Pstat 131 Homework 3

Kendall Brown 8564403

Winter 2018

```
library(tidyverse)
library(e1071)
library(cluster)
library(NbClust)
library(bmp)
library(imager)
library(tree)
library(randomForest)
library(gbm)
library(ROCR)
```

1a. In a sample of size n, the probability that the jth observation is not in the bootstrap is the product of the probability that each observation in the bootstrap is not the jth element of the sample. If each element has probability $\frac{1}{n}$ of being in the bootstrap, then each element has probability $(1 - \frac{1}{n})$ of not being in the bootstap. Thus the probability that the jth observation is not in the bootstrap is
$(1 - \frac{1}{n})^n$

1b. Probability when n=1000

```
(1-1/1000)^1000
```

```
## [1] 0.3676954
```

1c. Resampling numbers from 1 to 1000, then calculating observed probability that an element is not part of the bootstrap sample.

```
set.seed(300)
n.1c=c(1:1000)
sample.1c=sample(n.1c,1000,replace = T)
sample.1c.unique=unique(sample.1c)
p.n.1c=length(sample.1c.unique)/1000
1-p.n.1c
```

```
## [1] 0.366
```

Setting the seed to be 300, I observe that the probability of an element not being in the bootstrap is .366.

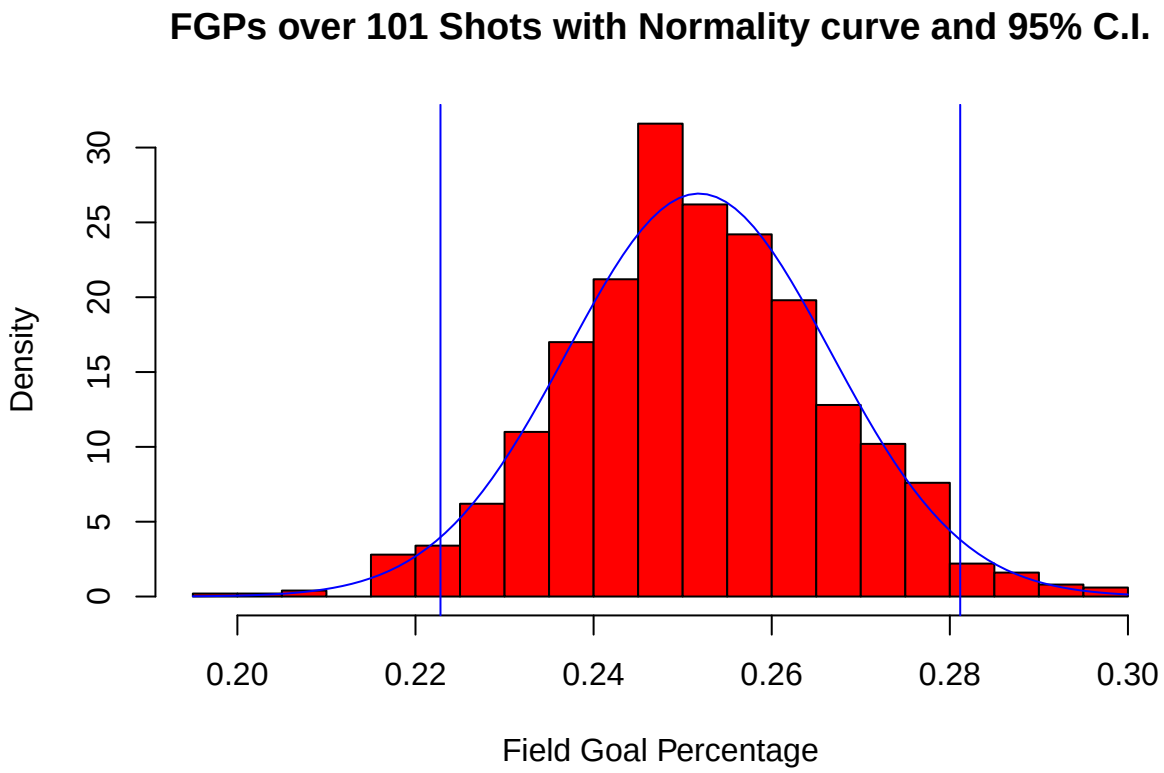
1d. Basketball shot field goal precentage.

```
n.1d=c(rep(1:50),rep(0:51))
fgp.1d=rep(0,1000)
for(i in 1:1000){
  sample.1d=sample(n.1d,101,replace=T)
  fgp.1d[i]=mean(sample.1d)/101
}
```

Histogram of Field Goal Percentages from a bootstrap sample of 101 shots. Overlayed with normality curve and 95% confidence interval.

```
fgp.ci=quantile(fgp.1d,probs=c(.025,.975))
hist(fgp.1d,breaks=15,col="red",
     main="FGPs over 101 Shots with Normality curve and 95% C.I.",
```

```
     xlab="Field Goal Percentage",probability = T)
curve(dnorm(x,mean(fgp.1d),sd(fgp.1d)),add=T,col="blue")
abline(v=fgp.ci,col="blue")
```



From the histogram and overlayed gausian curve, we see a strong indication of normality with little skew.

95% confidence interval for Rober Covington's FGP

```
fgp.ci#95% confidence Interval
```

```
##      2.5%     97.5%
## 0.2228066 0.2811661
```

```
fgp.ci.9999=quantile(fgp.1d,probs=c(.9999))#Upper 99.99% confidence bound of FGPs
fgp.ci.9999
```

```
##    99.99%
## 0.2969709
```

From this 95% confidence interval I must assume that Robert Covington's FGP will fall by the end of the competitive season. As of now he is shooting well beyond the realm of expectation with an average of .495. Such a value dwarfs the 99.99% confidence bound of .299 and should be viewed as a statistical outlier.

```
seeds = read.table('http://archive.ics.uci.edu/ml/machine-learning-databases/00236/seeds_dataset.txt')
names(seeds) = c('area', 'perimeter', 'compactness', 'length',
'width', 'asymmetry', 'groovelength', 'type')
seeds.label = factor(seeds$type)
seeds.orig = seeds[,-ncol(seeds)]
seeds = scale(seeds.orig)
```

```
seeds = as.data.frame(seeds)
```

2a. PCA on Seeds Data set

```
seeds.pca=prcomp(seeds)
summary(seeds.pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6
## Standard deviation     2.2430 1.0943 0.82341 0.26147 0.13680 0.07302
## Proportion of Variance 0.7187 0.1711 0.09686 0.00977 0.00267 0.00076
## Cumulative Proportion  0.7187 0.8898 0.98668 0.99645 0.99912 0.99988
##                           PC7
## Standard deviation     0.02850
## Proportion of Variance 0.00012
## Cumulative Proportion  1.00000
```

From the summary report we see that the firts three predictors account for 98.668% of the observed variance in the seeds data set.

Correlation matrix of seeds dataset rounded to 3 decimal places.

```
round(cor(seeds),3)
```

```
##                area perimeter compactness length  width asymmetry
## area          1.000     0.994       0.608  0.950  0.971    -0.230
## perimeter     0.994     1.000       0.529  0.972  0.945    -0.217
## compactness   0.608     0.529       1.000  0.368  0.762    -0.331
## length        0.950     0.972       0.368  1.000  0.860    -0.172
## width         0.971     0.945       0.762  0.860  1.000    -0.258
## asymmetry    -0.230    -0.217      -0.331 -0.172 -0.258     1.000
## groovelength  0.864     0.891       0.227  0.933  0.749    -0.011
##              groovelength
## area                0.864
## perimeter           0.891
## compactness         0.227
## length              0.933
## width               0.749
## asymmetry          -0.011
## groovelength        1.000
```

Here we confirm that a number of predictor variables are highly correlated with each other and that only a few of the variables provide meaningful insight.

2b. Finding out the best number of clusters for k-means clustering.

```
seeds.pc2=seeds.pca$x[,1:2]
seeds.nbclust=NbClust(seeds.pc2,method="kmeans",index="trcovw")
seeds.nbclust
```

```
## $All.index
##          2          3          4          5          6          7
## 26335.3992 19120.0357 10668.3865  4934.7128  6624.5561  3675.8460
##          8          9         10         11         12         13
##  2615.8625  1932.8255  1748.8097  1101.9779  1135.9410  1113.8106
##         14         15
##   988.1539   718.0355
##
```

```
## $Best.nc
## Number_clusters     Value_Index
##           4.000        8451.649
##
## $Best.partition
##   [1] 1 1 1 1 1 1 1 1 3 3 3 1 1 1 1 1 1 1 1 4 1 1 1 1 1 1 1 1 1 1 3 1 1 1
##  [36] 3 3 3 1 1 1 1 1 3 1 1 1 1 1 1 1 3 3 1 1 1 1 1 1 1 4 1 1 1 1 1 1 1 4
##  [71] 2 3 3 2 3 3 3 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 3 2 2 2 2
## [106] 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 1 2 2 2 2 3 2 2 3 3 3 1 3 3 3 3
## [141] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 1 4 4 4 4 4 4 4 4 4
## [176] 4 4 4 4 1 4 4 4 4 4 4 4 4 4 4 4 4 4 1 4 4 1 4 1 4 1 4 1 4 4 4 1 4 4 4 4
```

```
best.km=seeds.nbclust$Best.nc[1]
```

From the NbClust function we see that the best number of clusters for this data set is 4 and the index vlaue is 8451.649.

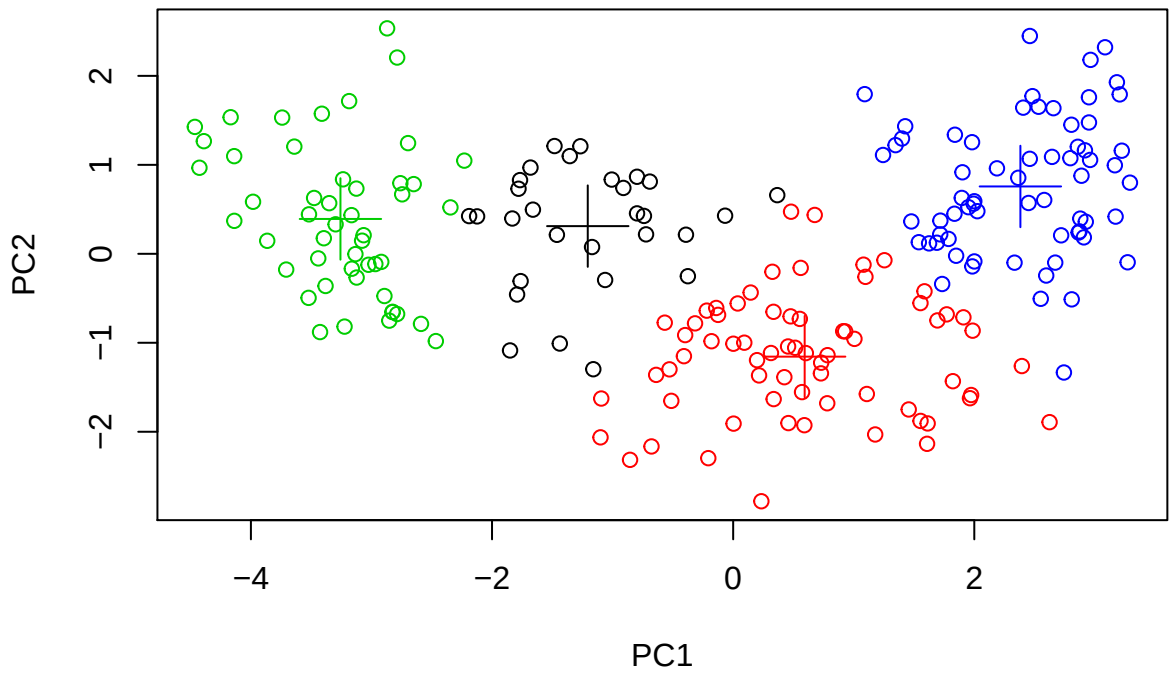
Trcovw is the sum of the diagonal of a pooled covariance matrix. The optimal solutution is then computed by finding the maximum difference between factor levels.

Plotting the clustered data, clusters and their centers distinguised by color.

```
set.seed(10)
seed.km=kmeans(seeds.pc2,best.km)
seed.km
```

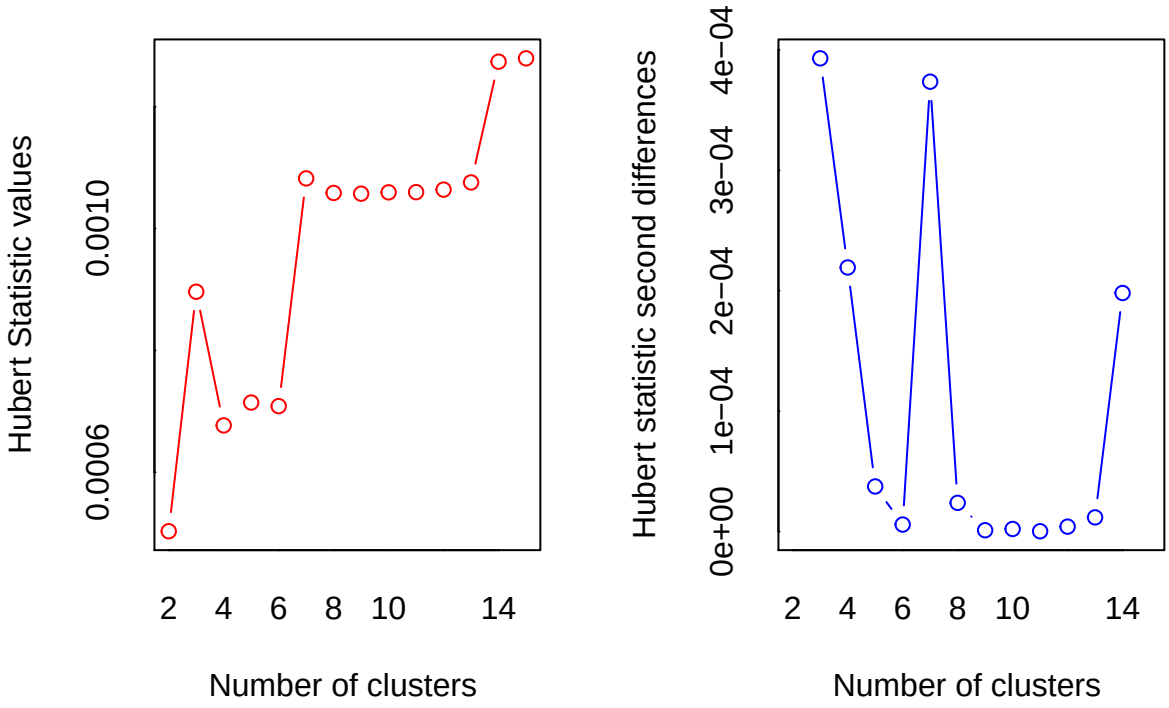
```
## K-means clustering with 4 clusters of sizes 29, 67, 49, 65
##
## Cluster means:
##          PC1        PC2
## 1 -1.2059792  0.3112454
## 2  0.5930386 -1.1557403
## 3 -3.2573633  0.3918498
## 4  2.3823171  0.7570437
##
## Clustering vector:
##   [1] 2 2 2 2 2 2 2 2 1 1 1 2 2 2 2 2 2 2 2 4 2 2 2 2 2 2 2 2 2 2 1 2 2 2
##  [36] 1 1 1 2 2 2 2 2 1 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 4 2 2 2 2 2 2 2 4
##  [71] 3 1 1 3 1 1 1 3 3 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 1 3 3 3 3
## [106] 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 2 3 3 3 3 1 3 3 1 1 1 2 1 1 1 1
## [141] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 4 4 4 4 4 4 4 4 4
## [176] 4 4 4 4 2 4 4 4 4 4 4 4 4 4 4 4 4 4 2 4 4 2 4 2 4 2 4 2 4 4 4 2 4 4 4 4
##
## Within cluster sum of squares by cluster:
## [1] 23.27612 78.65895 48.54999 59.69446
##  (between_SS / total_SS =  83.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
plot(seeds.pc2,col=seed.km$cluster)
points(seed.km$centers, col = 1:4, pch = 3, cex = 4)
```

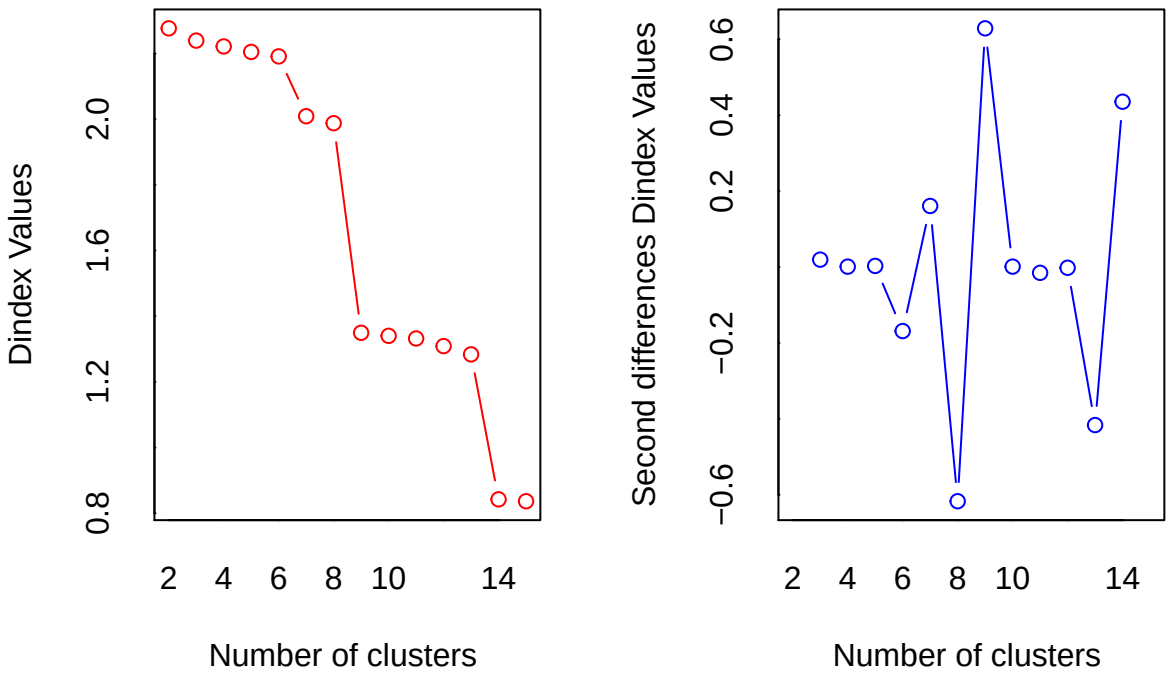


2c.

```
dist.pc2=dist(seeds.pc2)
best.s=NbClust(seeds.pc2,dist.pc2,distance=NULL,
               min.nc = 2,index="all",method="single")$Best.nc[1]
```

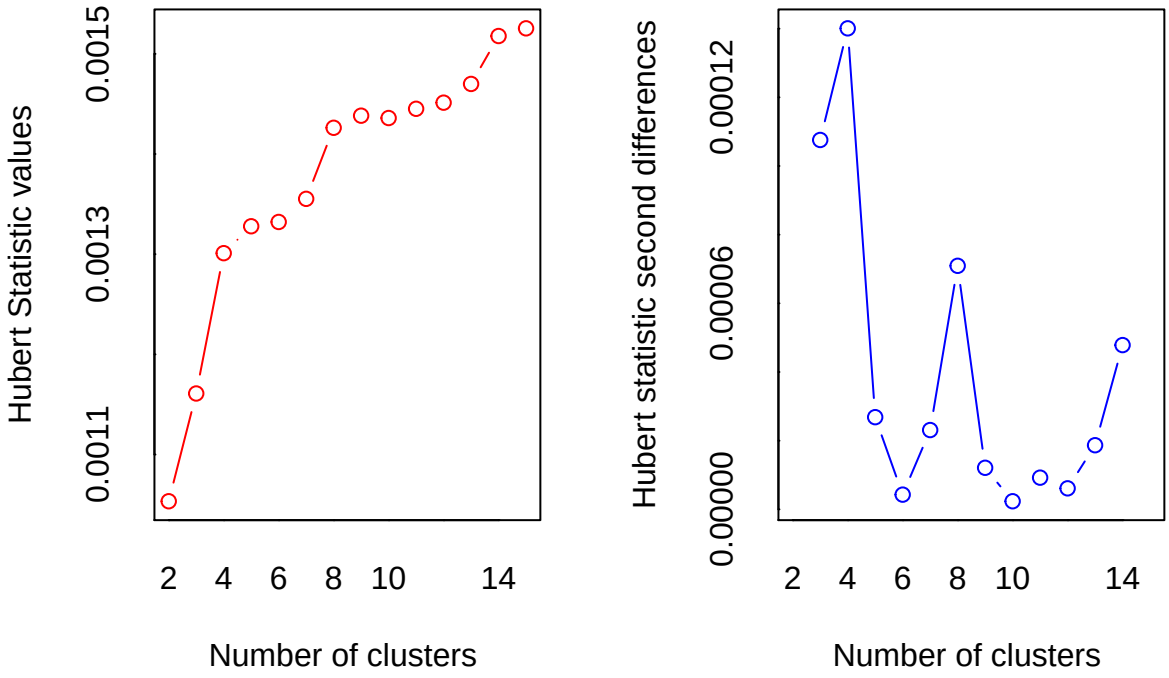



```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```

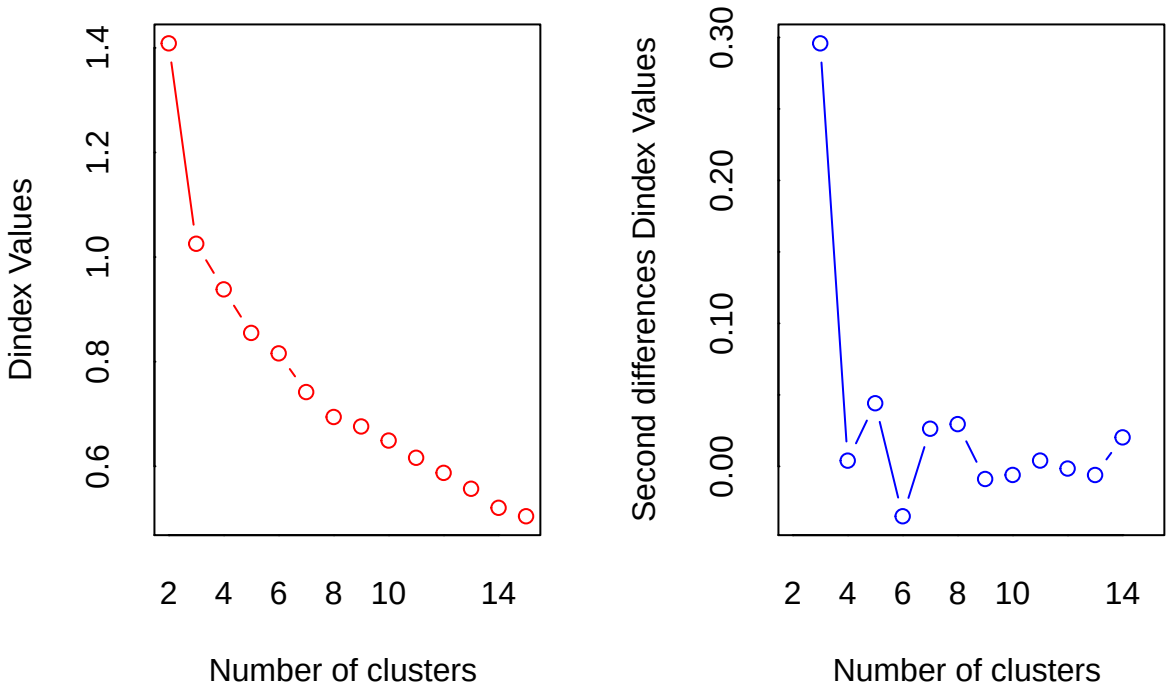



```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 5 proposed 2 as the best number of clusters
## * 1 proposed 3 as the best number of clusters
## * 1 proposed 6 as the best number of clusters
## * 1 proposed 7 as the best number of clusters
## * 3 proposed 9 as the best number of clusters
## * 11 proposed 14 as the best number of clusters
## * 1 proposed 15 as the best number of clusters
##
##                   ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  14
##
##
## *******************************************************************
```

```
best.c=NbClust(seeds.pc2,dist.pc2,distance=NULL,
               min.nc = 2,index="all",method="complete")$Best.nc[1]
```

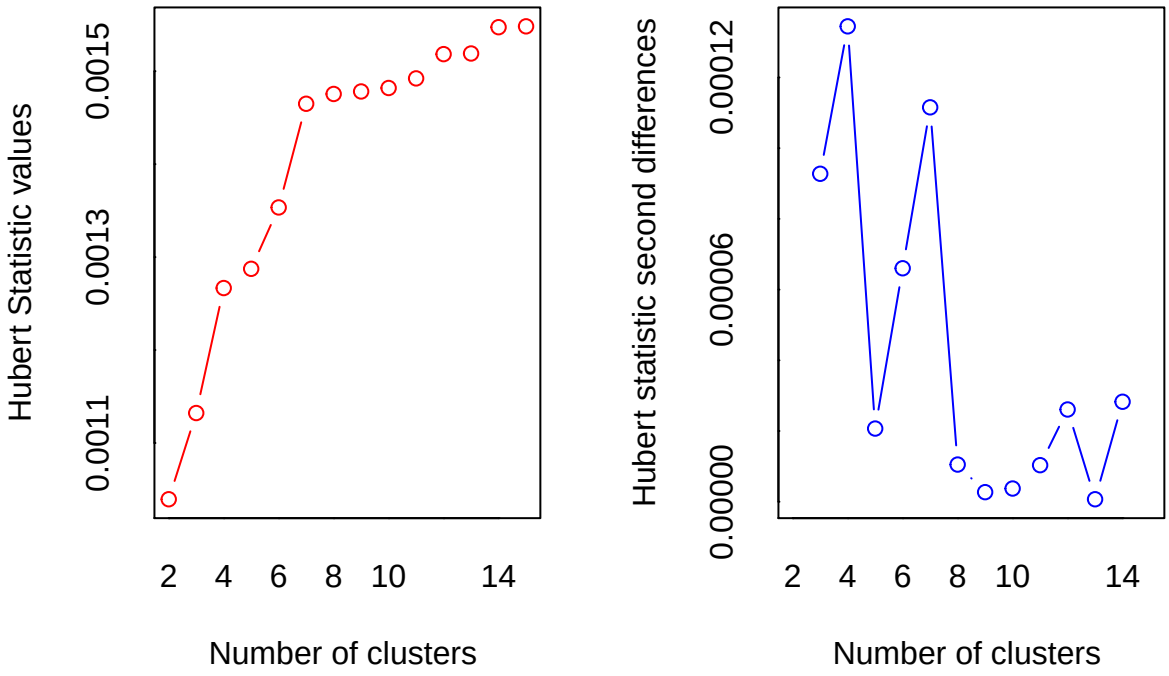



```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```

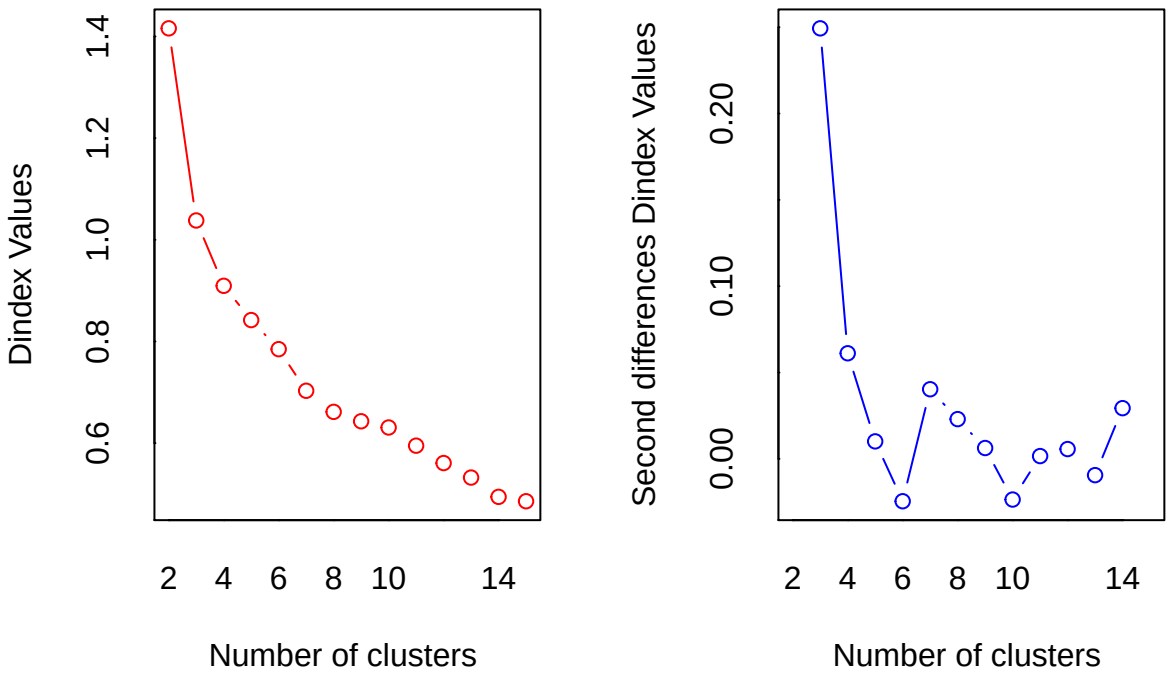



```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 7 proposed 2 as the best number of clusters
## * 11 proposed 3 as the best number of clusters
## * 1 proposed 4 as the best number of clusters
## * 1 proposed 5 as the best number of clusters
## * 2 proposed 14 as the best number of clusters
## * 1 proposed 15 as the best number of clusters
##
##                     ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  3
##
##
## *******************************************************************
```

```
best.a=NbClust(seeds.pc2,dist.pc2,distance=NULL,
               min.nc = 2,index="all",method="average")$Best.nc[1]
```




```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```

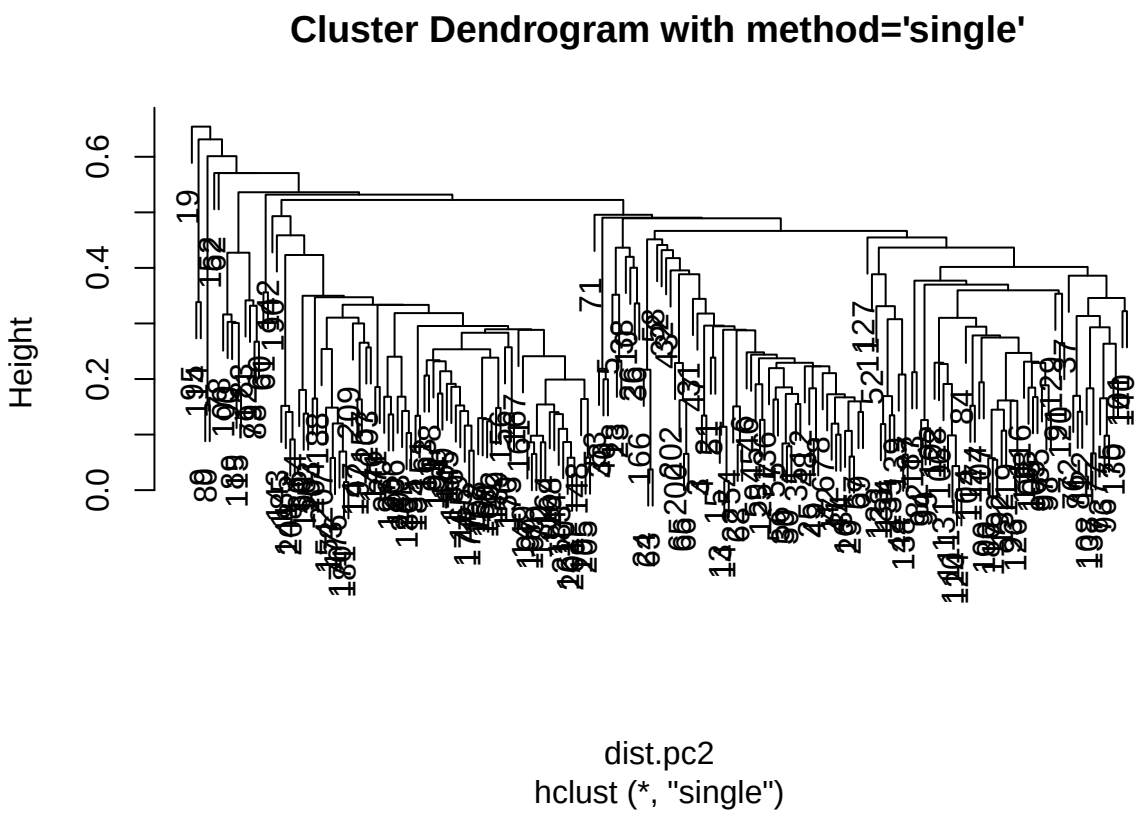



```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 6 proposed 2 as the best number of clusters
## * 11 proposed 3 as the best number of clusters
## * 1 proposed 4 as the best number of clusters
## * 1 proposed 7 as the best number of clusters
## * 1 proposed 13 as the best number of clusters
## * 2 proposed 14 as the best number of clusters
## * 1 proposed 15 as the best number of clusters
##
##                   ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  3
##
##
## *******************************************************************
```

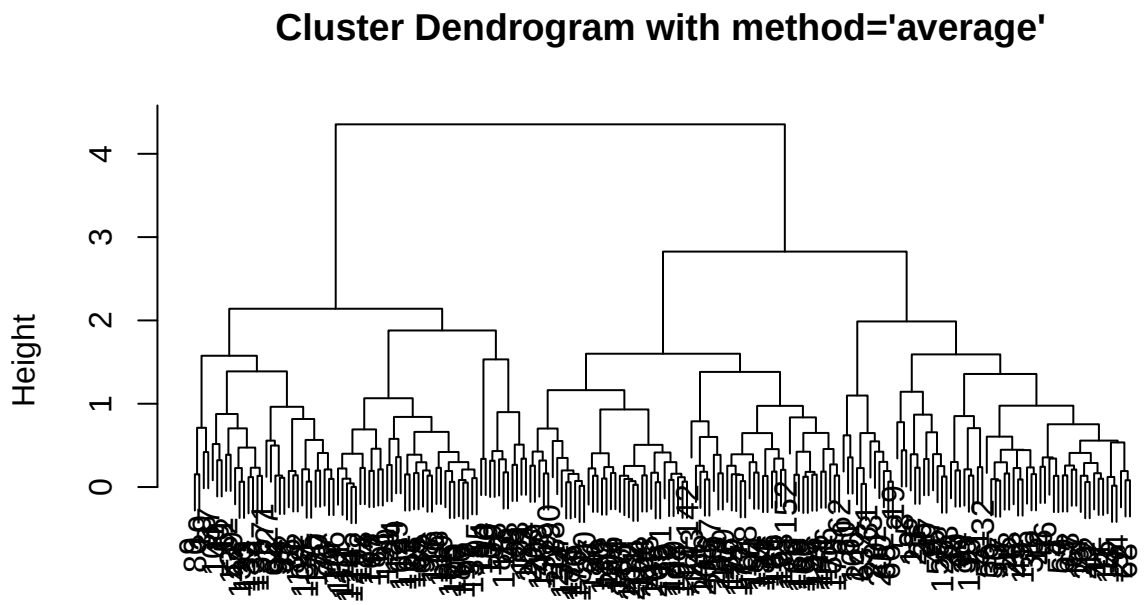
Under these methods, I find the best clusters for single, complete, and average methods to be 14,3, and 3 respectively.

```
set.seed(10)
hc.s=hclust(dist.pc2,method="single")
```

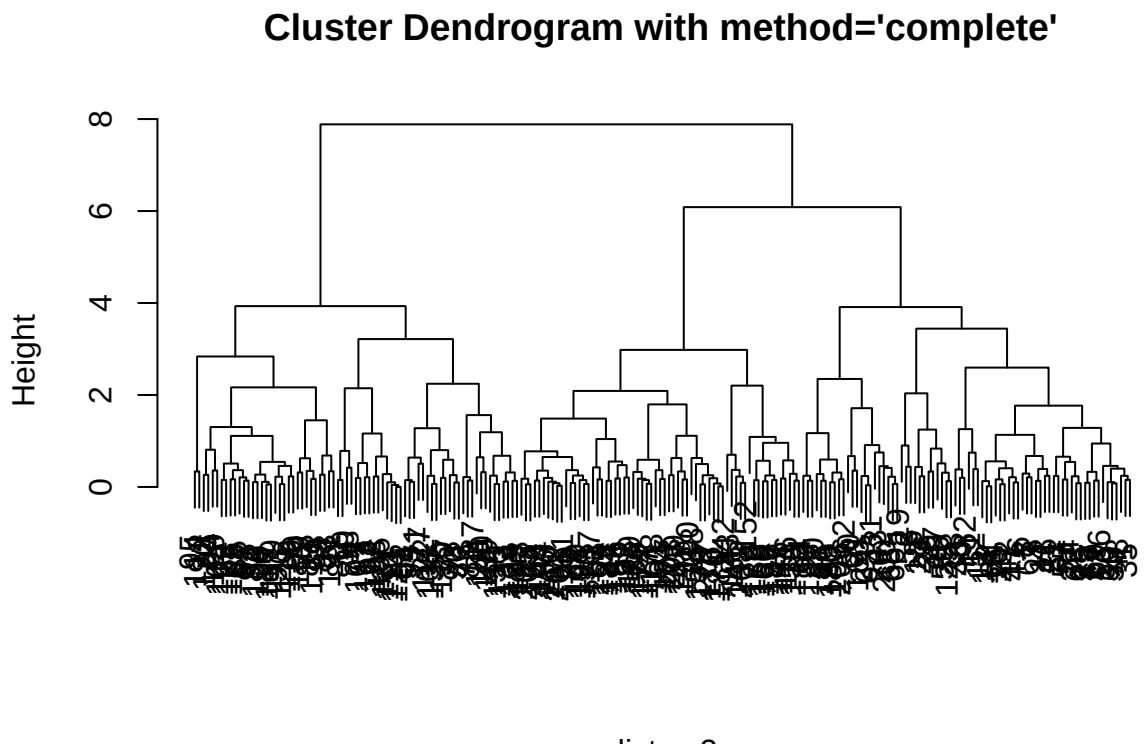
```
hc.c=hclust(dist.pc2,method="complete")
hc.a=hclust(dist.pc2,method="average")
plot(hc.s,main="Cluster Dendrogram with method='single'")
```



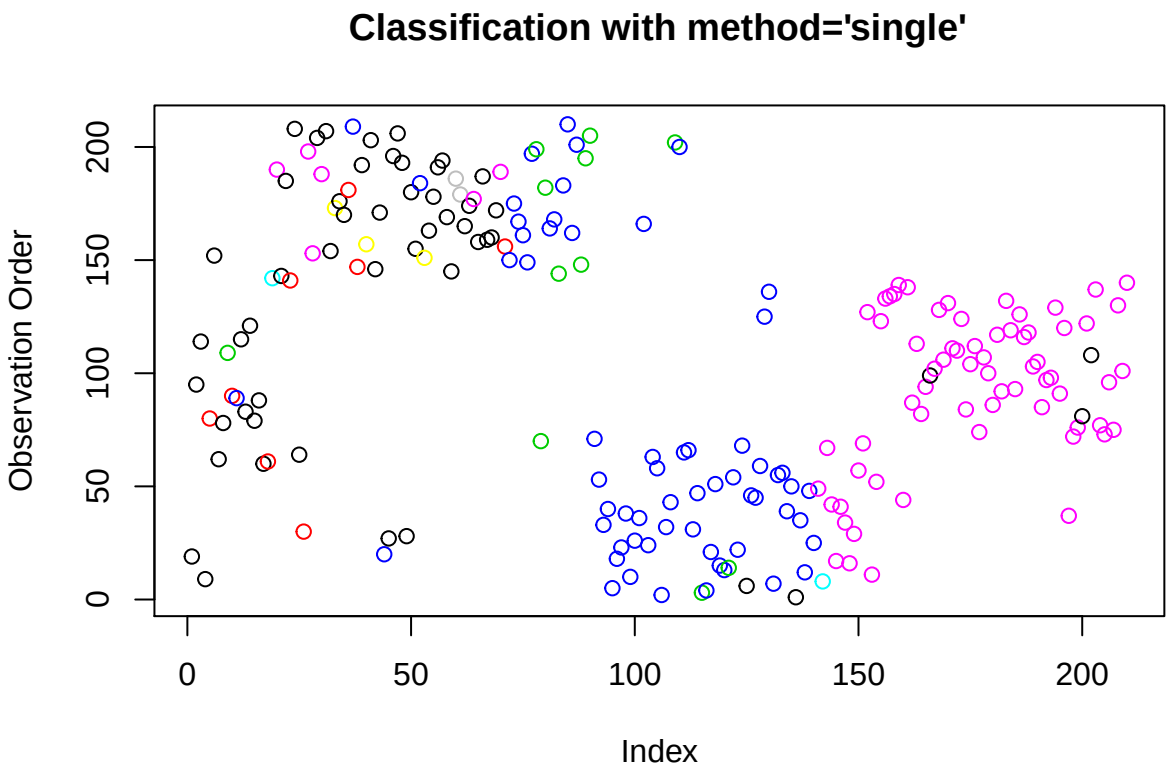

```
plot(hc.a,main="Cluster Dendrogram with method='average'")
```



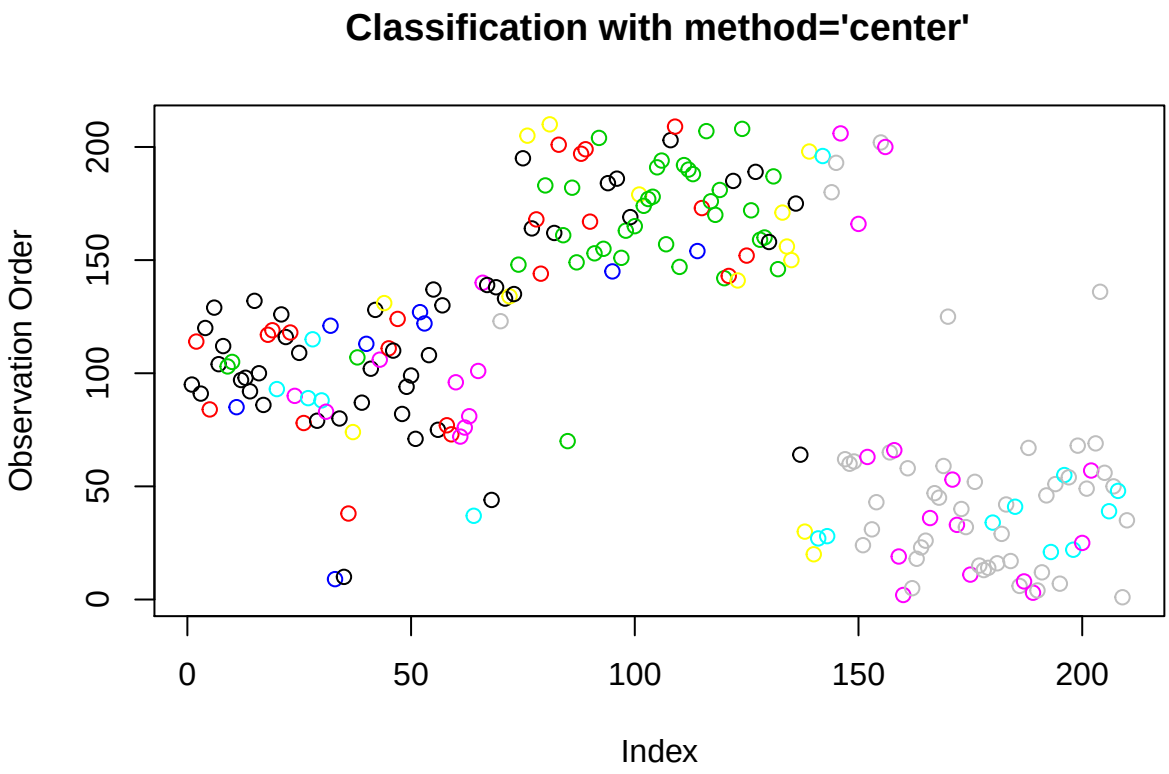

```
plot(hc.c,main="Cluster Dendrogram with method='complete'")
```



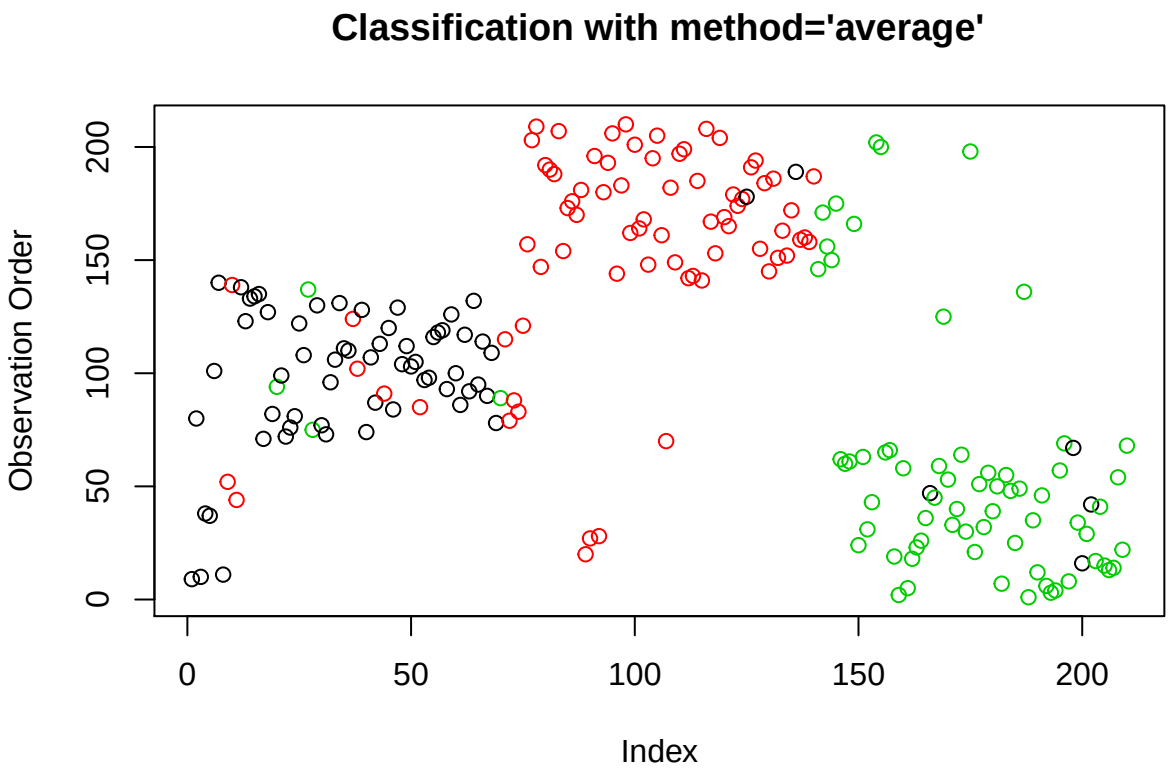

```
hc.s.cut=cutree(hc.s,k=best.s)
hc.a.cut=cutree(hc.a,k=best.a)
hc.c.cut=cutree(hc.c,k=best.c)
plot(hc.s$order,col=hc.s.cut,main="Classification with method='single'",
     ylab="Observation Order")
```




```
plot(hc.c$order,col=hc.c.cut,main="Classification with method='center'",
     ylab="Observation Order")
```




```
plot(hc.a$order,col=hc.a.cut,main="Classification with method='average'",
     ylab="Observation Order")
```



Here we see

Finding best cluster arrangement using index="ball"

```
best.s.b=NbClust(seeds.pc2,dist.pc2,distance=NULL,
                 min.nc = 2,index="ball",method="single")$Best.nc[1]
best.c.b=NbClust(seeds.pc2,dist.pc2,distance=NULL,
                 min.nc = 2,index="ball",method="complete")$Best.nc[1]
best.a.b=NbClust(seeds.pc2,dist.pc2,distance=NULL,
                 min.nc = 2,index="ball",method="average")$Best.nc[1]
best.s.b
```

```
## Number_clusters
##               3
```

```
best.c.b
```

```
## Number_clusters
##               3
```

```
best.c.b
```

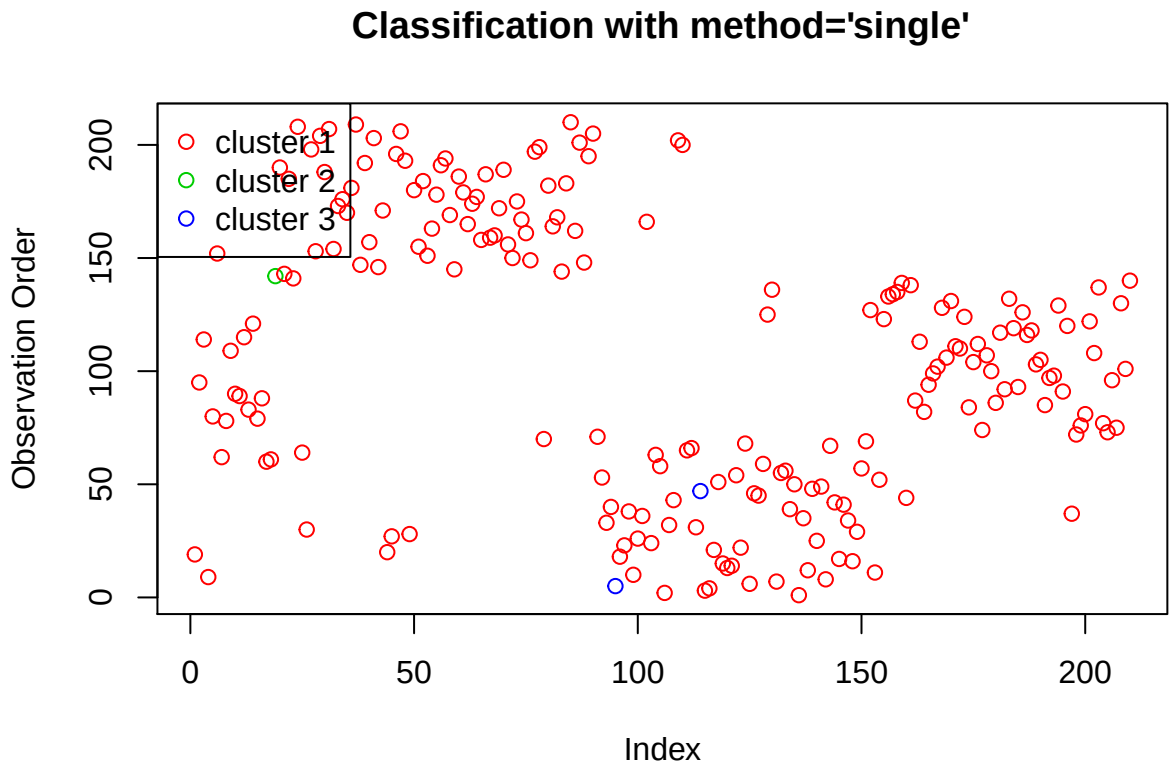
```
## Number_clusters
##               3
```

We observe the optimal number of clusters to be 3 for each method.

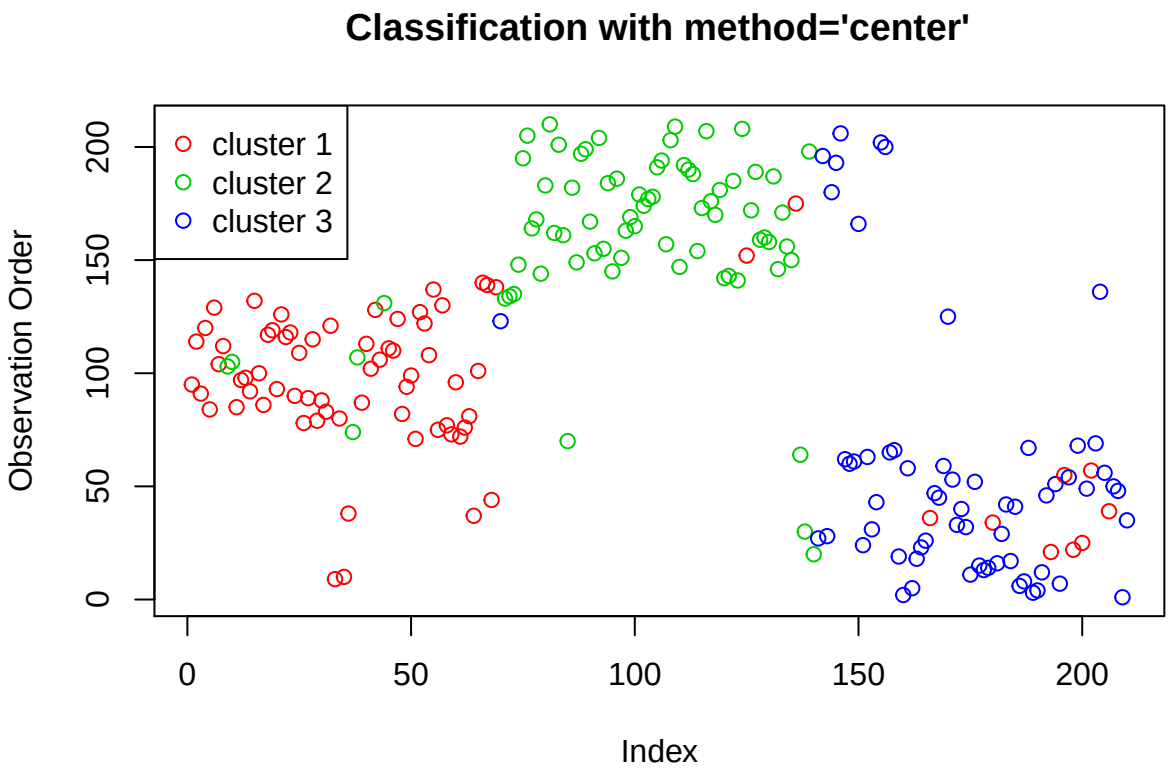
Cutting the dendrogram into three clusters then plotting cluster assignments

```
hc.sb.cut=cutree(hc.s,k=best.s.b)
hc.ab.cut=cutree(hc.a,k=best.a.b)
```

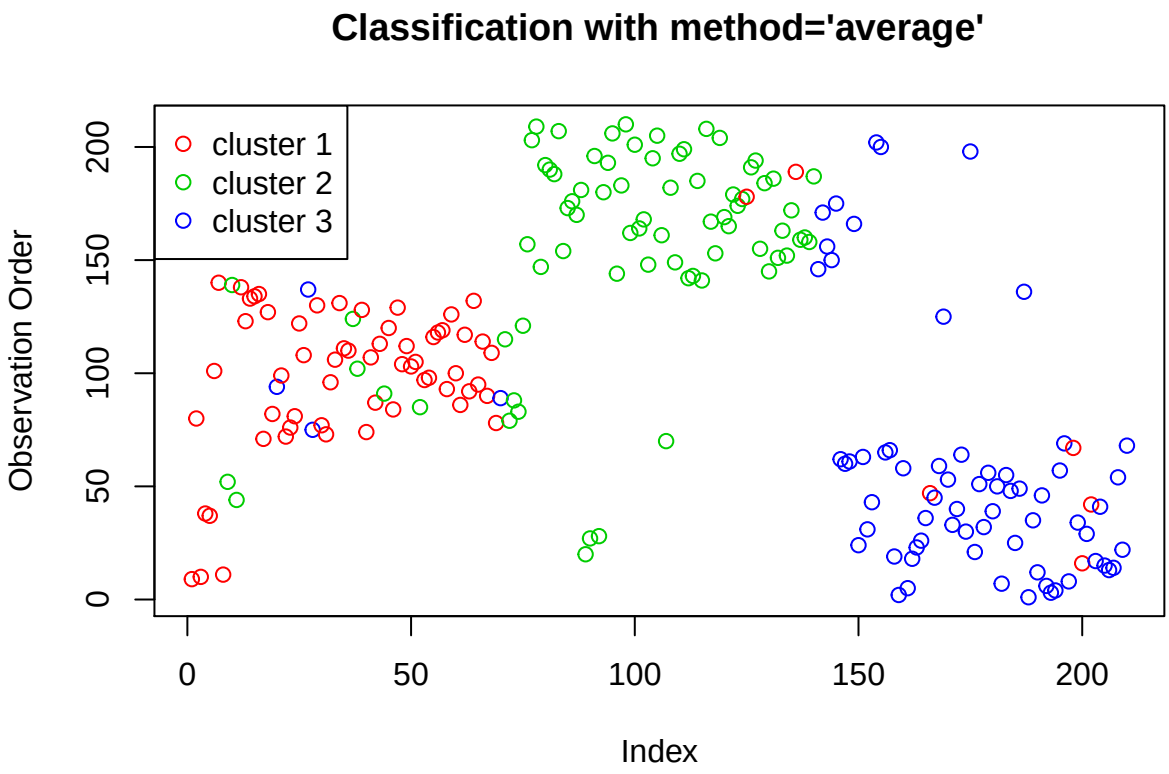
```
hc.cb.cut=cutree(hc.c,k=best.c.b)
plot(hc.s$order,col=hc.sb.cut+1,
     main="Classification with method='single'",
     ylab="Observation Order")
legend("topleft",legend = c("cluster 1","cluster 2","cluster 3"),
       pch=rep(1,3),col=c(2:4))
```




```
plot(hc.c$order,col=hc.cb.cut+1,
     main="Classification with method='center'",
     ylab="Observation Order")
legend("topleft",legend = c("cluster 1","cluster 2","cluster 3"),
       pch=rep(1,3),col=c(2:4))
```




```
plot(hc.a$order,col=hc.ab.cut+1,
     main="Classification with method='average'",
     ylab="Observation Order")
legend("topleft",legend = c("cluster 1","cluster 2","cluster 3"),
       pch=rep(1,3),col=c(2:4))
```



Here we observe that when method=single almost all observations are classified within the first cluster. However, the average and complete method appear to classify the data more evenly across all clusters.

3. Image to be used for analysis

```
img = read.bmp('image1.bmp')
img = t(img[ncol(img):1,])
img = img - mean(img)
gs = grey((0:255)/255)
image(img, asp=1, col=gs, yaxs='r', xaxt='n', yaxt='n')
```

3a. The xaxt and yaxt secifies the type of variable of the x and y axis. In the image reading xaxt and yaxt="n". This implies that the plotting is supressed when the function is called.

3b.

```
image.3b=prcomp(img,center=F)
str(summary(image.3b))
```

```
## List of 6
##  $ sdev      : num [1:512] 579 448 376 290 261 ...
##  $ rotation  : num [1:512, 1:512] 0.0286 0.0287 0.0287 0.0284 0.0291 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:512] "PC1" "PC2" "PC3" "PC4" ...
##  $ center    : logi FALSE
##  $ scale     : logi FALSE
##  $ x         : num [1:512, 1:512] -906 -907 -893 -885 -880 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:512] "PC1" "PC2" "PC3" "PC4" ...
##  $ importance: num [1:3, 1:512] 579.398 0.286 0.286 447.544 0.171 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "Standard deviation" "Proportion of Variance" "Cumulative Proportion"
##   .. ..$ : chr [1:512] "PC1" "PC2" "PC3" "PC4" ...
##  - attr(*, "class")= chr "summary.prcomp"
```

3c.

```
norm(image.3b$x%*%t(image.3b$rotation) - img,type="f")
```

```
## [1] 3.495589e-11
```

From this we see that the product of these two matracies is $3.4955 * 10^-11$ which we take to be approximately 0.

3d. Checking if the rotation of the prcomp matrix is a rotational matrix

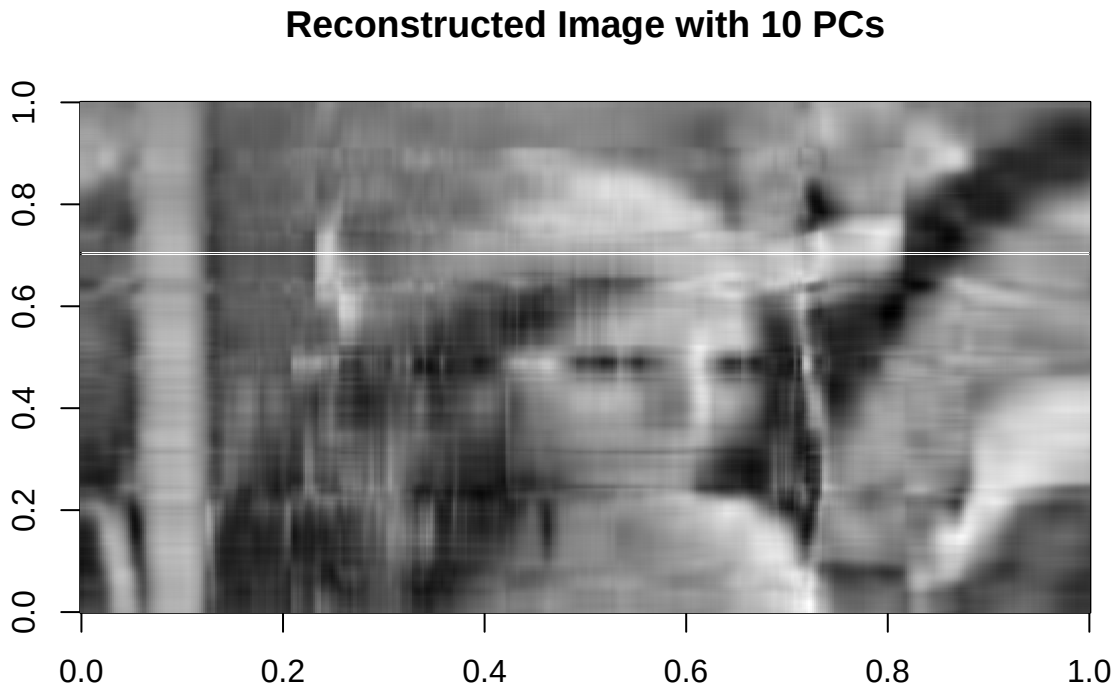
```
norm(t(image.3b$rotation)%*%image.3b$rotation-matrix(rep(c(1,rep(0,512)),512),nrow=512,ncol=512),type="
```

```
## [1] 5.522941e-14
```

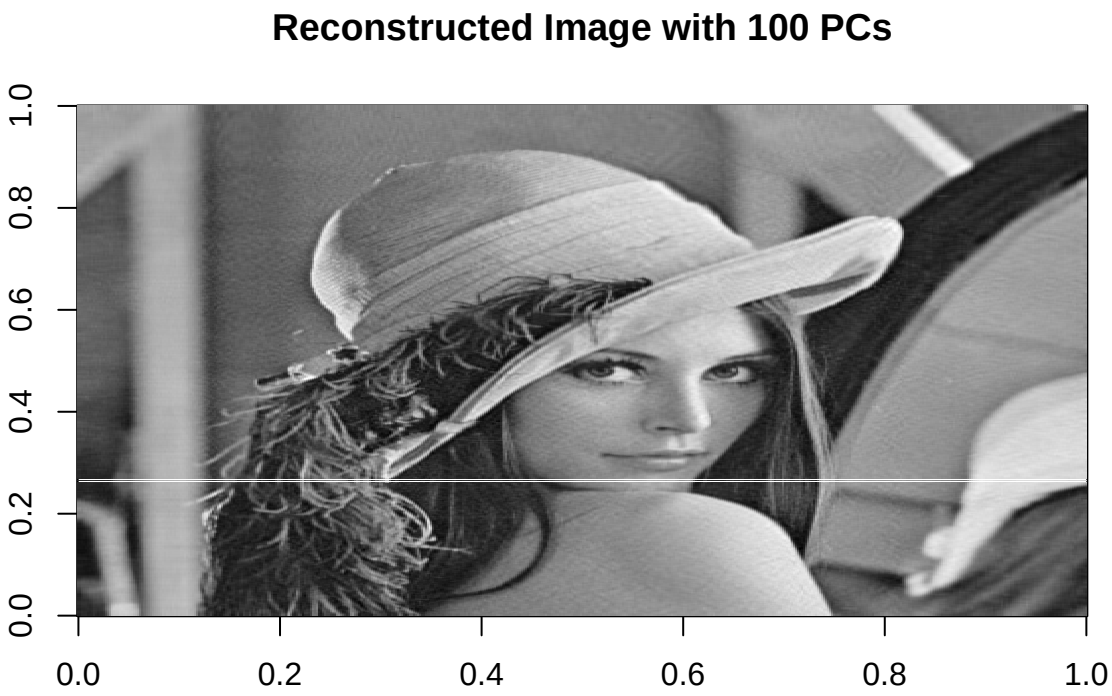
From this we calculate a value of $5.523 * 10^-14$ which we approximate to be 0. Thus confirming that image.3b$rotation is indead a rotational matrix.

3e. Reconstructing Image based on Principle Components

```
img.rec.10=image.3b$x[,1:10]%*%t(image.3b$rotation[,1:10])
img.rec.100=image.3b$x[,1:100]%*%t(image.3b$rotation[,1:100])
image(img.rec.10,col=gray((0:255)/255),main="Reconstructed Image with 10 PCs")
```

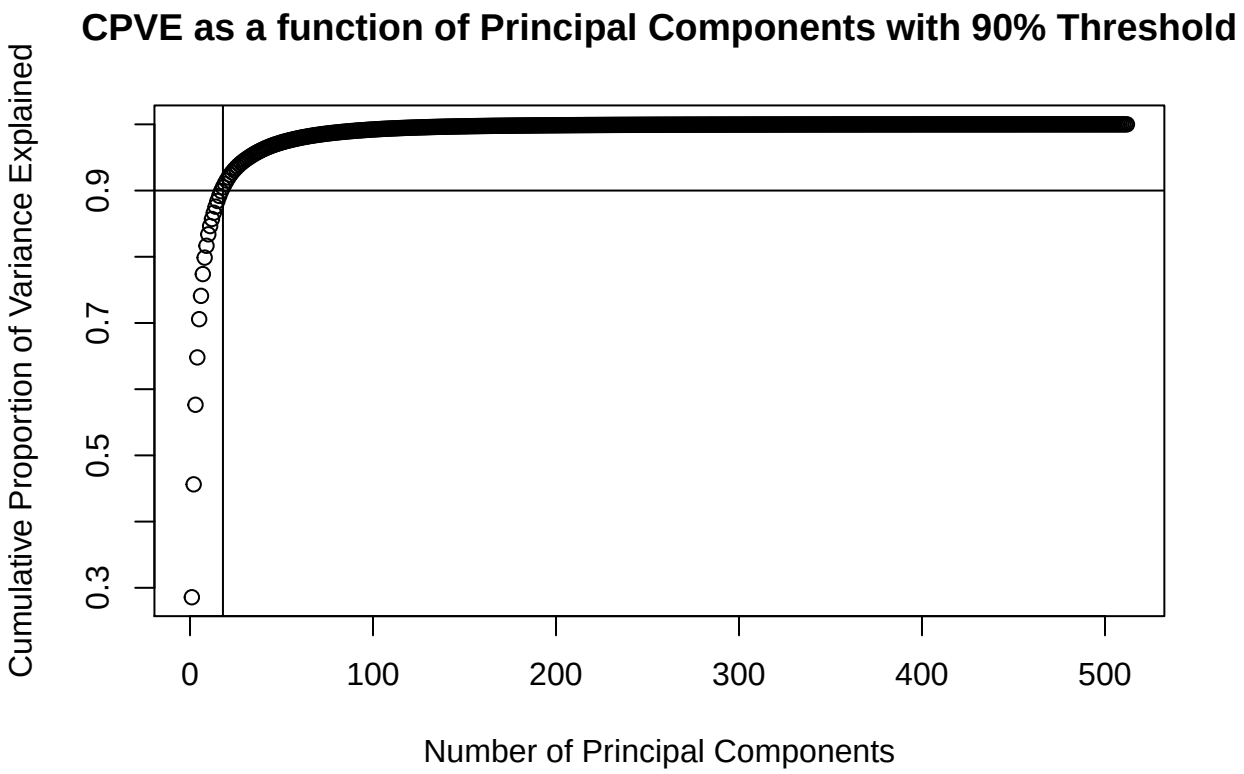



```
image(img.rec.100,col=gray((0:255)/255),main="Reconstructed Image with 100 PCs")
```



3f. Plot of Cumulative Proportion of Variance as a function of number of principal components.

```
plot(c(1:512),summary(image.3b)$importance[3,],
     xlab="Number of Principal Components",ylab="Cumulative Proportion of Variance Explained",
     main="CPVE as a function of Principal Components with 90% Threshold")
abline(h=.9,v=18)
```




```
pve.90=summary(image.3b)$importance[3,]
min(which(pve.90>=.9))
```

```
## [1] 18
```

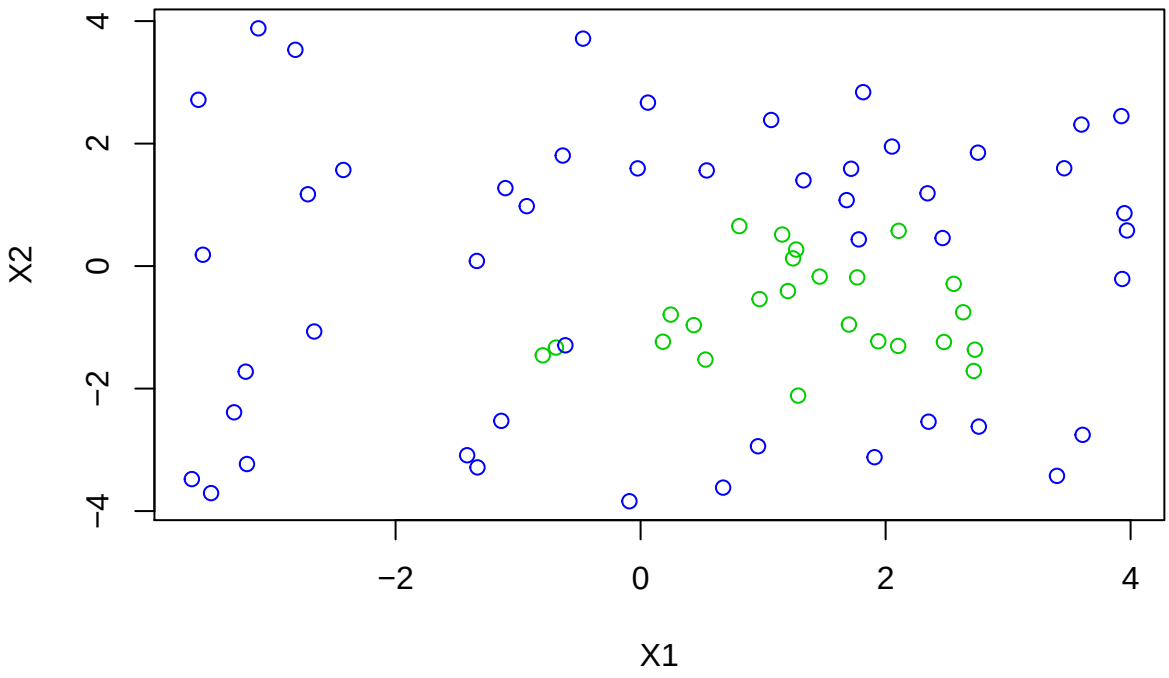
18 Principle components are required to explain at least 90% of the variance.

4a.

```
nonlinear=read_csv("nonlinear.csv")
```

```
## Parsed with column specification:
## cols(
##   Index = col_integer(),
##   X1 = col_double(),
##   X2 = col_double(),
##   Y = col_integer()
## )
```

```
plot(nonlinear$X1,nonlinear$X2,col=c(nonlinear$Y+3),xlab="X1",ylab="X2")
```



4b.Using logistic regresion to predict classification given X1 and X2

```
nlin.glm=glm(Y~X1+X2,family=binomial,data=nonlinear)
summary(nlin.glm)
```

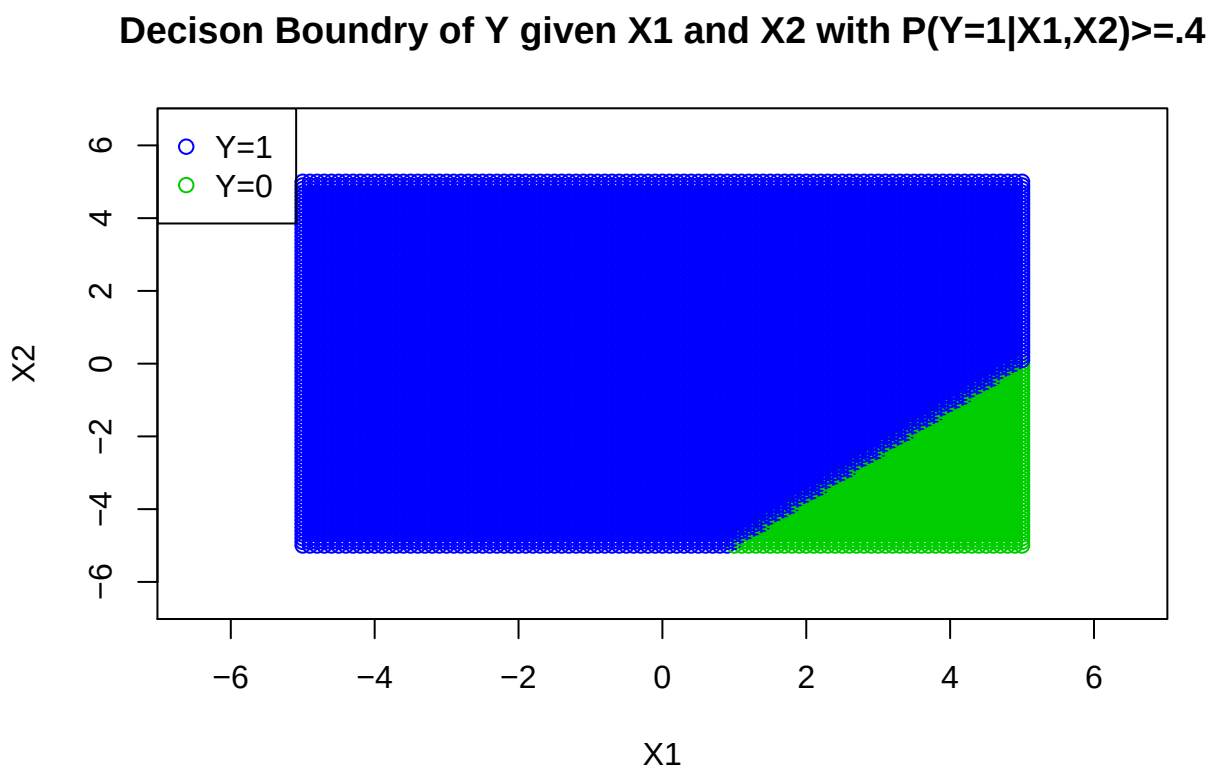
```
##
## Call:
## glm(formula = Y ~ X1 + X2, family = binomial, data = nonlinear)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5940  -1.2476   0.6256   0.9155   1.5108
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.0224     0.3137   3.259  0.00112 **
## X1           -0.2893     0.1360  -2.127  0.03341 *
## X2            0.2323     0.1435   1.618  0.10560
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 91.658  on 71  degrees of freedom
## Residual deviance: 84.523  on 69  degrees of freedom
## AIC: 90.523
##
```

```
## Number of Fisher Scoring iterations: 4
```

```
gr <- expand.grid(X1=seq(-5, 5, by=0.1), # sample points in X1
X2=seq(-5, 5, by=0.1)) # sample points in X2
nlin.glm.pred=predict(nlin.glm,gr,type="response")
```

Plot of decision boundry where boundry is decided where P(Y=1|X1,X2)>=.4

```
plot(gr,col=(nlin.glm.pred>=.4)+3,
     main="Decison Boundry of Y given X1 and X2 with P(Y=1|X1,X2)>=.4",
     xlim=c(-6.5,6.5),ylim=c(-6.5,6.5))
legend("topleft",legend=c("Y=1 ","Y=0"),pch=c(1,1),col=c(4,3))
```



4c.

```
nlin.glm.poly=glm(Y~poly(X1,2)*poly(X2,2),family=binomial,data=nonlinear)
```

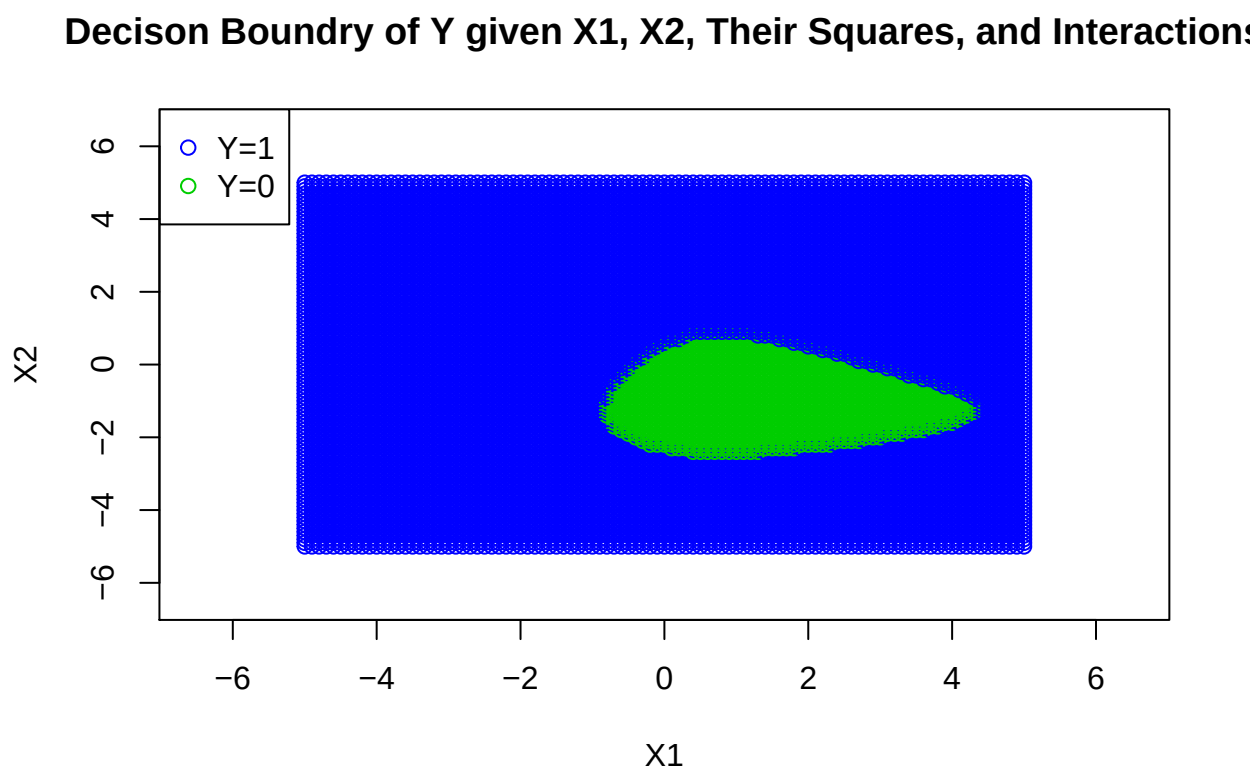
```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(nlin.glm.poly)
```

```
##
## Call:
## glm(formula = Y ~ poly(X1, 2) * poly(X2, 2), family = binomial,
##     data = nonlinear)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.57091  -0.09697   0.00000   0.01295   1.89656
##
```

```
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    14.37      15.55   0.924    0.355
## poly(X1, 2)1                  -51.11     147.45  -0.347    0.729
## poly(X1, 2)2                  103.41     134.54   0.769    0.442
## poly(X2, 2)1                   99.60     134.17   0.742    0.458
## poly(X2, 2)2                  113.85     117.09   0.972    0.331
## poly(X1, 2)1:poly(X2, 2)1  -181.63    1294.32  -0.140    0.888
## poly(X1, 2)2:poly(X2, 2)1   583.71    1165.55   0.501    0.617
## poly(X1, 2)1:poly(X2, 2)2   108.28    1072.06   0.101    0.920
## poly(X1, 2)2:poly(X2, 2)2   445.15    1127.08   0.395    0.693
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 91.658  on 71  degrees of freedom
## Residual deviance: 12.561  on 63  degrees of freedom
## AIC: 30.561
##
## Number of Fisher Scoring iterations: 14
```

```
nlin.glm.pred.poly=predict(nlin.glm.poly,gr,type="response")
plot(gr,col=(nlin.glm.pred.poly>=.4)+3,
     main="Decison Boundry of Y given X1, X2, Their Squares, and Interactions",
     xlim=c(-6.5,6.5),ylim=c(-6.5,6.5))
legend("topleft",legend=c("Y=1","Y=0"),pch=c(1,1),col=c(4,3))
```



From this plot we see that the classification of whether or not Y=1 becomes significantly less linear aswe we

account for the interactions of higher degree predictor variables.

4d.

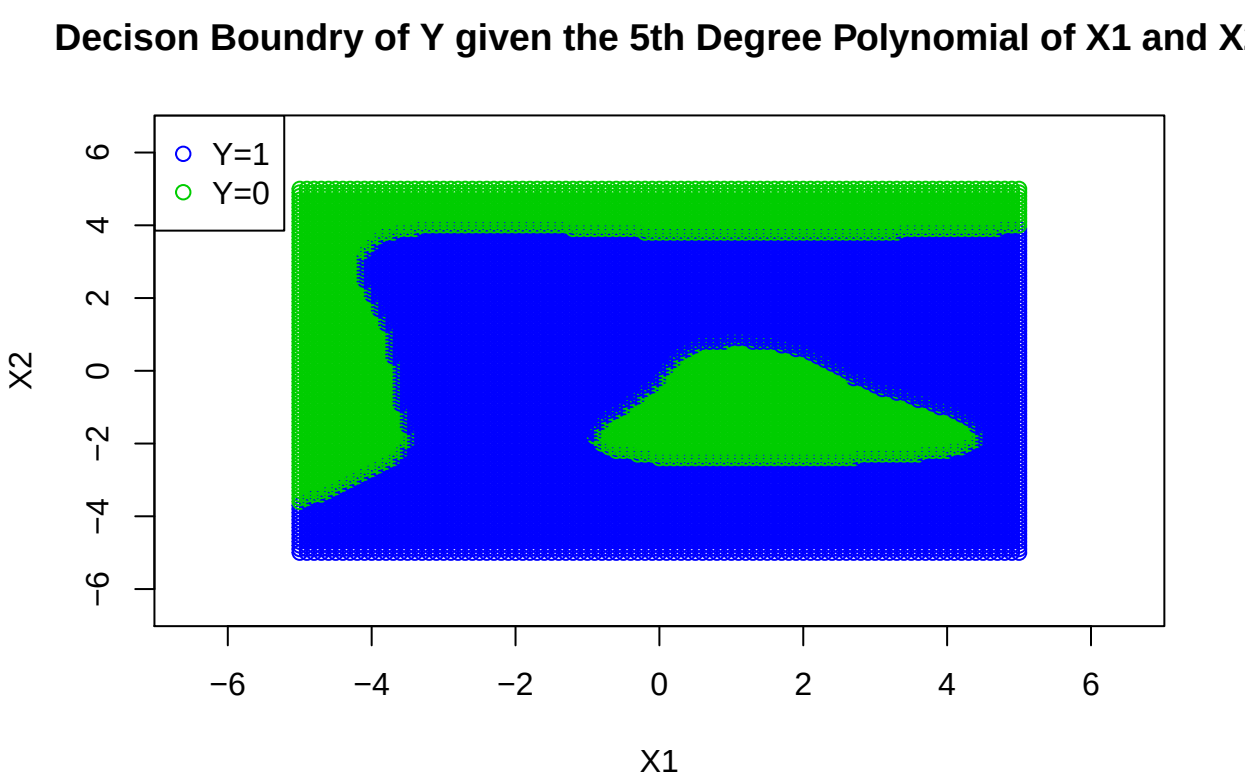
```
nlin.glm.poly.5=glm(Y~poly(X1,5)+poly(X2,5),family=binomial,data=nonlinear)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(nlin.glm.poly.5)
```

```
##
## Call:
## glm(formula = Y ~ poly(X1, 5) + poly(X2, 5), family = binomial,
##     data = nonlinear)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.24411  -0.02088   0.00000   0.00078   1.85481
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)     25.42      41.06   0.619    0.536
## poly(X1, 5)1   -49.29      88.35  -0.558    0.577
## poly(X1, 5)2    25.89      36.92   0.701    0.483
## poly(X1, 5)3    36.24      60.98   0.594    0.552
## poly(X1, 5)4   -34.71      64.85  -0.535    0.593
## poly(X1, 5)5    12.65      37.72   0.335    0.737
## poly(X2, 5)1  -174.38     386.21  -0.452    0.652
## poly(X2, 5)2   266.09     480.06   0.554    0.579
## poly(X2, 5)3  -228.97     422.75  -0.542    0.588
## poly(X2, 5)4    90.75     219.09   0.414    0.679
## poly(X2, 5)5  -101.31     203.20  -0.499    0.618
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 91.658  on 71  degrees of freedom
## Residual deviance: 12.494  on 61  degrees of freedom
## AIC: 34.494
##
## Number of Fisher Scoring iterations: 14
```

```
nlin.glm.pred.poly.5=predict(nlin.glm.poly.5,gr,type="response")
plot(gr,col=(nlin.glm.pred.poly.5>=.4)+3,
     main="Decison Boundry of Y given the 5th Degree Polynomial of X1 and X2",
     xlim=c(-6.5,6.5),ylim=c(-6.5,6.5))
legend("topleft",legend=c("Y=1","Y=0"),pch=c(1,1),col=c(4,3))
```



Much like part c, we see the linear relation of part a disapear significantly. Additionally, there appears to be an even greater amount of classification of Y=0 than the other two methods.

4e. From the summary reports we see that the coefficients of the linear model, in terms or magnitude, are significantly smaller than the polynomial regression coefficients. This implies that the polynomial models are subject to a lot of variance with their predictions, specifically the 5th degree polynomial model which does not appear to have much bias in its predicitions. The linear model does appear to have quite a bit of bias in its predictions. So we assume that as we expand the model from a linear to a polynomial format we lose bias at the cost of increased variance. However, as we take higher order polynomial models we increase the probability that we overfit the prediction model. This is because as we add predictior variables to the model their relative impact on explaining the variance diminishes. In the summary outputs above we see the calculated z-scores tend towards zero as more predictor variables are added. If the model groes to be too large the value of each predictor will tend towards 0 and we will not be able to construct any meaningful predictions from our model. As such as we take higher degree polynomials, we overfit our model with useless predictors.