# Question 3

*Bejan Sadeghian*

*August 7, 2015*

# Question 3 of Homework 1

## Objective

Use Clustering and PCA to evaluate the best way to describe different wines based on their chemical attributes. (Classify into Red or White and also the quality)

## Conclusion

- The K-Means clustering approach allowed me the ability to group wines with similar colours based on their chemical components with a relatively high accruacy. K=5 allowed from 95-99% accuracy for determining if a wine was red or white whereas K=10 ranged from 88-99% so it was less accurate.
    - Quality was not as effective, around 50% accuracy determineing the wine’s quality rating. I believe this was caused because of K-Means’ limitation to be able to cluster effectively when clusters are different sizes.
- Principal Component Analysis was able to capture 95% of the varaince from the original data however because of the high number of dimensions to plot based on, it was difficult to be able to plot the information from this analysis.
- My conclusion is clustering creates a much more intepretable result. I would recommend using K-Means to cluster the wines to predict their color. Quality is much more difficult given the amount of categories each can be in.

## Format for my report

Ive “stepped” through each part of my code below for both Clustering and PCA including a snap shot of the code itself. After the steps I provide the executable code with charts and tables to show.

## Clustering

Step 1: Import the mosaic and ggplot2 library

```
library(mosaic)
library(ggplot2)
```

Step 2: Read in the data and scale the numerical attributes

```r
rawdata = read.csv('wine.csv', header=TRUE)

wineData = scale(rawdata[,1:11]) #Scale the data
wineData = cbind(wineData,rawdata[,12:13])
```

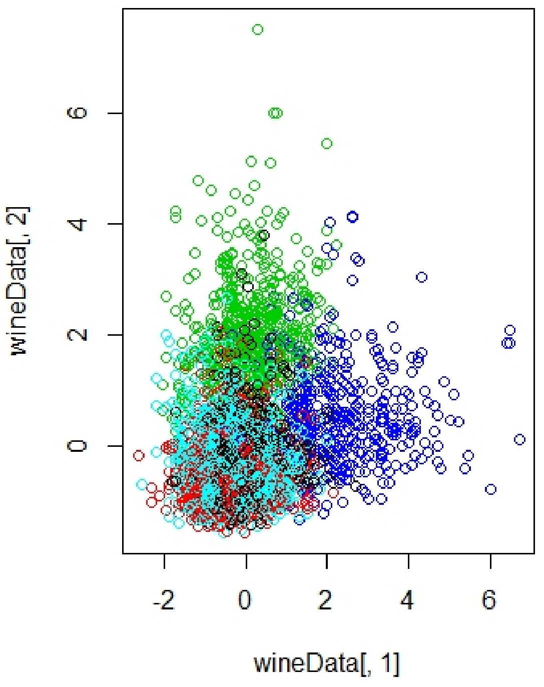
Step 3) What I did next was run a for loop to find the best number of clusters to decrease my total within cluster sum of square distances. I was careful to not go with a K number that was too high. The resulting plot for testing K=1 to K=10 is below the code. My decision for K was to use 5 centroids because that is the “knee” of the plot. I tried using Bayesian Information Criterion to determine the optimal K, but the BIC function I was using took upwards of 30 minutes to run so I decided to decide this the “dumb” way, knee.

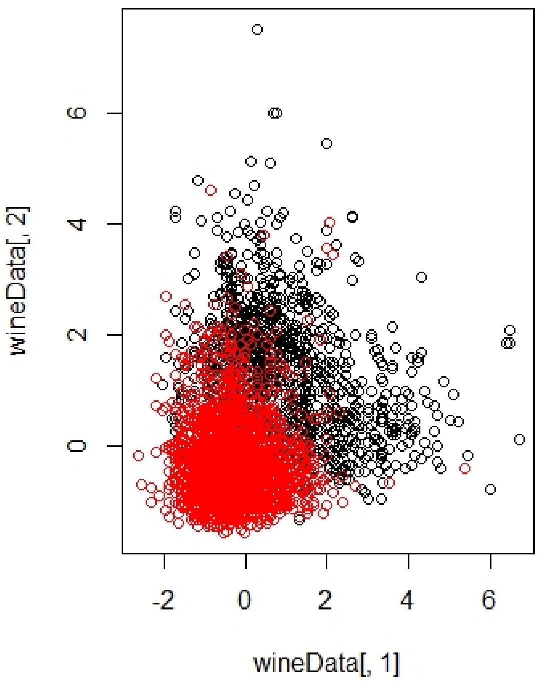
```r
#Find a K number that will lead to the
numK = 10
SSEArray = rep(0,numK)
#BICArray = rep(0,numK)
for(i in 1:numK){
  wineCluster = kmeans(wineData[1:11],centers = i, nstart = 10)
  SSEArray[i] = wineCluster$tot.withinss
}
plot(SSEArray)
```

Step 5) I then perform the K Means for 5 centroids and plot the data in a barchart showing each cluster’s count split apart by red and white color wine classification provided.

- I do run a K-Means of 10 centroids here as well specifically to show percent accuracy in predicting the Red vs White wine color. You can see here that the percent accuracy for 5 centroids was consistently higher across all clusters as opposed to K=10.

- The reason I did not do a scatter plot is because as you can see from one of the images I saved of a scatter plot, there does appear to be seperation, but we cant say for certain that one cluster does not overlap the other (Red vs White is on the right, Clusters K=5 is on the left)





```
#Performing K-means and plotting
numK = 5

for(r in c(10,5)){
numK = r
ClusterAll = kmeans(wineData[1:11], centers=numK, nstart=10)

finalwine = cbind(wineData, ClusterAll$cluster)
finalwine = finalwine[,c(13,14)]
names(finalwine)[2]<-'cluster'

qplot(cluster, data=finalwine, geom="bar", fill=as.factor(cluster)) + facet_wra
p(~ color, ncol = 6)

#Calculating the percent accuracy at guessing the color of wine per cluster

ResultsTable = table(finalwine)
red = rep(0,numK)
count = 1
for(i in seq(1,numK*2,2)){
  red[count] = ResultsTable[i]/sum(ResultsTable[i],ResultsTable[i+1])
  count = count+1
}

white = rep(0,numK)
count = 1
for(i in seq(1,numK*2,2)){
  white[count] = ResultsTable[i+1]/sum(ResultsTable[i],ResultsTable[i+1])
  count = count+1
}

ColorAccuracy = rbind(red,white)
print('Color Accuracy')
print(ColorAccuracy)

}
```

Step 6) Finally, to see how well I classified each quality rating provided by the experts, I did a similar thing that I did to color. The table below shows centroids of 5 and 10 for my K-Means evaluation however in both cases the accuracy was at best 40-50%

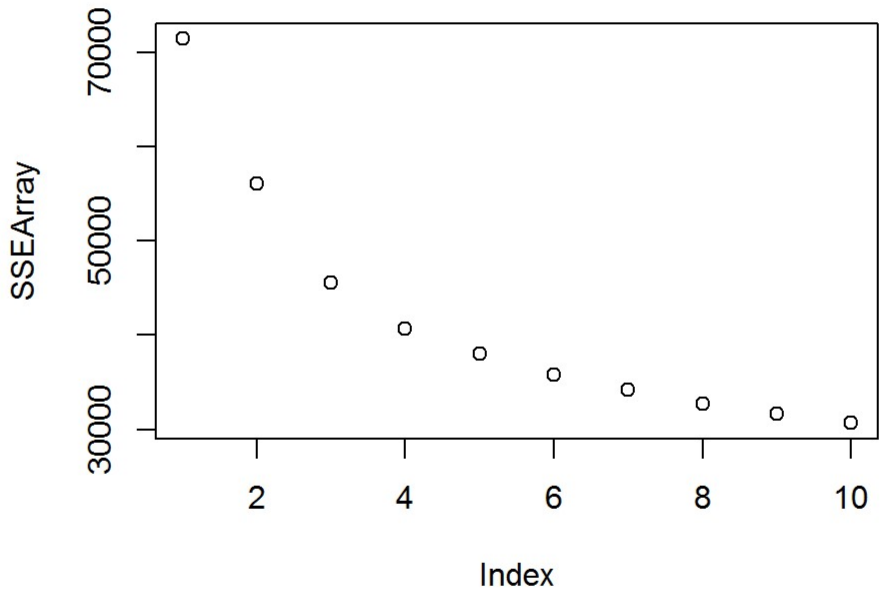
```
##Calculate the accuracy of quality clustering
for(r in c(5,10)){
  numK = r
  ClusterAll = kmeans(wineData[1:11], centers=numK, nstart=10)

  names(ClusterAll)

  finalwine = cbind(wineData, ClusterAll$cluster)
  finalwine = finalwine[,c(12,14)]
  names(finalwine)[2]<-'cluster'

  qplot(cluster, data=finalwine, geom="bar", fill=as.factor(cluster)) + facet_w
rap(~ quality, ncol = 6)

  #Calculating the percent accuracy at guessing the color of wine per cluster

  ResultsTable = as.data.frame(table(finalwine))

  QualityAccuracy = c('Quality 1','Quality 2','Quality 3','Quality 4','Quality
5','Quality 6','Quality 7','Quality 8','Quality 9','Quality 10')
  for(x in 1:9){
    cluster = rep(0,numK)
    for(y in 1:numK){
      cluster[y] = with(ResultsTable,sum(ResultsTable[quality==x & cluster==
y,'Freq']))/with(ResultsTable,sum(ResultsTable[cluster==y,'Freq']))
    }
    QualityAccuracy = rbind(QualityAccuracy,t(as.data.frame(cluster)))
  }
  rownames(QualityAccuracy) = c('Names',1,2,3,4,5,6,7,8,9)
  print('K')
  print(numK)
  print(QualityAccuracy)
  }
```

The Code for K-Means clustering is below with charts.

```r
library(mosaic)
library(ggplot2)
rawdata = read.csv('wine.csv', header=TRUE)

set.seed(1)
wineData = scale(rawdata[,1:11]) #Scale the data
wineData = cbind(wineData,rawdata[,12:13])

#Find a K number that will lead to the
numK = 10
SSEArray = rep(0,numK)
#BICArray = rep(0,numK)
for(i in 1:numK){
  wineCluster = kmeans(wineData[1:11],centers = i, nstart = 10)
  SSEArray[i] = wineCluster$tot.withinss
}
```

```
## Warning: did not converge in 10 iterations
```

```r
plot(SSEArray)
```



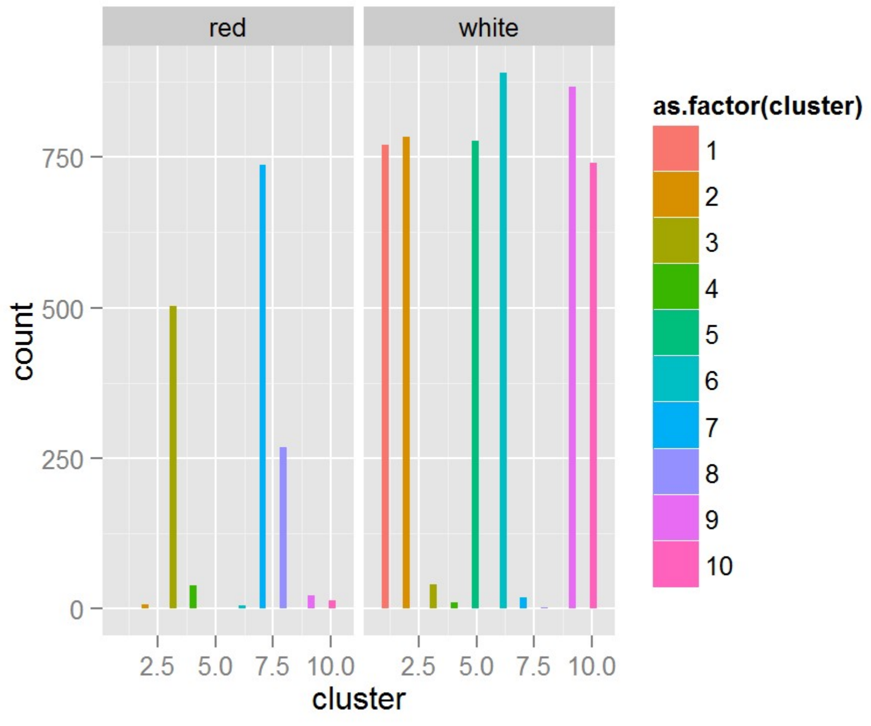
```r
#Performing K-means and plotting
numK = 5

for(r in c(5,10)){
numK = r
ClusterAll = kmeans(wineData[1:11], centers=numK, nstart=10)

finalwine = cbind(wineData, ClusterAll$cluster)
finalwine = finalwine[,c(13,14)]
names(finalwine)[2]<-'cluster'

qplot(cluster, data=finalwine, geom="bar", fill=as.factor(cluster)) + facet_wra
p(~ color, ncol = 6)

#Calculating the percent accuracy at guessing the color of wine per cluster

ResultsTable = table(finalwine)
red = rep(0,numK)
count = 1
for(i in seq(1,numK*2,2)){
  red[count] = ResultsTable[i]/sum(ResultsTable[i],ResultsTable[i+1])
  count = count+1
}

white = rep(0,numK)
count = 1
for(i in seq(1,numK*2,2)){
  white[count] = ResultsTable[i+1]/sum(ResultsTable[i],ResultsTable[i+1])
  count = count+1
}

ColorAccuracy = rbind(red,white)
print('Color Accuracy')
print(ColorAccuracy)

}
```

```
## [1] "Color Accuracy"
##              [,1]       [,2]       [,3]       [,4]        [,5]
## red   0.01948843 0.01511487 0.93860562 0.94783905 0.002549395
## white 0.98051157 0.98488513 0.06139438 0.05216095 0.997450605
## [1] "Color Accuracy"
##               [,1]       [,2]       [,3]      [,4]        [,5]        [,6]
## red   0.001297017 0.01011378 0.92633517 0.7959184 0.001287001 0.005586592
## white 0.998702983 0.98988622 0.07366483 0.2040816 0.998712999 0.994413408
##              [,7]       [,8]      [,9]      [,10]
## red   0.97486772 0.98892989 0.0258427 0.01856764
## white 0.02513228 0.01107011 0.9741573 0.98143236
```

```
print('K=5 bar chart')
```

```
## [1] "K=5 bar chart"
```

```
qplot(cluster, data=finalwine, geom="bar", fill=as.factor(cluster)) + facet_wra
p(~ color, ncol = 6)
```



```
##Calculate the accuracy of quality clustering
for(r in c(5,10)){
  numK = r
  ClusterAll = kmeans(wineData[1:11], centers=numK, nstart=10)

  names(ClusterAll)

  finalwine = cbind(wineData, ClusterAll$cluster)
  finalwine = finalwine[,c(12,14)]
  names(finalwine)[2]<-'cluster'

  qplot(cluster, data=finalwine, geom="bar", fill=as.factor(cluster)) + facet_w
rap(~ quality, ncol = 6)

  #Calculating the percent accuracy at guessing the color of wine per cluster

  ResultsTable = as.data.frame(table(finalwine))

  QualityAccuracy = c('Quality 1','Quality 2','Quality 3','Quality 4','Quality
5','Quality 6','Quality 7','Quality 8','Quality 9','Quality 10')
  for(x in 1:9){
    cluster = rep(0,numK)
    for(y in 1:numK){
      cluster[y] = with(ResultsTable,sum(ResultsTable[quality==x & cluster==
y,'Freq']))/with(ResultsTable,sum(ResultsTable[cluster==y,'Freq']))
    }
    QualityAccuracy = rbind(QualityAccuracy,t(as.data.frame(cluster)))
  }
  rownames(QualityAccuracy) = c('Names',1,2,3,4,5,6,7,8,9)
  print('K')
  print(numK)
  print(QualityAccuracy)
  }
```

```
## Warning in rbind(QualityAccuracy, t(as.data.frame(cluster))): number of
## columns of result is not a multiple of vector length (arg 1)
## [1] "K"
## [1] 5
##       [,1]                  [,2]                  [,3]
## Names "Quality 1"           "Quality 2"           "Quality 3"
## 1     "0"                   "0"                   "0"
## 2     "0"                   "0"                   "0"
## 3     "0.0030229746070133"  "0.0072840790842872"  "0.00894187779433681"
## 4     "0.0247883917775091"  "0.0697190426638918"  "0.0312965722801788"
## 5     "0.119105199516324"   "0.489073881373569"   "0.329359165424739"
## 6     "0.455864570737606"   "0.374609781477627"   "0.406855439642325"
## 7     "0.323458282950423"   "0.0541103017689906"  "0.205663189269747"
## 8     "0.0713422007255139"  "0.00520291363163371" "0.0178837555886736"
## 9     "0.00241837968561064" "0"                   "0"
##       [,4]                  [,5]
## Names "Quality 4"           "Quality 5"
## 1     "0"                   "0"
## 2     "0"                   "0"
## 3     "0.00243605359317905" "0.00509878903760357"
## 4     "0.0347137637028015"  "0.0191204588910134"
## 5     "0.342265529841657"   "0.438495857233907"
## 6     "0.465286236297199"   "0.436583811344806"
## 7     "0.135200974421437"   "0.0841300191204589"
## 8     "0.0200974421437272"  "0.0159337157425112"
## 9     "0"                   "0.000637348629700446"
```

```
## Warning: did not converge in 10 iterations
```

```
## Warning: did not converge in 10 iterations
## [1] "K"
## [1] 10
##       [,1]                 [,2]                 [,3]
## Names "Quality 1"          "Quality 2"          "Quality 3"
## 1     "0"                  "0"                  "0"
## 2     "0"                  "0"                  "0"
## 3     "0.00376293508936971" "0.00348027842227378" "0.00213219616204691"
## 4     "0.0573847601128881"  "0.0208816705336427"  "0.0490405117270789"
## 5     "0.373471307619944"   "0.412993039443155"   "0.654584221748401"
## 6     "0.436500470366886"   "0.424593967517401"   "0.277185501066098"
## 7     "0.110065851364064"   "0.117169373549884"   "0.0170575692963753"
## 8     "0.0188146754468485"  "0.0197215777262181"  "0"
## 9     "0"                   "0.00116009280742459" "0"
##       [,4]                 [,5]                 [,6]
## Names "Quality 4"          "Quality 5"          "Quality 6"
## 1     "0"                  "0"                  "0"
## 2     "0"                  "0"                  "0"
## 3     "0.00223463687150838" "0.00184672206832872" "0.00662983425414365"
## 4     "0.0256983240223464"  "0.012927054478301"   "0.0132596685082873"
## 5     "0.210055865921788"   "0.061865189289012"   "0.456353591160221"
## 6     "0.491620111731844"   "0.456140350877193"   "0.459668508287293"
## 7     "0.23463687150838"    "0.377654662973223"   "0.0497237569060773"
## 8     "0.0357541899441341"  "0.0858725761772853"  "0.0143646408839779"
## 9     "0"                   "0.00369344413665743" "0"
##       [,7]                [,8]                [,9]
## Names "Quality 7"         "Quality 8"         "Quality 9"
## 1     "0"                 "0"                 "0"
## 2     "0"                 "0"                 "0"
## 3     "0.0344827586206897" "0.0150943396226415" "0.0118845500848896"
## 4     "0.0689655172413793" "0.0226415094339623" "0.0848896434634974"
## 5     "0.655172413793103"  "0.256603773584906"  "0.427843803056027"
## 6     "0.206896551724138"  "0.471698113207547"  "0.397283531409168"
## 7     "0.0344827586206897" "0.211320754716981"  "0.0713073005093379"
## 8     "0"                  "0.0226415094339623" "0.0067911714770798"
## 9     "0"                  "0"                  "0"
##       [,10]
## Names "Quality 10"
## 1     "0"
## 2     "0"
## 3     "0"
## 4     "0.0207715133531157"
## 5     "0.210682492581602"
## 6     "0.477744807121662"
## 7     "0.267062314540059"
## 8     "0.0237388724035608"
## 9     "0"
```

## Principal Component Analysis

Step 1) First we read in the data again and remove the quality and color for our analysis

```
#Reimporting the data
rawdata = read.csv('wine.csv', header=TRUE)

#Remove color and quality rating
workingData = rawdata[,1:11]
```

Step 2) I center but dont scale the data here because of what was mentioned in class.

```
#Centering the data but not scaling
centeredData = scale(workingData, center=TRUE, scale=FALSE)
```

Step 3) I then use the prcomp() function to calculate the principal components (v)

```
#Calculating the principal components and extracting PC1
PC1 = prcomp(centeredData)
v_best = PC1$rotation[,1]
```

Step 4) Calculate the alpha (scalar score) of the projections by doing the innerproduct of the centered datapoints against the vector domain

```
#inner product of x and v (PC1) gives us the alpha, the scalar score, of each p
rojection
alpha_best = centeredData %*% v_best
```

Step 5) Finally to find out how much information our PC1 captured in terms of variance of the data, I calculate the variance of each piece of data and divide the variance of the alphas against that.

```
#Calculating the amount of variance captured
varofdata = apply(centeredData,2,var)
print(var(alpha_best)/sum(varofdata))
```

Below is the code for PCA, the output is the percent of variance captured by PC1. Because there were so many dimensions I was not sure what I could have plotted.

```r
#Reimporting the data
rawdata = read.csv('wine.csv', header=TRUE)

#Remove color and quality rating
workingData = rawdata[,1:11]

#Centering the data but not scaling
centeredData = scale(workingData, center=TRUE, scale=FALSE)

#Calculating the principal components and extracting PC1
PC1 = prcomp(centeredData)
v_best = PC1$rotation[,1]

#inner product of x and v (PC1) gives us the alpha, the scalar score, of each p
rojection
alpha_best = centeredData %*% v_best

#Calculating the amount of variance captured
varofdata = apply(centeredData,2,var)
print('Amount of variance captured')
```

```
## [1] "Amount of variance captured"
```

```r
print(var(alpha_best)/sum(varofdata))
```

```
##           [,1]
## [1,] 0.9537583
```