

Model Based Clustering

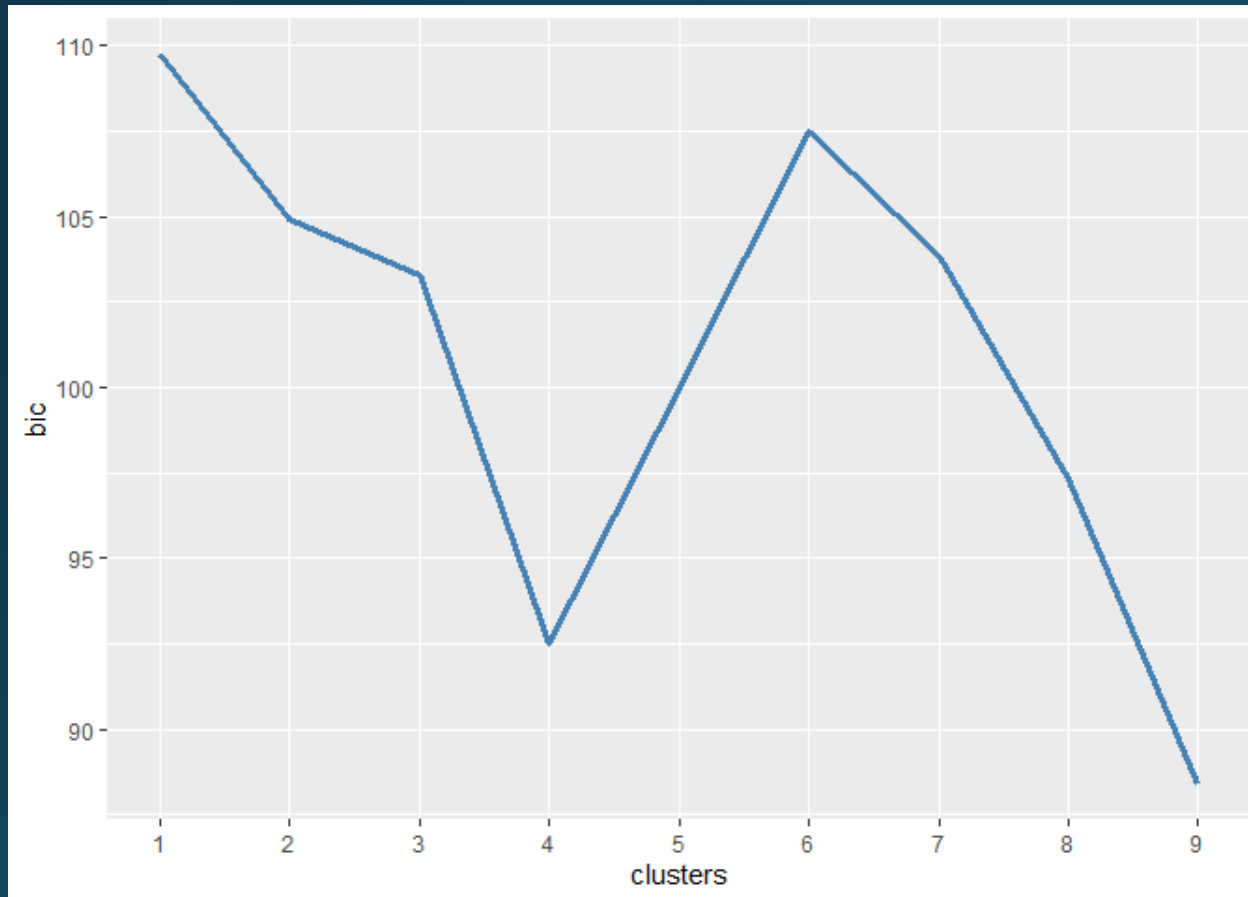
Model-Based Clustering

- Key idea of model-based clustering is that observations come from groups with different statistical distributions (such as different means and variances). The algorithms try to find the best set of such underlying distributions to explain the observed data.
- These methods are also known as mixture models because it is assumed that the data reflect a mixture of observations drawn from different populations.
- `mclust()` models such clusters as being drawn from a mixture of normal (also known as Gaussian) distributions.
- Only useful when data are numeric

Model-Based Clustering

- Infers ideal number of clusters by comparing a variety of different mixture shapes
- It is also possible to specify number of clusters
- Different solutions may be compared using log-likelihood or BIC

Number of clusters based on bic



Other Clustering Techniques

- DBSCAN
- OPTICS

R Illustration: Clustering for Segmentation

Dataset

Read Data

Select Variables

Prepare Data

Clustering Methods

Hierarchical Cluster Analysis

K-means Clustering

Model-based clustering

Contrast Results

Profile Clusters



Model-based clustering

Key idea of model-based clustering is that observations come from groups with different statistical distributions (such as different means and variances). The algorithms try to find the best set of such underlying distributions to explain the observed data. These methods are also known as mixture models because it is assumed that the data reflect a mixture of observations drawn from different populations.

Clustering Method


`mclust()` models such clusters as being drawn from a mixture of normal (also known as Gaussian) distributions.

```
library(mclust)
clusters_mclust = Mclust(data_cluster)
```

The optimal cluster solution is the one that performs best on BIC and log.likelihood

```
summary(clusters_mclust)
```

```
## -----  
## Gaussian finite mixture model fitted by EM algorithm  
## -----  
##  
## Mclust EEV (ellipsoidal, equal volume and shape) model with 6 components:  
##  
##   log-likelihood    n  df      BIC      ICL  
##   -3903.212 500 485 -10820.51 -10840.66  
##  
## Clustering table:  
##    1    2    3    4    5    6  
##  12 198   15 166   92   17
```



Of course, one can impose a cluster solution and examine how much worse the log.likelihood and BIC get. Now, a word of caution, some R functions including `mclust` generate negative of BIC (Chapman and Feit 2015) which makes interpretation a bit confusing. To interpret BIC, reverse the sign of the bic reported. Now, lower the bic, better the cluster solution. This means, a solution of $-(-11309.5)$ is better than $-(-14546.64)$.

```
clusters_mclust_3 = Mclust(data_cluster, G=3)
summary(clusters_mclust_3)
```

```
## -----
## Gaussian finite mixture model fitted by EM algorithm
## -----
##
## Mclust EEV (ellipsoidal, equal volume and shape) model with 3 components:
##
## log-likelihood  n  df      BIC      ICL
##      -6170.399 500 248 -13882.02 -13882.57
##
## Clustering table:
##   1   2   3
## 200  43 257
```

```
clusters_mclust_3$bic
```

```
## [1] -13882.02
```

Interpret Results

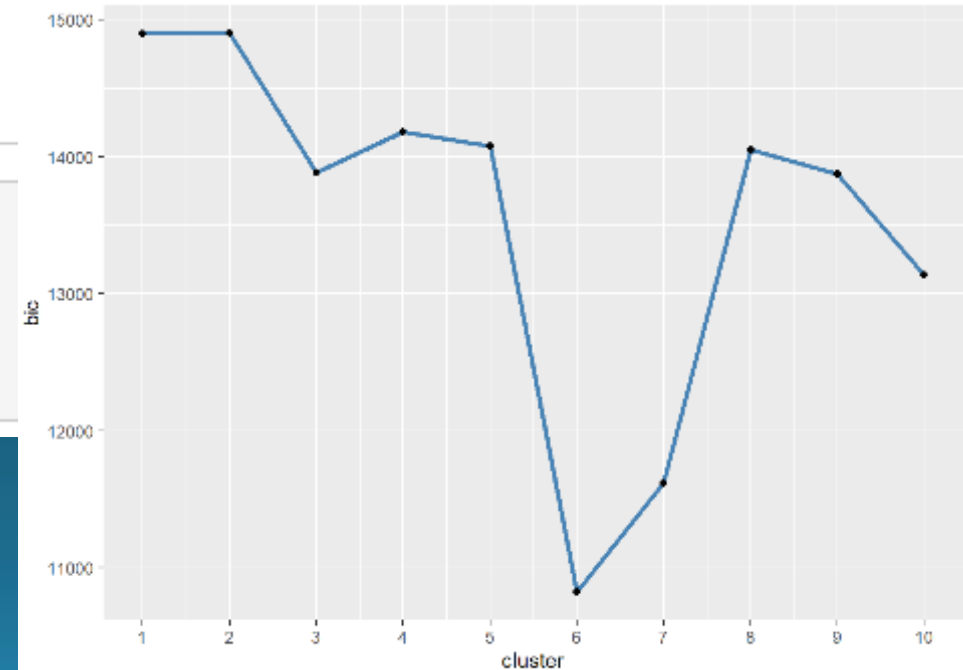
Plot of bic (Bayesian Information Criterion)

Let us generate the Bayesian Information Criterion (bic) for a series of cluster solutions and plot them. Note, the sign of the default bic has been reversed, so we will be looking for the lowest bic in the line graph.

```
mclust_bic = sapply(1:10, FUN = function(x) -Mclust(data_cluster, G=x)$bic)  
mclust_bic
```

```
##      EEE,1  
## 14895.79 14900.60 13882.02 14182.20 14077.96 10820.51 11614.45 14048.39  
##  
## 13872.31 13138.63
```

```
ggplot(data=data.frame(cluster = 1:10, bic = mclust_bic), aes(x=cluster,y=bic))+  
  geom_line(col='steelblue',size=1.2)+  
  geom_point()+  
  scale_x_continuous(breaks=seq(1,10,1))
```



The six-cluster solution has the lowest bic but catering to so many segments may not be practical. Moreover, both hierarchical and k-means favored a four-cluster solution, so we are going to go with a four-cluster solution.

```
m_clusters = Mclust(data=data_cluster, G = 4)
```

Size of the clusters

```
m_segments = m_clusters$classification ←  
table(m_segments)
```

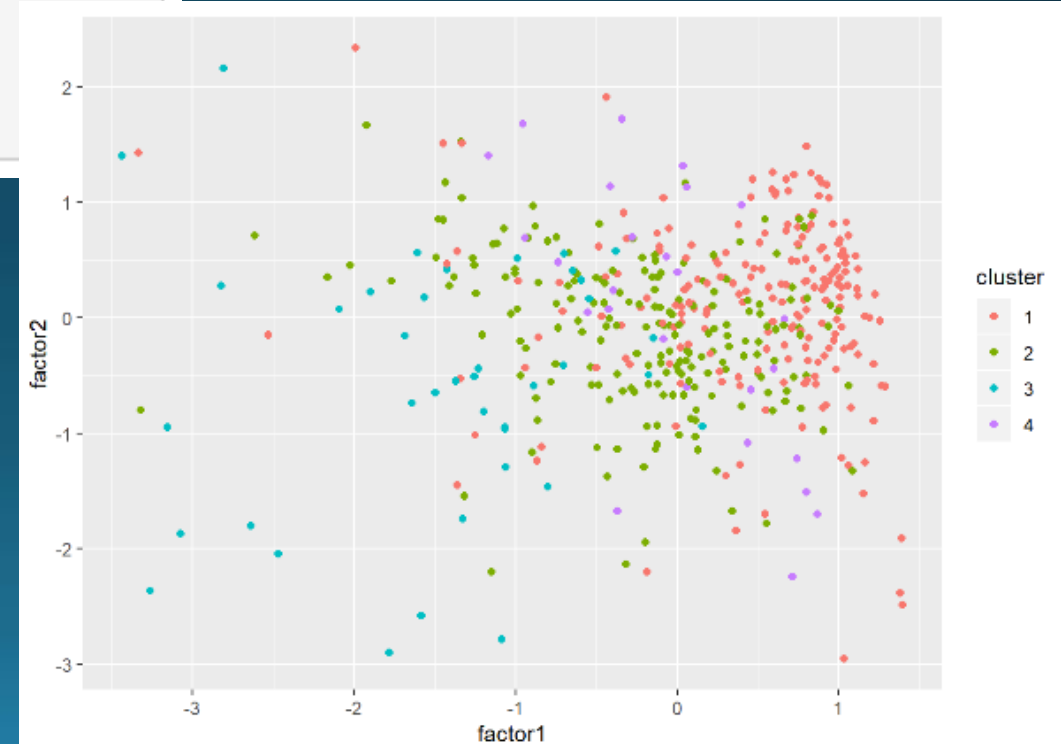
```
## m_segments  
##    1    2    3    4  
## 231 200  43  26
```

Visualize

To express the clusters on a scatterplot, we flatten the data from 12 dimensions onto 2 by conducting a factor analysis with varimax rotation. This is done because it is not possible to visualize 12-dimensional data.

```
library(psych)
temp = data.frame(cluster = factor(m_segments),
  factor1= fa(data_cluster, nfactors=2, rotate='varimax')$scores[,1],
  factor2= fa(data_cluster, nfactors=2, rotate='varimax')$scores[,2])

ggplot(temp, aes(x=factor1, y=factor2, col=cluster))+
  geom_point()
```

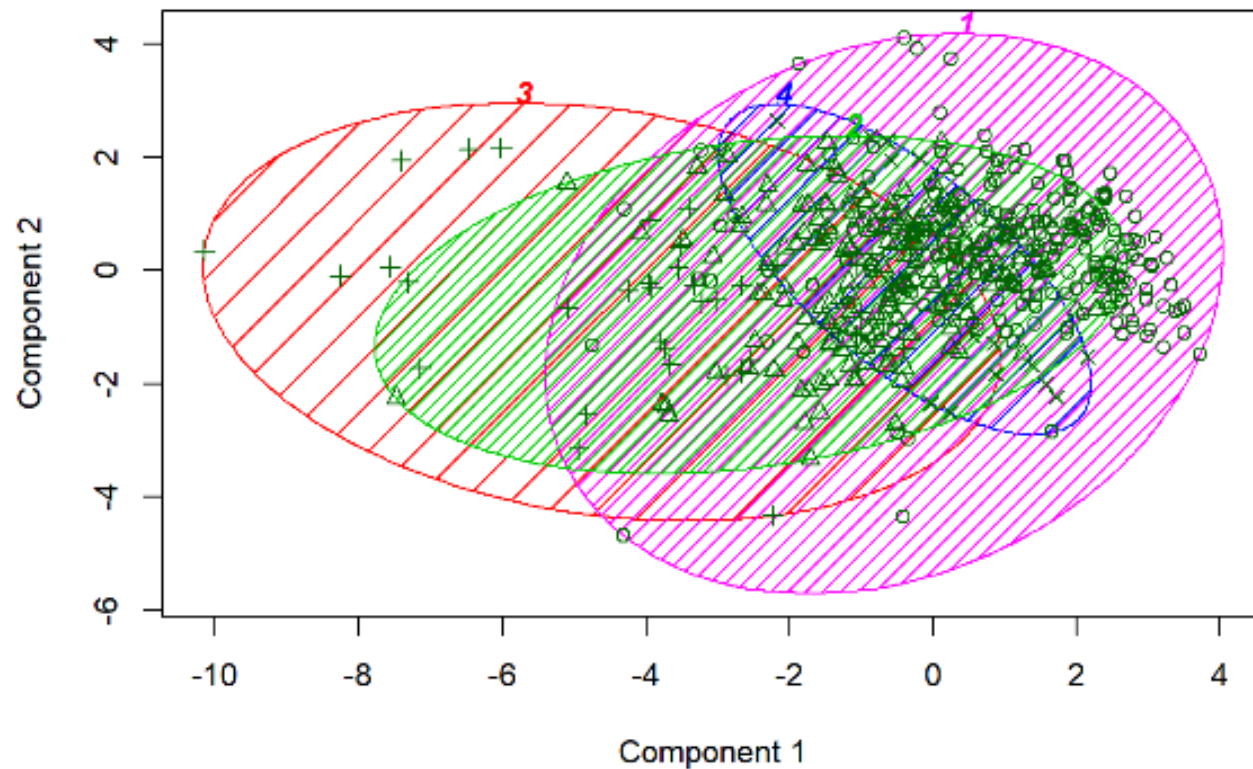


Using clusplot

`clusplot` does something similar.

```
library(cluster)
clusplot(data_cluster,
         m_segments,
         color=T, shade=T, labels=4, lines=0, main='mclust Cluster Plot')
```

mclust Cluster Plot



These two components explain 56.4 % of the point variability.

Contrast Results

The pattern of cluster assignments for hierarchical and k-means are quite similar (although the actual cluster numbers are not the same). The different results of model-based clustering may have to do with the way mixture models work.

```
table(h_segments)
```

```
## h_segments
##    1    2    3    4
## 188 106 190  16
```

```
table(k_segments)
```

```
## k_segments
##    1    2    3    4
## 221 110 155  14
```

```
table(m_segments)
```

```
## m_segments
##    1    2    3    4
## 231 200  43  26
```

Profile Cluster

R Illustration: Clustering for Segmentation

Dataset

Read Data

Select Variables

Prepare Data

Clustering Methods

Hierarchical Cluster Analysis

K-means Clustering

Model-based clustering

Contrast Results

Profile Clusters



Profile Clusters

Let us use the k-means clusters to inspect the characteristics of each cluster based on (1) needs-based variables, and (2) demographics. Profiling clusters by needs-based variables will help us understand how the four market segments differ in their needs which in turn will aid the firm in offering products and services to match the unique needs of the segment. Profiling clusters on demographics or observable variables will help identify customers and target them with the right offer.


Combine segment membership with original data

```
data2 = cbind(data, h_segments, k_segments, m_segments)
```

Profile Segments by Needs

Use k-means segments to examine means of each needs-based variable. Of course, in examining profiles of these segments, bear in mind that segment 4 with only 14 observations (14/500 = 2.8%) is at best a niche segment.



```
library(dplyr)
data2 %>%
  select(performance:quality_of_service, k_segments)%>%
  group_by(k_segments)%>%
  summarize_all(function(x) round(mean(x,na.rm=T),2))%>%
  data.frame()
```




For each group, what is the need based variable with highest value 

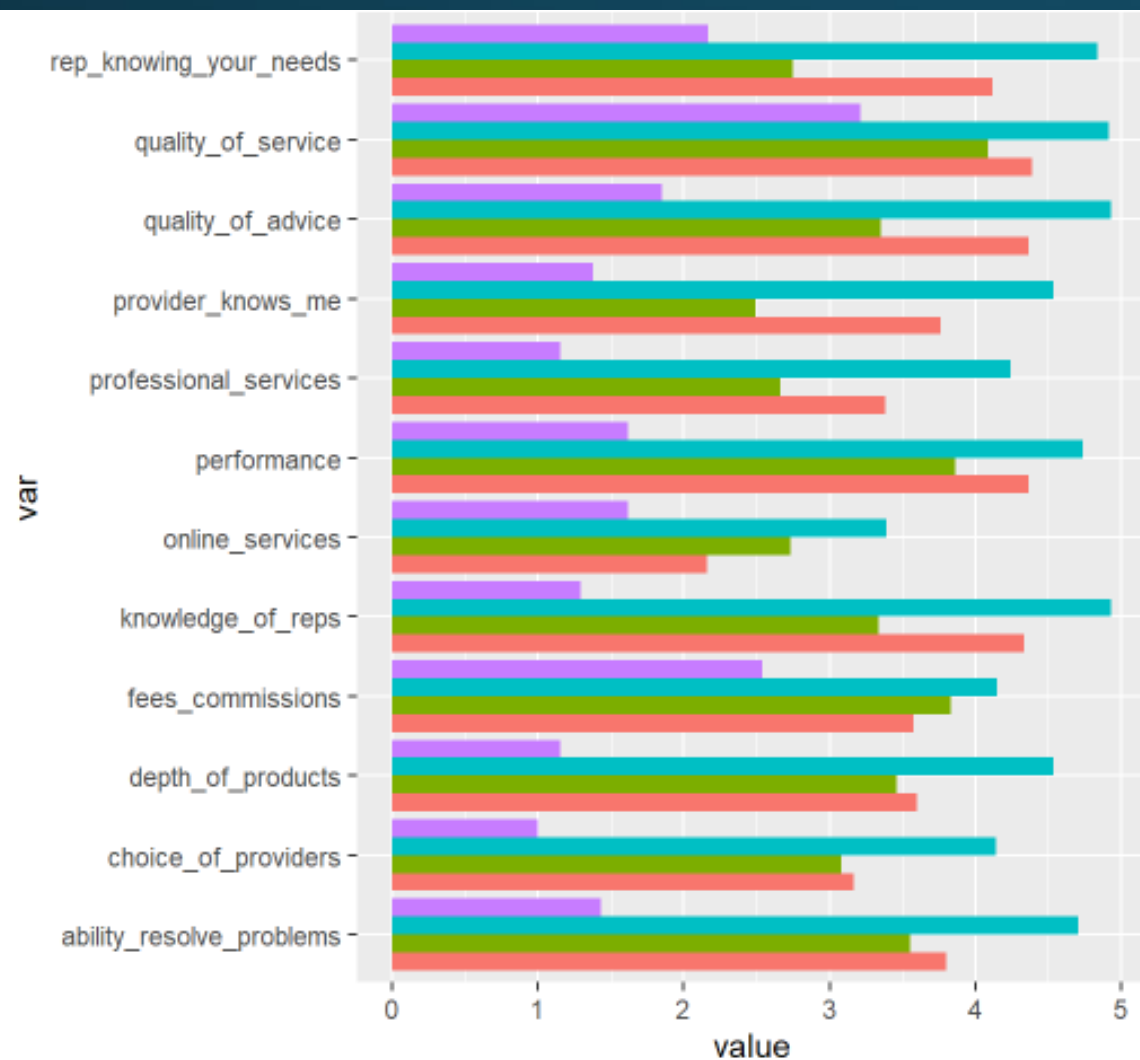
Compare to other groups, group 3 has the highest value in which of the need based variables?

k_segments <int>	performance <dbl>	fees_commissions <dbl>	depth_of_products <dbl>
1	4.37	3.58	3.60
2	3.86	3.83	3.46
3	4.74	4.15	4.54
4	1.62	2.54	1.15

ability_resolve_problems <dbl>	online_services <dbl>	choice_of_providers <dbl>
3.80	2.16	3.17
3.55	2.73	3.08
4.71	3.39	4.14
1.43	1.62	1.00

quality_of_advice <dbl>	knowledge_of_reps <dbl>	rep_knowing_your_needs <dbl>
4.37	4.34	4.12
3.35	3.34	2.75
 4.93	 4.93	4.84
1.85	1.29	2.17

professional_services <dbl>	provider_knows_me <dbl>	quality_of_service <dbl>
3.38	3.76	 4.39
2.66	2.49	 4.09
4.24	4.54	4.92
1.15	1.38	 3.21



```
library(dplyr); library(ggplot2); library(tidyr)
data2 %>%
  select(performance:quality_of_service, k_segments)%>%
  group_by(k_segments)%>%
  summarize_all(function(x) round(mean(x,na.rm=T),2))%>%
  gather(key = var,value = value,performance:quality_of_service)%>%
  ggplot(aes(x=var,y=value,fill=factor(k_segments)))+
    geom_col(position='dodge')+
    coord_flip()
```

Can you describe what each segment values? How do these segments differ in the things they value?

Profile Segments by Demographics

Since data on age, marital status and education are ordinal, it is best to examine counts or proportions. Using `prop.table`, we can find the proportion of each age group for each segment.

```
prop.table(table(data2$k_segments,data2[,14]), 1)
```

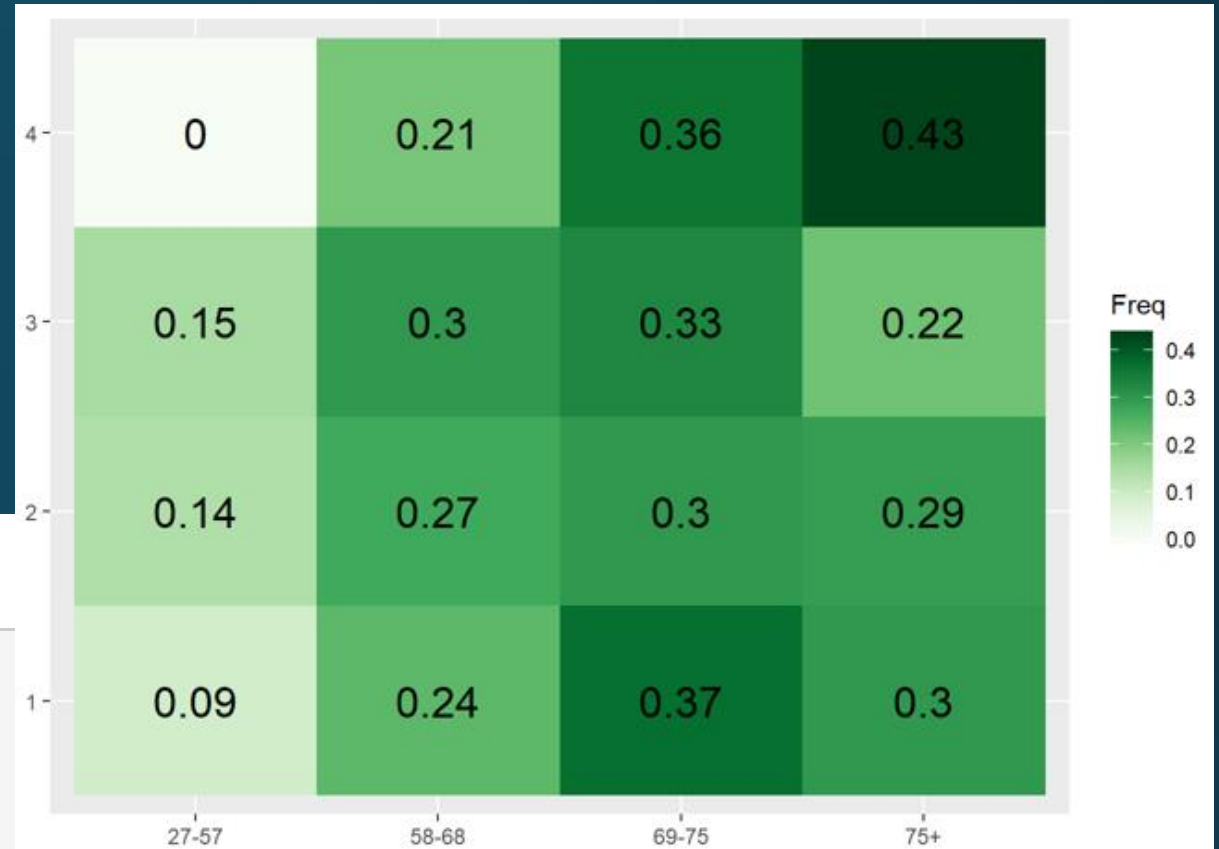
```
##
##          27-57          58-68          69-75          75+
##  1 0.08962264 0.24056604 0.36792453 0.30188679
##  2 0.14285714 0.26666667 0.30476190 0.28571429
##  3 0.14666667 0.30000000 0.33333333 0.22000000
##  4 0.00000000 0.21428571 0.35714286 0.42857143
```

A heatmap may be easier to interpret.

```
library(ggplot2)
library(RColorBrewer)

tab = prop.table(table(data2$k_segments,data2[,14]), 1)
tab2 = data.frame(round(tab, 2))

ggplot(data=tab2,aes(x=Var2,y=Var1,fill=Freq))+
  geom_tile()+
  geom_text(aes(label=Freq),size=6)+
  xlab(label = '')+
  ylab(label = '')+
  scale_fill_gradientn(colors=brewer.pal(n=9,name = 'Greens'))
```



```
## [[1]]
##
##      27-57 58-68 69-75 75+
## 1      9    24    37    30
## 2     14    27    30    29
## 3     15    30    33    22
## 4      0    21    36    43
##
```

```
## [[2]]
##
##      married not married
## 1         84         16
## 2         92          8
## 3         86         14
## 4         93          7
##
```

```
## [[3]]
##
##      no college some college college graduate some graduate school
## 1          8          15          25          12
## 2          8          20          30           8
## 3          9          23          31          11
## 4         14           0          36           0
##
##      masters degree doctorate
## 1          20          19
## 2          20          14
## 3          17           9
## 4          29          21
```

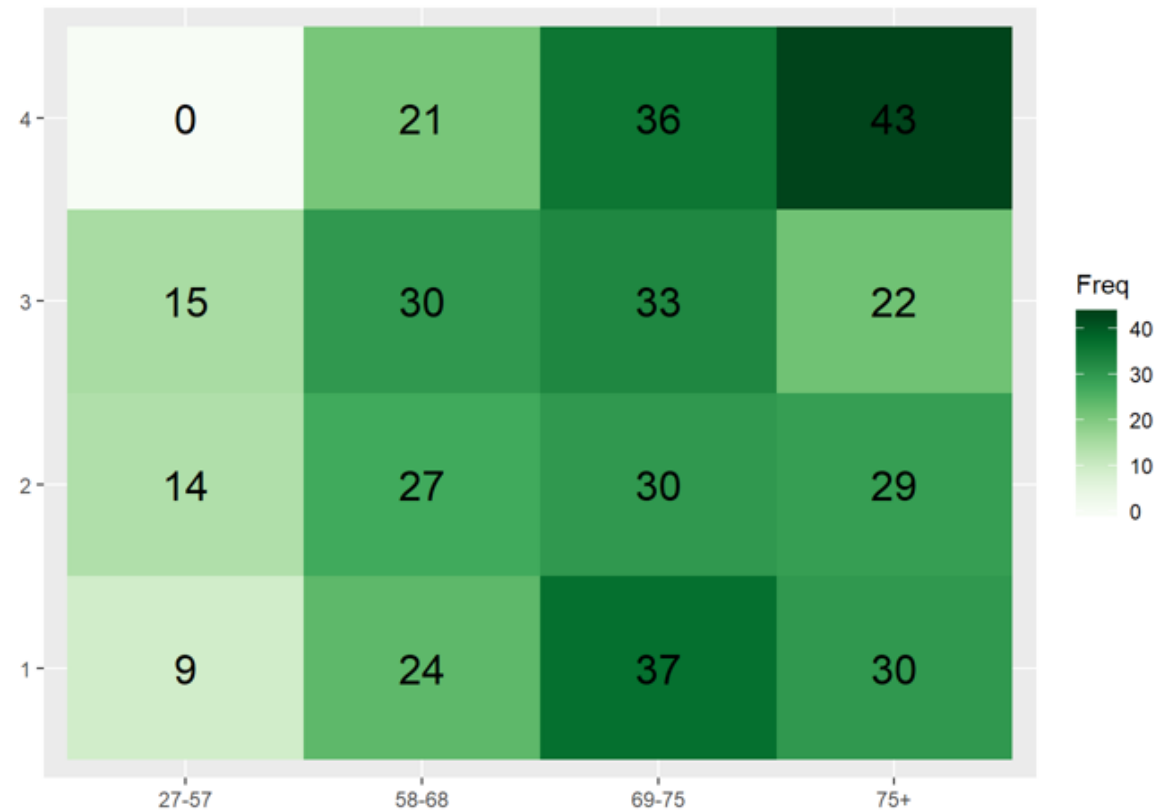
Lets clean up the table for the age groups by rounding and expressing as percentages. And, using lapply, we can apply this function to all three demographic variables at one go, so we can now examine age, marital status and education.

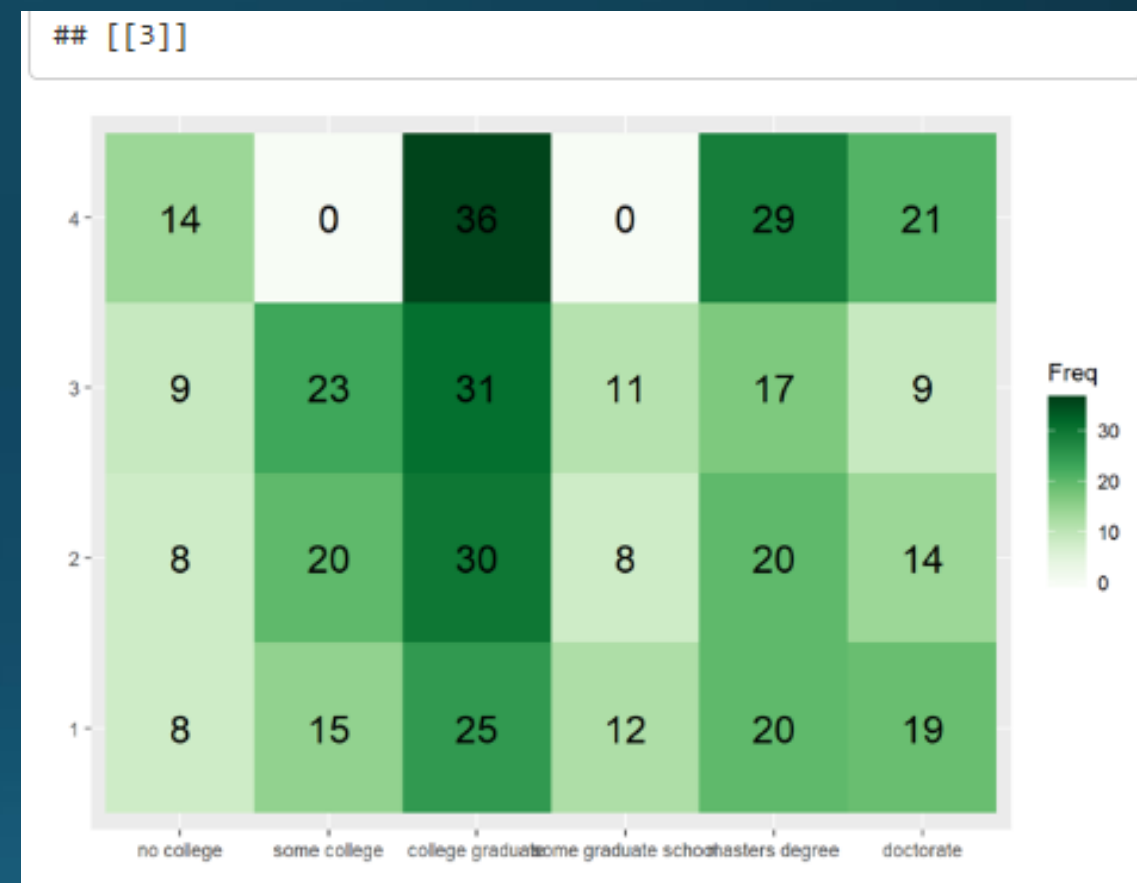
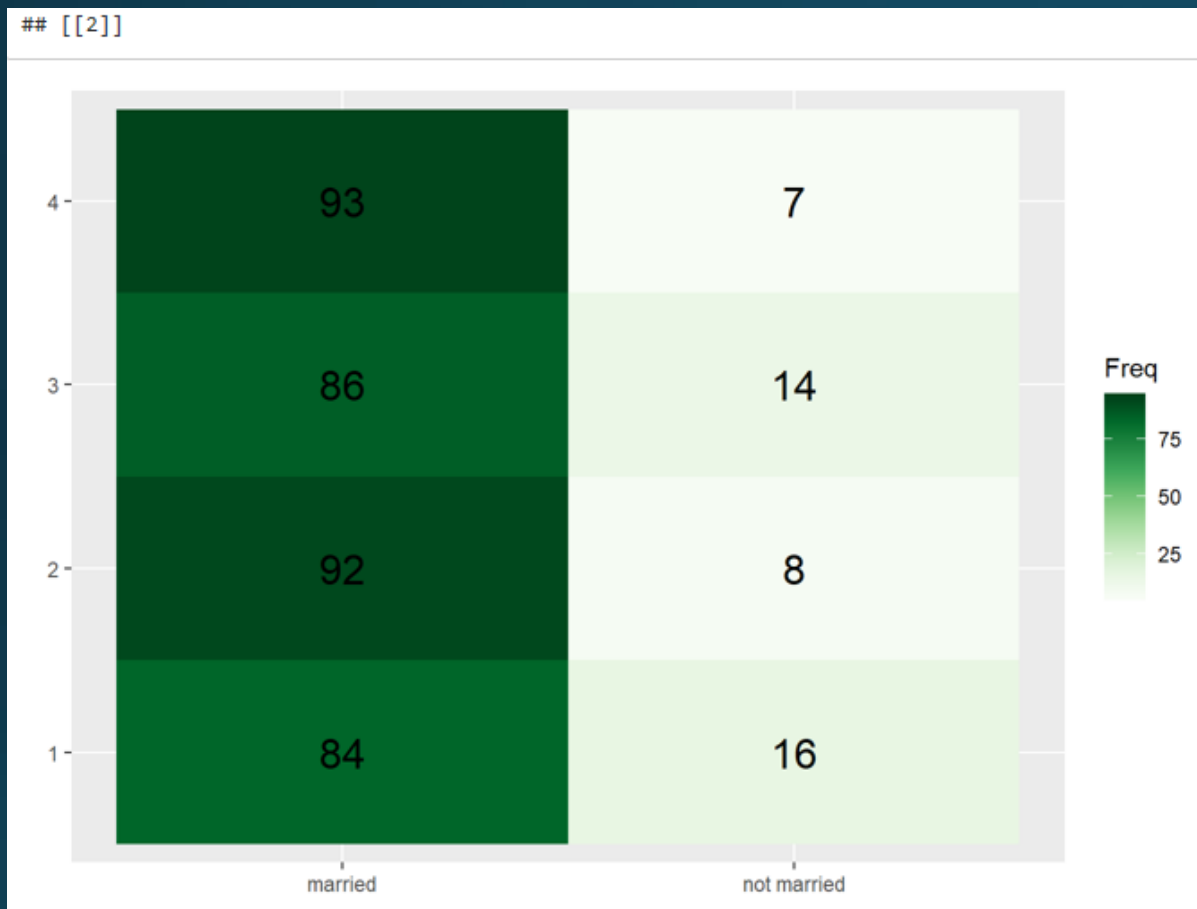
```
lapply(14:16,function(x) round(prop.table(table(data2$k_segments,data2[,x])), 1), 2)*100)
```

Heat Maps of above crostabulations.

```
lapply(14:16,function(x) {
  dat = round(prop.table(table(data2$k_segments,data2[,x]), 1), 2)*100
  dat = data.frame(dat)
  ggplot(data=dat,aes(x=Var2,y=Var1,fill=Freq))+
    geom_tile()+
    geom_text(aes(label=Freq),size=6)+
    xlab(label = '')+
    ylab(label = '')+
    scale_fill_gradientn(colors=brewer.pal(n=9,name = 'Greens'))
})
```

[[1]]





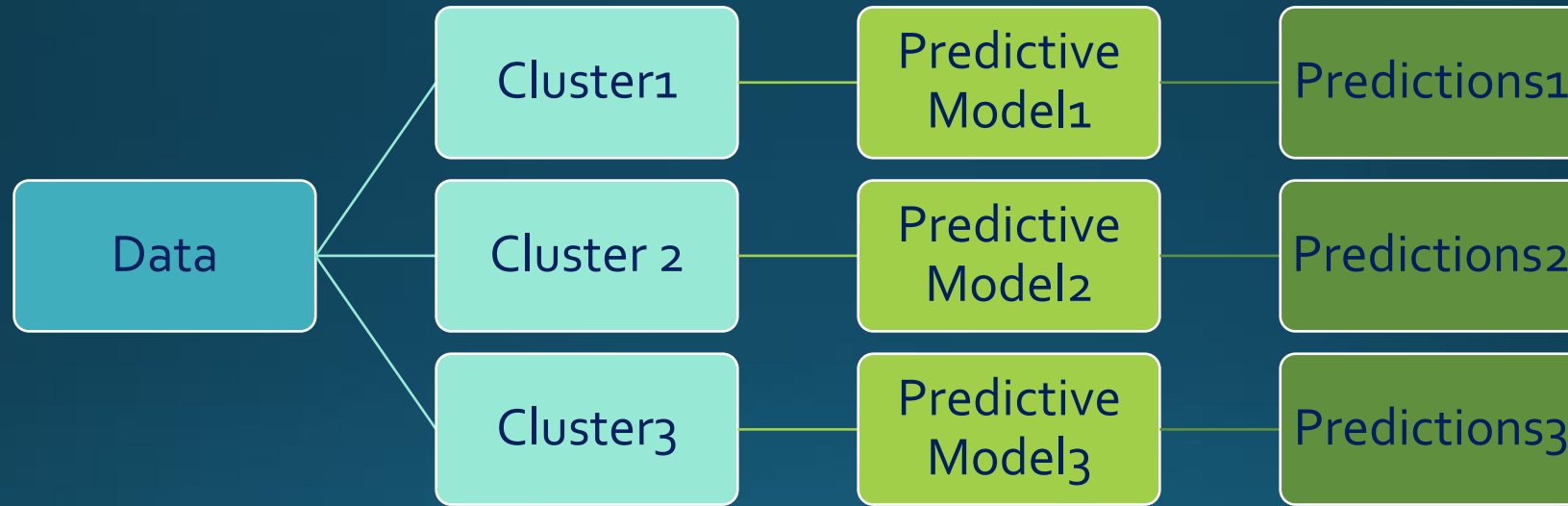
Can you describe the customers in each segment? How do these segments differ?

Input to Predictive Modeling

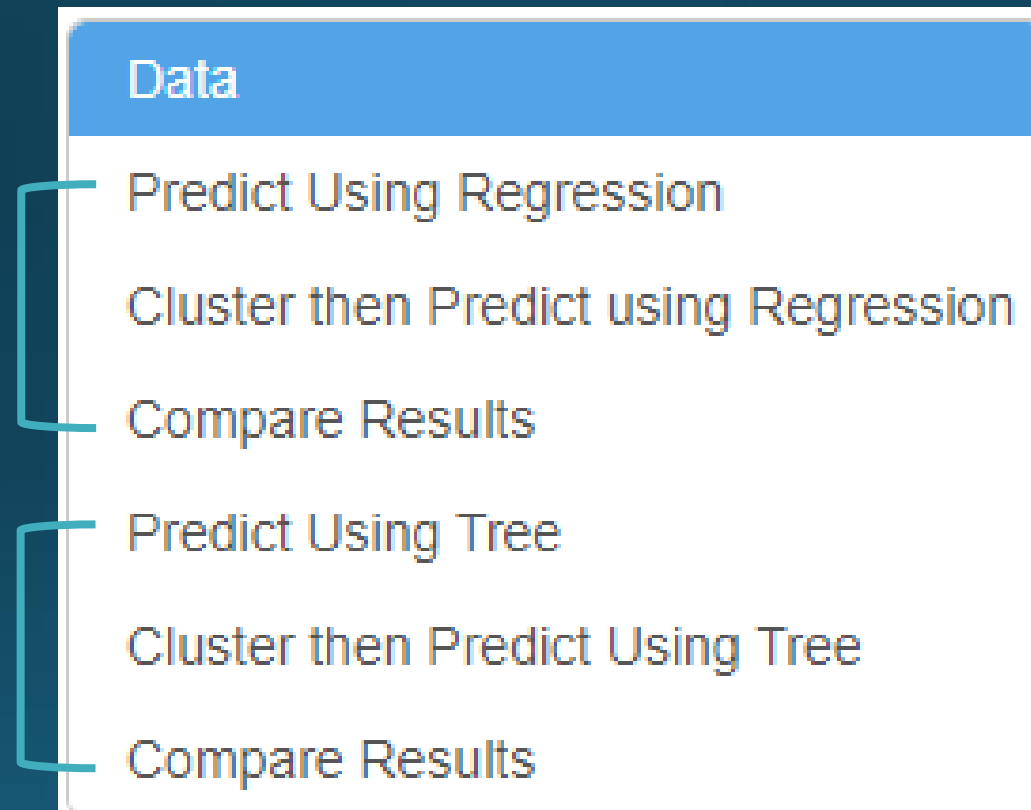
Cluster before Predictive Modeling

- Clustering data before predictive modelling can improve prediction quality
- This approach involves generating clusters of observations and predictive models for each cluster. The predictions are then combined into a single vector.
- Cluster Data – Predictive Model for each Cluster – Combine Predictions

Clustering before Predictive Modeling



R Illustration: Cluster then Predict



Cluster then Predict

In this illustration, we will cluster our data before constructing predictive models.

Data

Read Data

You must read the data before trying to run code on your own machine. To read data use the following code after setting your working directory. To set your working directory, modify the following to set the file path for the folder where the data file resides. `setwd('c:/thatawesomeclass/')`

```
wine = read.table('wine.csv',header=TRUE,sep=';')
```

Explore Data

This dataset contains information on chemical properties of a set of wines (e.g., fixed acidity, alcohol) and a rating of quality.

```
str(wine)
```

```
## 'data.frame':  4898 obs. of  12 variables:
## $ fixed.acidity      : num  7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
## $ volatile.acidity   : num  0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
## $ citric.acid        : num  0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
## $ residual.sugar     : num  20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
## $ chlorides          : num  0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.044 ...
## $ free.sulfur.dioxide : num  45 14 30 47 47 30 30 45 14 28 ...
## $ total.sulfur.dioxide: num  170 132 97 186 186 97 136 170 132 129 ...
## $ density            : num  1.001 0.994 0.995 0.996 0.996 ...
## $ pH                 : num  3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
## $ sulphates          : num  0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
## $ alcohol            : num  8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
## $ quality            : int  6 6 6 6 6 6 6 6 6 6 ...
```

Split Data

```
library(caret)
set.seed(1706)
split = createDataPartition(y=wine$quality,p = 0.7,list = F,groups = 100)
train = wine[split,]
test = wine[-split,]
```

Predict Using Regression

```
linear = lm(quality~.,train)
summary(linear)
```

```
##
## Call:
## lm(formula = quality ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.8275 -0.4929 -0.0384  0.4640  3.0883
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.590e+02  2.132e+01   7.458 1.11e-13 ***
## fixed.acidity    7.730e-02  2.476e-02   3.122  0.00181 **
## volatile.acidity -1.918e+00  1.382e-01 -13.877 < 2e-16 ***
## citric.acid      3.931e-02  1.143e-01   0.344  0.73092
## residual.sugar    8.751e-02  8.746e-03  10.006 < 2e-16 ***
## chlorides        1.878e-01  6.814e-01   0.276  0.78284
## free.sulfur.dioxide 4.252e-03  9.986e-04   4.258 2.12e-05 ***
## total.sulfur.dioxide -7.686e-04  4.459e-04  -1.724  0.08484 .
## density          -1.593e+02  2.165e+01  -7.360 2.29e-13 ***
## pH                7.509e-01  1.264e-01   5.939 3.15e-09 ***
## sulphates         7.120e-01  1.195e-01   5.960 2.78e-09 ***
## alcohol           1.782e-01  2.754e-02   6.472 1.10e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7549 on 3419 degrees of freedom
## Multiple R-squared:  0.2781, Adjusted R-squared:  0.2757
## F-statistic: 119.7 on 11 and 3419 DF,  p-value: < 2.2e-16
```

```
sseLinear = sum(linear$residuals^2); sseLinear
```

```
## [1] 1948.546
```

```
predLinear = predict(linear,newdata=test)
sseLinear = sum((predLinear-test$quality)^2); sseLinear
```

```
## [1] 812.489
```

Cluster then Predict using Regression

Let us cluster first and then run a model to see if there is an improvement. To cluster the wines, we have to remove the outcome variable

```
trainMinusDV = subset(train,select=-c(quality))  
testMinusDV = subset(test,select=-c(quality))
```

Prepare Data for Clustering

Cluster Analysis is sensitive to scale. Standardize the data.

```
library(caret)  
preproc = preProcess(trainMinusDV)  
trainNorm = predict(preproc,trainMinusDV)  
testNorm = predict(preproc,testMinusDV)  
mean(trainNorm$chlorides)
```

```
## [1] 6.795492e-17
```

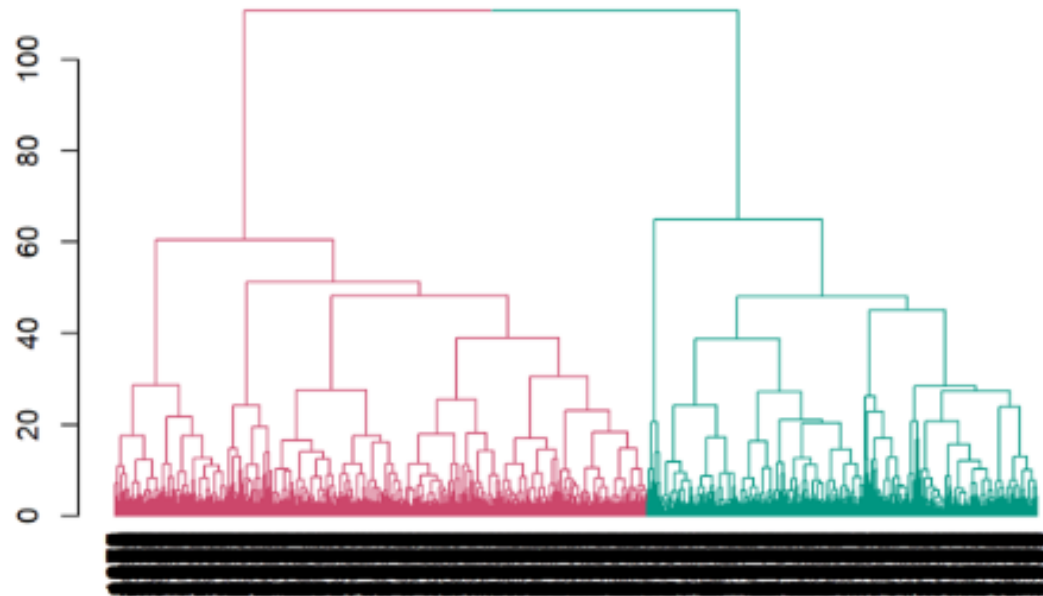
```
mean(testNorm$chlorides)
```

```
## [1] 0.03580296
```


Hierarchical Cluster Analysis

```
distances = dist(trainNorm,method = 'euclidean')
clusters = hclust(d = distances,method = 'ward.D2')

library(dendextend)
plot(color_branches(as.dendrogram(clusters), k=2, groupLabels = F))
```



```
h_segments = cutree(clusters,k=2)
table(h_segments)
```

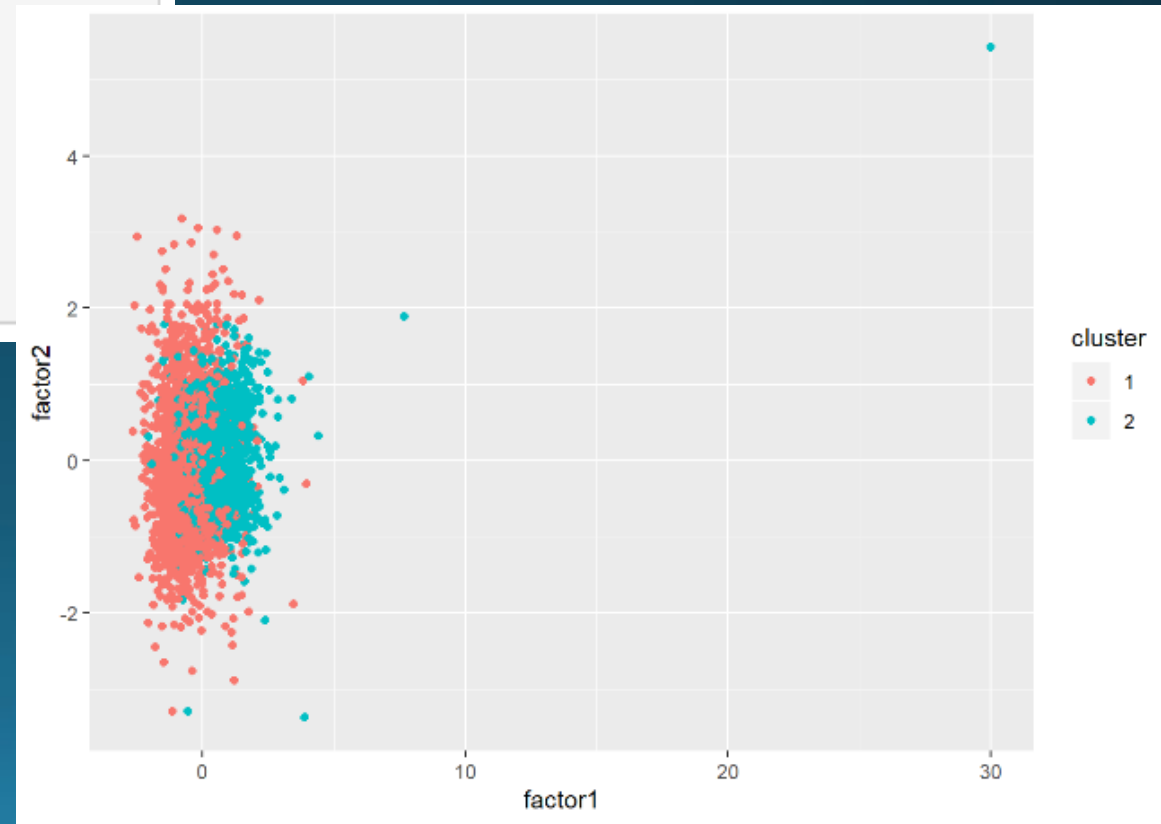
```
## h_segments
##      1      2
## 1981 1450
```

Visualize

To express the clusters on a scatterplot, we flatten the data with eleven variables into 2 dimensions by conducting a factor analysis with varimax rotation. This is done because it is not possible to visualize 11-dimensional data.

```
library(psych)
temp = data.frame(cluster = factor(clusterGroups),
                  factor1 = fa(trainNorm,nfactors =2, rotate ='varimax')$scores[,1],
                  factor2 = fa(trainNorm,nfactors =2, rotate ='varimax')$scores[,2])

library(ggplot2)
ggplot(temp,aes(x=factor1,y=factor2,col=cluster))+
  geom_point()
```



k-means Clustering

```
set.seed(1706)
km = kmeans(x=trainNorm, centers=2, iter.max=10000, nstart=100)
k_segments = km$cluster
table(k_segments)
```

```
## k_segments
##      1      2
## 1359 2072
```

Let's us compare the two-cluster solutions obtained from hierarchical cluster to k-means.

```
table(h_segments)
```

```
## h_segments
##      1      2
## 1981 1450
```

```
table(k_segments)
```

```
## k_segments
##      1      2
## 1359 2072
```

```
table(h_segments, k_segments)
```

```
##           k_segments
## h_segments      1      2
##           1  190 1791
##           2 1169  281
```

```
mean(k_segments==h_segments) # %match between results of hclust and kmeans
```

```
## [1] 0.1372778
```

The minimum of
the number of cases that match with the original labels, and
the number of cases that do not match with the original labels.

The maximum of
the number of cases that match with the original labels, and
the number of cases that do not match with the original labels.

```
min(sum(h_segments==k_segments),
    sum(h_segments!=k_segments))
```

```
## [1] 471
```

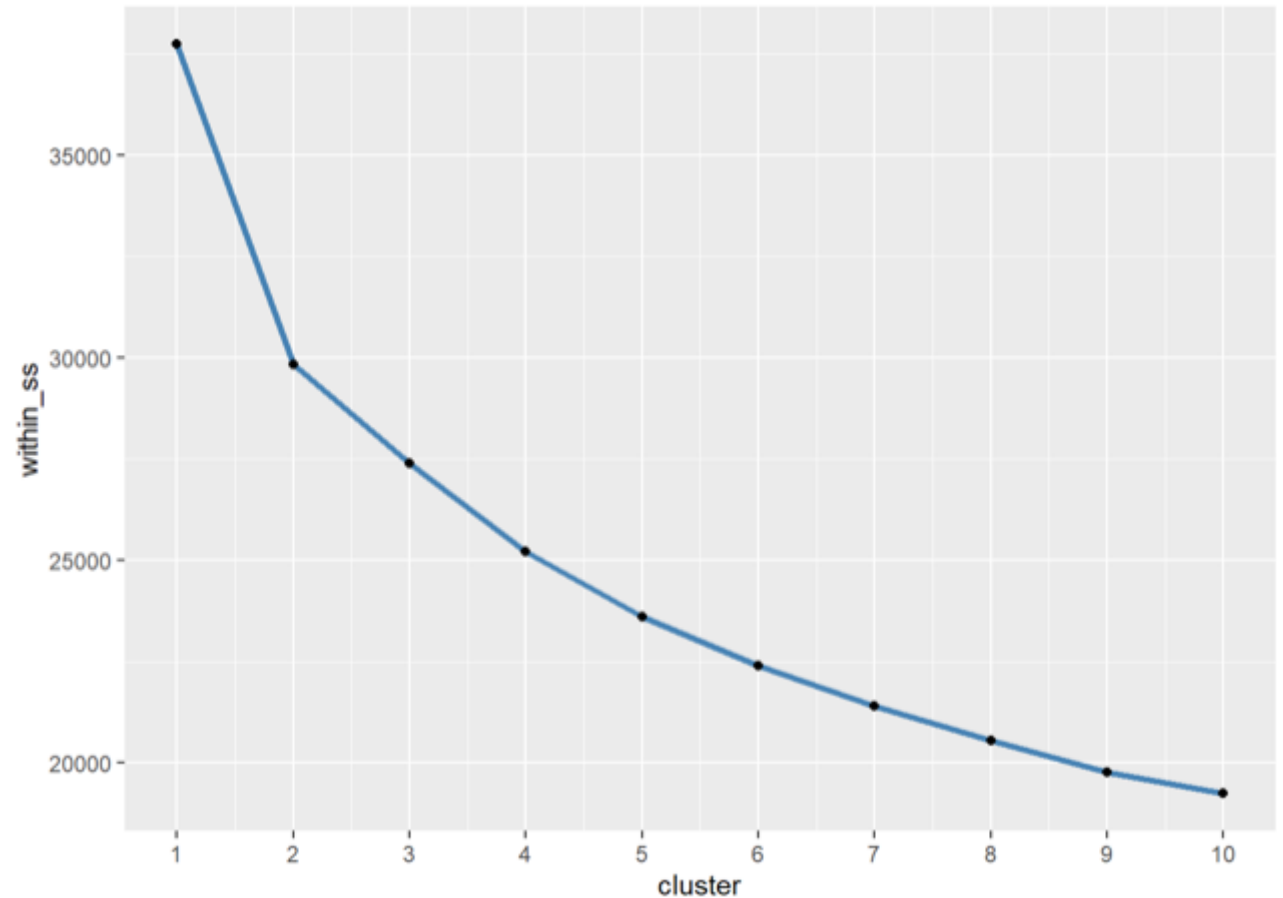
```
max(sum(h_segments==k_segments),
    sum(h_segments!=k_segments))
```

```
## [1] 2960
```

Total within sum of squares Plot

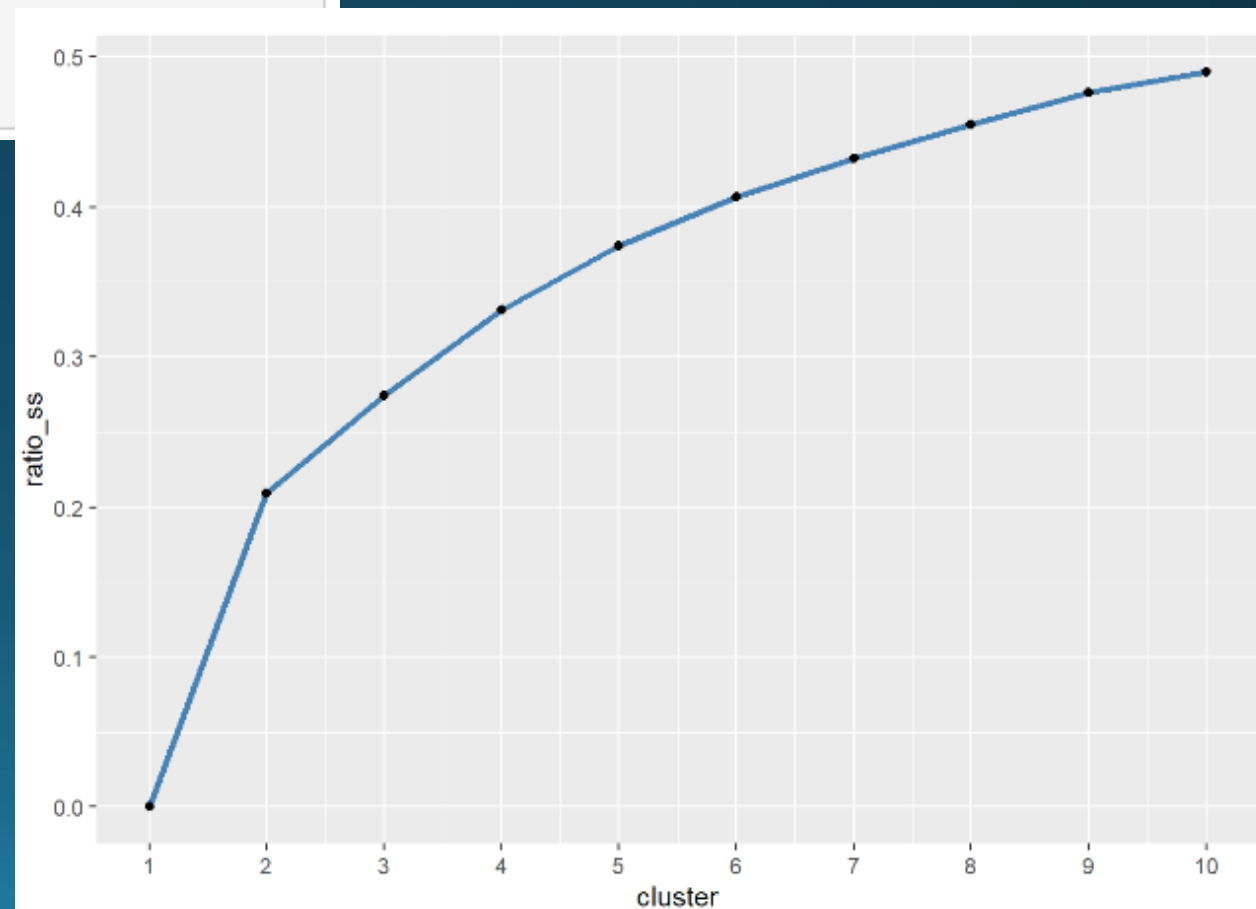
```
within_ss = sapply(1:10,FUN = function(x) kmeans(x=trainNorm, centers = x,
                                                  iter.max=1000, nstart=25)$tot.withinss)

ggplot(data=data.frame(cluster = 1:10,within_ss),aes(x=cluster,y=within_ss))+
  geom_line(col='steelblue',size=1.2)+
  geom_point()+
  scale_x_continuous(breaks=seq(1,10,1))
```

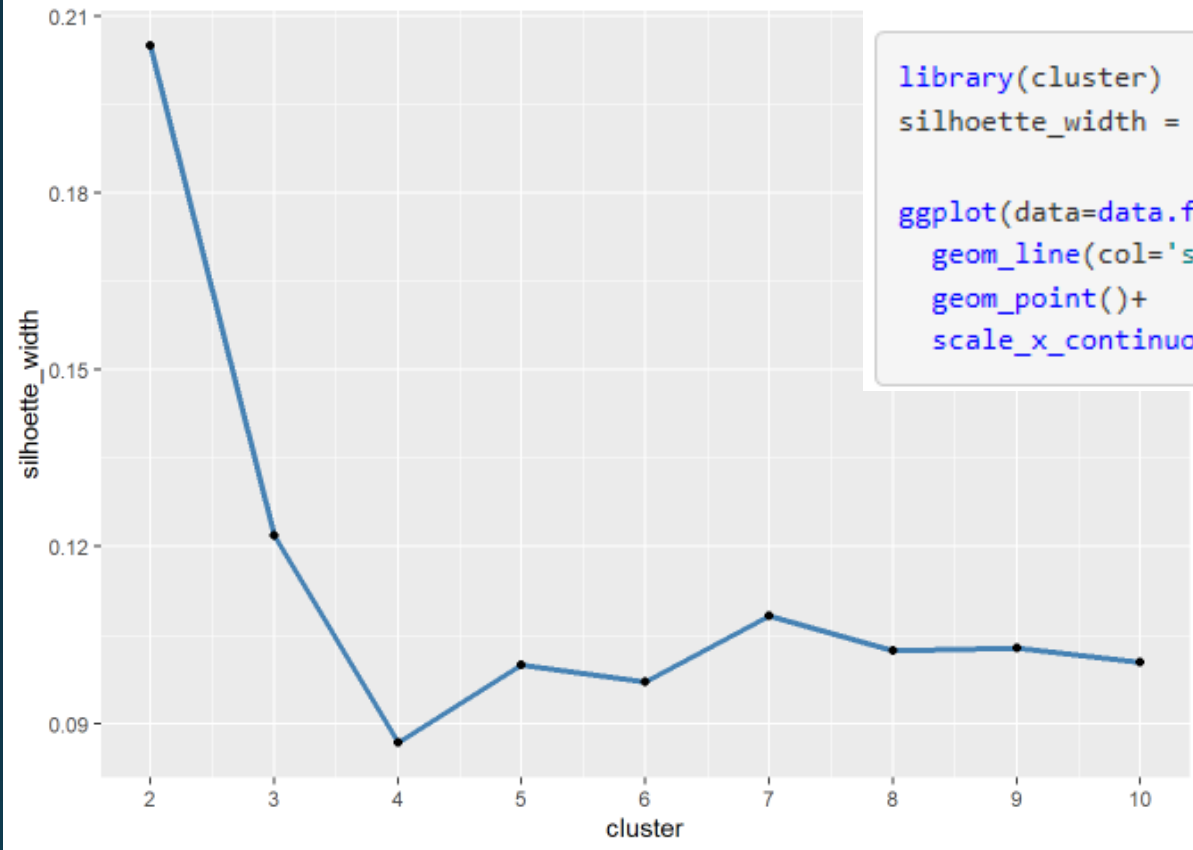


Ratio Plot

```
ratio_ss = sapply(1:10,FUN = function(x) {km=kmeans(x=trainNorm, centers = x,  
                                                    iter.max=1000, nstart=25)  
                                                    km$betweenss/km$totss})  
  
ggplot(data=data.frame(cluster = 1:10,ratio_ss),aes(x=cluster,y=ratio_ss))+  
  geom_line(col='steelblue',size=1.2)+  
  geom_point()+  
  scale_x_continuous(breaks=seq(1,10,1))
```



Silhouette Plot



```
library(cluster)
silhouette_width = sapply(2:10,FUN = function(x) pam(x=trainNorm, k=x)$silinfo$avg.width)

ggplot(data=data.frame(cluster = 2:10,silhouette_width),aes(x=cluster,y=silhouette_width))+
  geom_line(col='steelblue',size=1.2)+
  geom_point()+
  scale_x_continuous(breaks=seq(2,10,1))
```

Both, the elbow plot and silhouette plot recommend a two-cluster solution, so we will group the data into two clusters.

```
set.seed(1706)
km = kmeans(x = trainNorm,centers = 2,iter.max=10000,nstart=100)
```

Apply Clustering Solution from Train to Test

```
library(flexclust)
km_kcca = as.kcca(km,trainNorm) # flexclust uses objects of the classes kcca
clusterTrain = predict(km_kcca)
clusterTest = predict(km_kcca,newdata=testNorm)
```

Distribution of wines across clusters in Train

```
table(clusterTrain)
```

```
## clusterTrain
##      1      2
## 1359 2072
```

Distribution of wines across clusters in Test

```
table(clusterTest)
```

```
## clusterTest
##      1      2
## 588 879
```

Split Train and Test Based on Cluster Membership

```
train1 = subset(train,clusterTrain==1)
train2 = subset(train,clusterTrain==2)

test1 = subset(test,clusterTest==1)
test2 = subset(test,clusterTest==2)
```


Predict for Each Cluster then Combine

```
lm1 = lm(quality~.,train1)
lm2 = lm(quality~.,train2)
pred1 = predict(lm1,newdata=test1)
pred2 = predict(lm2,newdata=test2)
sse1 = sum((test1$quality-pred1)^2); sse1
```

```
## [1] 275.1254
```

```
sse2 = sum((test2$quality-pred2)^2); sse2
```

```
## [1] 510.5178
```

```
predOverall = c(pred1,pred2)
qualityOverall = c(test1$quality,test2$quality)
sseOverall = sum((predOverall - qualityOverall)^2); sseOverall
```

```
## [1] 785.6432
```

Compare Results

Let us compare the sse for model on the entire data to the sse for models on clusters.

```
paste('SSE for model on entire data',sseLinear)
```

```
## [1] "SSE for model on entire data 812.489032059707"
```

```
paste('SSE for model on clusters',sseOverall)
```

```
## [1] "SSE for model on clusters 785.643176502731"
```

Predict Using Tree

```
library(rpart)
library(rpart.plot)
tree = rpart(quality~.,train,minbucket=10)
predTree = predict(tree,newdata=test)
sseTree = sum((predTree - test$quality)^2); sseTree
```

```
## [1] 866.0674
```

Cluster then Predict Using Tree

```
tree1 = rpart(quality~.,train1,minbucket=10)
tree2 = rpart(quality~.,train2,minbucket=10)
pred1 = predict(tree1,newdata=test1)
pred2 = predict(tree2,newdata=test2)
sse1 = sum((test1$quality-pred1)^2); sse1
```

```
## [1] 268.4452
```

```
sse2 = sum((test2$quality-pred2)^2); sse2
```

```
## [1] 553.1455
```

```
predTreeCombine = c(pred1,pred2)
qualityOverall = c(test1$quality,test2$quality)
sseTreeCombine = sum((predTreeCombine - qualityOverall)^2); sseTreeCombine
```

```
## [1] 821.5907
```

Compare Results

Let us compare the sse for model on the entire data to the sse for models on clusters.

```
paste('SSE for model on entire data',sseTree)
```

```
## [1] "SSE for model on entire data 866.06744900816"
```

```
paste('SSE for model on clusters',sseTreeCombine)
```

```
## [1] "SSE for model on clusters 821.590740434687"
```

Simple Illustration of Cluster Analysis - R

R Illustration: Simple Illustration of Cluster Analysis

Simple Illustration of Cluster Analysis

- Cluster Analysis
- Illustration
- Create data
- Hierarchical Clustering
- K-Means Clustering
- Model-Based Clustering

Cluster Analysis

is a class of techniques used to classify objects or cases into relatively homogeneous groups called clusters. Objects in each cluster tend to be similar to each other and dissimilar to objects in the other clusters.

Illustration

What follows is an illustration of three clustering algorithms: *hierarchical*, *k-means*, and *model-based clustering*.

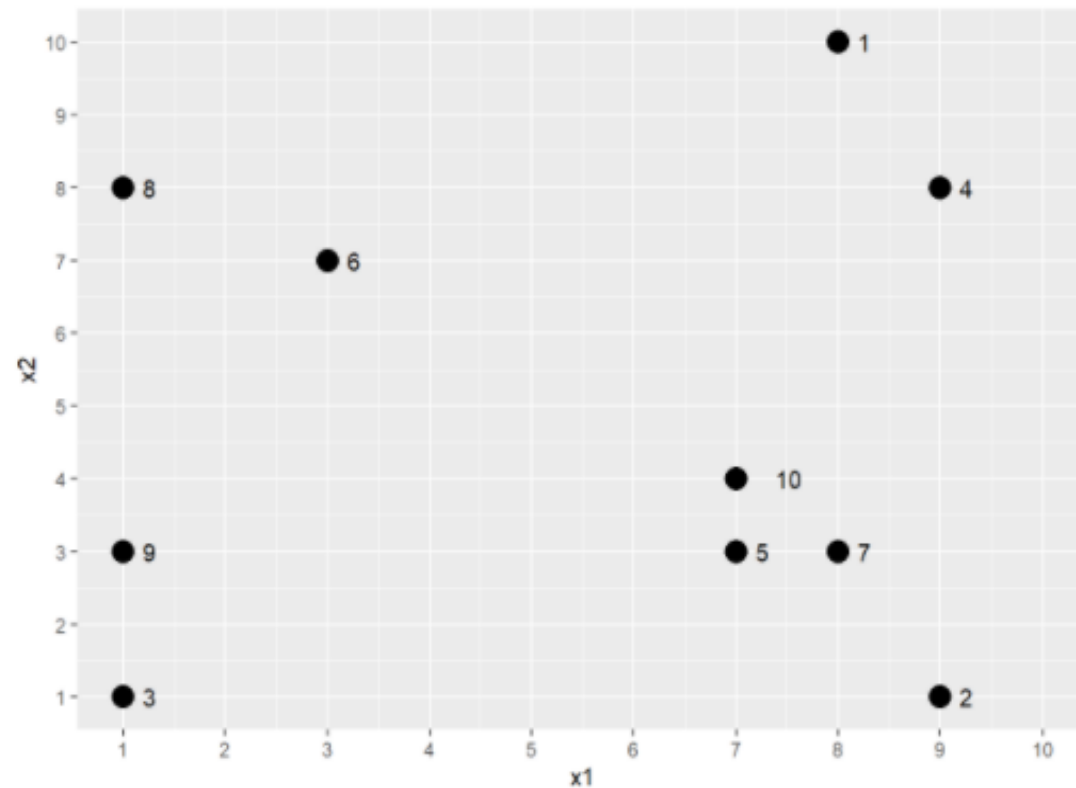
Create data

```
library(ggplot2)
set.seed(1031)
data = data.frame(x1=sample(1:10,10,replace = T),
                  x2=sample(1:10,10,replace = T))
rownames(data) = 1:10
data
```

```
##      x1 x2
## 1      8 10
## 2      9  1
## 3      1  1
## 4      9  8
## 5      7  3
## 6      3  7
## 7      8  3
## 8      1  8
## 9      1  3
## 10     7  4
```

Scatter Plot

```
ggplot(data=data,aes(x=x1,y=x2))+
  geom_point(aes(size=1.2))+
  scale_x_continuous(limits=c(1,10),breaks=1:10)+
  scale_y_continuous(limits=c(1,10),breaks=1:10)+
  guides(size=F)+
  geom_text(aes(label=rownames(data)),hjust=-1.5,vjust=0.5)
```



Hierarchical Clustering

Hierarchical Clustering

Compute distances

```
distances = round(dist(data,method = "euclidean"),2)
distances
```

##	1	2	3	4	5	6	7	8	9
## 2	9.06								
## 3	11.40	8.00							
## 4	2.24	7.00	10.63						
## 5	7.07	2.83	6.32	5.39					
## 6	5.83	8.49	6.32	6.08	5.66				
## 7	7.00	2.24	7.28	5.10	1.00	6.40			
## 8	7.28	10.63	7.00	8.00	7.81	2.24	8.60		
## 9	9.90	8.25	2.00	9.43	6.00	4.47	7.00	5.00	
## 10	6.08	3.61	6.71	4.47	1.00	5.00	1.41	7.21	6.08

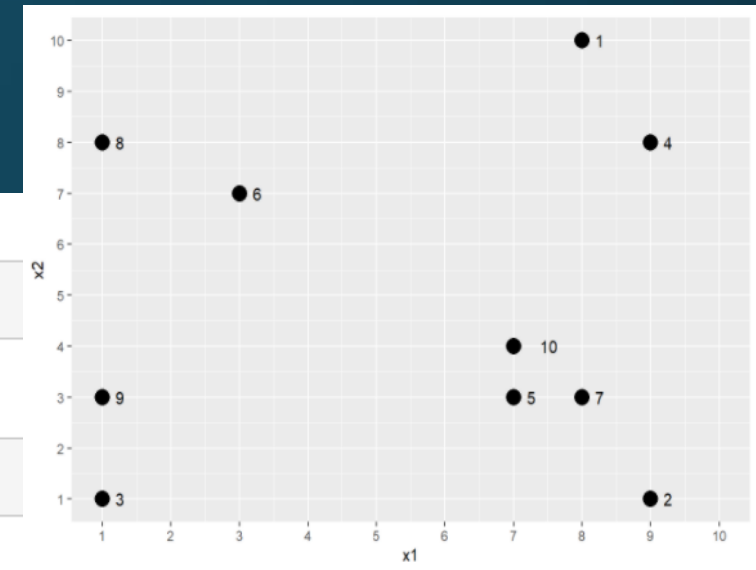
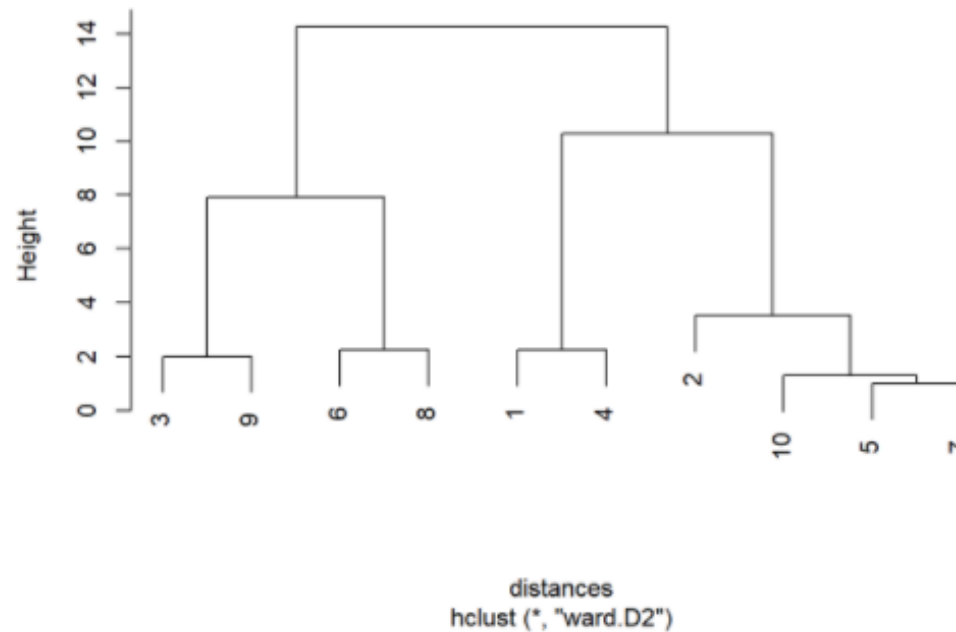
Cluster data based on distances

```
clust = hclust(distances,method = "ward.D2")
```

Dendrogram: Tree presentation of cluster results

```
plot(clust)
```

Cluster Dendrogram

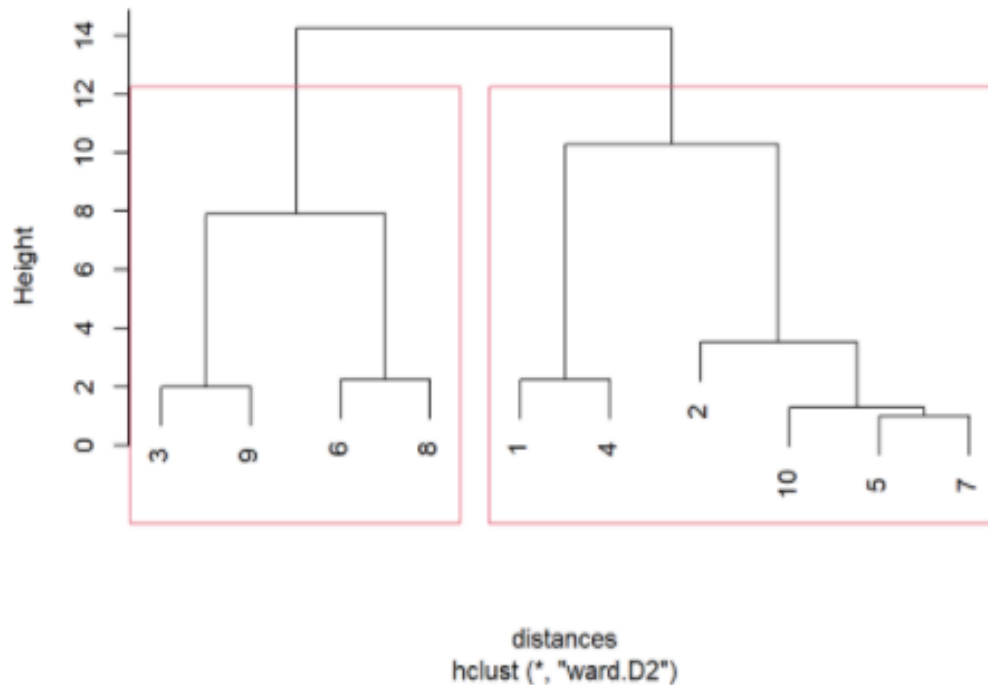


Number of Clusters

Unlike predictive modeling techniques, clusters analyses do not yield a unique solution. The number of clusters chosen is based on, (a) Distance: Clusters should be far apart relative to within cluster distances (b) Meaningfulness: Clusters should be meaningful within the domain of inquiry. Trying out a 2-cluster solution

```
plot(clust)
rect.hclust(tree=clust,k = 2)
```

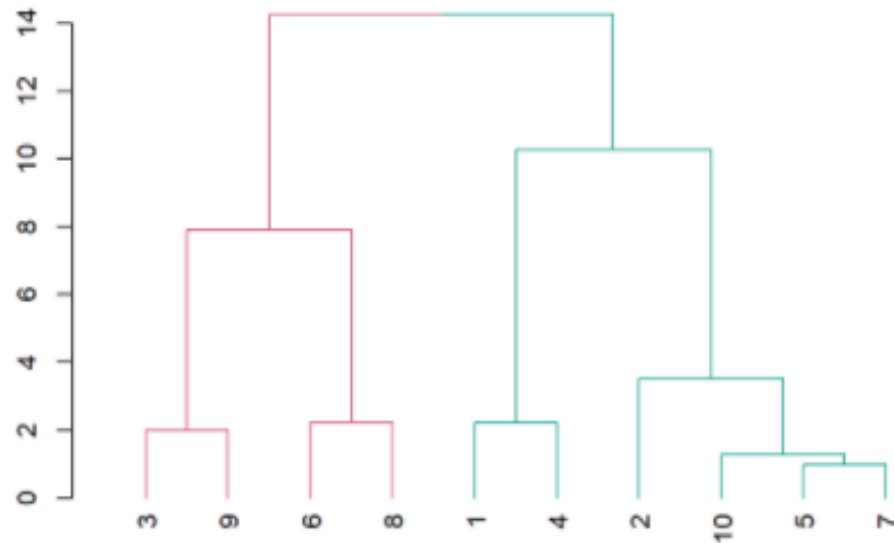
Cluster Dendrogram



2-cluster solution

Coloring the clusters in the Dendrogram. The library dendextend can be used to color clusters in the dendrogram

```
library(dendextend)
plot(color_branches(as.dendrogram(clust),k=2))
```



2-cluster solution

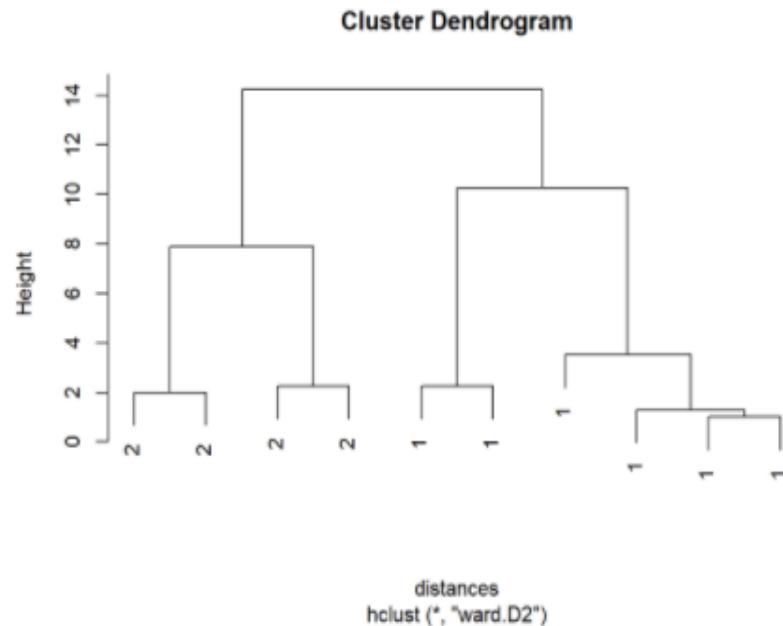
Now, let us cut the data at two clusters. All clustering algorithms are similar in that they yield a fairly boring vector of cluster memberships.

```
clusters = cutree(clust,k = 2)
clusters
```

```
## 1 2 3 4 5 6 7 8 9 10
## 1 1 2 1 1 2 1 2 2 1
```

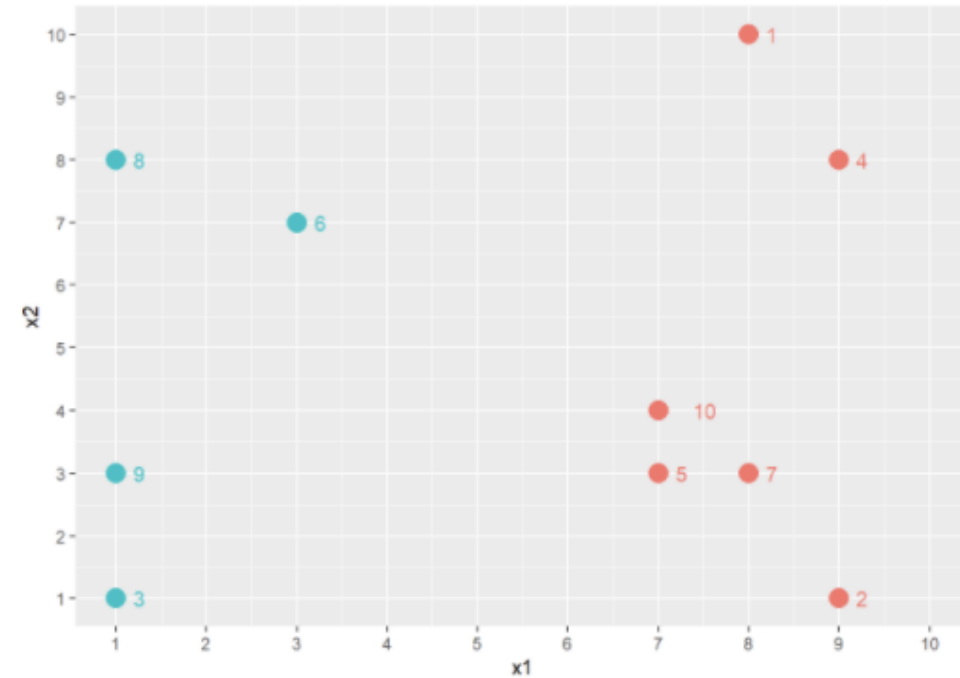
This cluster membership can be highlighted in the original dendrogram

```
plot(clust,label=clusters)
```



Let us visualize the a plot of Clusters

```
data2 = cbind(data,clusters)
ggplot(data=data2,aes(x=x1,y=x2,color=factor(clusters)))+
  geom_point(aes(size=1.2))+
  scale_x_continuous(limits=c(1,10),breaks=1:10)+
  scale_y_continuous(limits=c(1,10),breaks=1:10)+
  guides(size=F,color=F)+
  geom_text(aes(label=rownames(data2)),hjust=-1.5,vjust=0.5)
```



It is also possible to cut based on height

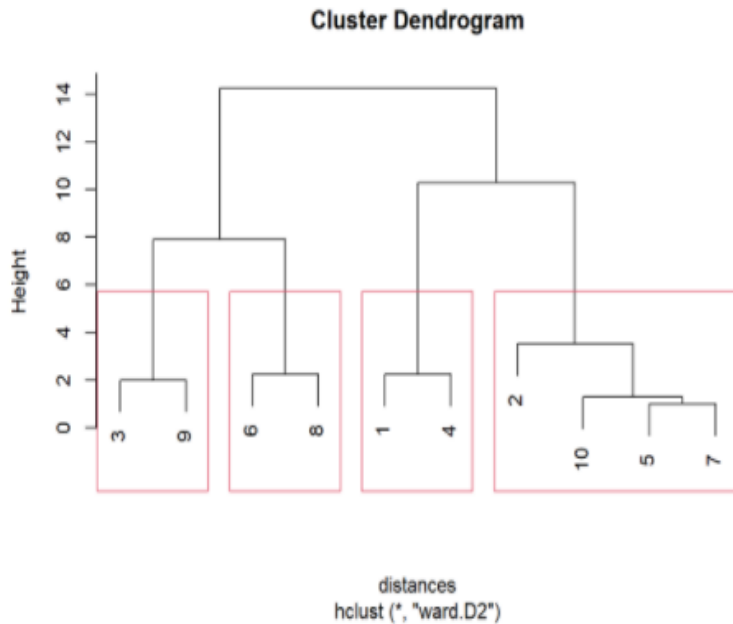
```
clust$height
```

```
## [1] 1.000000 1.287918 2.000000 2.240000 2.240000 3.516336 7.902712
## [8] 10.279743 14.240875
```

4-cluster solution

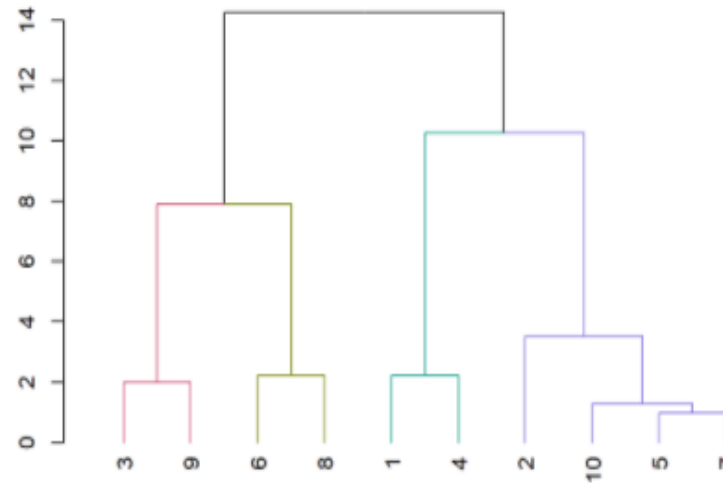
Now, let us try out a 4-cluster solution

```
plot(clust)
rect.hclust(tree=clust,k = 4)
```



Coloring the clusters in the Dendrogram

```
library(dendextend)
plot(color_branches(as.dendrogram(clust),k=4))
```



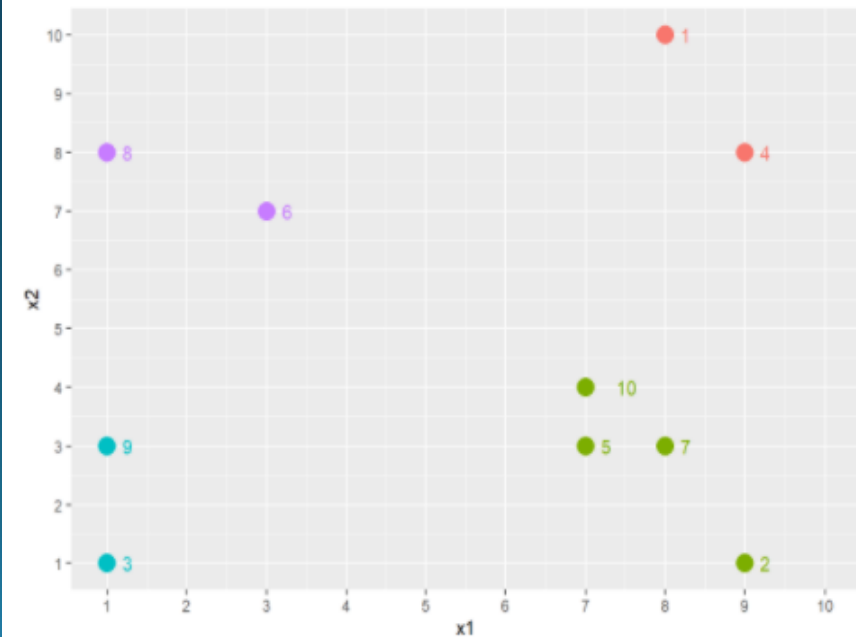
Cut the data at 4 clusters

```
clusters = cutree(clust,k = 4)
clusters
```

```
##  1  2  3  4  5  6  7  8  9 10
##  1  2  3  1  2  4  2  4  3  2
```

Let us visualize the a plot of a 4-Cluster Solution

```
data2 = cbind(data,clusters)
ggplot(data=data2,aes(x=x1,y=x2,color=factor(clusters)))+
  geom_point(aes(size=1.2))+
  scale_x_continuous(limits=c(1,10),breaks=1:10)+
  scale_y_continuous(limits=c(1,10),breaks=1:10)+
  guides(size=F,color=F)+
  geom_text(aes(label=rownames(data2)),hjust=-1.5,vjust=0.5)
```



K-Means Clustering

K-Means Clustering

Hierarchical clustering requires computing distances between every pair of observations. For large datasets, this can be a very expensive process, (for n observations have $n(n-1)/2$ distances). K-means clustering begins by arbitrarily placing centroids in the data and then iterating from that point to the final solution. This disorganized approach to clustering produces similar quality of clusters to hierarchical clustering but much faster.

It is worth noting that K-Means is sensitive to starting point, so the seed can influence final solution. Also, for k-means, one has to provide the number of centers.

```
set.seed(100)
km = kmeans(data, centers = 4, iter.max = 1000)
```

A good cluster solution maximizes between cluster distance related to within cluster distances. The kmeans object generates this ratio which is useful in evaluating the suitability of cluster solution.

km

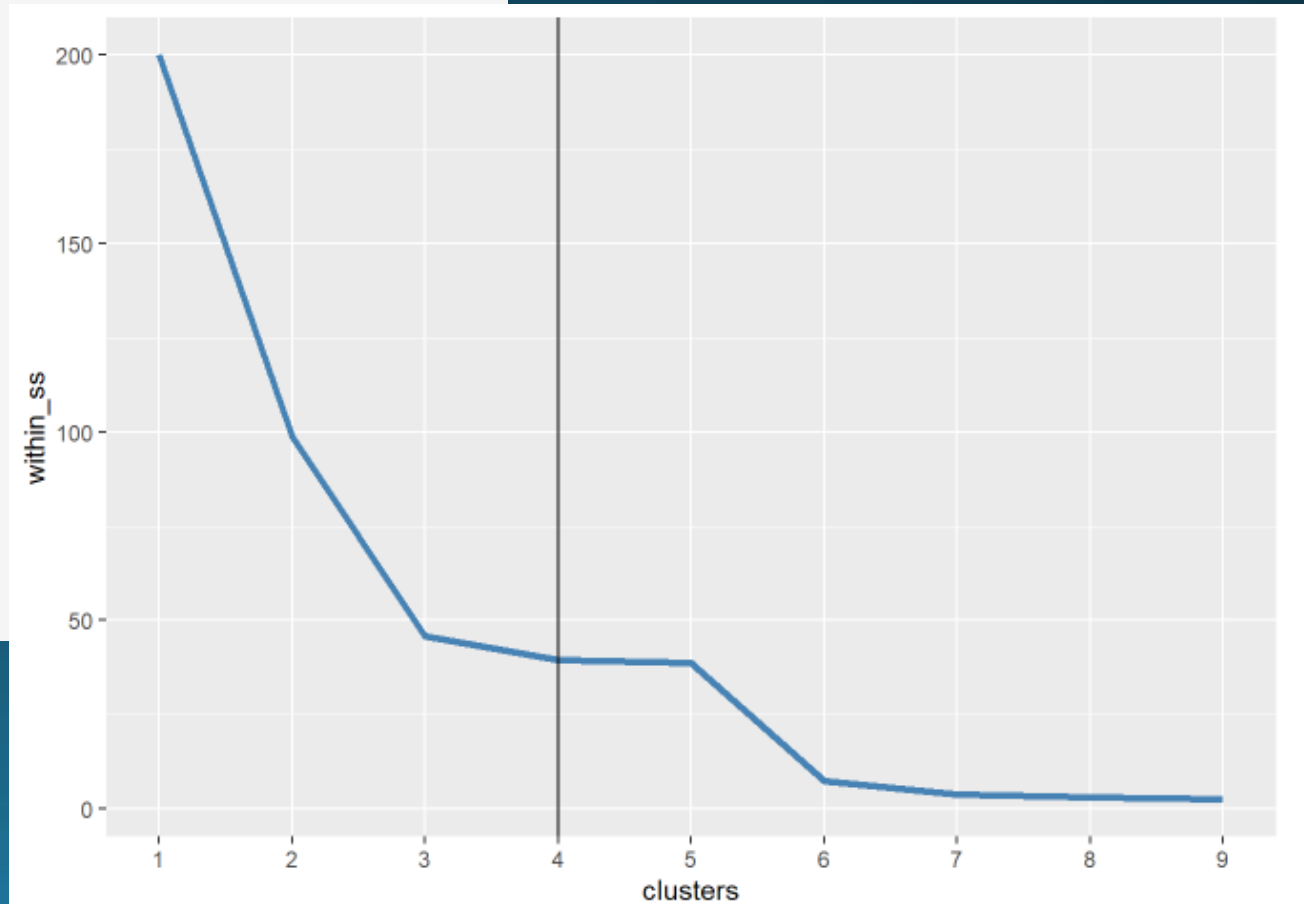
```
## K-means clustering with 4 clusters of sizes 2, 4, 2, 2
##
## Cluster means:
##      x1  x2
## 1 8.50 9.00
## 2 7.75 2.75
## 3 2.00 7.50
## 4 1.00 2.00
##
## Clustering vector:
##  1  2  3  4  5  6  7  8  9 10
##  1  2  4  1  2  3  2  3  4  2
##
## Within cluster sum of squares by cluster:
## [1] 2.5 7.5 2.5 2.0
## (between_SS / total_SS =  92.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

In fact, this ratio can be used to generate a scree plot. A sudden change in ratio marks the optimal cluster scheme.

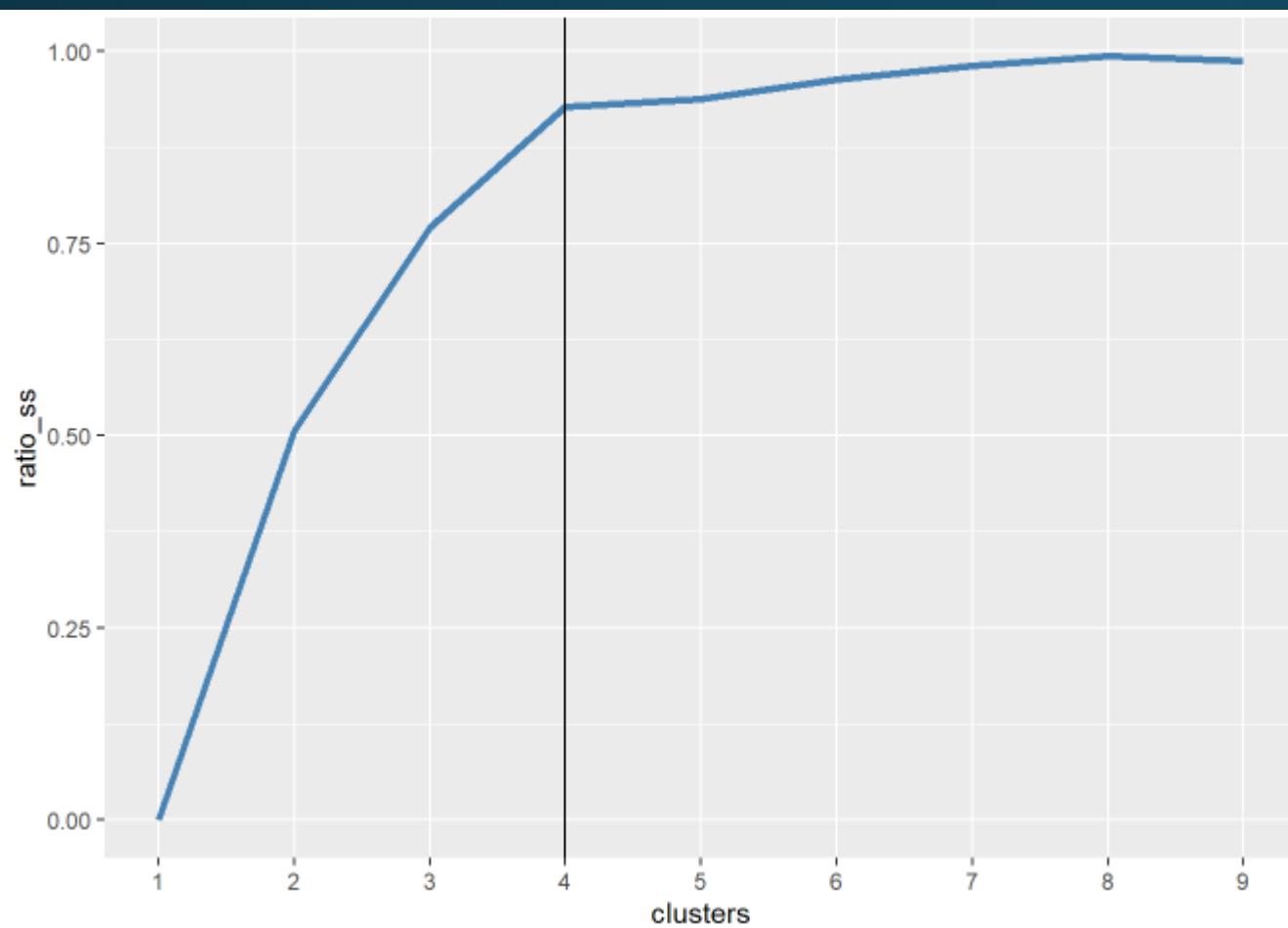
```
within_ss = sapply(X = 1:9,
  FUN = function(x) kmeans(data,centers = x,iter.max = 100)$tot.withinss)

ratio_ss = sapply(X = 1:9,
  FUN = function(x) {
    km = kmeans(data,centers = x,iter.max = 100)
    ratio = km$betweenss/km$totss
    return(ratio)
  })
dat = data.frame(clusters=1:9,within_ss, ratio_ss)

# Elbow in Within_ss
library(ggplot2)
ggplot(dat,aes(x=clusters,y=within_ss))+
  geom_line(color='steelblue',size=1.4)+
  scale_x_continuous(breaks=1:9,minor_breaks = 1:9)+
  geom_vline(xintercept=4)
```



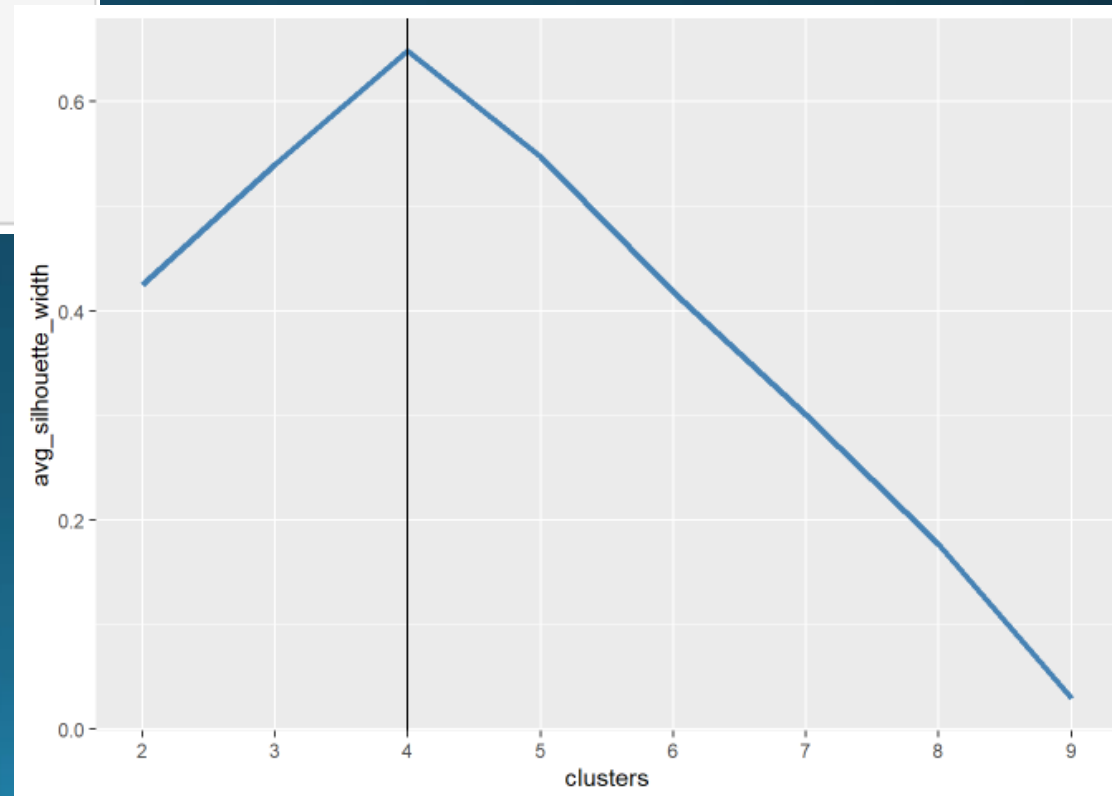
```
# Elbow in Ratio  
ggplot(dat,aes(x=clusters,y=ratio_ss))+  
  geom_line(color='steelblue',size=1.4)+  
  scale_x_continuous(breaks=1:9,minor_breaks = 1:9)+  
  geom_vline(xintercept=4)
```



Another method used for determining the optimal number of cluster is the Silhouette method. Here we use the pam function to determine the average silhouette width. Note, pam runs an algorithm like k-means but not exactly the same as k-means. Since we are only using it to determine ideal number of cluster, it should be okay.

```
library(cluster)
sil = sapply(2:9,FUN=function(x){
  pam(data,k=x)$silinfo$avg.width
})

ggplot(data.frame(clusters=2:9,avg_silhouette_width = sil),aes(x=clusters,y=avg_silhouette_width))+
  geom_line(color='steelblue',size=1.4)+
  scale_x_continuous(breaks=1:9,minor_breaks = 1:9)+
  geom_vline(xintercept=4)
```



The k-means object has some nice features including a summary of clusters.

```
km$centers
```

```
##      x1    x2
## 1 8.50 9.00
## 2 7.75 2.75
## 3 2.00 7.50
## 4 1.00 2.00
```

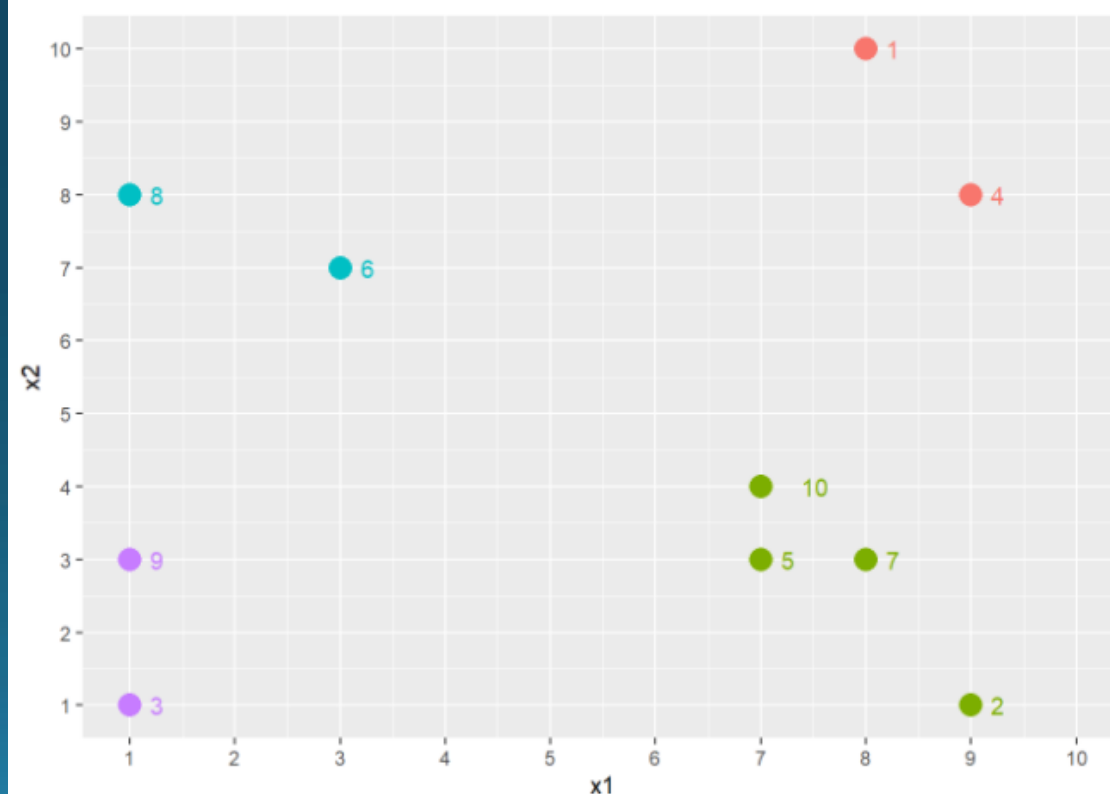
Just like `hclust()`, `kmeans()` also yields a vector of cluster assignments

```
km$cluster
```

```
##  1  2  3  4  5  6  7  8  9 10
##  1  2  4  1  2  3  2  3  4  2
```

Finally, we can visualize the solution

```
data3 = cbind(data2, kmc=km$cluster)
ggplot(data=data3, aes(x=x1, y=x2, color=factor(kmc)))+
  geom_point(aes(size=1.2))+
  scale_x_continuous(limits=c(1,10), breaks=1:10)+
  scale_y_continuous(limits=c(1,10), breaks=1:10)+
  guides(size=F, color=F)+
  geom_text(aes(label=rownames(data3)), hjust=-1.5, vjust=0.5)
```



Model-Based Clustering

Model-Based Clustering

Key idea of model-based clustering is that Observations come from groups with different statistical distributions (such as different means and variances). The algorithms try to find the best set of such underlying distributions to explain the observed data.

```
library(mclust)
clus = Mclust(data)
summary(clus)
```

```
## -----
## Gaussian finite mixture model fitted by EM algorithm
## -----
##
## Mclust EII (spherical, equal volume) model with 9 components:
##
## log-likelihood  n df      BIC      ICL
##      -13.12953 10 27 -88.42886 -88.42886
##
## Clustering table:
## 1 2 3 4 5 6 7 8 9
## 1 1 1 1 2 1 1 1 1
```

Model-based clustering infers ideal number of clusters by comparing a variety of different mixture shapes.

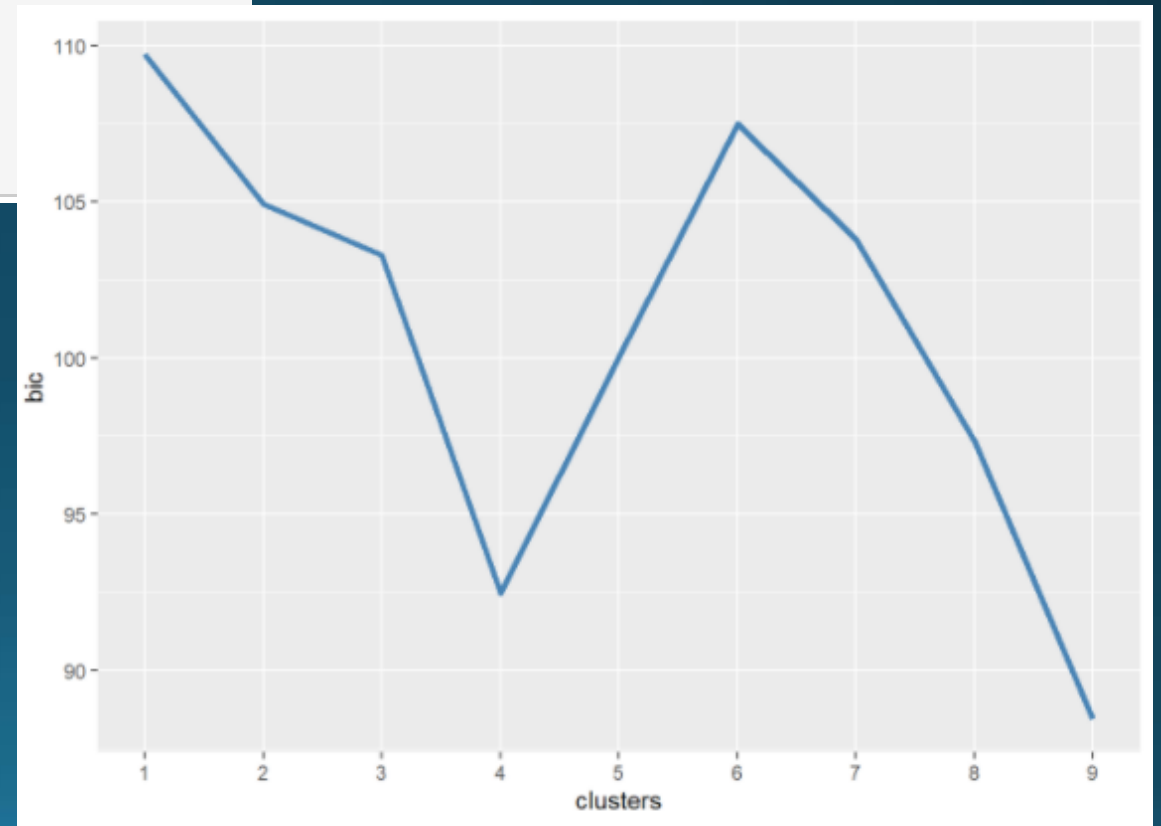
But, one could also specify the number of clusters. Quality of a cluster solution is inferred from log-likelihood and BIC

```
clus4 = Mclust(data,G = 4)
summary(clus4)
```

```
## -----
## Gaussian finite mixture model fitted by EM algorithm
## -----
##
## Mclust EEV (ellipsoidal, equal volume and shape) model with 4 components:
##
## log-likelihood  n df      BIC      ICL
##      -26.66904 10 17 -92.48203 -92.48204
##
## Clustering table:
## 1 2 3 4
## 2 4 2 2
```

One could also compare BIC for a set of clustering solutions as above. Note, `Mclust()` generates negative of bic, so the sign has been flipped.

```
bic = sapply(1:9,FUN=function(x){
  -Mclust(data,G=x)$bic
})
dat = data.frame(clusters=1:9,bic)
ggplot(dat,aes(x=clusters,y=bic))+
  geom_line(color='steelblue',size=1.4)+
  scale_x_continuous(breaks=1:9,minor_breaks = 1:9)
```

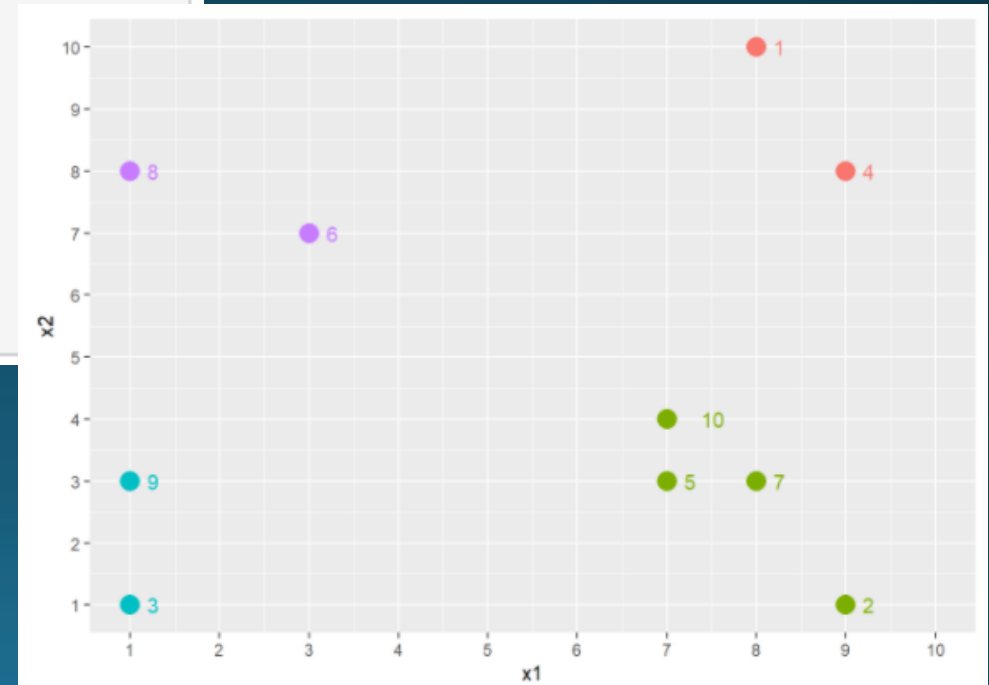


Let us extract cluster memberships for a four cluster solution

```
mcluster = Mclust(data,G=4)
```

Finally, we can visualize the solution

```
data4 = cbind(data3,mclust=mcluster$classification)
ggplot(data=data4,aes(x=x1,y=x2,color=factor(mclust)))+
  geom_point(aes(size=1.2))+
  scale_x_continuous(limits=c(1,10),breaks=1:10)+
  scale_y_continuous(limits=c(1,10),breaks=1:10)+
  guides(size=F,color=F)+
  geom_text(aes(label=rownames(data3)),hjust=-1.5,vjust=0.5)
```



Cluster Analysis: An Iris Story - R

R Illustration: Cluster Analysis: An Iris Story

iris
Visualize
Hierarchical Cluster Analysis
k-means
Model-based
Results

Cluster Analysis: An Iris Story

iris

iris is a simple dataset containing four flower characteristics and the names of the flowers. This dataset has been further simplified to illustrate three popular cluster analysis techniques. Specifically, only Sepal.Length, Sepal.Width for two flowers are being used.

```
dat = iris[1:100,c(1,2,5)]  
head(dat)
```

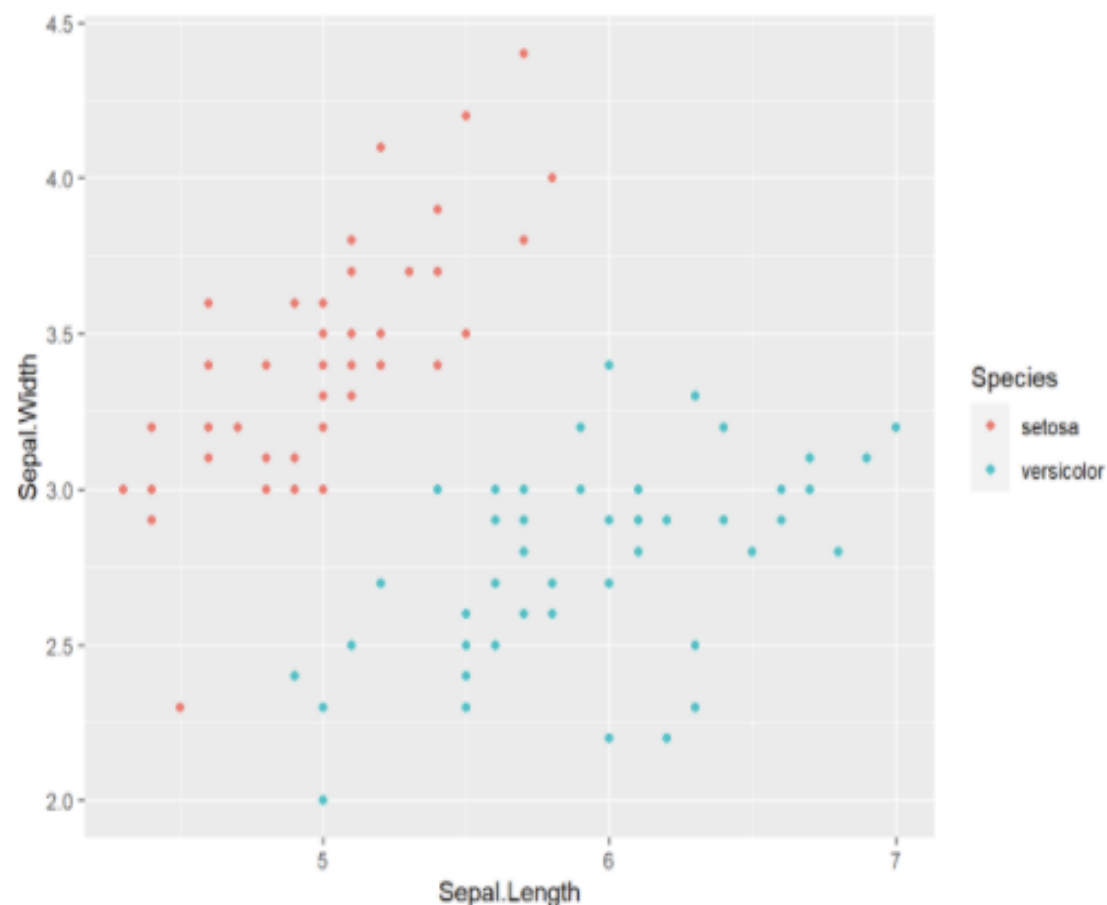
	Sepal.Length <dbl>	Sepal.Width <dbl>	Species <fct>
1	5.1	3.5	setosa
2	4.9	3.0	setosa
3	4.7	3.2	setosa
4	4.6	3.1	setosa
5	5.0	3.6	setosa
6	5.4	3.9	setosa

6 rows

Visualize

Let's visualize the two Species in terms of Sepal.Length and Sepal.Width

```
library(ggplot2)  
ggplot(dat, aes(x=Sepal.Length, y=Sepal.Width, color=Species))+  
  geom_point()
```

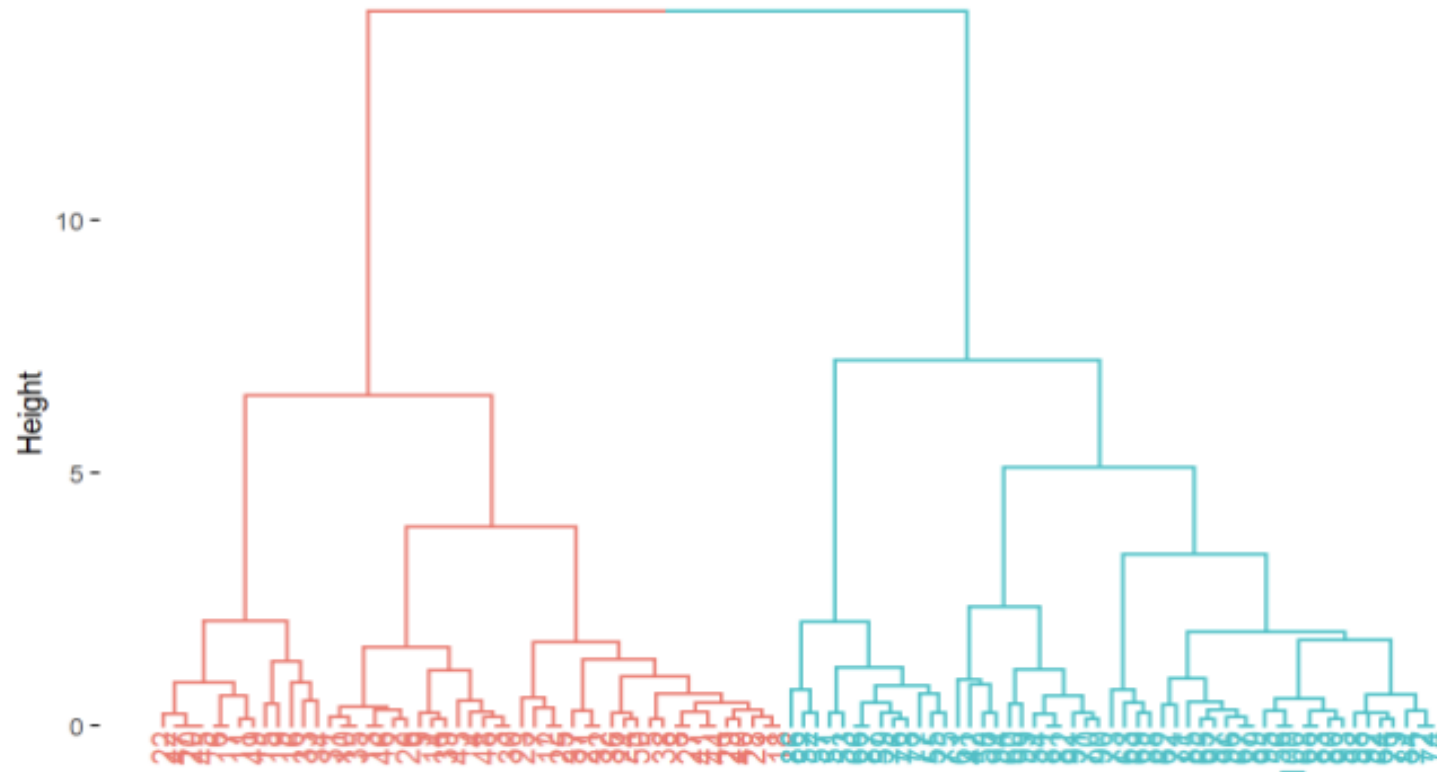


Hierarchical Cluster Analysis

We are going to use Sepal.Width and Sepal.Length to cluster the 100 flowers. To conduct a hierarchical cluster analysis, we standardize the two columns, and then use Ward's method to cluster the flowers based on their euclidean distance. The dendrogram illustrates a nice separation into two groups.

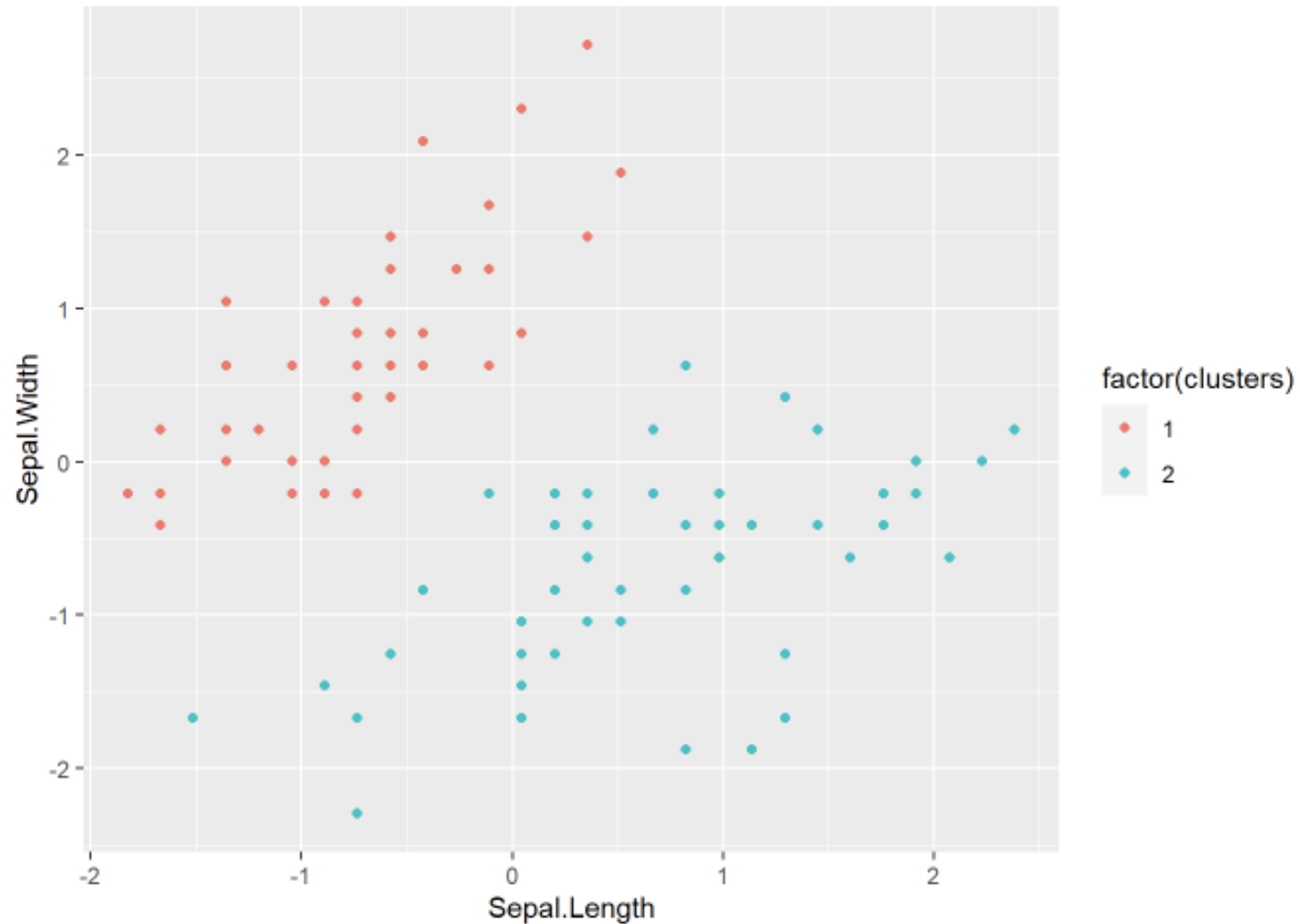
```
dat[,1:2] = scale(dat[,1:2])  
clusters = hclust(d = dist(dat[,1:2]),method = 'euclidean'),method = 'ward.D2')  
  
library(factoextra)  
fviz_dend(clusters,k = 2)
```

Cluster Dendrogram



Next, lets extract the cluster assignments and plot them. Hopefully, our clusters clearly distinguish the two flower species.

```
h_clusters = cutree(clusters,k = 2)
ggplot(data=cbind(dat,clusters = h_clusters), aes(x=Sepal.Length,y=Sepal.Width,color=factor(clusters)))+
  geom_point()
```

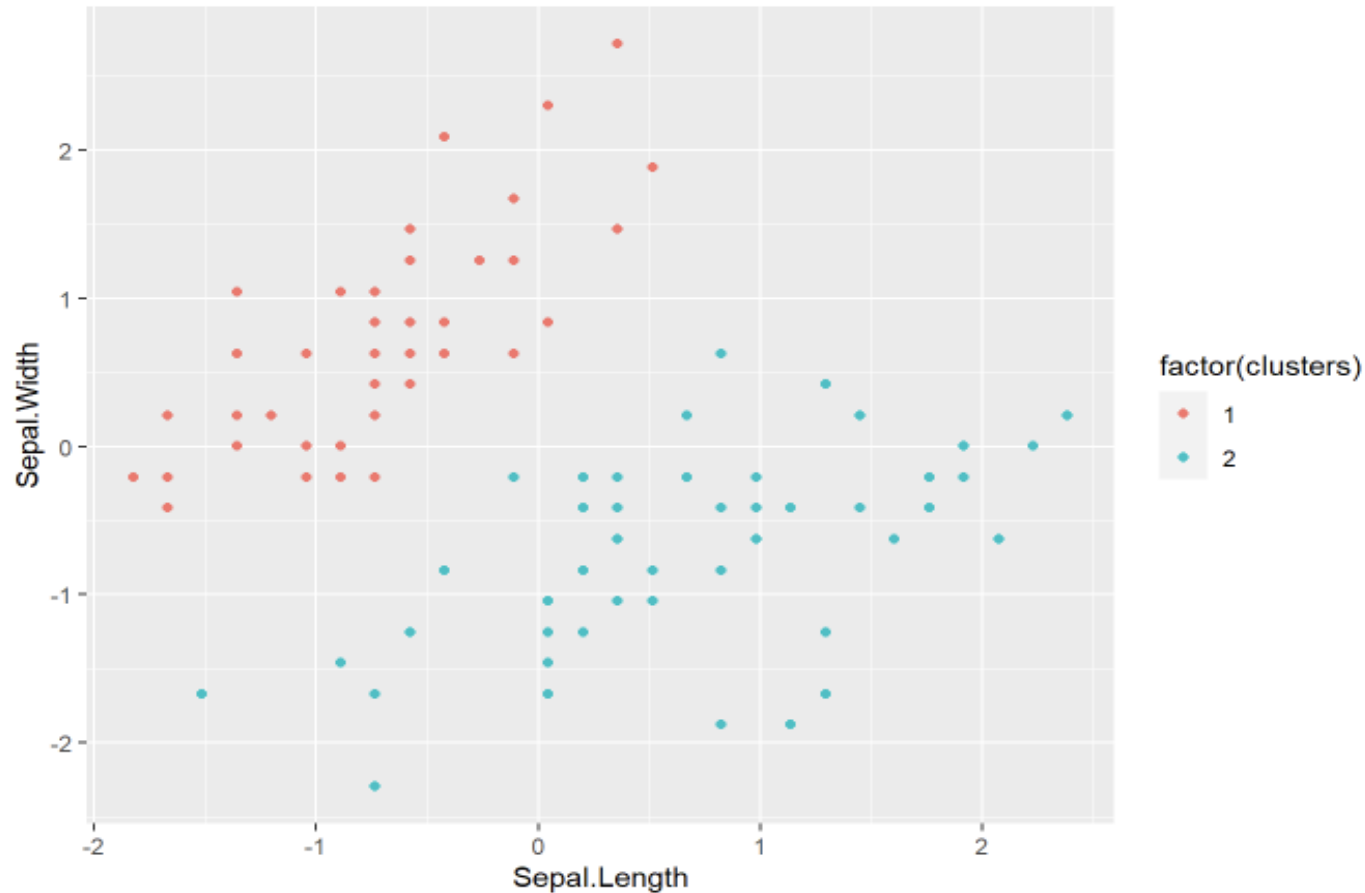


k-means

Let's try k-means algorithm, specifically asking it to give us two clusters. Again, we are hoping the cluster assignments clearly distinguish the two species of flowers.

```
set.seed(617)
km = kmeans(x = dat[,1:2],centers = 2,iter.max = 1000,nstart = 25)
k_cluster = km$cluster

ggplot(data=cbind(dat,clusters = k_cluster), aes(x=Sepal.Length,y=Sepal.Width,color=factor(clusters)))+
  geom_point()
```

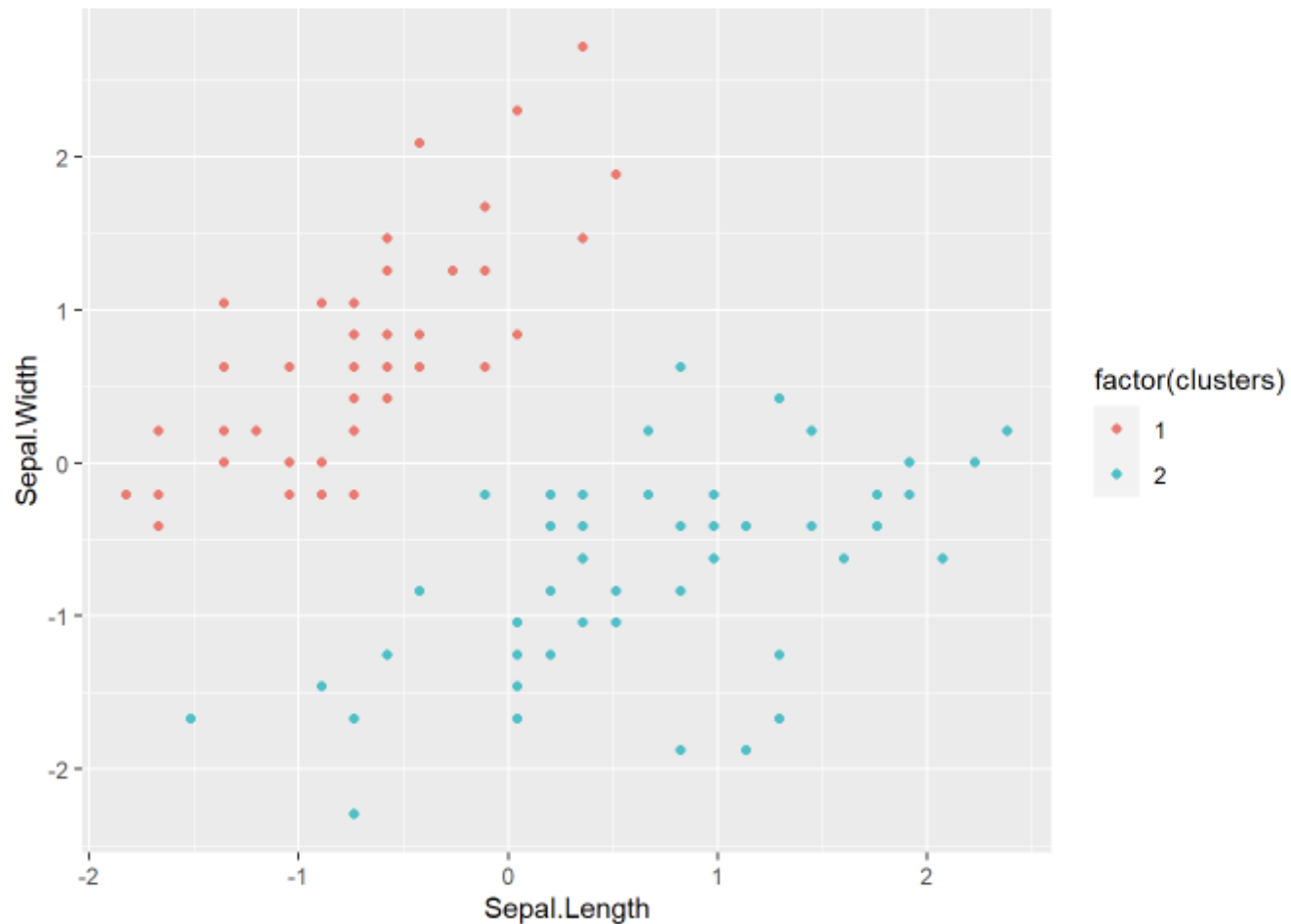


Model-based

Lastly, let us try a mixture model to see if we can generate clusters that map well to the original flower species.

```
library(mclust)
m_cluster = Mclust(dat[,1:2])$classification

ggplot(data=cbind(dat,clusters = m_cluster), aes(x=Sepal.Length,y=Sepal.Width,color=factor(clusters)))+
  geom_point()
```



Results

The three algorithms yielded identical results. All three made one error in classifying the two flowers.

Conclusion

- In the basic and advanced clustering modules, we
 - discussed the concept of clustering
 - examined application of clustering for market segmentation
 - reviewed process of clustering
 - discussed three clustering algorithms: hierarchical clustering, k-means and model-based clustering
 - examined application of clustering as an input to predictive modelling
 - R Illustrations:
 - Clustering for Segmentation
 - Cluster then Predict
 - Simple Illustration of Cluster Analysis
 - Cluster Analysis: An Iris Story