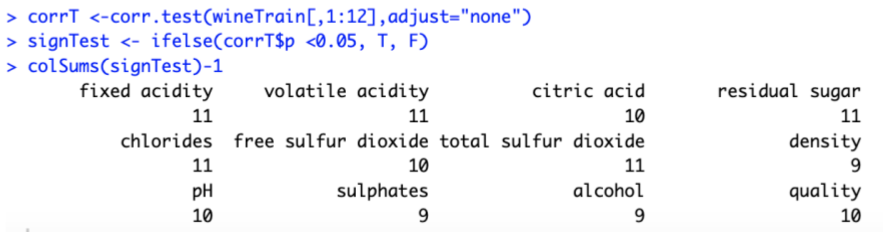
FUNDAMENTALS OF DATA SCIENCE – DSC 441
HW4 Xuyang Ji

Problem #1

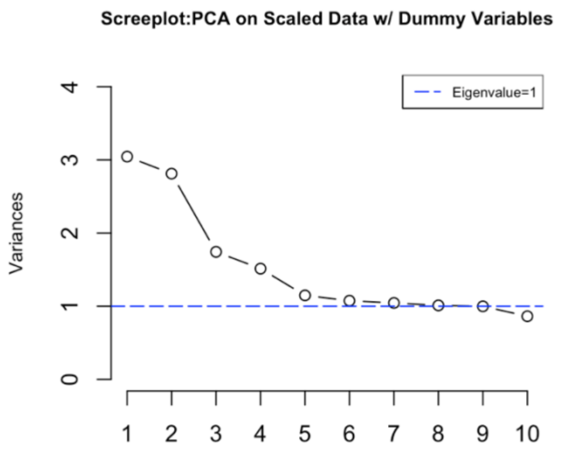
I. Data Exploratory

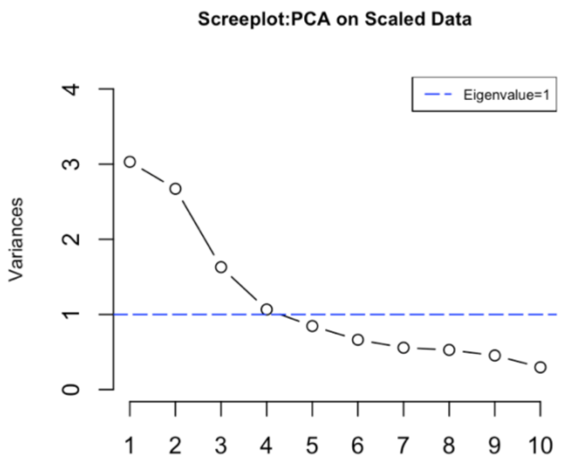
While the goal of PCA is to summarize the correlations among the projected variables with a smaller set of linear combinations, the correlation matrix for entries that are significant at a 95% confidence level is examined as follows. Based on the output of the population correlation coefficient p at an alpha level of 0.05, all predictors are significantly linearly correlated with another variable. Recall that for principal component analysis, the PC scores are calculated based on components considered to account for the correlational structure, whereas trivial PCs can be estimated from fields without any correlation structure among the original datasets. In order to find informative relationships between the fields and reveal the underlying structure among the set of highly correlated explanatory variables, the PCA based on the correlation matrix is used.



```
> corrT <-corr.test(wineTrain[,1:12],adjust="none")
> signTest <- ifelse(corrT$p <0.05, T, F)
> colSums(signTest)-1
       fixed acidity     volatile acidity          citric acid       residual sugar 
                  11                   11                   10                   11 
           chlorides  free sulfur dioxide total sulfur dioxide              density 
                  11                   10                   11                    9 
                  pH            sulphates              alcohol              quality 
                  10                    9                    9                   10 
```

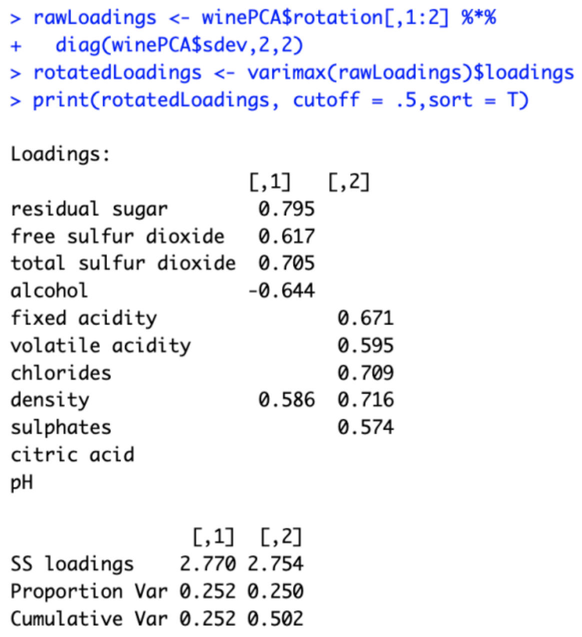
While being very effective with numerical data, PCA cannot take into consideration categorical data as is. To deal with the factor variable, *quality*, three methods are compared based on the final projected inertia. By applying PCA with the one-hot encoding of the factor variable, removing the factor variable, and converting the variable into numeric in three independent runs, the trial with one-hot encoding modalities reveals that the inertia given to a factor level/column would be based on the size and probabilities of the modalities, which did nothing but introduced more biases to the model. In the other two models, there is no significant difference in terms of the component structure. Considering the subtle importance of the *quality* variable in explaining wine types, it is dropped to avoid any future biases.



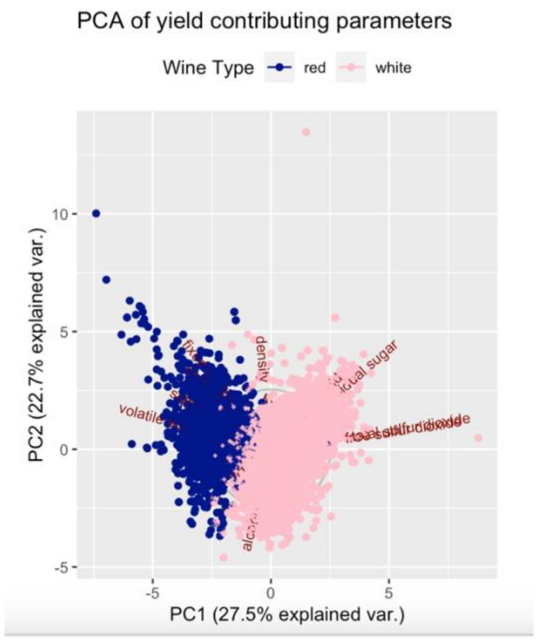


Since the goal of the analysis is to examine observations that have a similar pattern, and then distill the variables down to the most important features so that the data is simplified without losing significant traits, data interpretation is the key for this particular dataset. Using the proportion of variance explained criterion, 2 PCs is optimal for a cumulative variance of 50.2% of the original dataset. The Varimax factor rotation is applied to clarify the relationship among factors. By maximizing the variance shared among selected features, results more discretely represent how features correlated with each principal component. Based on the output, PC1 is strongly correlated with *sugar and sulfur dioxide*, whereas PC2 is most strongly associated with *density*, *chlorides*, *and acidity*.

The PCA scatter plot showed that white wines were placed on the right-hand side and red wines were placed on the left-hand side. Thus, the taste difference between white and red wines was explained using 11 properties. Looking in detail, white wines, clustered at the higher tail of PC1, have larger *residual sugar*, *sulfur dioxide*, *and less alcohol percentage*, than red wine.

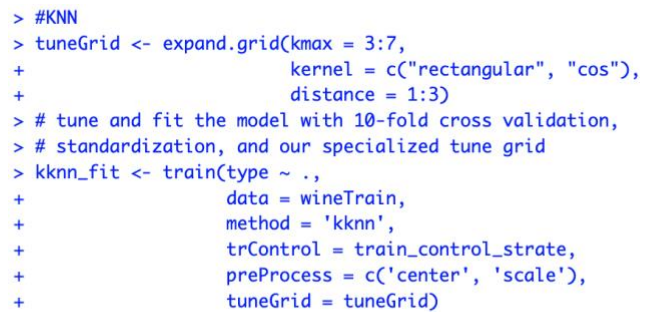


```
> rawLoadings <- winePCA$rotation[,1:2] %*%
+   diag(winePCA$sdev,2,2)
> rotatedLoadings <- varimax(rawLoadings)$loadings
> print(rotatedLoadings, cutoff = .5,sort = T)

Loadings:
                     [,1]   [,2]  
residual sugar        0.795       
free sulfur dioxide   0.617       
total sulfur dioxide  0.705       
alcohol              -0.644       
fixed acidity                0.671
volatile acidity             0.595
chlorides                    0.709
density               0.586  0.716
sulphates                    0.574
citric acid                       
pH                                

                [,1]  [,2]
SS loadings    2.770 2.754
Proportion Var 0.252 0.250
Cumulative Var 0.252 0.502
```

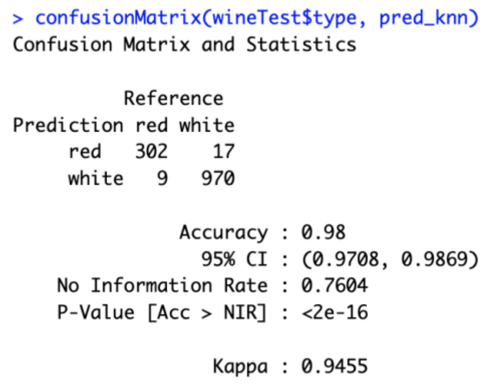


II. KNN

The following KNN model uses a 10-fold stratified cross-validation *trainControl*, and adds normalization through *preProcess* with *tuneGrid* parameters. The output suggests that the optimal KNN model achieves an accuracy rate of approximately 98.67%, with kmax = 6, distance = 1, and kernel = rectangular. The Kappa value of 0.96 indicates a strong agreement between the predicted and actual labels. Looking at the confusion matrix, the accuracy rate of the KNN classifier on the testing set is 98%, much higher than the no information rate.



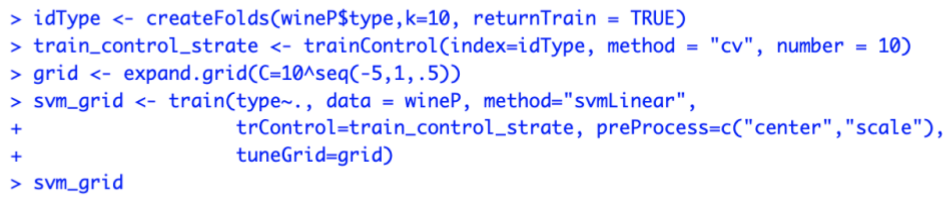
```
> #KNN
> tuneGrid <- expand.grid(kmax = 3:7,
+                         kernel = c("rectangular", "cos"),
+                         distance = 1:3)
> # tune and fit the model with 10-fold cross validation,
> # standardization, and our specialized tune grid
> kknn_fit <- train(type ~ .,
+                   data = wineTrain,
+                   method = 'kknn',
+                   trControl = train_control_strate,
+                   preProcess = c('center', 'scale'),
+                   tuneGrid = tuneGrid)
```



```
> confusionMatrix(wineTest$type, pred_knn)
Confusion Matrix and Statistics

          Reference
Prediction red white
     red   302    17
     white   9   970

               Accuracy : 0.98
                 95% CI : (0.9708, 0.9869)
    No Information Rate : 0.7604
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9455
```

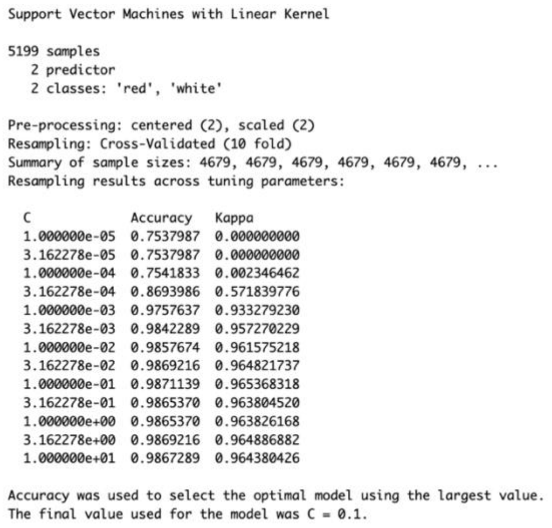
III. SVM on PCA Dataset

To avoid potential overfitting presented in the *wineP* dataset, a stratified random sampling using train-test splitting is applied as follows. The SVM model is built using the training dataset, with a 10-fold cross-validation method along with grid search on C-parameter. The output suggests that the optimal SVM model achieves an accuracy rate of approximately 98.7%, with C-parameter set to 0.1. The Kappa value of 0.55, indicates an agreement between the predicted and the actual.

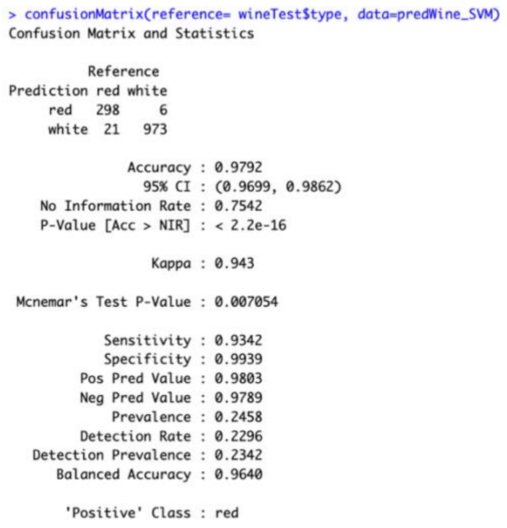


```
> idType <- createFolds(wineP$type,k=10, returnTrain = TRUE)
> train_control_strate <- trainControl(index=idType, method = "cv", number = 10)
> grid <- expand.grid(C=10^seq(-5,1,.5))
> svm_grid <- train(type~., data = wineP, method="svmLinear",
+                   trControl=train_control_strate, preProcess=c("center","scale"),
+                   tuneGrid=grid)
> svm_grid
```

Looking at the confusion matrix, the accuracy rate of the SVM classifier on the testing set is approximately 97.92%, much higher than the no information rate. Although the model has erroneously labeled 6 observations from *white wine* as *red*, and 21 *red wine* as *white*, the model is strong despite the class imbalance.



```
Support Vector Machines with Linear Kernel

5199 samples
   2 predictor
   2 classes: 'red', 'white'

Pre-processing: centered (2), scaled (2)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 4679, 4679, 4679, 4679, 4679, 4679, ...
Resampling results across tuning parameters:

  C             Accuracy   Kappa
  1.000000e-05  0.7537987  0.000000000
  3.162278e-05  0.7537987  0.000000000
  1.000000e-04  0.7541833  0.002346462
  3.162278e-04  0.8693986  0.571839776
  1.000000e-03  0.9757637  0.933279230
  3.162278e-03  0.9842289  0.957270229
  1.000000e-02  0.9857674  0.961575218
  3.162278e-02  0.9869216  0.964821737
  1.000000e-01  0.9871139  0.965368318
  3.162278e-01  0.9865370  0.963804520
  1.000000e+00  0.9865370  0.963826168
  3.162278e+00  0.9869216  0.964886882
  1.000000e+01  0.9867289  0.964380426

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was C = 0.1.
```

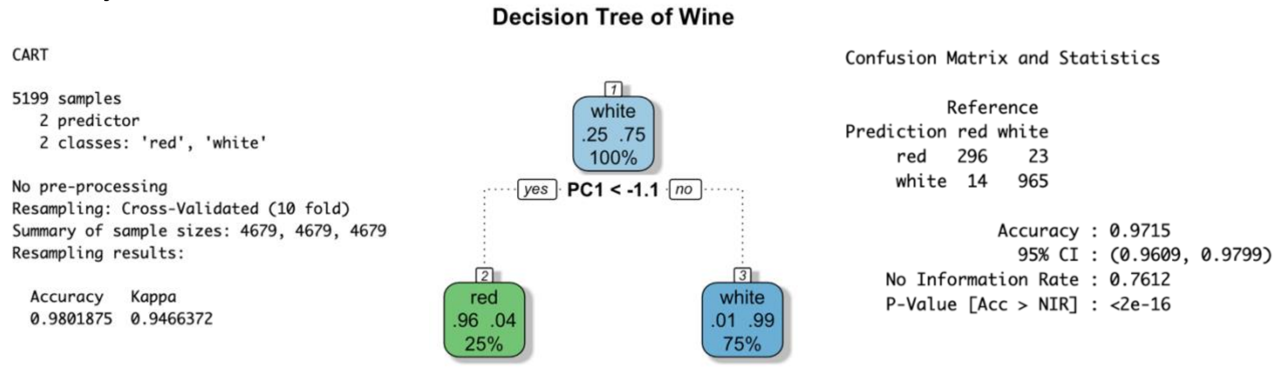


```
> confusionMatrix(reference= wineTest$type, data=predWine_SVM)
Confusion Matrix and Statistics

          Reference
Prediction red white
     red   298     6
     white  21   973

               Accuracy : 0.9792
                 95% CI : (0.9699, 0.9862)
    No Information Rate : 0.7542
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.943

 Mcnemar's Test P-Value : 0.007054

            Sensitivity : 0.9342
            Specificity : 0.9939
         Pos Pred Value : 0.9803
         Neg Pred Value : 0.9789
             Prevalence : 0.2458
         Detection Rate : 0.2296
   Detection Prevalence : 0.2342
      Balanced Accuracy : 0.9640

       'Positive' Class : red
```

IV. Decision Trees with Tuned Hyperparameters

To predict the class label attribute- *type,* of an unseen wine, the decision tree model is built based on the 11 predictors. Based on the previous step, the white wines strongly outnumbered the red wines. The highly imbalanced distributions of classes suggest that, when applying cross-validation, random sampling would be prone toward imbalanced samples, resulting in a low model quality in predicting items in minority classes. Hence, stratified cross-validation is applied for the following model-building steps.

Using 10-fold stratified cross-validation with tuned hyperparameters, the resulting decision tree model on the basis of Gini's impurity index, has an accuracy rate of approximately 98%, with a kappa value of 0.94, indicating a high agreement between the predicted and actual label for the model. However, as class imbalance is presented in the dataset, the model quality is expected to be heavily biased.



```
CART

5199 samples
   2 predictor
   2 classes: 'red', 'white'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 4679, 4679, 4679
Resampling results:

  Accuracy   Kappa
  0.9801875  0.9466372
```

```
Confusion Matrix and Statistics

          Reference
Prediction red white
     red   296    23
     white  14   965

               Accuracy : 0.9715
                 95% CI : (0.9609, 0.9799)
    No Information Rate : 0.7612
    P-Value [Acc > NIR] : <2e-16
```

With tuned hyperparameters, the minimum number of items in the parent node that could be split further is set at 10, the *maxdepth* parameter prevents the tree from growing past a depth of 5, and the *minbucket* of 3 provides the smallest number of items that are allowed in a terminal node. Using the Gini index method, the attribute *PC1* at a value of -1.1 is selected as the first split point with the maximized reduction in impurity. The leaf node 3 indicates there are 75% of wines with *PC1* equal to or greater than the threshold of -1.1 is classified as *white.* Meantime, leaf node 2 indicates that there are 25% of the wines with PC1 smaller than the threshold, are classified as *red.* Based on the confusion matrix, the Decision Tree model achieves an accuracy rate of approximately 97.15%, indicating the classifier is stronger than expected. In other words, the model ends up with meaningful regularity that is relevant to the true, important, and distinguishing features of all wine types.

V. Model Comparison

The results show that the KNN algorithm has the best accuracy in predicting with a precision value of 98%. The Decision Tree algorithm has a prediction accuracy of 97.15 % and the SVM algorithm has a prediction accuracy value of 97.9%. When classifying, the kNN will generally classify more accurately although with more misclassifications in the majority class. In comparison, the SVM will occasionally misclassify an item in the minority class by looking at the confusion table of the test set. Meantime, decision trees are able to generate understandable rules without requiring much computation and prior knowledge. While all algorithms yield positive results regarding the accuracy in which they classify wines, SVM provides slightly better classification accuracy and speed.

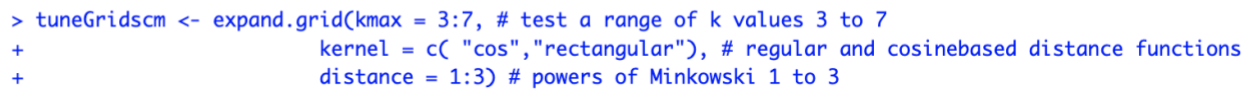
Problem #2 KNN on Sacramento

I. Feature Selection

In the *Sacramento* dataset that focuses on the house *type*, 932 observations have been investigated across 9 attributes. The variables include numerical information about the size of the housing and its price, as well as categorical information like zip code, city, and the type of unit. Noted here, the variable *city* has been dropped for future analysis, since it is inherently reflected by its zip code variable. One important aspect of this dataset after transforming all categorical variables into numeric is that it contains many variables and many of these variables have extremely low variances. This means that there is very little information in these dummy variables because they mostly consist of a single value, such as zero. To remove such variables before modelling, nearZeroVar() is used with default parameter including a freqCut = 19 and uniqueCut = 10. The resulting data frame *scmDummy* contains only *zip.z95823* as the significant variable out of all the dummies. A possible alternative method to deal with near zero variance is to use PCA in the preprocess in model building, which usually improve the performance of the KNN classifiers significantly.

II. Distance Function

Regarding the high level of dimensionality, the Manhattan distance works better than the Euclidean distance. With its high dimensionality, everything is far from everything, so another option is usually to look at the direction of the vectors, that is, use the ***cosine distance***.

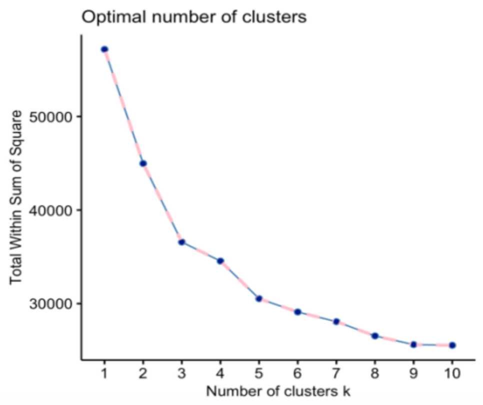

```
> tuneGridscm <- expand.grid(kmax = 3:7, # test a range of k values 3 to 7
+                            kernel = c( "cos","rectangular"), # regular and cosinebased distance functions
+                            distance = 1:3) # powers of Minkowski 1 to 3
```

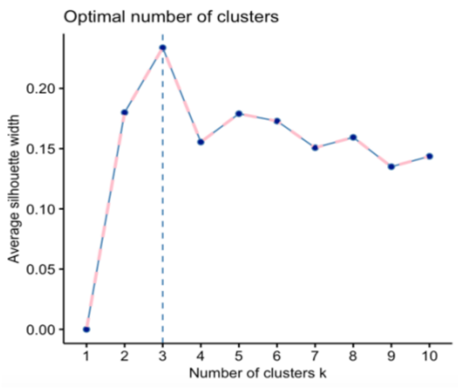
Here we tried different Minkowski distances from 1 to 3, and tested regular and cosine distance functions. The cosine's distance function Kappa value generally performed better, although accuracy was comparable on both. Accuracy was used to select the optimal model using the largest value, where the final values used for the model were kmax = 6, distance = 1 and kernel = cos. The accuracy is 94.91% on the final model

Problem #3. Clustering on Wine Data

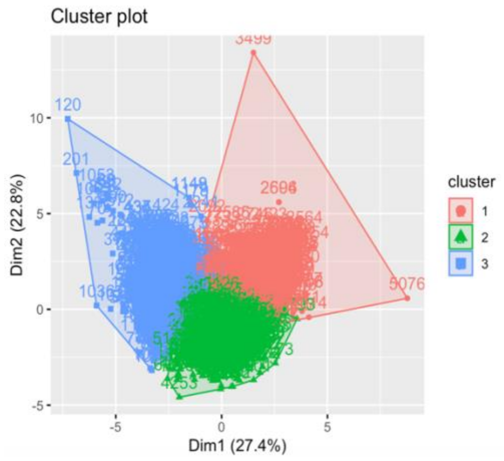
I. K-Means Clustering

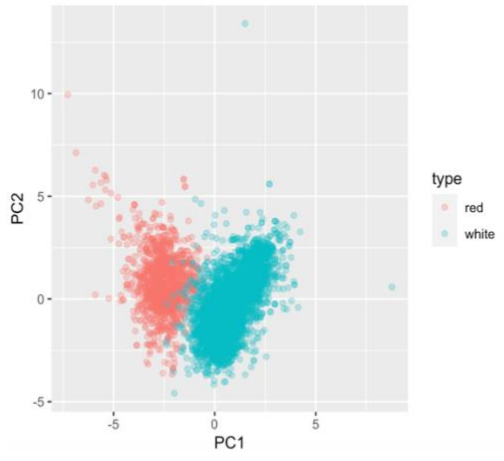
The quality of wine is determined by various factors such as alcohol content, the presence of sulfates, and pH values. However, the taste, aroma, and potency of wine are primarily determined by the chemical ingredients and their proportions in the wine. In the *wine* dataset that focuses on the wine *type*, 6,497 observations have been investigated across 12 attributes, of which 4,898 observations are white wine. To classify wines into distinct clusters based on their properties, the K-means clustering algorithm can be used. This algorithm groups data into K clusters by iteratively improving the clustering until no further improvement is possible. The elbow method and silhouette scores indicate that K=3 is an optimal value for the KMeans model. Therefore, the model is fit using K=3 as suggested by the plots.





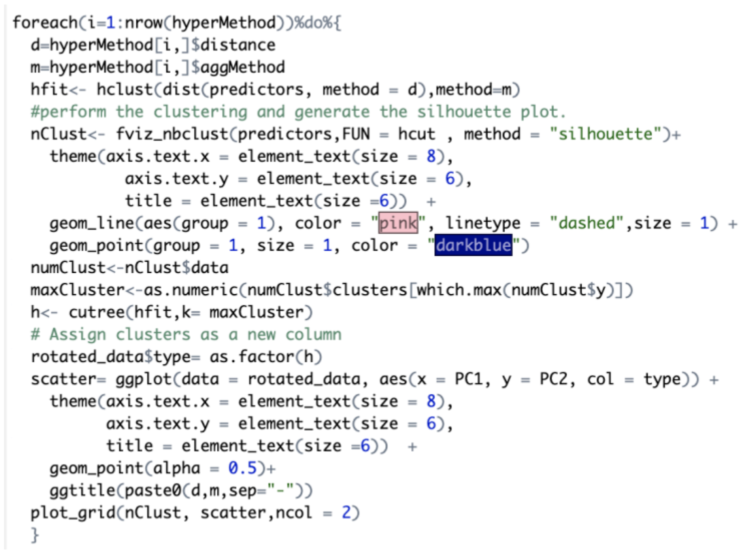
Based on the output, the three clusters each with the size of 1499, 2421, and 1279 observations. Using the *fviz_cluster* function, we can visualize how the clusters are formed. Recall that, PC1 is strongly correlated with *sugar and sulfur dioxide*, whereas PC2 is most strongly associated with *density*, *chlorides*, *and acidity.* Compared with the PCA plot, which has a clear linear separation between classes, the K-Mean model makes a further classification within the white wine, regarding wine with different chloride and density, which mainly because K Means mostly considers circular areas.



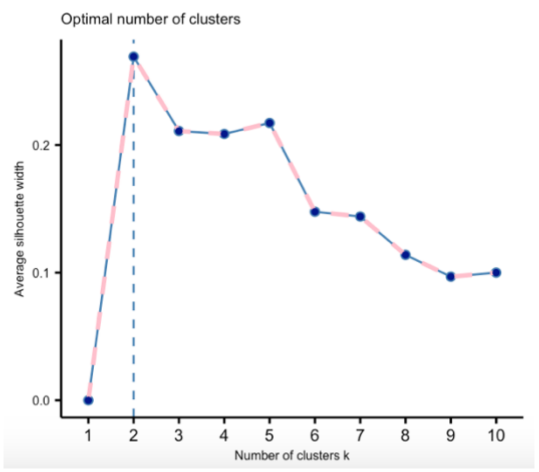


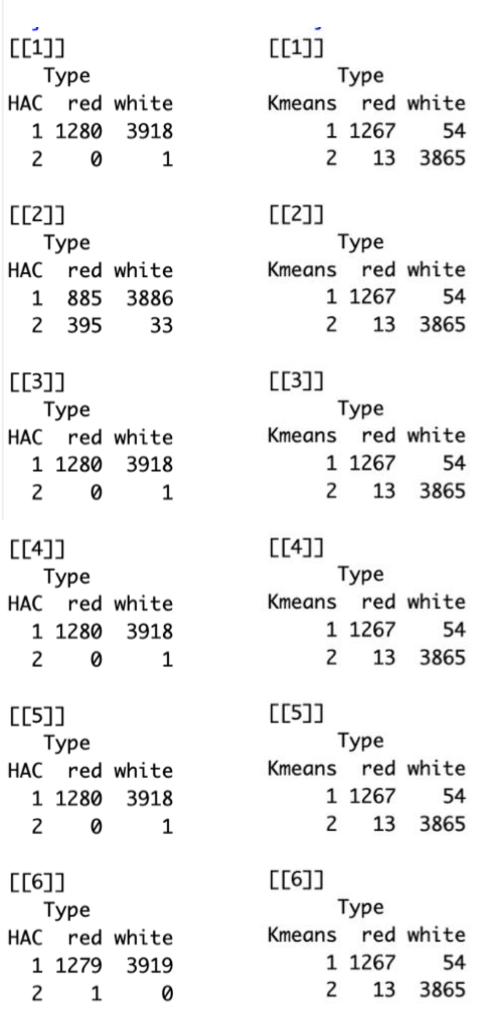
II. Hierarchical Agglomerative Clustering (HAC)

Hierarchical clustering groups objects in clusters based on their similarity, which starts by treating each point as a single cluster, then merging clusters until all clusters have been merged into one big cluster containing all points. The goal of the following configuration is to try different distance functions, in combination with various linkage functions; and gain intuition about the tradeoffs of model quality and model complexity. The algorithm below firstly allocates a grid of 2 hyperparameters – the distance parameter that includes Euclidean Distance and Manhattan Distance, and the agglomerative methods which include average, complete, and centroid, resulting in a total of 6 configurations. The best HAC clustering is based on the maximum value in the average silhouette width, which is 2 in each loop run. The following confusion tables are based on k-means and HAC clustering in terms of their silhouette score.

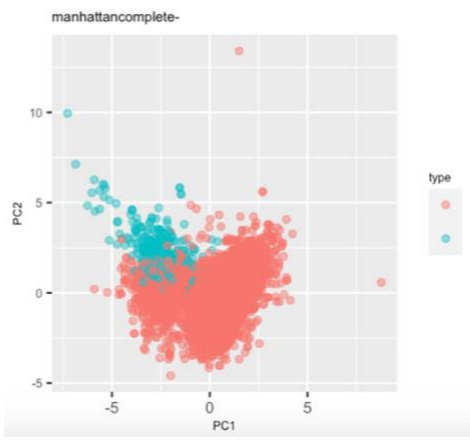


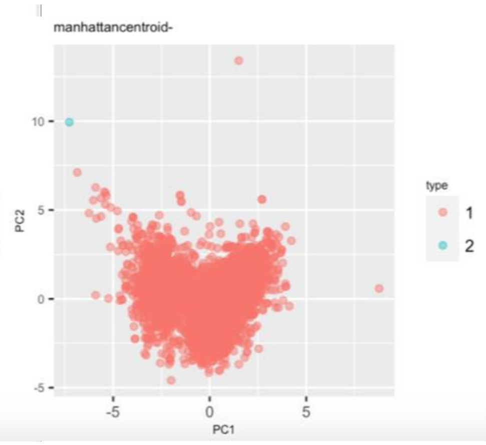
```
foreach(i=1:nrow(hyperMethod))%do%{
  d=hyperMethod[i,]$distance
  m=hyperMethod[i,]$aggMethod
  hfit<- hclust(dist(predictors, method = d),method=m)
  #perform the clustering and generate the silhouette plot.
  nClust<- fviz_nbclust(predictors,FUN = hcut , method = "silhouette")+
    theme(axis.text.x = element_text(size = 8),
          axis.text.y = element_text(size = 6),
          title = element_text(size =6))  +
    geom_line(aes(group = 1), color = "pink", linetype = "dashed",size = 1) +
    geom_point(group = 1, size = 1, color = "darkblue")
  numClust<-nClust$data
  maxCluster<-as.numeric(numClust$clusters[which.max(numClust$y)])
  h<- cutree(hfit,k= maxCluster)
  # Assign clusters as a new column
  rotated_data$type= as.factor(h)
  scatter= ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = type)) +
    theme(axis.text.x = element_text(size = 8),
        axis.text.y = element_text(size = 6),
        title = element_text(size =6))  +
    geom_point(alpha = 0.5)+
    ggtitle(paste0(d,m,sep="-"))
  plot_grid(nClust, scatter,ncol = 2)
  }
```

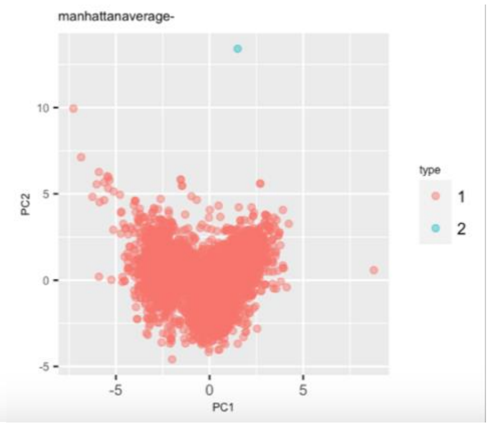


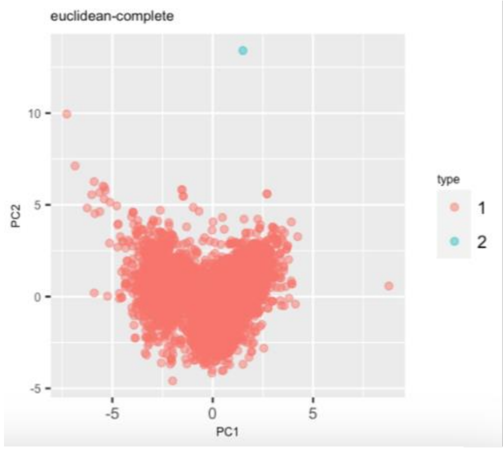


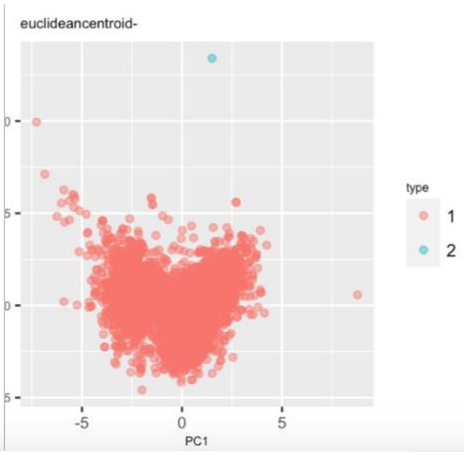
```
[[1]]                   [[1]]
   Type                     Type
HAC  red white          Kmeans  red white
  1 1280  3918               1 1267    54
  2    0     1               2   13  3865

[[2]]                   [[2]]
   Type                     Type
HAC  red white          Kmeans  red white
  1  885  3886               1 1267    54
  2  395    33               2   13  3865

[[3]]                   [[3]]
   Type                     Type
HAC  red white          Kmeans  red white
  1 1280  3918               1 1267    54
  2    0     1               2   13  3865

[[4]]                   [[4]]
   Type                     Type
HAC  red white          Kmeans  red white
  1 1280  3918               1 1267    54
  2    0     1               2   13  3865

[[5]]                   [[5]]
   Type                     Type
HAC  red white          Kmeans  red white
  1 1280  3918               1 1267    54
  2    0     1               2   13  3865

[[6]]                   [[6]]
   Type                     Type
HAC  red white          Kmeans  red white
  1 1279  3919               1 1267    54
  2    1     0               2   13  3865
```

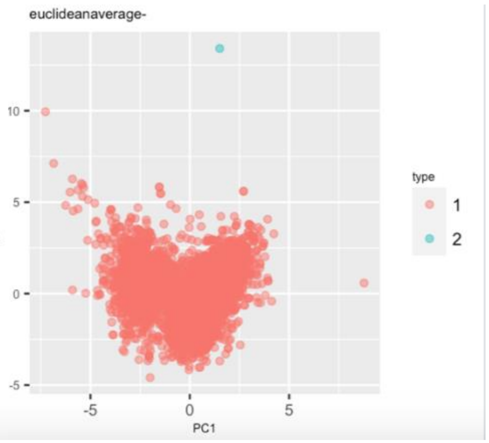








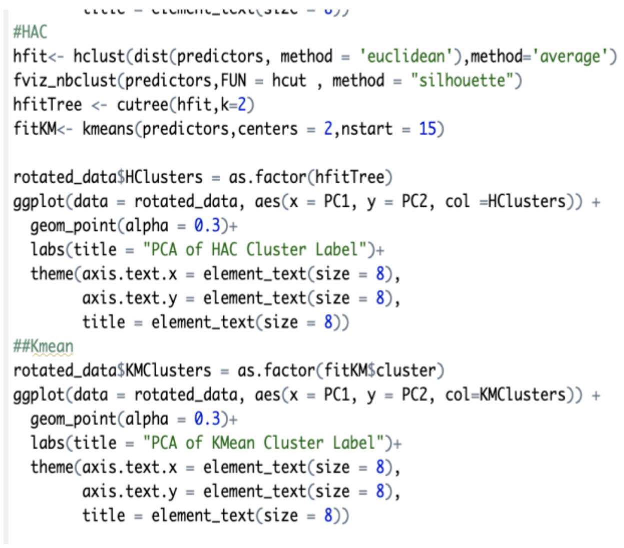




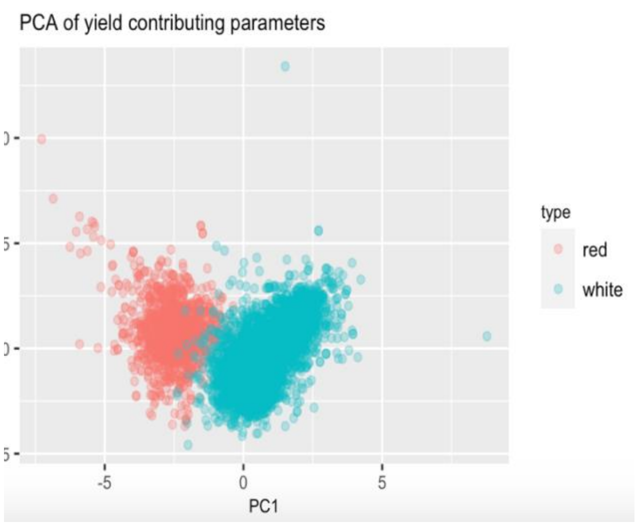
III. Model Comparison

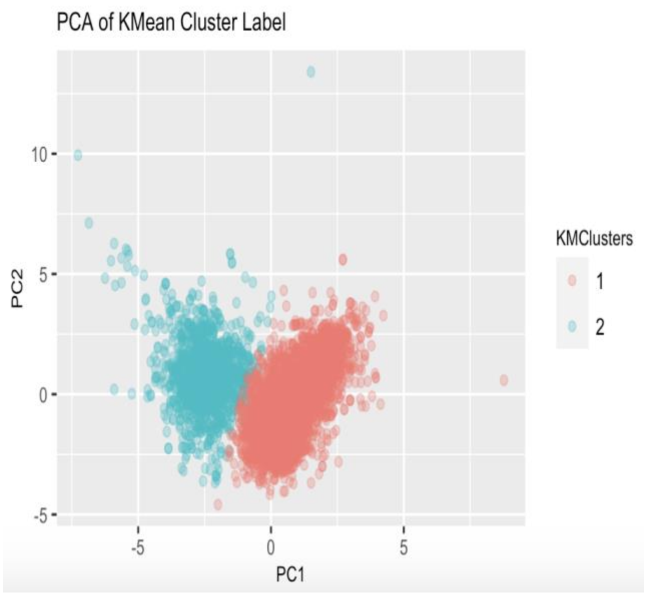
Considering the hierarchical agglomerative clustering output based on different linkage and distance methods, each of the algorithms is examined. As for the method of complete linkage, known as the farthest neighbor method, tend to measure the proximity between two clusters based on the distance between their two farthest objects. In other words, the distance between two clusters is defined as the maximum distance between any two points in the two clusters. However, using this method, clusters tend to form compact contours by their borders, but they are not necessarily compact inside. Hence, it may not be suitable for our wine dataset with irregular shapes or clusters that overlap.

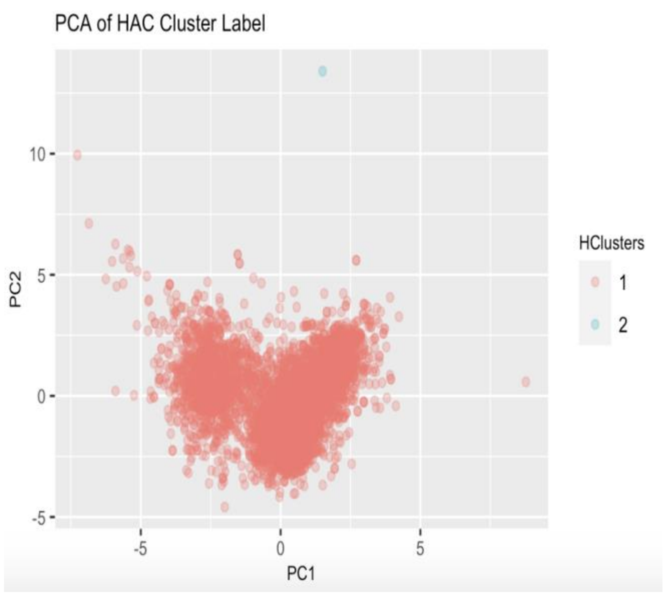
Meantime, the average method considers the average proximity between all pairs of objects in the two clusters and gives equal weight to the subclusters that were recently merged to form the clusters. However, it can result in clusters that are not as compact as those produced by the complete linkage method.
Lastly, if there is a specific number of true clusters, one should always K-Means, as HAC performs much worse in terms of classifying since it uses a bottom-up unsupervised approach.



```
          title = element_text(size = 8))
#HAC
hfit<- hclust(dist(predictors, method = 'euclidean'),method='average')
fviz_nbclust(predictors,FUN = hcut , method = "silhouette")
hfitTree <- cutree(hfit,k=2)
fitKM<- kmeans(predictors,centers = 2,nstart = 15)

rotated_data$HClusters = as.factor(hfitTree)
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col =HClusters)) +
  geom_point(alpha = 0.3)+
  labs(title = "PCA of HAC Cluster Label")+
  theme(axis.text.x = element_text(size = 8),
        axis.text.y = element_text(size = 8),
        title = element_text(size = 8))
##Kmean
rotated_data$KMClusters = as.factor(fitKM$cluster)
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col=KMClusters)) +
  geom_point(alpha = 0.3)+
  labs(title = "PCA of KMean Cluster Label")+
  theme(axis.text.x = element_text(size = 8),
        axis.text.y = element_text(size = 8),
        title = element_text(size = 8))
```



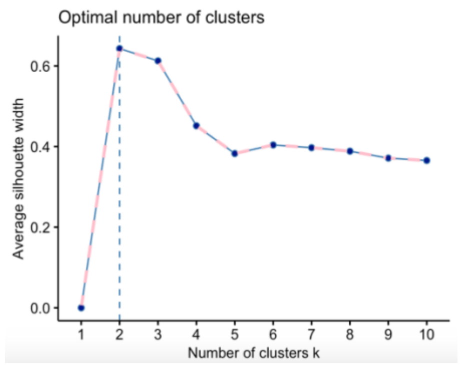


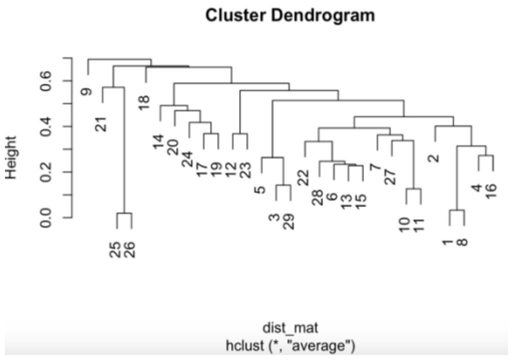


Problem #4 HAC vs. K-Mean Clustering

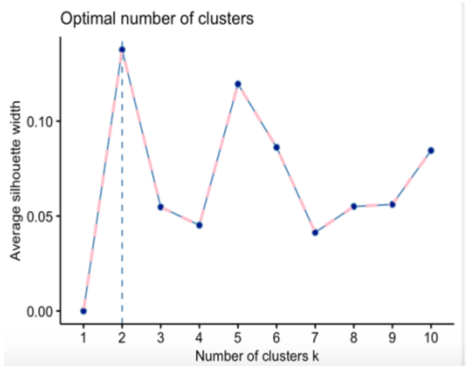
I. HAC Clustering

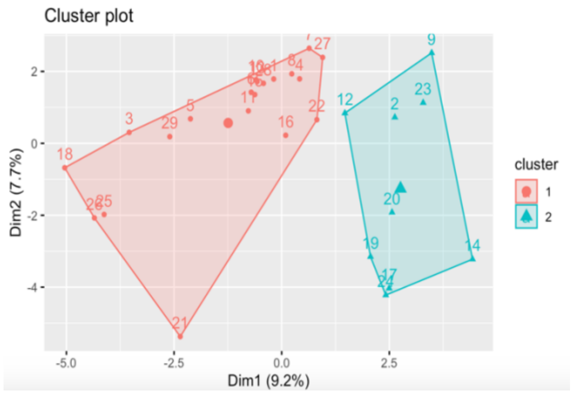
An anomaly might show up in the dendrogram as its own branch has no relationship to other members of the dataset. Based on this dataset, Wookiee and Ewok might be the two main anomalies based on height. The dendrogram output of the algorithm can be used to understand the big picture as well as the groups in the data. One key advantage is the ability to view anomalies directly without reviewing the actual dataset in a tabulation. However, the dendrogram can only show the anomalies here based on a certain attribute.

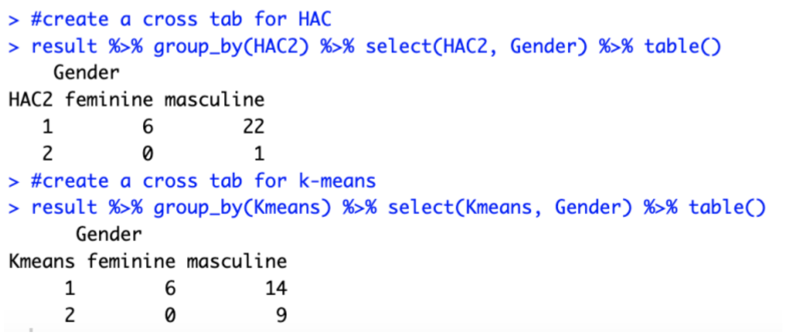




II. K- Means Clustering







```
> #create a cross tab for HAC
> result %>% group_by(HAC2) %>% select(HAC2, Gender) %>% table()
    Gender
HAC2 feminine masculine
   1        6        22
   2        0         1
> #create a cross tab for k-means
> result %>% group_by(Kmeans) %>% select(Kmeans, Gender) %>% table()
      Gender
Kmeans feminine masculine
     1        6        14
     2        0         9
```