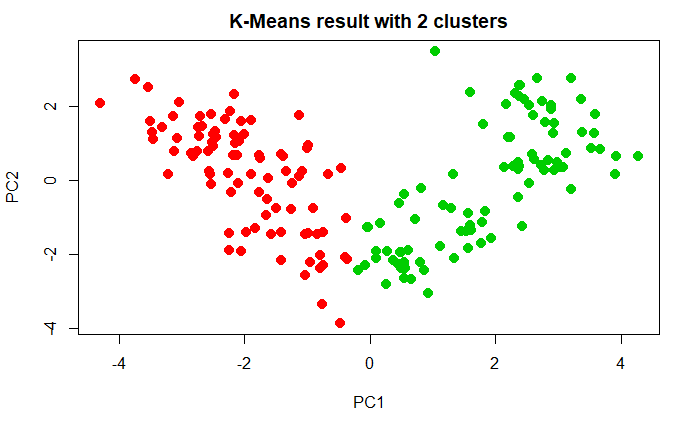


Figure 11 K-centroid for 2 clusters

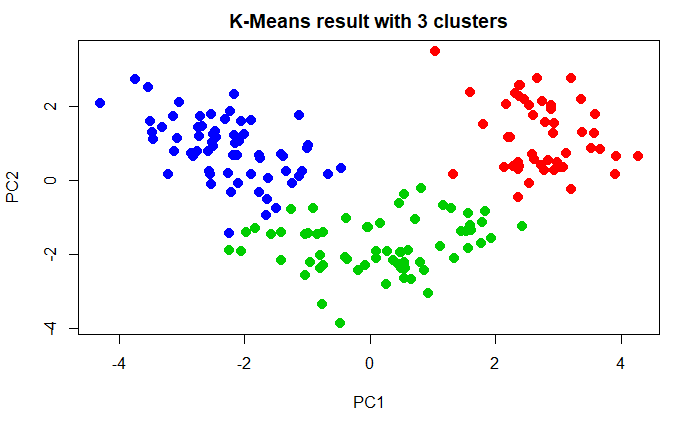


Figure 12 K-centroid for 3 clusters

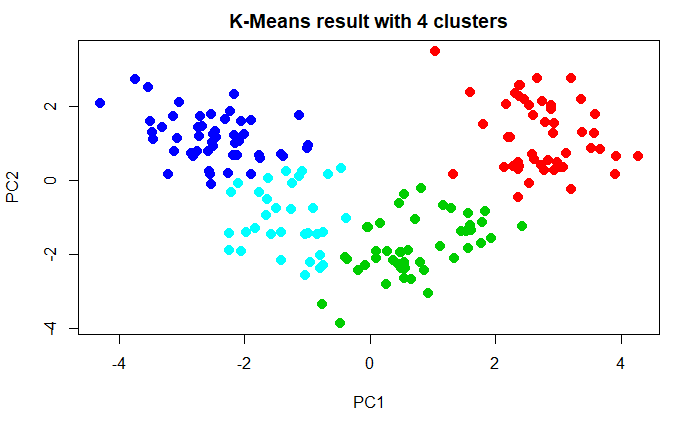


Figure 13 K-centroid for 4 clusters

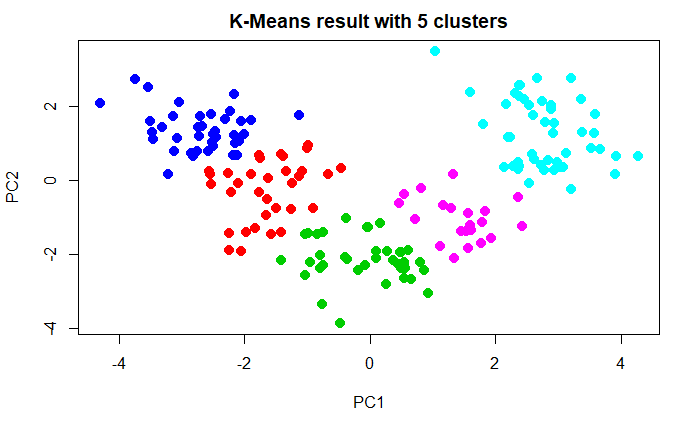


Figure 14 K-centroid for 5 clusters

By setting the K value as 2-5, the corresponding unsupervised clustering result for K-centroid is listed as following.

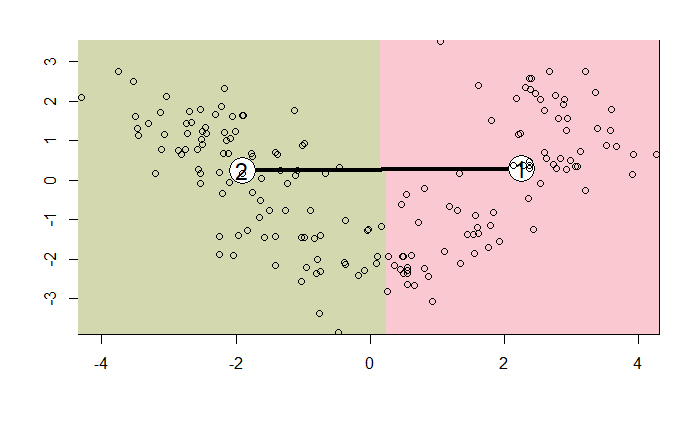


Figure 16 K-centroid for 2 clusters

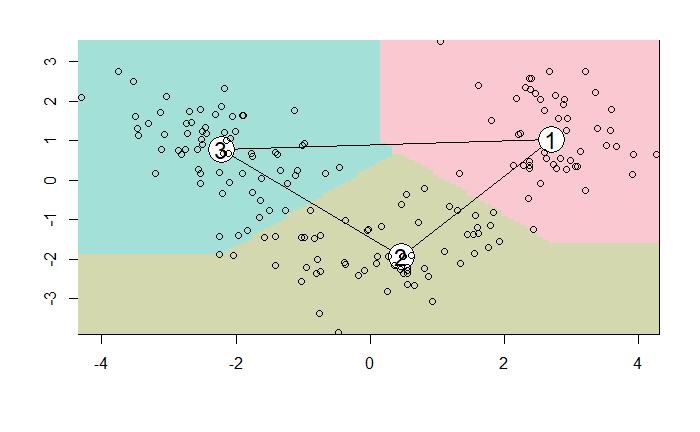


Figure 17 K-centroid for 3 clusters

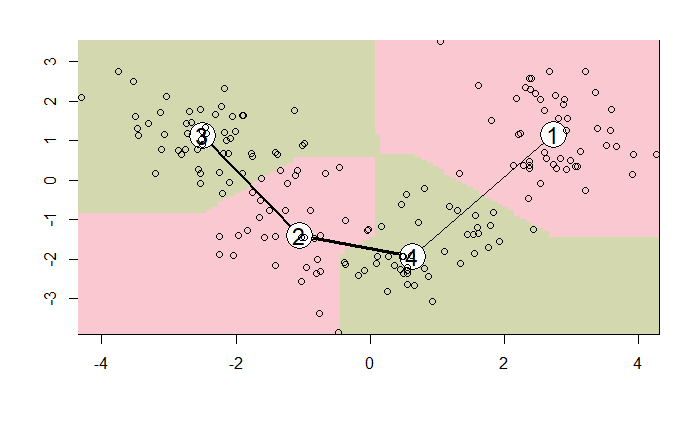


Figure 18 K-centroid for 4 clusters

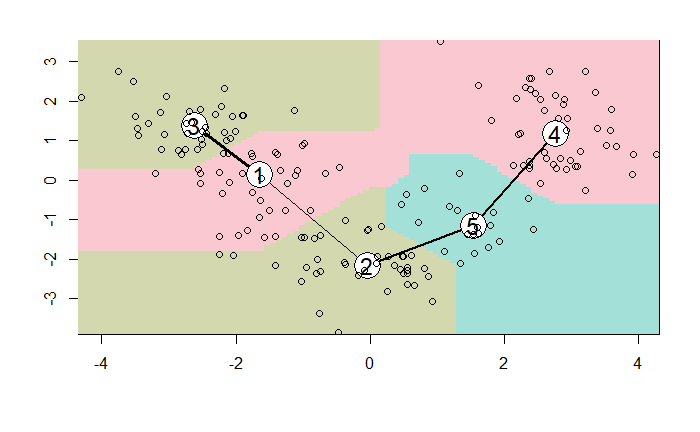


Figure 19 K-centroid for 5 clusters

Based on the comparison of the scatter plot for the two clustering method with different number of clusters, the data points is distributed in the same way as the same principle components are used as the coordinates for each point. Through visual inspection, the two clustering method differ at the cluster boundary, and classified some of the boundary points into different clusters, while the cluster centers are positioned similarly comparing the scatterplots with same K between two clustering method.9. Contribution of each team member:

**Sayak De:**

**Junqi Zhao:** Explore the error rate variation by tuning the parameter of K in the KNN method, and explored the best value of K in this dataset. Conducted the PCA for extracting the first two principle components, explored the two unsupervised clustering by plotting the cluster results with various values of K.

**Code in R:**

# plotting the elbow chart

> mydata <- wine

> wss <- (nrow(mydata)-1)\*sum(apply(mydata,2,var))

> for (i in 2:15) wss[i] <- sum(kmeans(mydata,

+ centers=i)$withinss)

> plot(1:15, wss, type="b", xlab="Number of Clusters",

+ ylab="Within groups sum of squares")

# K-means to the complete data set

> km = kmeans(wine, 2, nstart=100)

> table(wine$class,km$cluster)

> km = kmeans(wine, 3, nstart=100)

> table(wine$class,km$cluster)

#cross validation, 5-fold

library("dismo", lib.loc="D:/R-3.4.1/library")

set.seed(500)

folds <- kfold(wine, k=5, by=wine$class)

#**for k=2**

#Fold1: 36 data

#training

> km = kmeans(wine[folds==1,], 2, nstart=100)

> table(wine[folds==1,]$class,km$cluster)

Error rate: 33%

#Testing:

> km = kmeans(wine[folds!=1,], 2, nstart=100)

> table(wine[folds!=1,]$class,km$cluster)

#Error rate 92.2%

#Fold2: 35 data

#training

> km = kmeans(wine[folds==2,], 2, nstart=100)

> table(wine[folds==2,]$class,km$cluster)

#Error rate 28.5%

#Testing

> km = kmeans(wine[folds!=2,], 2, nstart=100)

> table(wine[folds!=2,]$class,km$cluster)

#Error rate 33.56%

#Fold 3: 36 data

#Training

> km = kmeans(wine[folds==3,], 2, nstart=100)

> table(wine[folds==3,]$class,km$cluster)

#Error rate 97%

#Testing:

> km = kmeans(wine[folds!=3,], 2, nstart=100)

> table(wine[folds!=3,]$class,km$cluster)

#Error rate 91.54%

#Fold4: 35 data

#training

> km = kmeans(wine[folds==4,], 2, nstart=100)

> table(wine[folds==4,]$class,km$cluster)

#Error rate 37.17%

#testing

> km = kmeans(wine[folds!=4,], 2, nstart=100)

> table(wine[folds!=4,]$class,km$cluster)

#Error rate 94.40%

#Fold5: 35 data

#Training

> km = kmeans(wine[folds==5,], 2, nstart=100)

> table(wine[folds==5,]$class,km$cluster)

#Error rate 97%

#Testing

> km = kmeans(wine[folds!=5,], 2, nstart=100)

> table(wine[folds!=5,]$class,km$cluster)

#Error rate 33%

#Average training error 58.5%

#Testing 68.86%

**#For k=3**

#Fold1:36 data

#training

> km = kmeans(wine[folds==1,], 3, nstart=100)

> table(wine[folds==1,]$class,km$cluster)

#Error rate 69.44%%

#Testing

> km = kmeans(wine[folds!=1,], 3, nstart=100)

> table(wine[folds!=1,]$class,km$cluster)

#Error rate 26.87%

#Fold2: 35 data

#Training

> km = kmeans(wine[folds==2,], 3, nstart=100)

> table(wine[folds==2,]$class,km$cluster)

#Error rate 97%

#Testing

> km = kmeans(wine[folds!=2,], 3, nstart=100)

> table(wine[folds!=2,]$class,km$cluster)

#Error rate 87%

#Fold3: 36data

#training

> km = kmeans(wine[folds==3,], 3, nstart=100)

> table(wine[folds==3,]$class,km$cluster)

#Error rate 75%

#Testing

> km = kmeans(wine[folds!=3,], 3, nstart=100)

> table(wine[folds!=3,]$class,km$cluster)

#Error rate 62%

#Fold4:35 data

#Training

> km = kmeans(wine[folds==4,], 3, nstart=100)

> table(wine[folds==4,]$class,km$cluster)

#Error rate 94%

#Testing

> km = kmeans(wine[folds!=4,], 3, nstart=100)

> table(wine[folds!=4,]$class,km$cluster)

#Error rate 55.2%

#Fold5:36 data

> km = kmeans(wine[folds==5,], 3, nstart=100)

> table(wine[folds==5,]$class,km$cluster)

#Error rate 77%

#Testing

> km = kmeans(wine[folds!=5,], 3, nstart=100)

> table(wine[folds!=5,]$class,km$cluster)

#Error rate 87.4%

#Avg training error rate 82.4%

#Avg testing error rate 63%

**#K=4**

#Fold1

#Training

> km = kmeans(wine[folds==1,], 4, nstart=100)

> table(wine[folds==1,]$class,km$cluster)

#Error rate 52%

#Testing

> km = kmeans(wine[folds!=1,], 4, nstart=100)

> table(wine[folds!=1,]$class,km$cluster)

#Error rate 84%

#Fold2

#Training

> km = kmeans(wine[folds==2,], 4, nstart=100)

> table(wine[folds==2,]$class,km$cluster)

#Error rate 82%

#Testing

> km = kmeans(wine[folds!=2,], 4, nstart=100)

> table(wine[folds!=2,]$class,km$cluster)

#Error rate 98%

#Fold3

#training

> km = kmeans(wine[folds==3,], 4, nstart=100)

> table(wine[folds==3,]$class,km$cluster)

#Error rate 100%

#Testing

> km = kmeans(wine[folds!=3,], 4, nstart=100)

> table(wine[folds!=3,]$class,km$cluster)

#Error rate 65.99%

#Fold4

#Training

> km = kmeans(wine[folds==4,], 4, nstart=100)

> table(wine[folds==4,]$class,km$cluster)

#Error rate 74%

#Testing

> km = kmeans(wine[folds!=4,], 4, nstart=100)

> table(wine[folds!=4,]$class,km$cluster)

#Error rate 67%

#Fold5

#Training

> km = kmeans(wine[folds==5,], 4, nstart=100)

> table(wine[folds==5,]$class,km$cluster)

#Error rate 61%

#Testing

> km = kmeans(wine[folds!=5,], 4, nstart=100)

> table(wine[folds!=5,]$class,km$cluster)

#Error rate 67%

#Avg training error 73.8%

#Avg testing error 76.2%

**#K=5**

**#Fold1**

#Training

> km = kmeans(wine[folds==1,], 5, nstart=100)

> table(wine[folds==1,]$class,km$cluster)

#Error rate 100%

#Testing

> km = kmeans(wine[folds!=1,], 5, nstart=100)

> table(wine[folds!=1,]$class,km$cluster)

#Error rate 97.18%

#Fold2

#Training

> km = kmeans(wine[folds==2,], 5, nstart=100)

table(wine[folds==2,]$class,km$cluster)

#error rate 82%

#testing

> km = kmeans(wine[folds!=2,], 5, nstart=100)

> table(wine[folds!=2,]$class,km$cluster)

#Error rate 93%

#Fold3

#Training

> km = kmeans(wine[folds==3,], 5, nstart=100)

> table(wine[folds==3,]$class,km$cluster)

#Error rate 83%

#Testing

> km = kmeans(wine[folds!=3,], 5, nstart=100)

> table(wine[folds!=3,]$class,km$cluster)

#Error rate 98%

#Fold4

#training

> km = kmeans(wine[folds==4,], 5, nstart=100)

> table(wine[folds==4,]$class,km$cluster)

#Error rate 66%

#Testing

> km = kmeans(wine[folds!=4,], 5, nstart=100)

> table(wine[folds!=4,]$class,km$cluster)

#Error rate 96%

#Fold5

#Training

> km = kmeans(wine[folds==5,], 5, nstart=100)

> table(wine[folds==5,]$class,km$cluster)

#Error rate 94%

#Testing

> km = kmeans(wine[folds!=5,], 5, nstart=100)

> table(wine[folds!=5,]$class,km$cluster)

#Error rate 83%

#Avg training error 85%

#Avg testing error 93.4%

# Validation set approach

**#Validation set approach**

# forming training and testing sets

> set.seed(500)

> tab<- sample(2, nrow(wine),replace=TRUE,prob=c(0.8,0.2))

> train<-wine[tab==1,]

> test<-wine[tab==2,]

#K=2

#Training

> km = kmeans(train, 2, nstart=100)

> table(train$class,km$cluster)

#Error rate 33%

#Testing

> km = kmeans(test, 2, nstart=100)

> table(test$class,km$cluster)

#Error rate 34%

#K=3

#Training

> km = kmeans(train, 3, nstart=100)

> table(train$class,km$cluster)

#Error rate 62%

#Testing

> km = kmeans(test, 3, nstart=100)

> table(test$class,km$cluster)

#Error rate 68%

#K=4

> km = kmeans(train, 4, nstart=100)

> table(train$class,km$cluster)

#Error rate 77%

#Testing

> km = kmeans(test, 4, nstart=100)

> table(test$class,km$cluster)

#Error rate 92%

#K=5

#Training

> km = kmeans(train, 5, nstart=100)

> table(train$class,km$cluster)

#Error rate 97%

#Testing

> km = kmeans(test, 5, nstart=100)

> table(test$class,km$cluster)

#Error rate 81%

**#Principal components:**

**#plotting the components**

> pcawine=wine[,2:14]

> prin\_comp=prcomp(pcawine,scale. = T)

> std\_dev <- prin\_comp$sdev

> pr\_var <- std\_dev^2

> prop\_varex <- pr\_var/sum(pr\_var)

> plot(prop\_varex, xlab = "Principal Component",

+ ylab = "Proportion of Variance Explained",

+ type = "b")

>

> plot(cumsum(prop\_varex), xlab = "Principal Component",

+ ylab = "Cumulative Proportion of Variance Explained",

+ type = "b")

#applying PCA in k means

#PCA=2

#K=2

> pc=prin\_comp$x[,1:2]

> km = kmeans(pc, 2, nstart=100)

> table(wine$class,km$cluster)

#Error rate= 42%

#K=3

> pc=prin\_comp$x[,1:2]

> km = kmeans(pc, 3, nstart=100)

> table(wine$class,km$cluster)

> plot(pc, col =(km$cluster +1) , main="K-Means result with 3 clusters", pch=20, cex=2)

#Error rate 61%

**#PCA=3**

#K=2

> pc=prin\_comp$x[,1:3]

> km = kmeans(pc, 2, nstart=100)

> plot(pc, col =(km$cluster +1) , main="K-Means result with 2 clusters", pch=20, cex=2)

> table(wine$class,km$cluster)

#Error rate 42%

#K=3

> km = kmeans(pc, 3, nstart=100)

> plot(pc, col =(km$cluster +1) , main="K-Means result with 3 clusters", pch=20, cex=2)

> table(wine$class,km$cluster)

#Error rate 70%

**#PCA=4**

#K=2

> pc=prin\_comp$x[,1:4]

> km = kmeans(pc, 2, nstart=100)

> plot(pc, col =(km$cluster +1) , main="K-Means result with 2 clusters", pch=20, cex=2)

> table(wine$class,km$cluster)

#Error rate 82%

#K=3

> km = kmeans(pc, 3, nstart=100)

> table(wine$class,km$cluster)

#Error rate 41%

**#PCA=5**

#K=2

> pc=prin\_comp$x[,1:5]

> km = kmeans(pc, 2, nstart=100)

> table(wine$class,km$cluster)

#Error rate 82%

#K=3

> km = kmeans(pc, 3, nstart=100)

> plot(pc, col =(km$cluster +1) , main="K-Means result with 3 clusters", pch=20, cex=2)

> table(wine$class,km$cluster)

#Error rate 71%

**#PCA=6**

#K=2

> pc=prin\_comp$x[,1:6]

> km = kmeans(pc, 2, nstart=100)

> plot(pc, col =(km$cluster +1) , main="K-Means result with 2 clusters", pch=20, cex=2)

> table(wine$class,km$cluster)

#Error rate 70%

#K=3

> km = kmeans(pc, 3, nstart=100)

> plot(pc, col =(km$cluster +1) , main="K-Means result with 3 clusters", pch=20, cex=2)

> table(wine$class,km$cluster)

#Error rate 63.43%

**#PCA=7**

#K=2

> pc=prin\_comp$x[,1:7]

> km = kmeans(pc, 2, nstart=100)

> plot(pc, col =(km$cluster +1) , main="K-Means result with 2 clusters", pch=20, cex=2)

> table(wine$class,km$cluster)

#Error rate 70%

#K=3

> km = kmeans(pc, 3, nstart=100)

> plot(pc, col =(km$cluster +1) , main="K-Means result with 3 clusters", pch=20, cex=2)

> table(wine$class,km$cluster)

#Error rate 63%

**#PCA=8**

#K=2

> pc=prin\_comp$x[,1:8]

> km = kmeans(pc, 2, nstart=100)

> table(wine$class,km$cluster)

#Error rate 70%

#K=3

> km = kmeans(pc, 3, nstart=100)

> table(wine$class,km$cluster)

#Error rate 65%

**#PCA=9**

#K=2

> pc=prin\_comp$x[,1:9]

> km = kmeans(pc, 2, nstart=100)

> table(wine$class,km$cluster)

#Error rate 57%

#K=3

> km = kmeans(pc, 3, nstart=100)

> table(wine$class,km$cluster)

#Error rate 63%

**#PCA=10**

#K=2

> pc=prin\_comp$x[,1:10]

> km = kmeans(pc, 2, nstart=100)

> table(wine$class,km$cluster)

#Error rate 84%

#K=3

> km = kmeans(pc, 3, nstart=100)

> table(wine$class,km$cluster)

#Error rate 63%

**#PCA=11**

#K=2

> pc=prin\_comp$x[,1:11]

> km = kmeans(pc, 2, nstart=100)

> table(wine$class,km$cluster)

#Error rate 84%

#K=3

> km = kmeans(pc, 3, nstart=100)

> table(wine$class,km$cluster)

#Error rate 98%

**#PCA=12**

#K=2

> pc=prin\_comp$x[,1:12]

> km = kmeans(pc, 2, nstart=100)

> table(wine$class,km$cluster)

#Error rate 84%

#K=3

> km = kmeans(pc, 3, nstart=100)

> table(wine$class,km$cluster)

#Error rate 71%

**#PCA=13**

#K=2

> pc=prin\_comp$x[,1:13]

> km = kmeans(pc, 2, nstart=100)

> table(wine$class,km$cluster)

#Error rate 84%

#K=3

> km = kmeans(pc, 3, nstart=100)

> table(wine$class,km$cluster)

#Error rate 98%

#plotting the first two principal components

#k=2

> pc=prin\_comp$x[,1:13]

> km = kmeans(pc, 2, nstart=100)

> plot(pc, col =(km$cluster +1) , main="K-Means result with 2 clusters", pch=20, cex=2)

#k=3

> pc=prin\_comp$x[,1:13]

> km = kmeans(pc, 3, nstart=100)

> plot(pc, col =(km$cluster +1) , main="K-Means result with 3 clusters", pch=20, cex=2)

#KNN clustering

#load data

library(caret)

wine=read.csv("wine.csv",header=T)

wine=transform(wine,class=as.factor(as.character(wine$class))) #Transform class into factor for classification

#prepare dataset

#rawdata[rawdata=="?"]<-NA

#ad=na.omit(rawdata)

#constructing the training and testing set by 80% to 20%

library("caret")

index=createDataPartition(wine$class,p=0.8, list = FALSE)

wine.training=wine[index,]

wine.testing=wine[-index,]

#train the model

trctrl=trainControl(method="repeatedcv", number=10,repeats = 3)

set.seed(9)

grid=expand.grid(k=c(1:178))

set.seed(9)

model.knn=train(wine.training[, 2:14], wine.training[, 1], method='knn',trControl=trctrl,tuneGrid = grid,tuneLength = 2)

#Evaluate the model

#inspect the performance of training set

model.knn

plot(model.knn)

#inspect the performance of testing set

set.seed(9)

predictions=predict(object = model.knn,wine.testing[,2:14])

table(predictions)

confusionMatrix(predictions,wine.testing[,1])

#Unsupervised Clustering

#Conduct Feature Selection with PCA

pcawine=wine[,2:14]

#PCA

prin\_comp=prcomp(pcawine, scale. = TRUE)

#Extract first two components without class labelW

pc=prin\_comp$x[,1:2]

##K-means

# Perform K-Means with 2 clusters

set.seed(7)

km2 = kmeans(pc, 2, nstart=100)

plot(pc, col =(km2$cluster +1) , main="K-Means result with 2 clusters", pch=20, cex=2)

# Perform K-Means with 3 clusters

set.seed(7)

km3 = kmeans(pc, 3, nstart=100)

plot(pc, col =(km3$cluster +1) , main="K-Means result with 3 clusters", pch=20, cex=2)

# Perform K-Means with 4 clusters

set.seed(7)

km4 = kmeans(pc, 4, nstart=100)

plot(pc, col =(km4$cluster +1) , main="K-Means result with 4 clusters", pch=20, cex=2)

km5 = kmeans(pc, 5, nstart=100)

plot(pc, col =(km5$cluster +1) , main="K-Means result with 5 clusters", pch=20, cex=2)

##K-center

# Perform K-center with 2 clusters

library("flexclust")

set.seed(7)

kc2=kcca(pc,k=2,family = kccaFamily("kmedians"),control = list(initcent="kmeanspp"))

image(kc2)

points(pc)

barplot(kc2)

# Perform K-center with 3 clusters

library("flexclust")

set.seed(7)

kc3=kcca(pc,k=3,family = kccaFamily("kmedians"),control = list(initcent="kmeanspp"))

image(kc3)

points(pc)

barplot(kc3)

# Perform K-center with 4 clusters

library("flexclust")

set.seed(7)

kc4=kcca(pc,k=4,family = kccaFamily("kmedians"),control = list(initcent="kmeanspp"))

image(kc4)

points(pc)

barplot(kc4)

# Perform K-center with 5 clusters

library("flexclust")

kc5=kcca(pc,k=5,family = kccaFamily("kmedians"),control = list(initcent="kmeanspp"))

image(kc5)

points(pc)

barplot(kc5)

**References:**

**[1] Elbow method web article:** [**https://bl.ocks.org/rpgove/0060ff3b656618e9136b**](https://bl.ocks.org/rpgove/0060ff3b656618e9136b)

**[2] Cross validation guide:** [**https://stats.stackexchange.com/questions/61090/how-to-split-a-data-set-to-do-10-fold-cross-validation**](https://stats.stackexchange.com/questions/61090/how-to-split-a-data-set-to-do-10-fold-cross-validation)

**[3] Applying PCA :** [**https://www.r-bloggers.com/principal-component-analysis-in-r/**](https://www.r-bloggers.com/principal-component-analysis-in-r/)

**[4] Validation set approach:** [**https://stackoverflow.com/questions/36068963/r-how-to-split-a-data-frame-into-training-validation-and-test-sets**](https://stackoverflow.com/questions/36068963/r-how-to-split-a-data-frame-into-training-validation-and-test-sets)

**[5] k-centroid clustering: https://cran.r-project.org/web/packages/flexclust/flexclust.pdf**

1. https://en.wikipedia.org/wiki/Classification\_of\_wine [↑](#footnote-ref-1)
2. https://archive.ics.uci.edu/ml/datasets/wine [↑](#footnote-ref-2)