Data Mining Project 4

Paichana Kularb 57090015

Import libraries and dataset

```
library(cluster)
library(factoextra)
```

```
## Loading required package: ggplot2
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```
library(dbscan)
library(seriation)
library(fpc)
```

```
##
## Attaching package: 'fpc'
```

```
## The following object is masked from 'package:dbscan':
##
##     dbscan
```

```
df = faithful
```

Set random seed so that the results can be reproducible

```
set.seed(0)
```

Check for NAs in the dataset

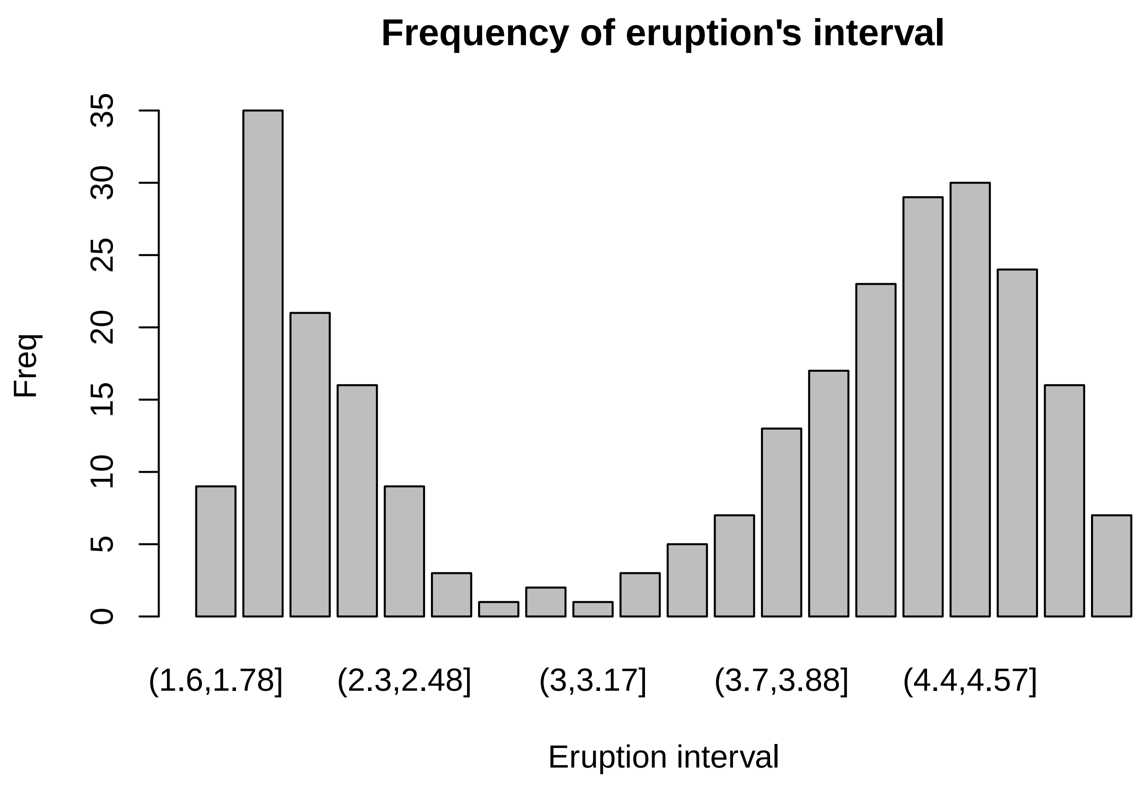
```
summary(is.na(faithful))
```

```
##  eruptions        waiting
##  Mode :logical   Mode :logical
##  FALSE:272       FALSE:272
```

Finding suitable number of clusters

Histogram with 20 bins that is group according to eruption time can be show by:

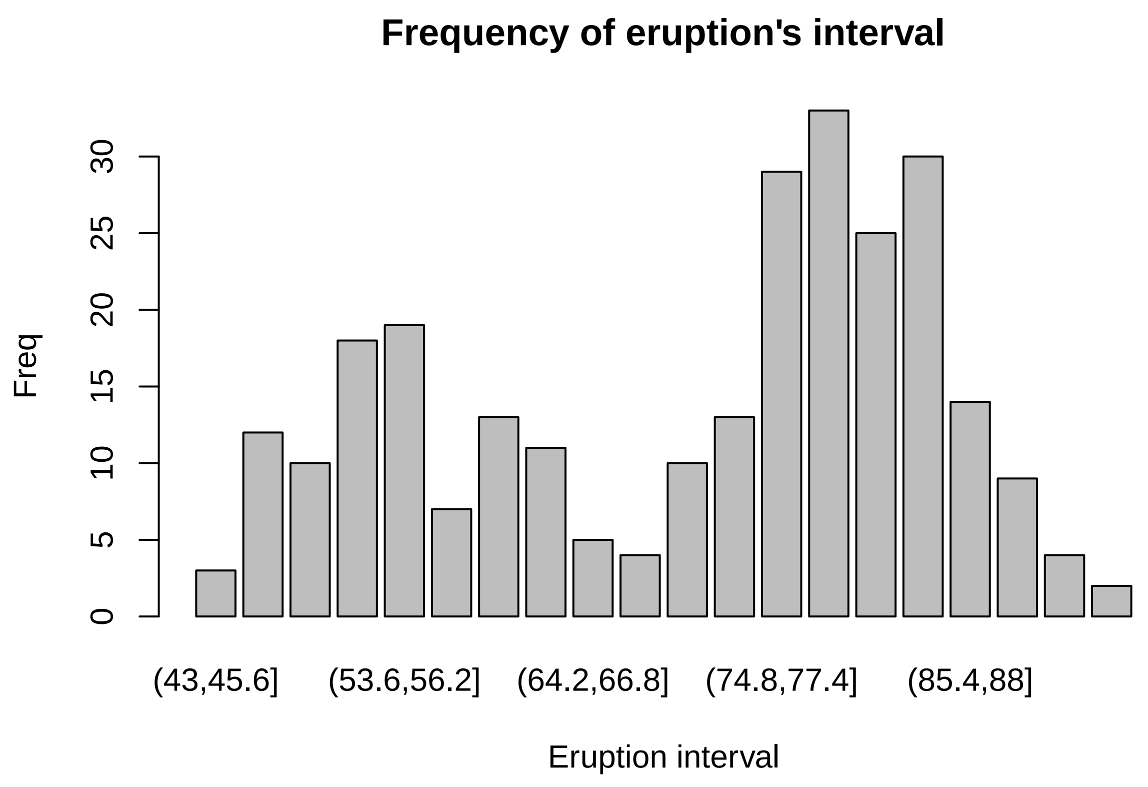
```
erup = df$eruptions
rangErup = range(erup)
interval = seq(rangErup[1], rangErup[2], by=((rangErup[2]-rangErup[1])/20))
erup.interval = cut(erup,interval)
plot(erup.interval,xlab = "Eruption interval",ylab = "Freq",main = "Frequency of eruption's interval")
```



2 groups of data can be seen

Histogram with 20 bins that is group according to waiting time can be show by:

```
wait = df$waiting
rangeWait = range(wait)
interval =seq(rangeWait[1], rangeWait[2], by=((rangeWait[2]-rangeWait[1])/20))
erup.interval = cut(wait,interval)
plot(erup.interval,xlab = "Eruption interval",ylab = "Freq",main = "Frequency of eruption's interval")
```



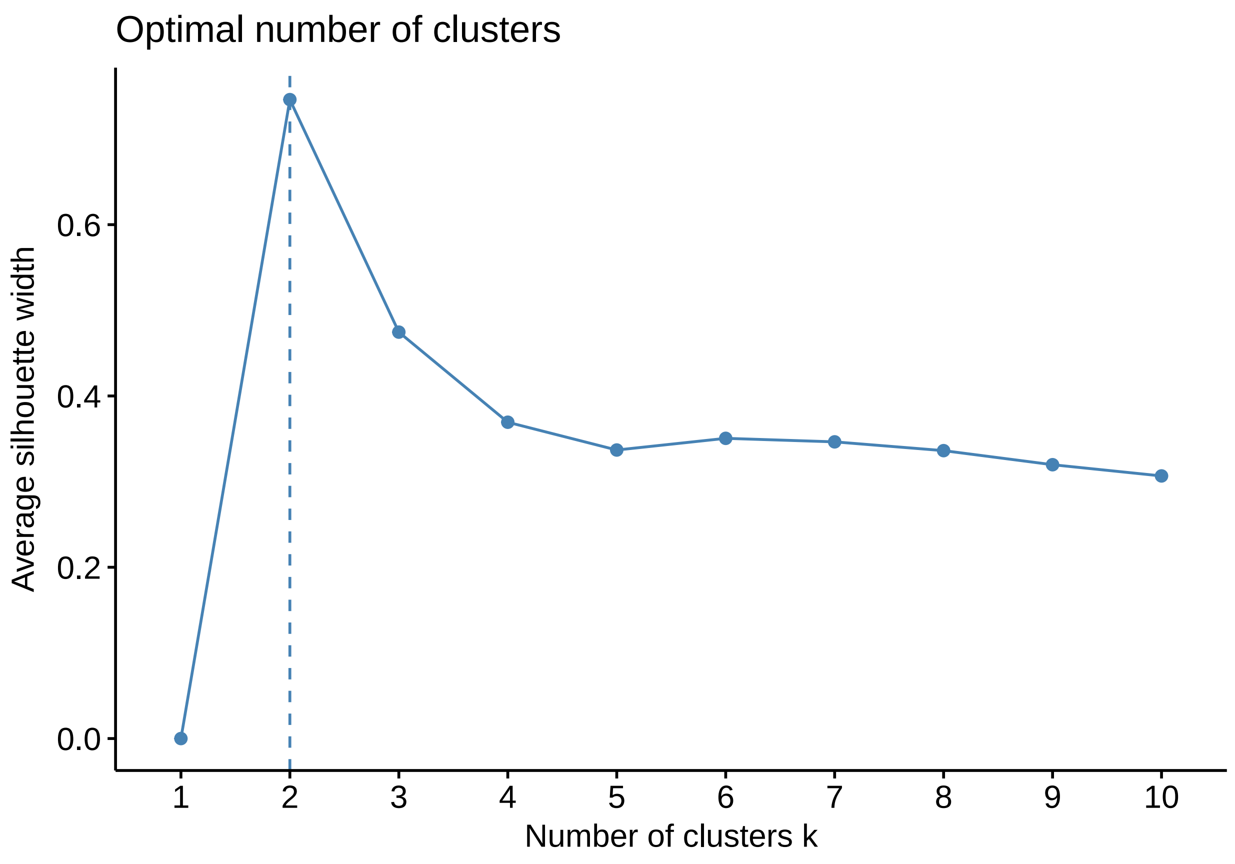
2 groups of data can be seen

Scale data in to standard normal distribution with mean of 0 and sd of 1

```
df = scale(df)
```

The optimal number of clusters can also be shown by using the Average Silhouette Method.

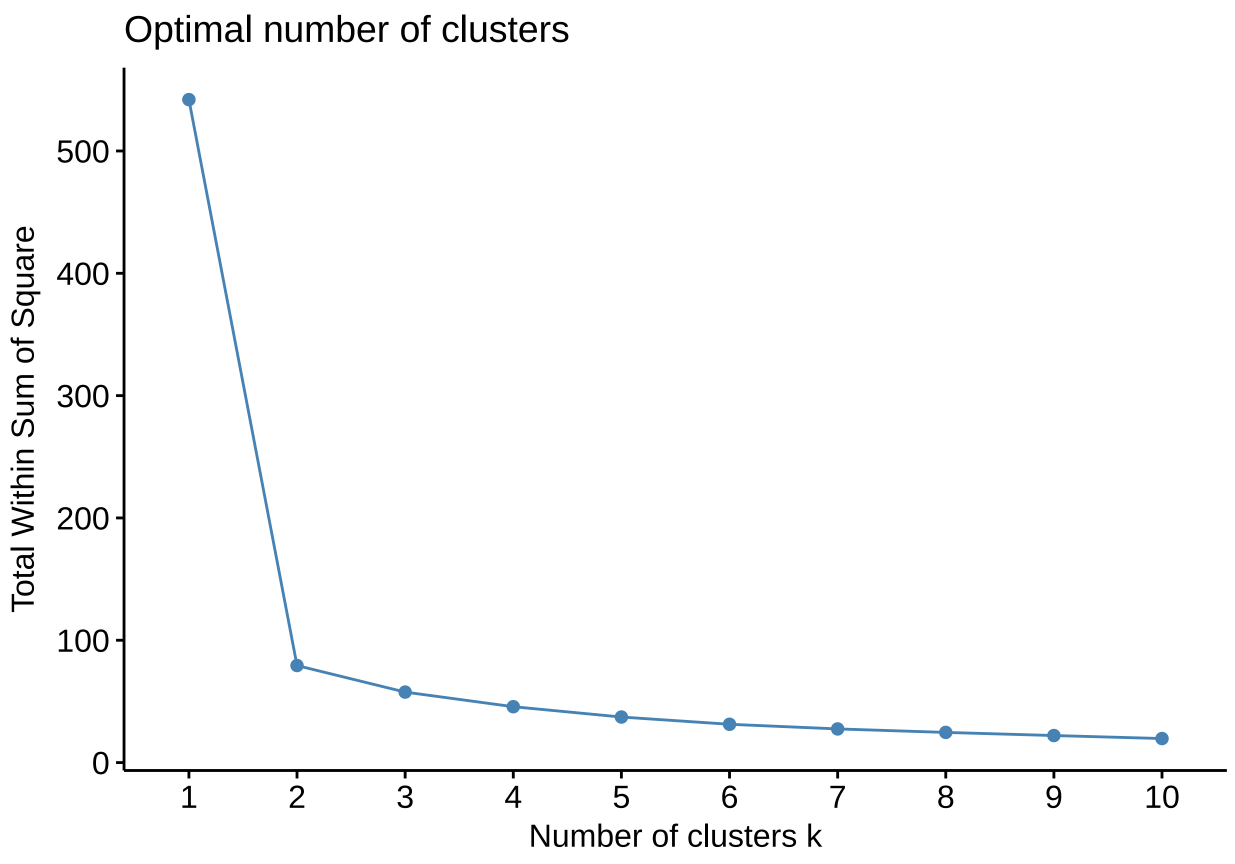
```
fviz_nbclust(df, FUN = hcut, method = "silhouette")
```



The Average Silhouette Method determines the quality of each cluster. The higher the number the better the clustering. From the graph 2 cluster maximizes the value.

Another way to determine the optimal is Elbow Method.

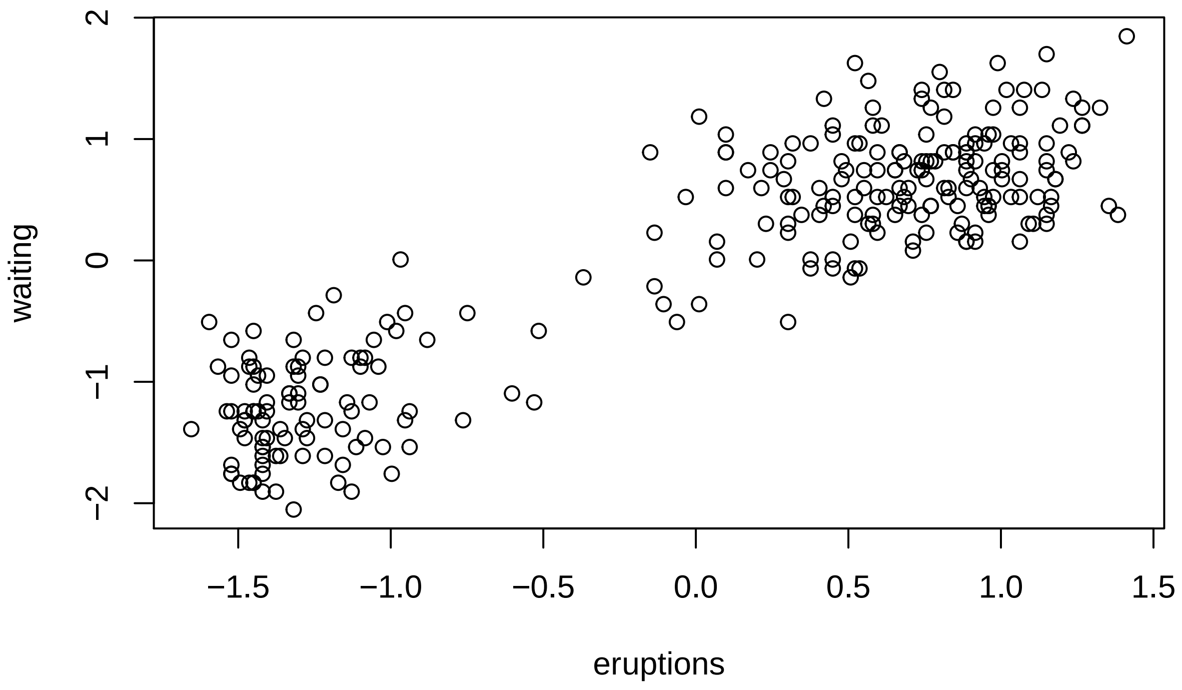
```r
fviz_nbclust(df, FUN = hcut, method = "wss")
```



The Elbow Method computes how well the clusters is grouped together. The smaller the number means that the variance in each group is smaller. The value decrease abruply at 2 clusters.

Plot data so that groups can be seen, since there is only 2 dimension scatter plot will be used

```r
plot(df,type = 'p')
```



The

plot shows that the data can be seperated by 2 groups

All of the above method suggest that the suitable number of clusters is 2

K-means clustering

K-mean with 2 clusters. The algorithm will be run 10 times randomly choosing different centroids, the best one will be kept.

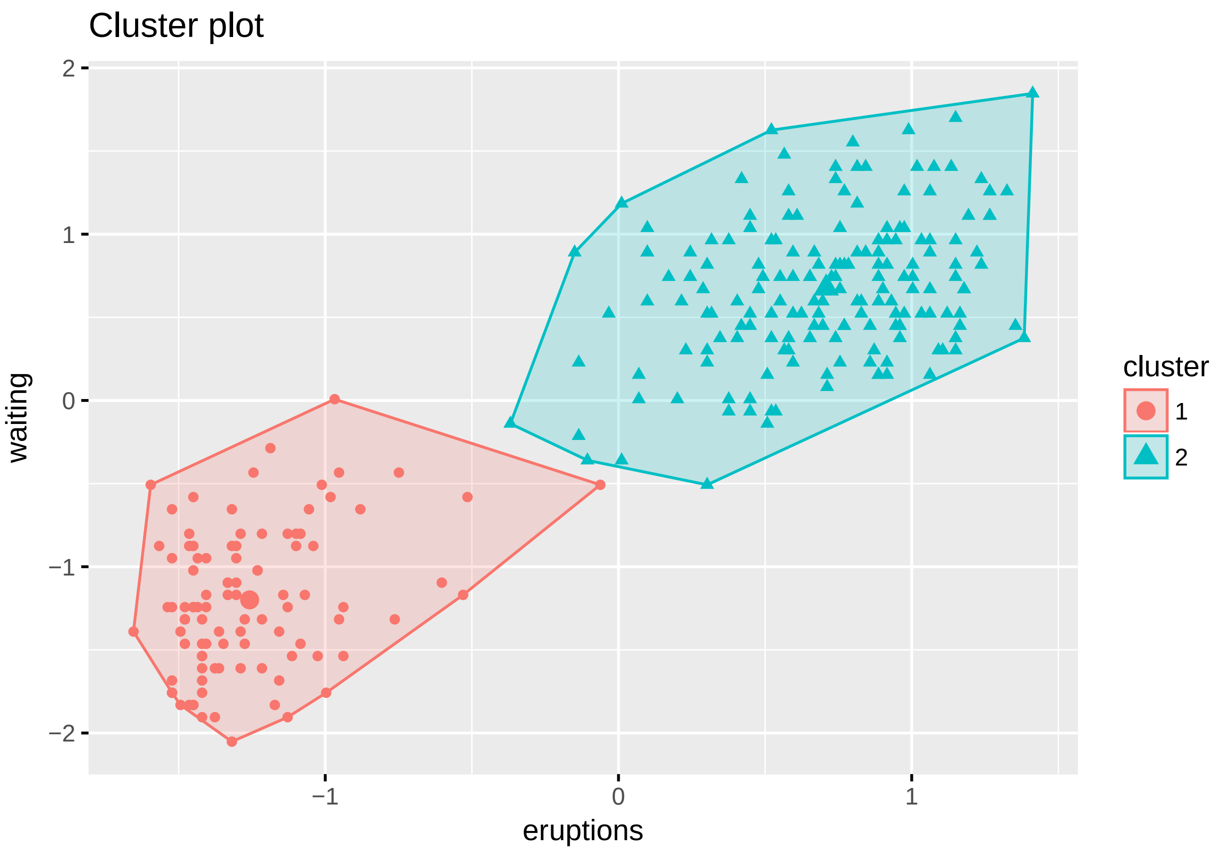
```
km = kmeans(df,centers = 2,nstart = 10)
km
```

```
## K-means clustering with 2 clusters of sizes 98, 174
##
## Cluster means:
##    eruptions    waiting
## 1 -1.2577669 -1.1993566
## 2  0.7083975  0.6754997
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##   2   1   2   1   2   1   2   2   1   2   1   2   2   1   2   1   1   2
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##   1   2   1   1   2   2   2   2   1   2   2   2   2   2   2   2   2   1
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##   1   2   1   2   2   1   2   1   2   2   2   1   2   1   2   2   1   2
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##   1   2   2   1   2   2   1   2   1   2   1   2   2   2   1   2   2   1
##  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##   2   2   1   2   1   2   2   2   2   2   2   1   2   2   2   2   1   2
##  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
##   1   2   1   2   1   2   2   2   1   2   1   2   1   2   2   1   2   1
## 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
##   2   2   2   1   2   2   1   2   1   2   1   2   1   2   2   1   2   2
## 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
##   1   2   1   2   1   2   1   2   1   2   1   2   1   2   2   1   2   2
## 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
##   2   1   2   1   2   1   2   2   1   2   2   2   2   2   1   2   1   2
## 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##   1   2   2   2   1   2   1   2   1   1   2   2   2   2   2   1   2   2
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
##   1   2   2   2   1   2   2   1   2   1   2   1   2   2   2   2   2   2
## 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
##   1   2   1   2   2   1   2   1   2   2   1   2   1   2   1   2   1   2
## 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234
##   1   2   1   2   1   2   1   2   2   2   2   2   2   2   2   1   2   1
## 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252
##   2   1   1   2   2   1   2   1   2   1   2   2   1   2   1   2   1   2
## 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
##   2   2   2   2   2   2   1   2   2   2   1   2   1   1   2   2   1   2
## 271 272
##   1   2
##
## Within cluster sum of squares by cluster:
```

```
## [1] 24.89206 54.39134
##  (between_SS / total_SS =  85.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

The data can be plot as follows:

```
fviz_cluster(list(data = df, cluster = km$cluster),labelsize = 0)
```



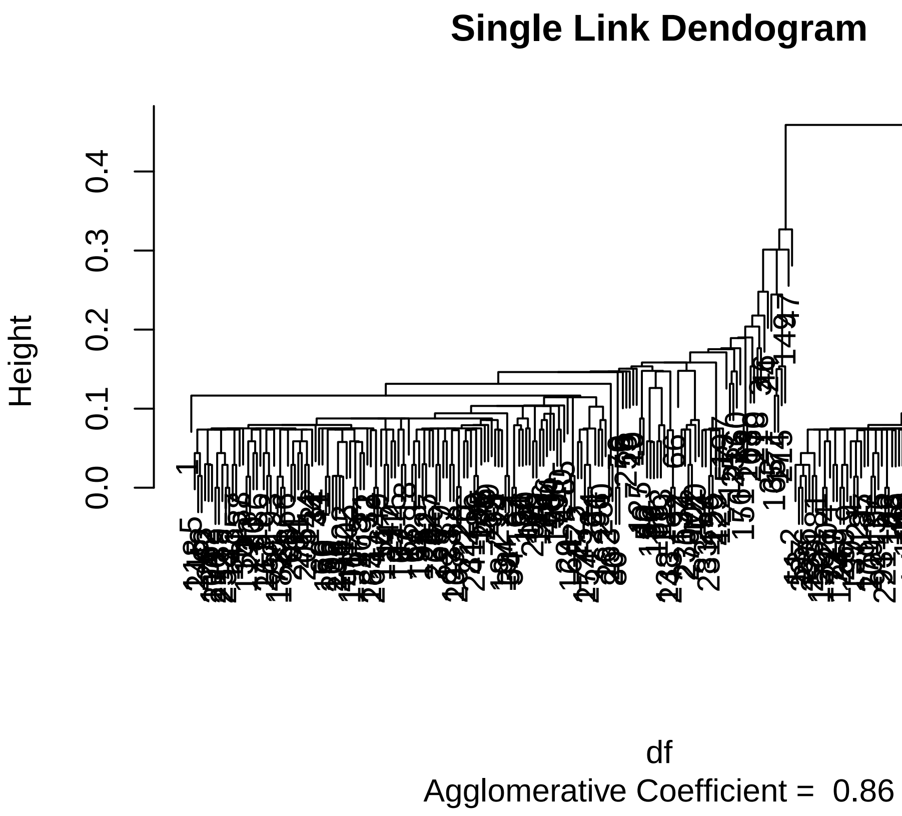
Hierarchical clustering

Euclidean distance will be used as the distance metric for all of the clustering

Single Link

The dendogram of Hierarchical clustering with single link can be shown by:

```
clusSingle = agnes(df,method = 'single',metric = "euclidean")
plot(clusSingle,which.plot = 2,main = "Single Link Dendogram")
```



Agglomerative coefficient using single link

```
sl.ac = clusSingle$ac
sl.ac
```

```
## [1] 0.8593234
```

Cophenetic Correlation Coefficient using single link

```
copSingle = cophenetic(clusSingle)
sl.ccc = cor(dist(df,"euclidean") ,copSingle)
sl.ccc
```

```
## [1] 0.91519
```

Average silhouette coefficient using single link

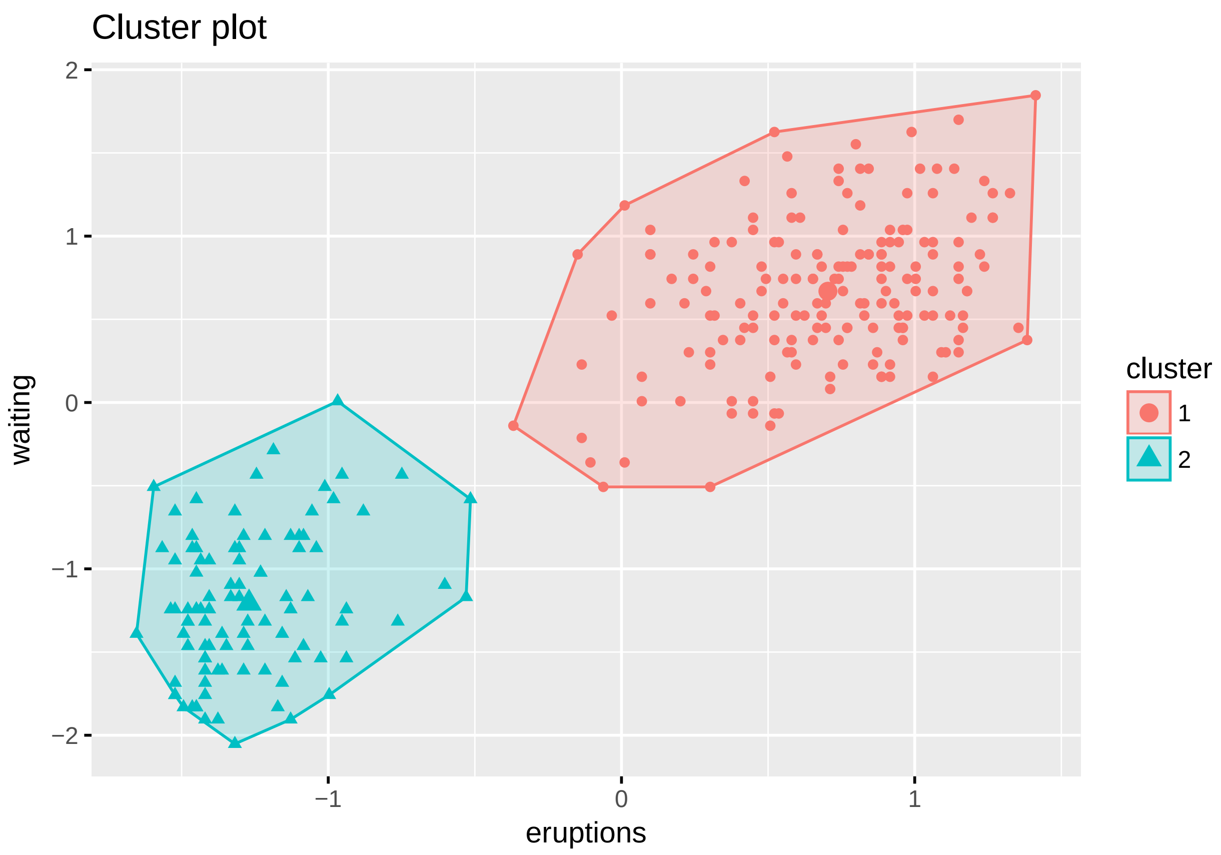
```
silSL = silhouette(cutree(clusSingle,2),dist(df))
sl.avs = mean(silSL[,3])
```

2 Clusters from Single Link method

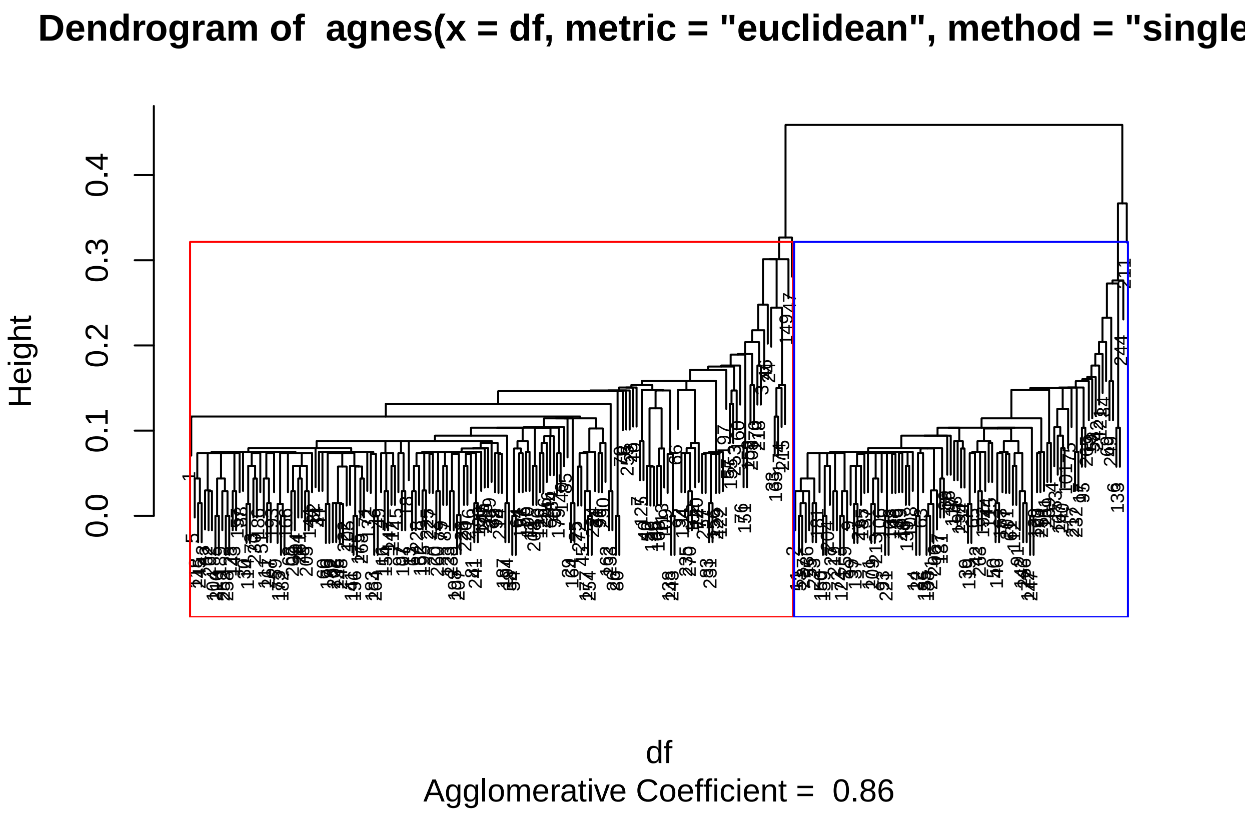
```
clusGroupSL = cutree(clusSingle,k = 2)
table(clusGroupSL)
```

```
## clusGroupSL
##   1   2
## 175  97
```

```
fviz_cluster(list(data = df, cluster = clusGroupSL),labelsize = 0)
```



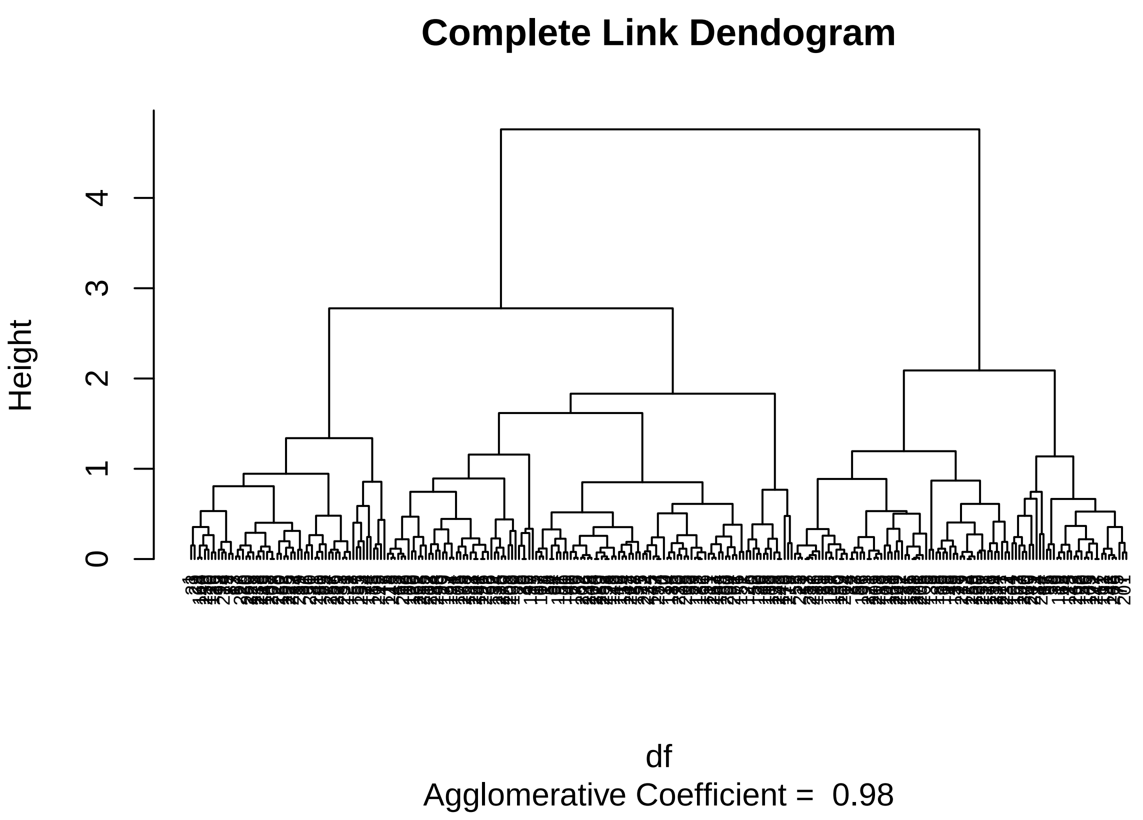
```r
plot(clusSingle,which.plot = 2, cex = 0.6)
rect.hclust(clusSingle, k = 2, border = c(2,4))
```



Complete Link

The dendogram of Hierarchical clustering with complete link can be shown by:

```r
clusComplete = agnes(df,method = 'complete',metric = "euclidean")
plot(clusComplete,which.plot = 2,main = "Complete Link Dendogram", cex = 0.6, hang = -1)
```



Agglomerative coefficient using complete link

```r
cl.ac = clusComplete$ac
```

Cophenetic Correlation Coefficient using complete link

```r
copComplete = cophenetic(clusComplete)
cl.ccc = cor(dist(df,"euclidean") ,copComplete)
cl.ccc
```

```
## [1] 0.8838317
```

Average silhouette coefficient using complete link

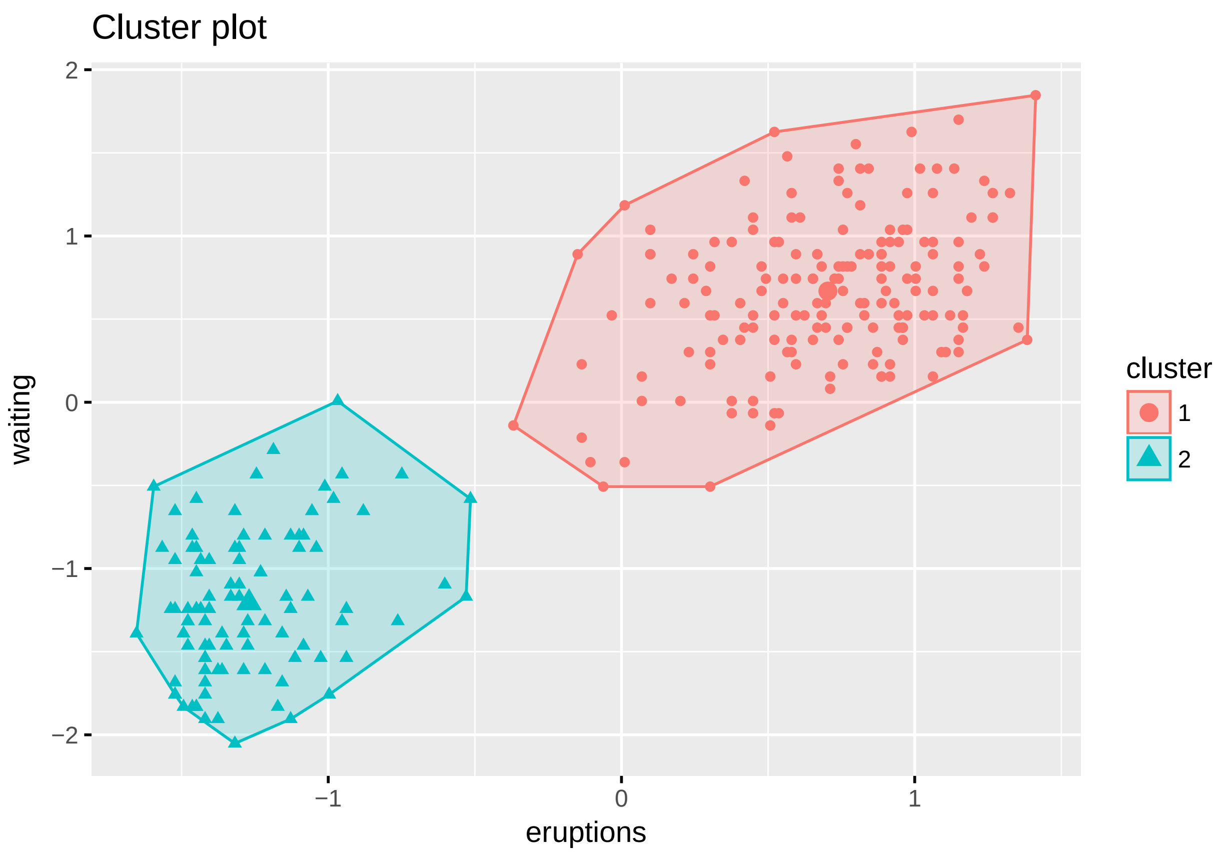
```r
silCL = silhouette(cutree(clusComplete,2),dist(df))
cl.avs= mean(silCL[,3])
```

2 Clusters from Complete Link method

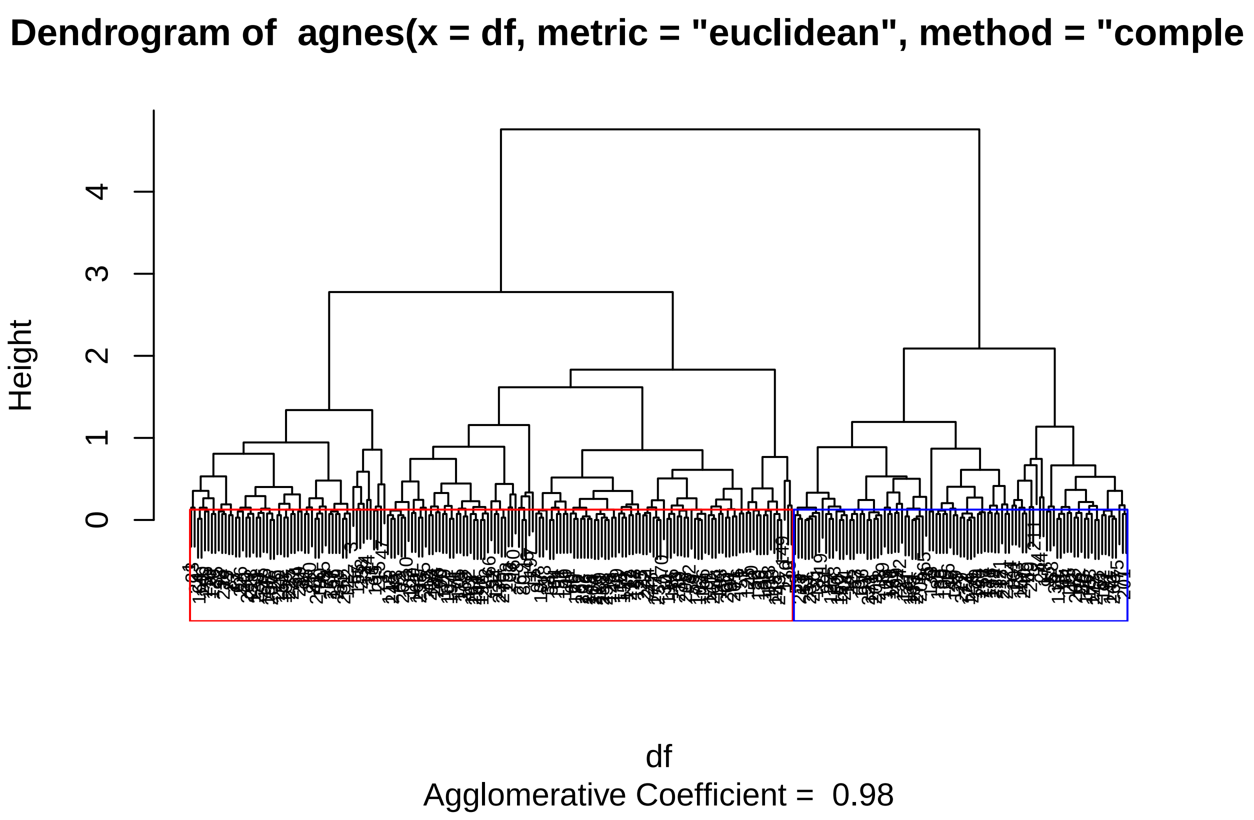
```r
clusGroupCL = cutree(clusComplete,k = 2)
table(clusGroupCL)
```

```
## clusGroupCL
##   1   2
## 175  97
```

```r
fviz_cluster(list(data = df, cluster = clusGroupCL),labelsize = 0)
```



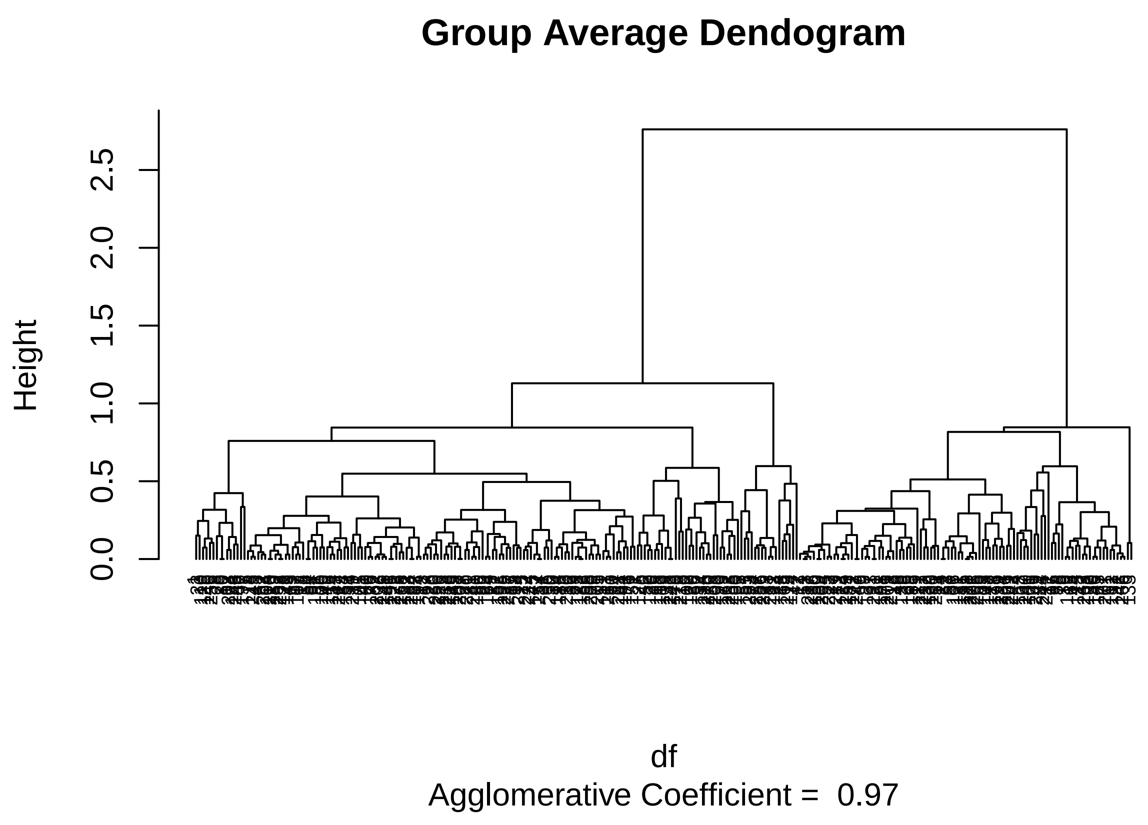
```
plot(clusComplete,which.plot = 2, cex = 0.6)
rect.hclust(clusComplete, k = 2, border = c(2,4))
```



Group Average

The dendogram of Hierarchical clustering with group average can be shown by:

```r
clusAVG = agnes(df,method = 'average',metric = "euclidean")
plot(clusAVG,which.plot = 2,main = "Group Average Dendogram", cex = 0.6, hang = -1)
```



Agglomerative coefficient using group average

```
ga.ac = clusAVG$ac
ga.ac
```

```
## [1] 0.9734022
```

Cophenetic Correlation Coefficient using group average

```
copAVG = cophenetic(clusAVG)
ga.ccc = cor(dist(df,"euclidean") ,copAVG)
ga.ccc
```

```
## [1] 0.9207053
```

Average silhouette coefficient using group average

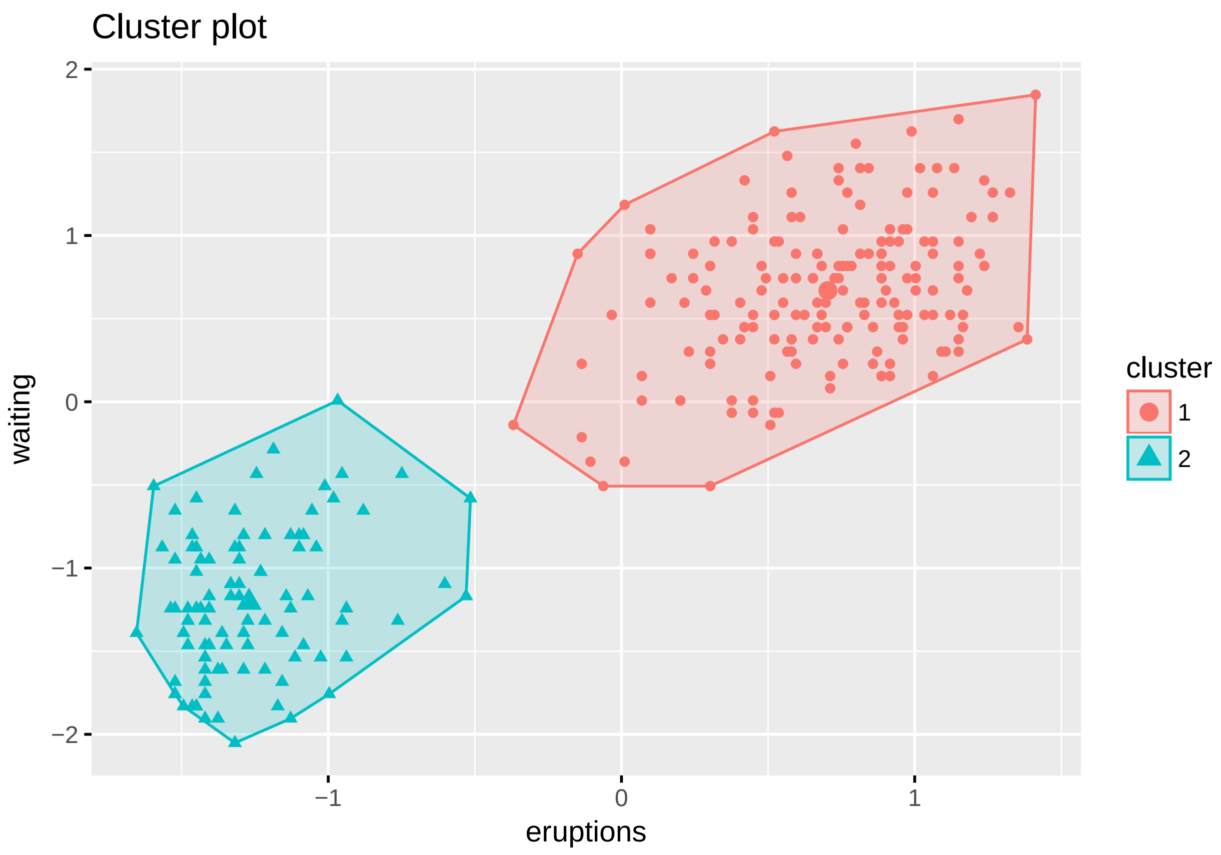
```
silAVG = silhouette(cutree(clusAVG,2),dist(df))
ga.avs= mean(silAVG[,3])
```

2 Clusters from group average method

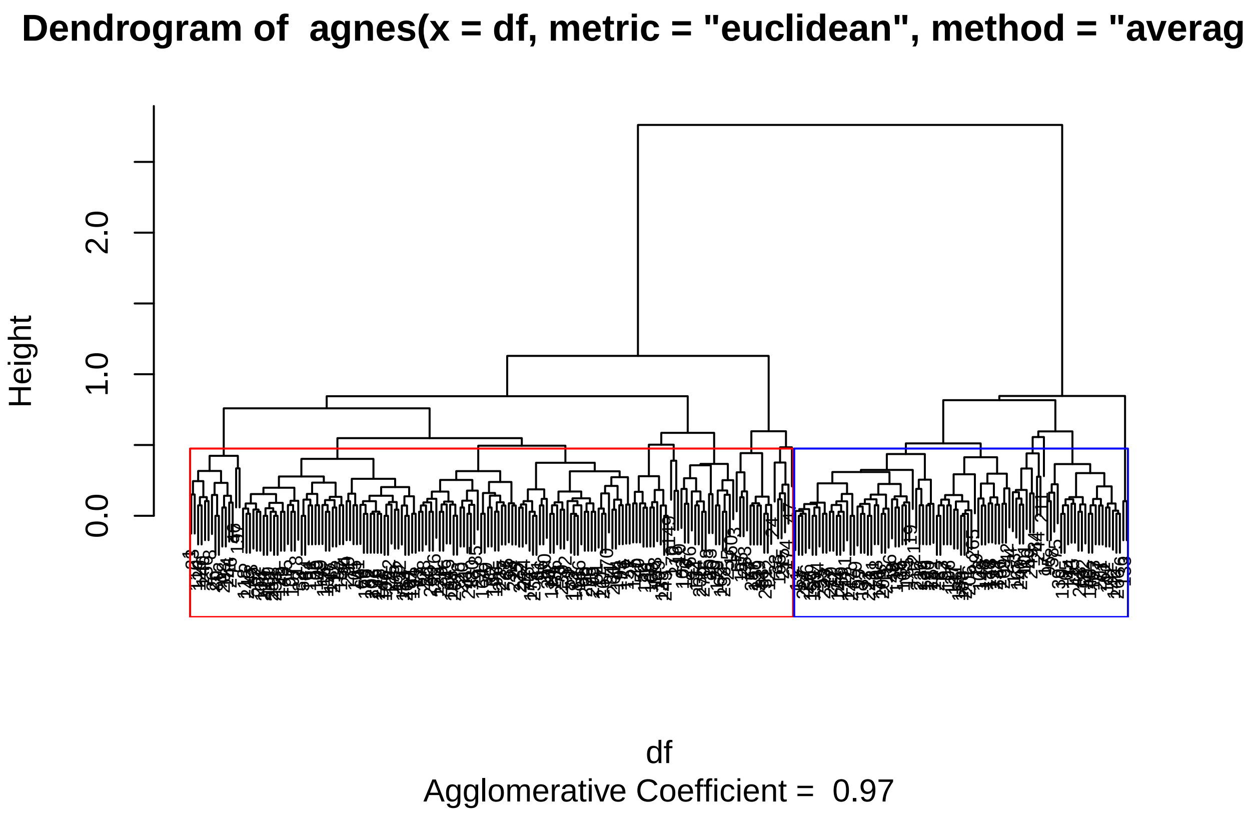
```
clusGroupGA = cutree(clusAVG,k = 2)
table(clusGroupGA)
```

```
## clusGroupGA
##   1   2
## 175  97
```

```
fviz_cluster(list(data = df, cluster = clusGroupGA),labelsize = 0)
```



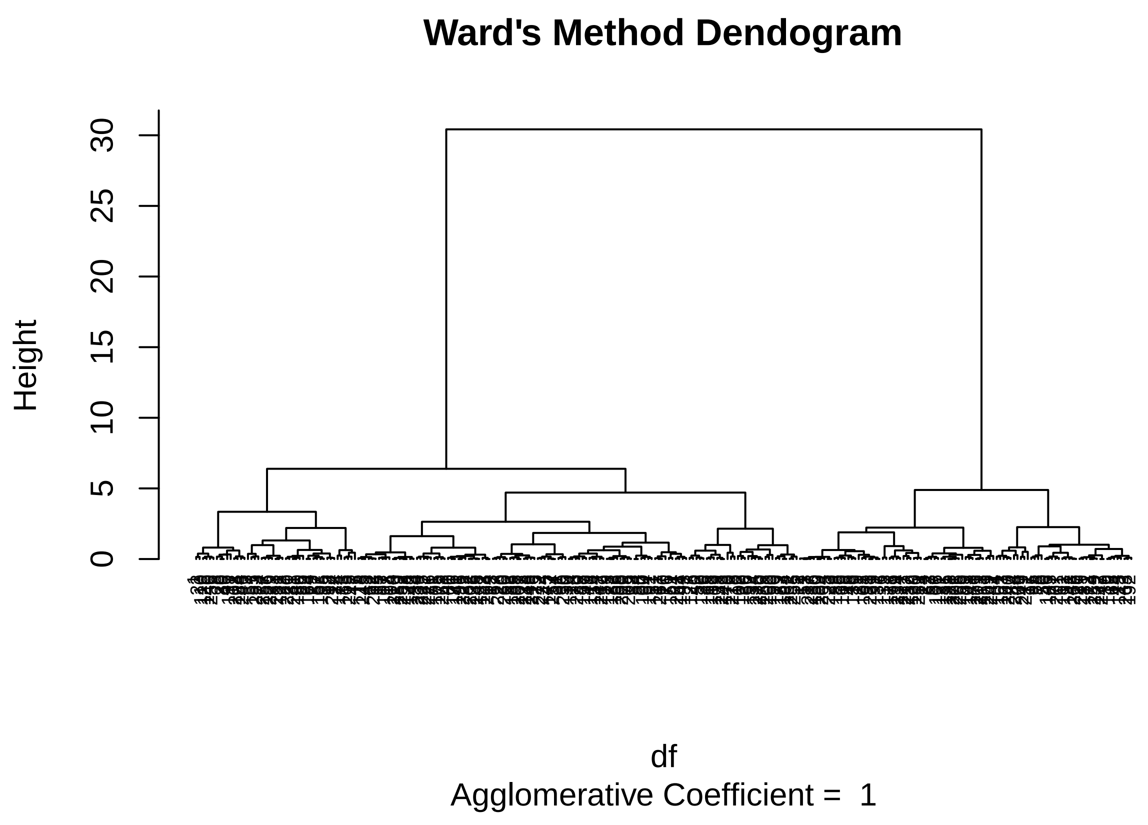
```
plot(clusAVG,which.plot = 2, cex = 0.6)
rect.hclust(clusAVG, k = 2, border = c(2,4))
```



Ward’s Method

The dendogram of Hierarchical clustering with group average can be shown by:

```
clusWard = agnes(df,method = 'ward',metric = "euclidean")
plot(clusWard,which.plot = 2,main = "Ward's Method Dendogram", cex = 0.6, hang = -1)
```



Agglomerative coefficient using Ward's method

```
wm.ac = clusWard$ac
wm.ac
```

```
## [1] 0.9975117
```

Cophenetic Correlation Coefficient using Ward's method

```
copWard = cophenetic(clusWard)
wm.ccc = cor(dist(df,"euclidean") ,copWard)
wm.ccc
```

```
## [1] 0.915404
```

Average silhouette coefficient using Ward's method

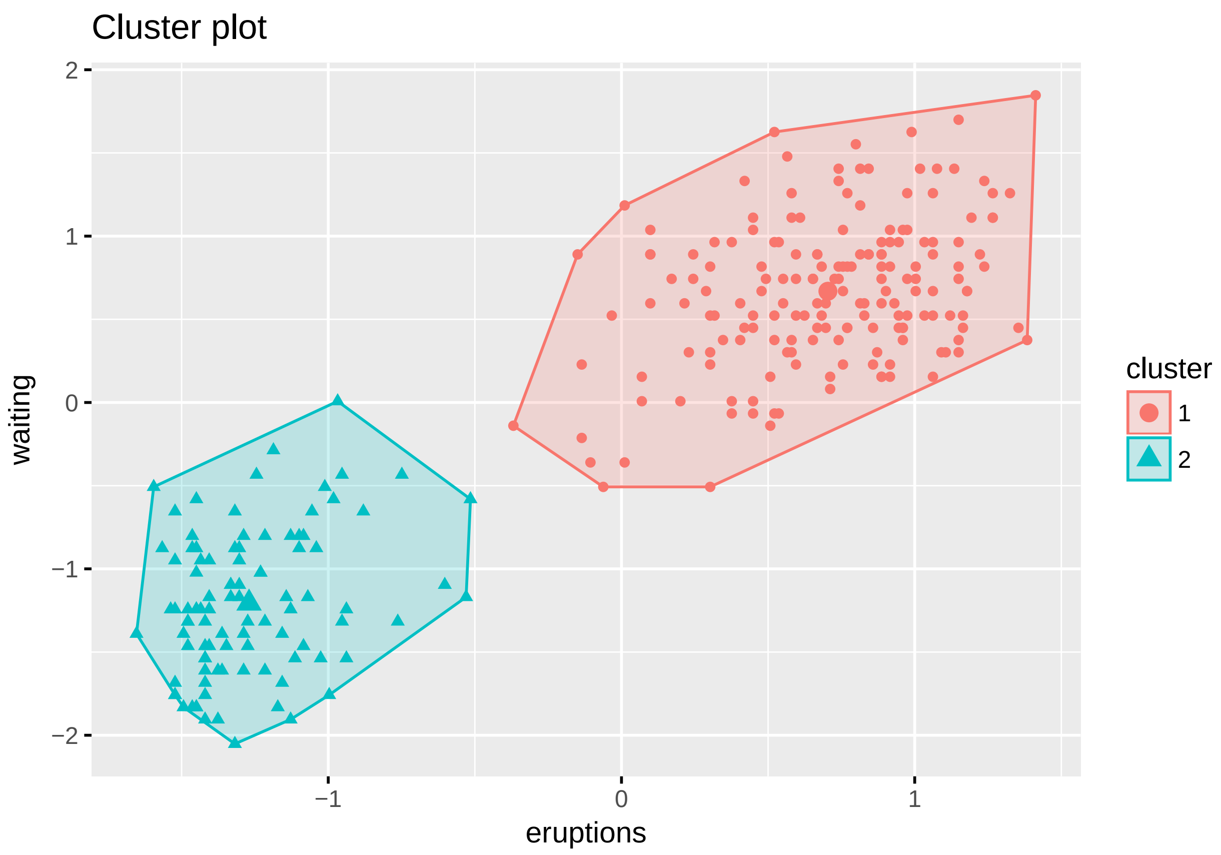
```
silWM = silhouette(cutree(clusWard,2),dist(df))
wm.avs= mean(silWM[,3])
```

2 Clusters from Ward's method

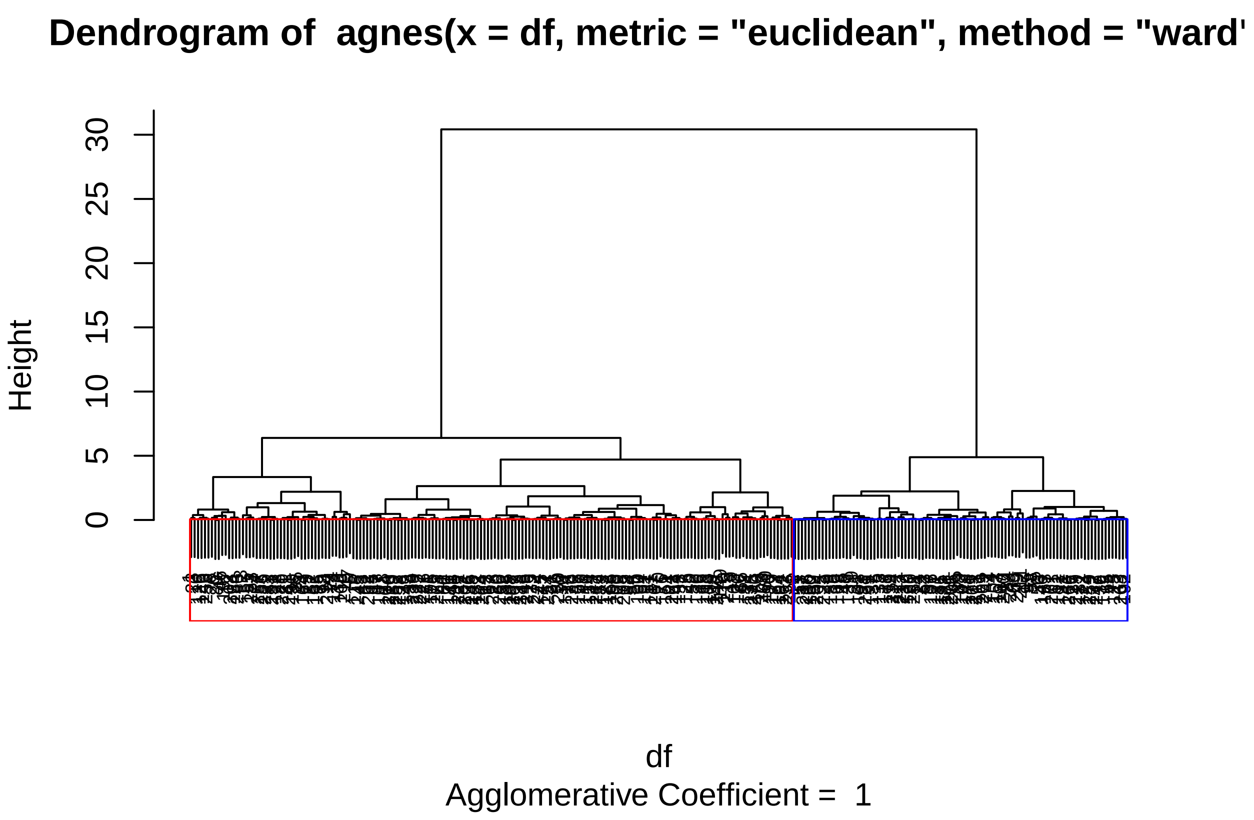
```
clusGroupWM = cutree(clusWard,k = 2)
table(clusGroupWM)
```

```
## clusGroupWM
##   1   2
## 175  97
```

```
fviz_cluster(list(data = df, cluster = clusGroupWM),labelsize = 0)
```



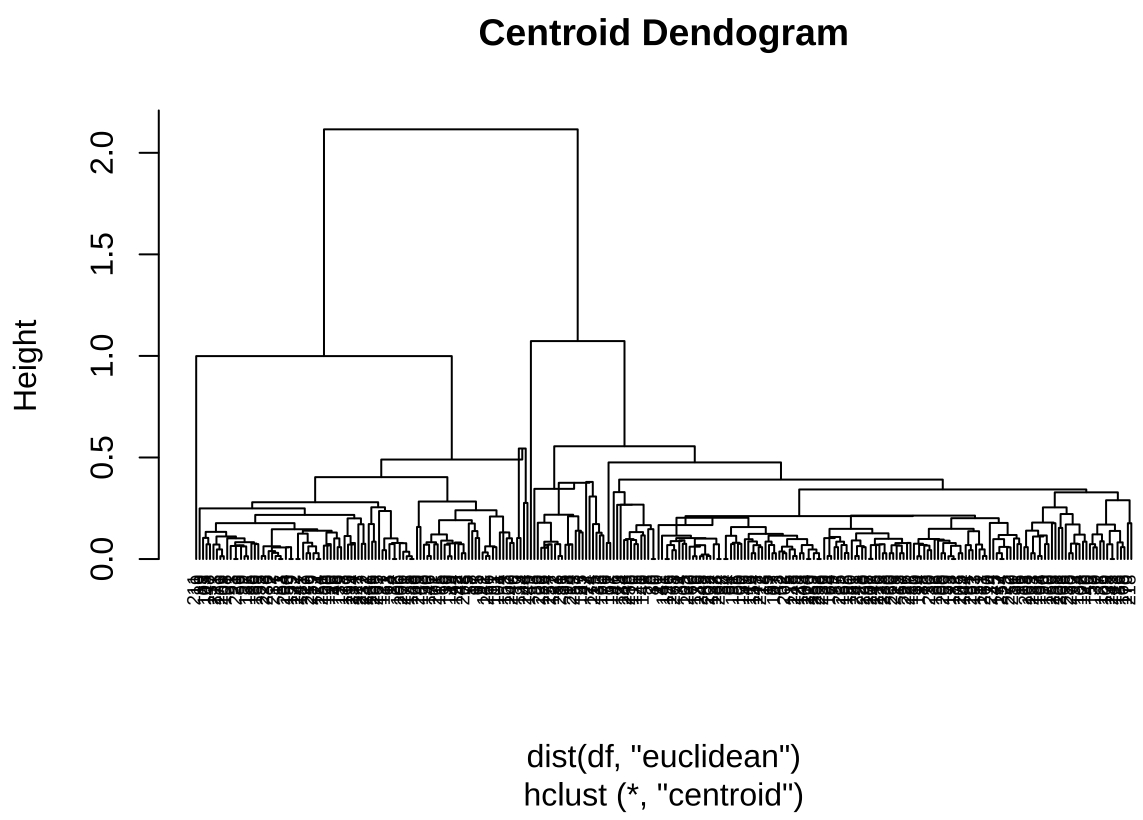
```r
plot(clusWard,which.plot = 2, cex = 0.6)
rect.hclust(clusWard, k = 2, border = c(2,4))
```



Centroid

The dendogram of Hierarchical clustering with centroid can be shown by:

```r
clusCentroid <- hclust(dist(df,"euclidean"), method = "centroid" )
plot(clusCentroid,main = "Centroid Dendogram", cex = 0.6, hang = -1)
```



Agglomerative coefficient using centroid

```
cen.ac = coefHier(clusCentroid)
cen.ac
```

```
## [1] 0.9445889
```

Cophenetic Correlation Coefficient using centroid

```
copCentroid = cophenetic(clusCentroid)
cen.ccc = cor(dist(df,"euclidean") ,copCentroid)
cen.ccc
```

```
## [1] 0.9182128
```

Average silhouette coefficient using centroid

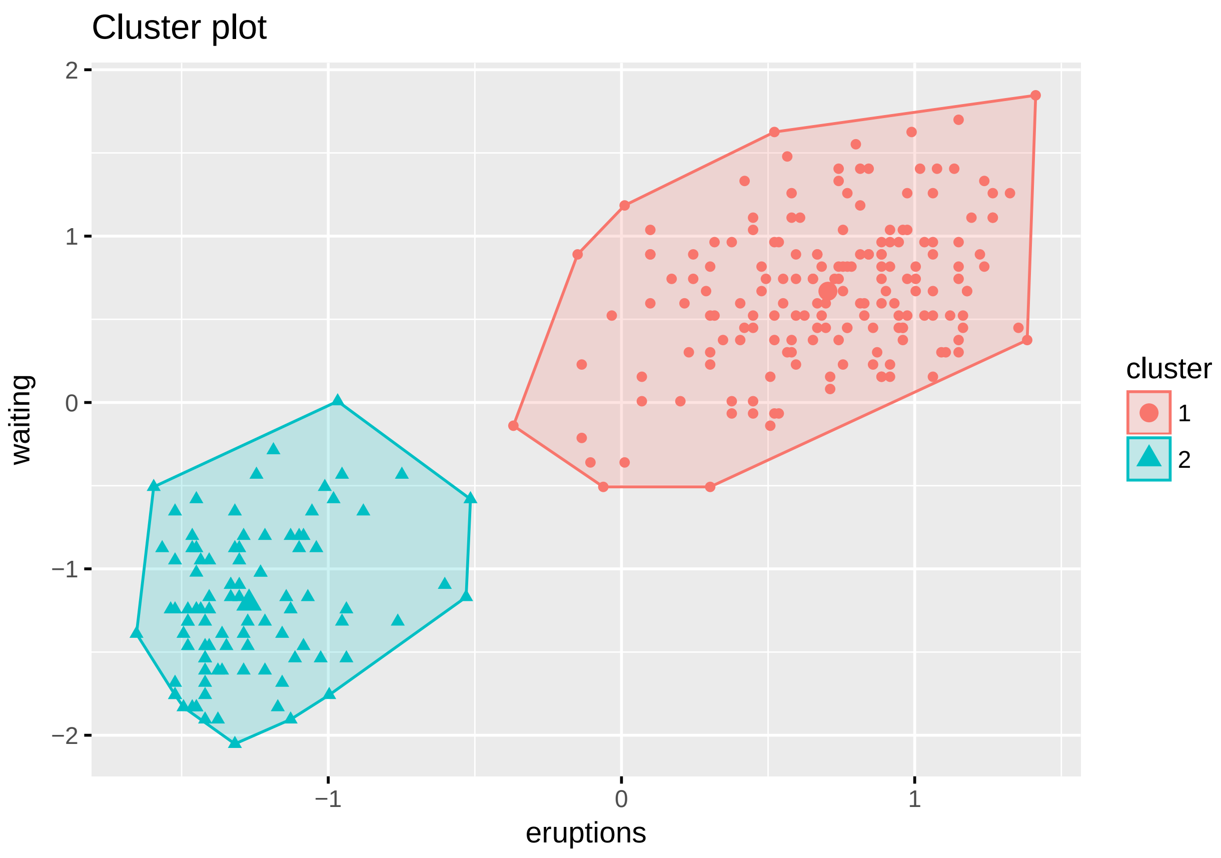
```
silCEN = silhouette(cutree(clusCentroid,2),dist(df))
cen.avs= mean(silCEN[,3])
```

2 Clusters from centroid method

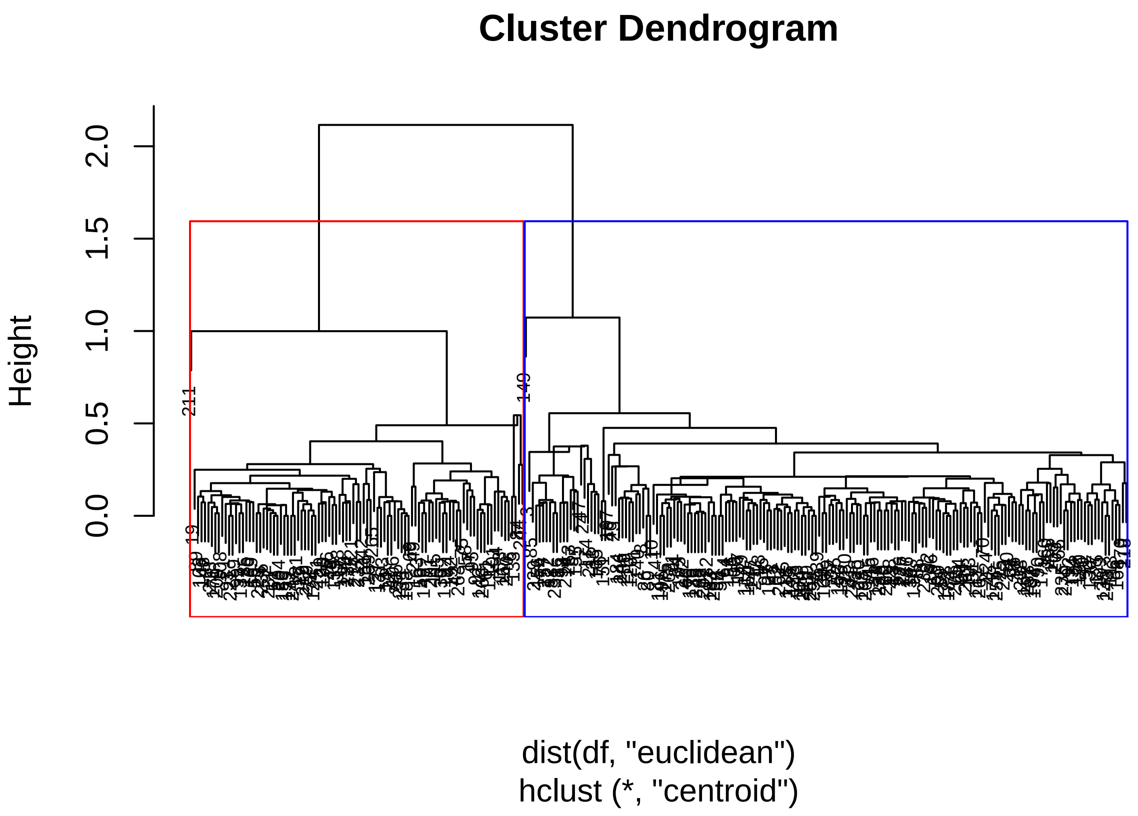
```
clusGroupCEN = cutree(clusCentroid,k = 2)
table(clusGroupCEN)
```

```
## clusGroupCEN
##   1   2
## 175  97
```

```
fviz_cluster(list(data = df, cluster = clusGroupCEN),labelsize = 0)
```



```r
plot(clusCentroid, cex = 0.6)
rect.hclust(clusCentroid, k = 2, border = c(2,4))
```



Comparing different methods

```
result = cbind(c("single","complete","group","ward","centroid"),c(sl.ac,cl.ac,ga.ac,wm.ac,cen.ac),c(sl.

colnames(result) = c('Method','Agglomerative','Cophenetic Correlation',"Silhouette")
result
```

```
##      Method     Agglomerative       Cophenetic Correlation
## [1,] "single"   "0.85932336504865"  "0.915189986436428"
## [2,] "complete" "0.984171810674116" "0.88383168817658"
## [3,] "group"    "0.973402218910261" "0.920705331748472"
## [4,] "ward"     "0.997511676748446" "0.915404044170347"
## [5,] "centroid" "0.944588945385072" "0.918212830625851"
##      Silhouette
## [1,] "0.746002489669941"
## [2,] "0.746002489669941"
## [3,] "0.746002489669941"
## [4,] "0.746002489669941"
## [5,] "0.746002489669941"
```

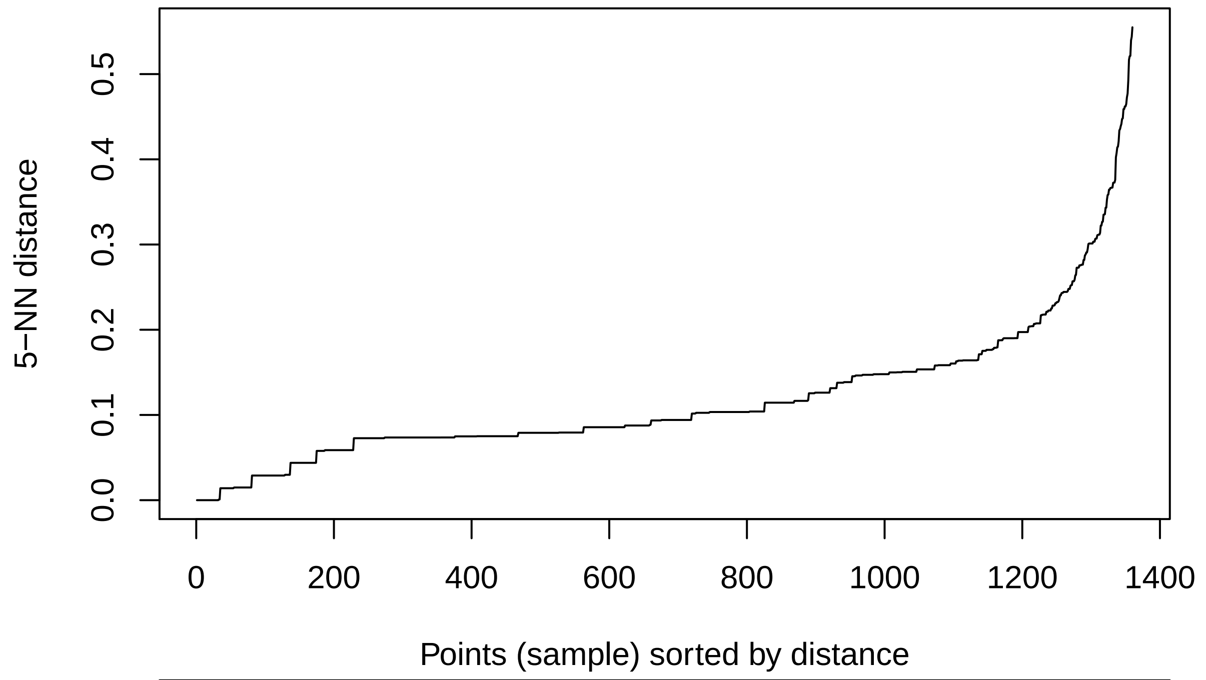
The Average Silhouette coefficient of all points are the same because all of the algorithm group data in the same manner. The cophenetic correllation coefficient is used to evalute the hierarchical clusters, The higher the value the better. Using group average as the linking criteria maximizes the value.

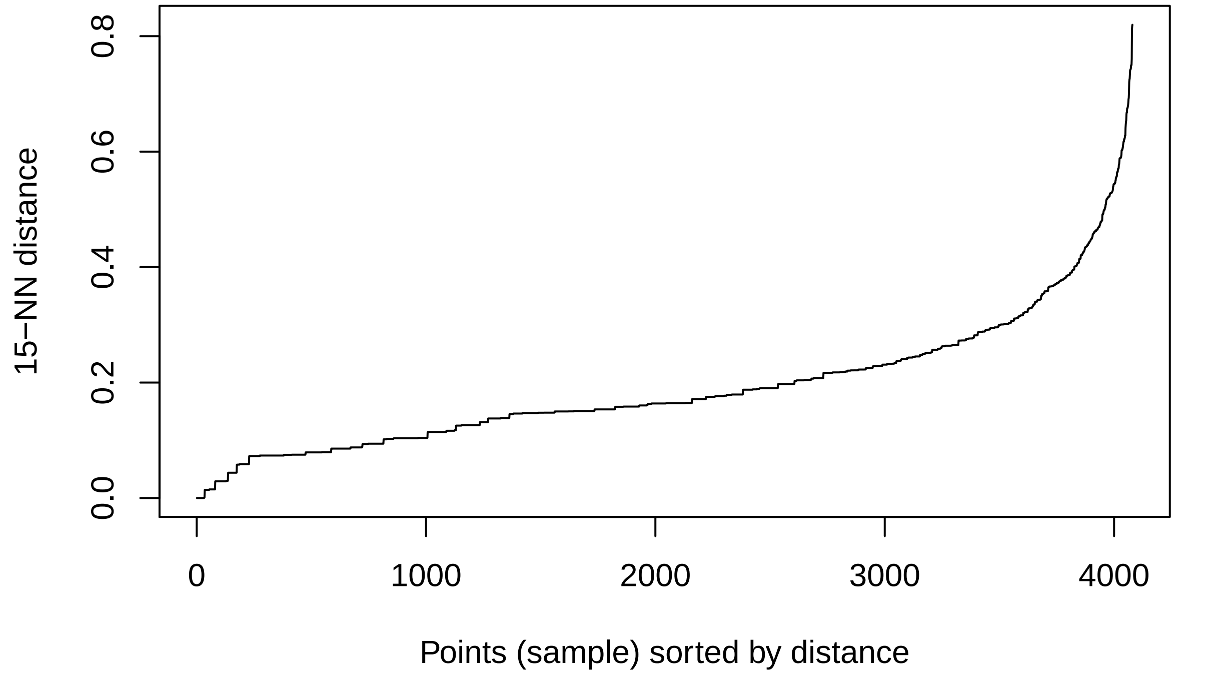
Agglomerative coefficient shows the strenght of the cluster. Ward's method as the linkage criteria result in the highest Aggolomerative coeeficient.

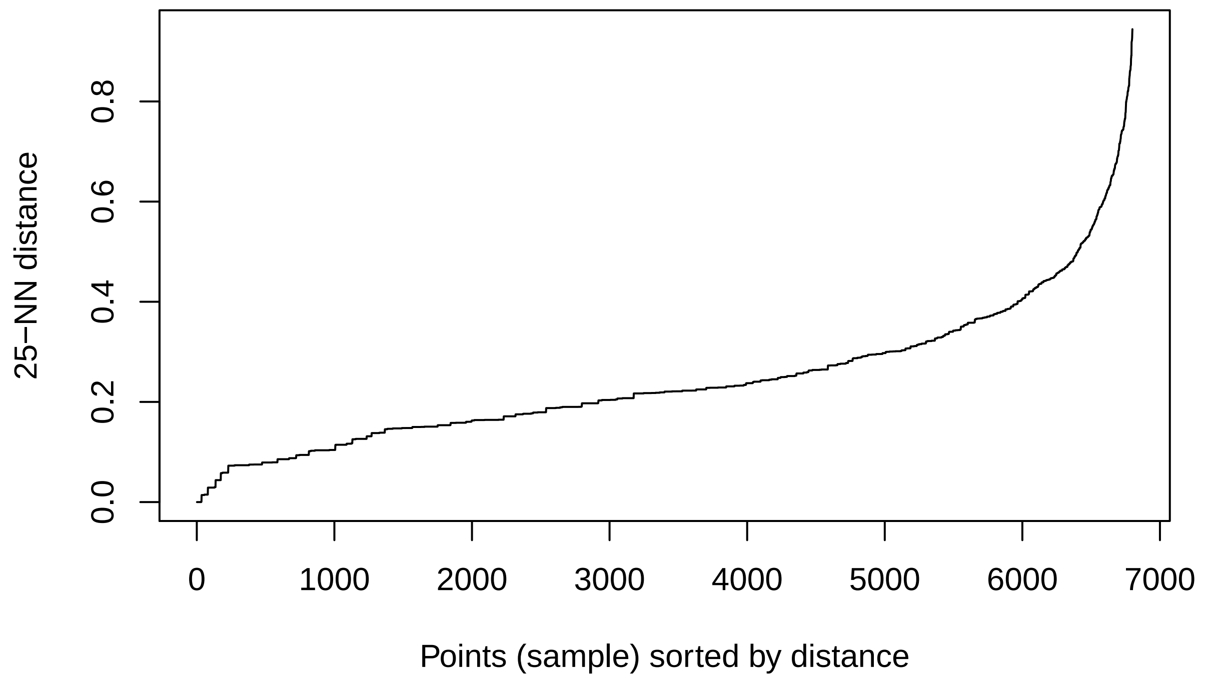
DBSCAN

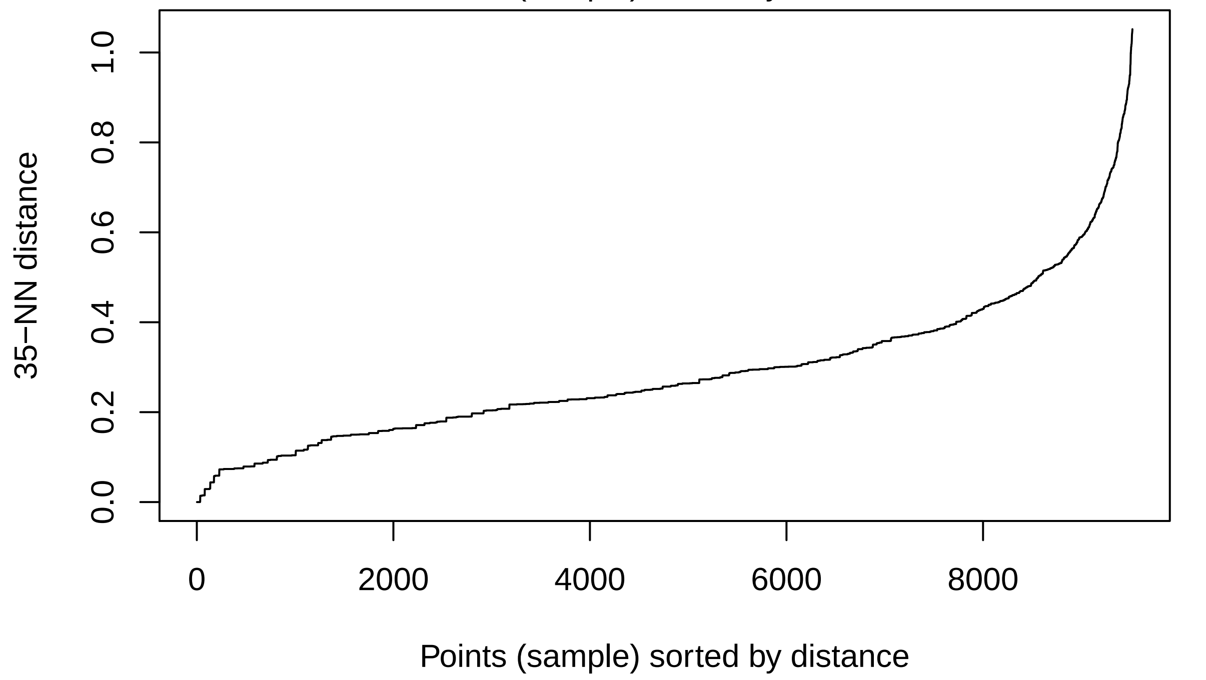
Plot K Nearest Neighbor

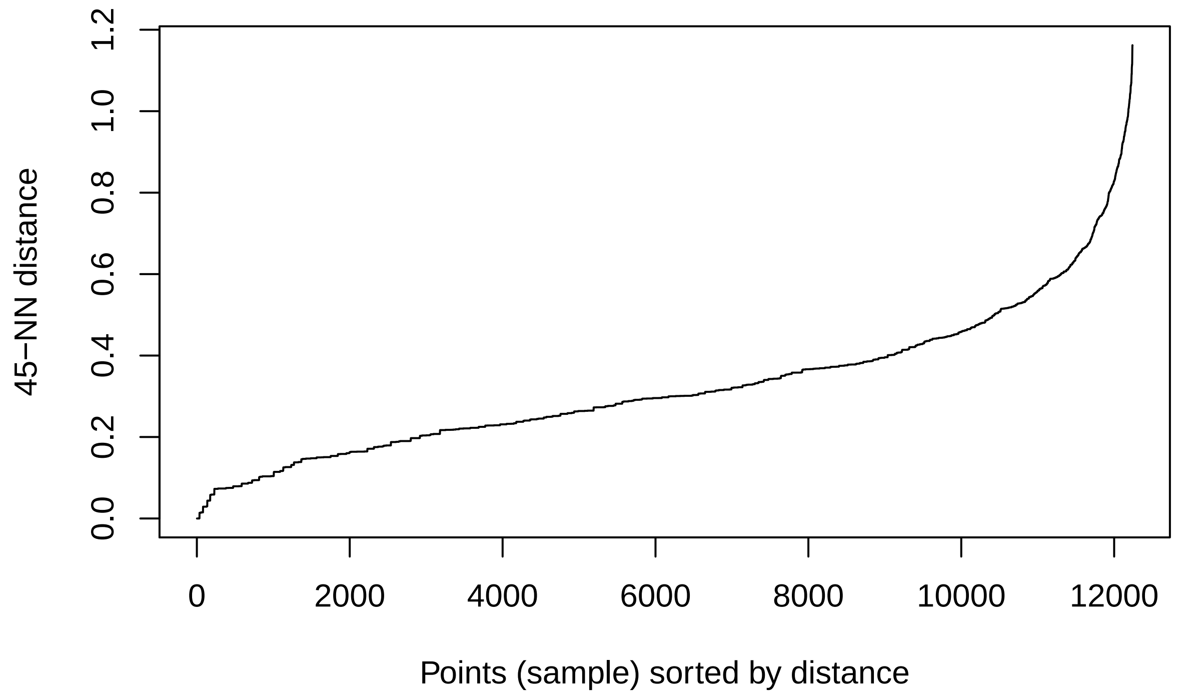
```
kSeq = seq(5,100,10)

for (minK in kSeq){
  kNNdist(df, k=minK, search="kd")
  kNNdistplot(df, k=minK)
}
```



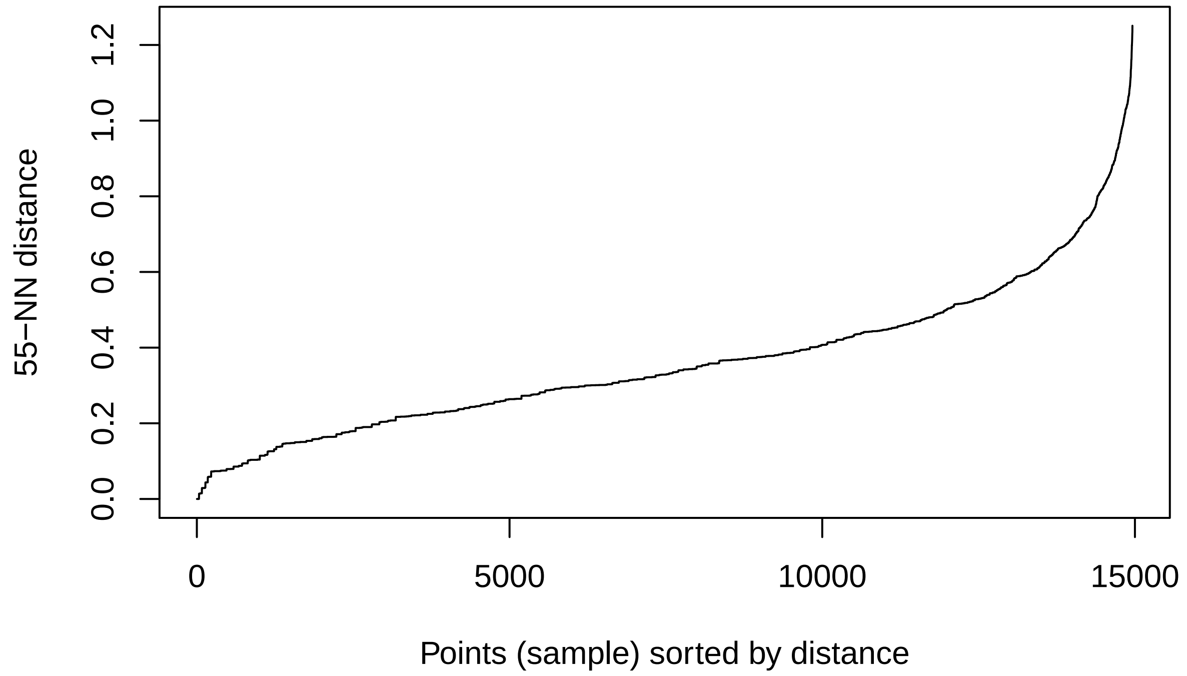



25−NN distance
0.0
0.2
0.4
0.6
0.8
0
1000
2000
3000
4000
5000
6000
7000
Points (sample) sorted by distance

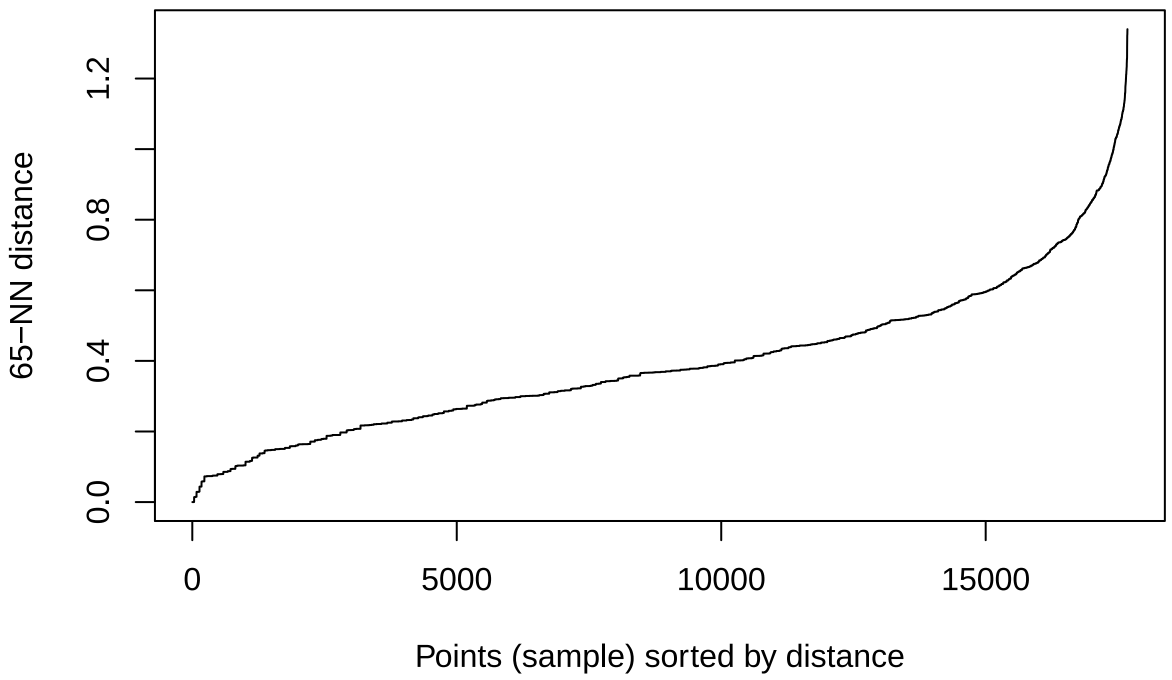

35−NN distance
0.0
0.2
0.4
0.6
0.8
1.0
0
2000
4000
6000
8000
Points (sample) sorted by distance

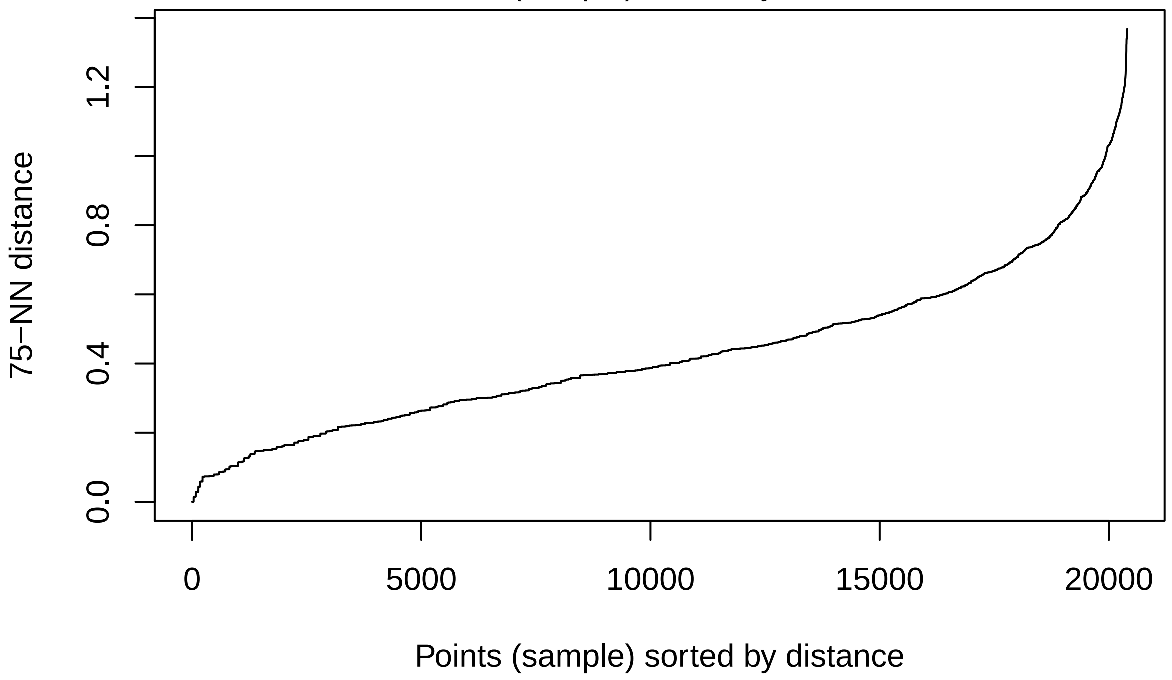


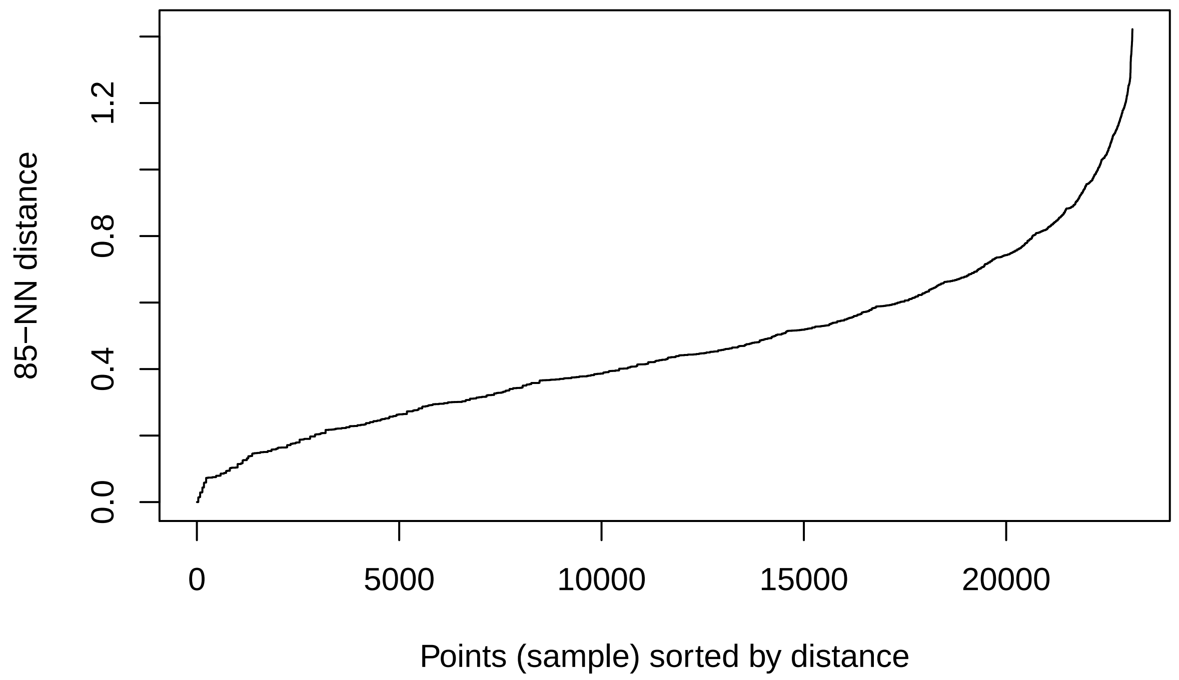
45−NN distance
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
2000
4000
6000
8000
10000
12000
Points (sample) sorted by distance

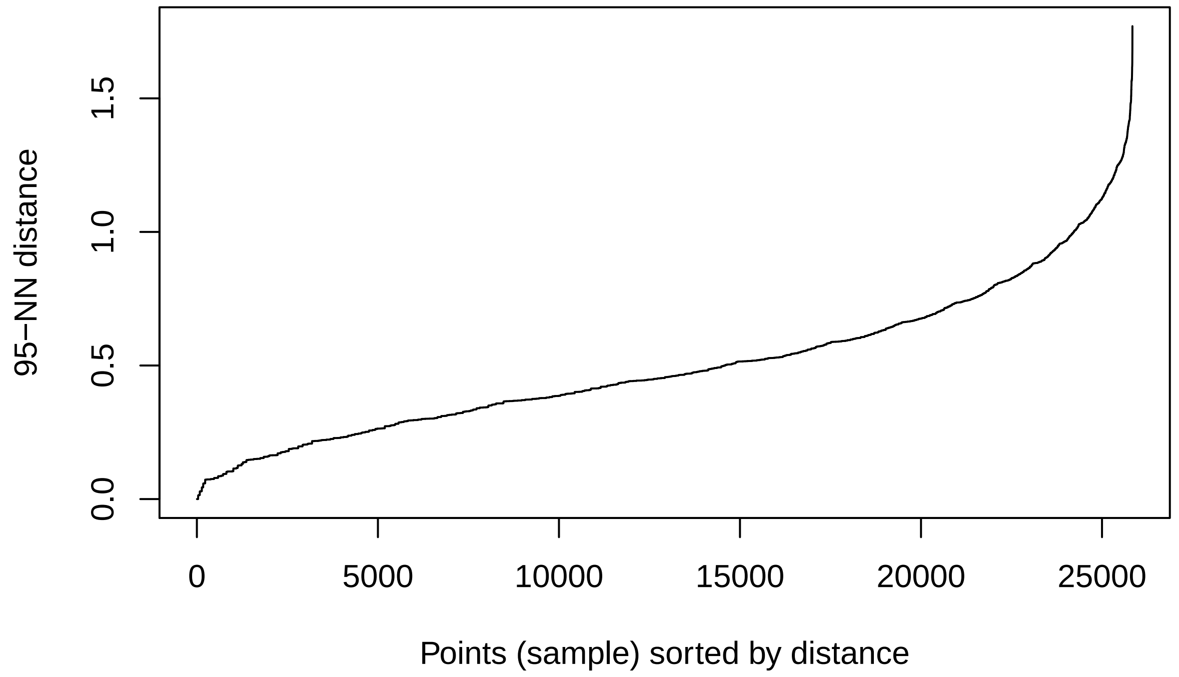


55−NN distance
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
5000
10000
15000
Points (sample) sorted by distance


65−NN distance
0.0
0.4
0.8
1.2
0
5000
10000
15000
Points (sample) sorted by distance


75−NN distance
0.0
0.4
0.8
1.2
0
5000
10000
15000
20000
Points (sample) sorted by distance





I have decided to select where K = 45 and the knee is around 0.5. So the DBSCAN will have the parameter of MinPts = 45 and Eps = 0.5

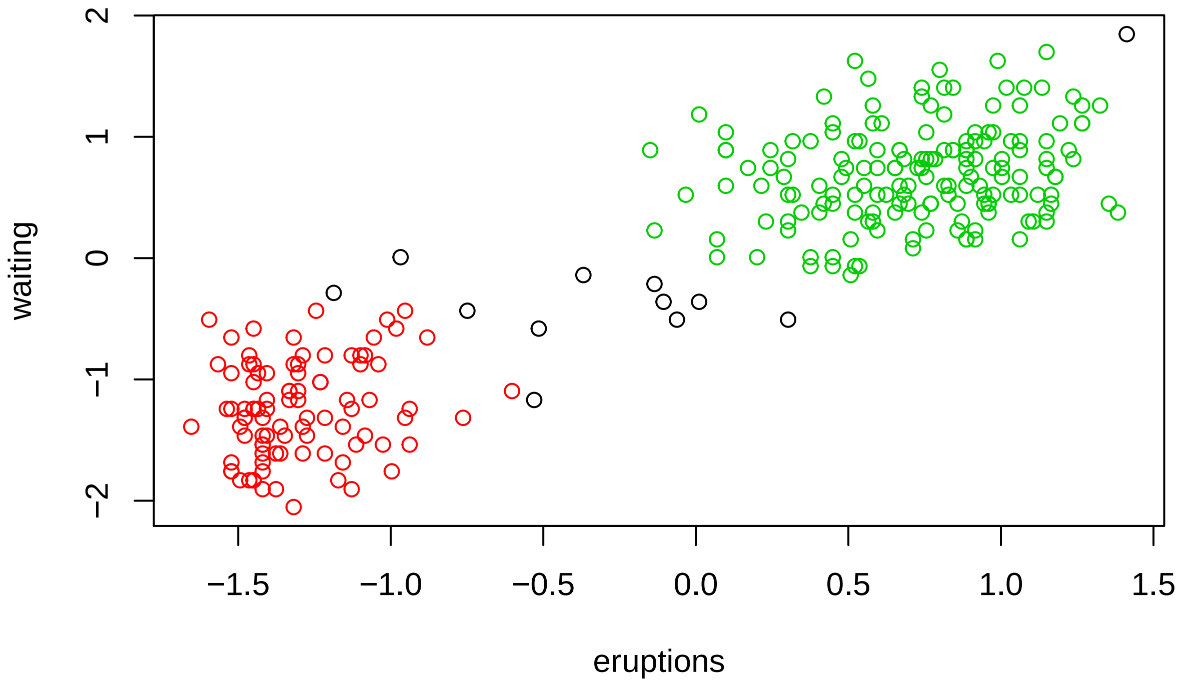
Create DBSCAN with above parameter

```
db = dbscan(df,eps = 0.5,MinPts = 45)
db
```

```
## dbscan Pts=272 MinPts=45 eps=0.5
##         0  1   2
## border 12 38  44
## seed    0 54 124
## total  12 92 168
```

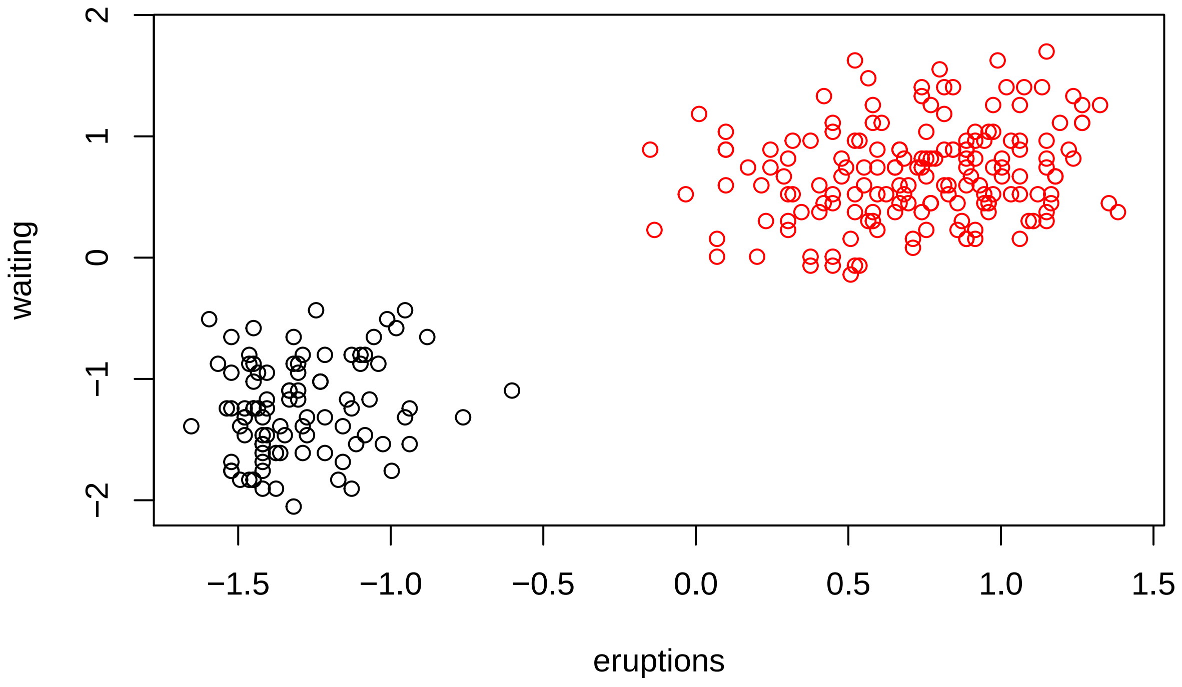
Plot result of clusters from DBScan. The document stated that outliers in the DBSCAN will be 0. The cluster values will be added 1 so that the outlier will be plot in black.

```r
plot(df, col = db$cluster + 1)
```



Plot without outliers

```r
plot(df, col = db$cluster)
```



Best Method For Faithful Dataset

Compare between DBSCAN, K-Mean and Hierachical Clustering (Group Average and Ward's Method)

Compare each method using various evaluation measures

```
results = c('average.between','average.within','avg.silwidth','within.cluster.ss')

statSum = cbind(cluster.stats(dist(df), clusGroupGA)[results],
cluster.stats(dist(df), clusGroupWM)[results],
cluster.stats(dist(df), db$cluster+1)[results],
cluster.stats(dist(df), -(db$cluster+1),noisecluster=TRUE)[results],
cluster.stats(dist(df), km$cluster)[results]
)
```

```
## Warning in cluster.stats(dist(df), -(db$cluster + 1), noisecluster = TRUE):
## clustering renumbered because maximum != number of clusters
```

```
colnames(statSum)  = c('Group Average','Ward\'s Method',"DBSCAN with Outlier","DBSCAN without Outlier",

t(statSum)
```

```
##                        average.between average.within avg.silwidth
## Group Average          2.761157        0.6784107      0.7460025
## Ward's Method          2.761157        0.6784107      0.7460025
## DBSCAN with Outlier    2.633671        0.6255499      0.5819516
## DBSCAN without Outlier 2.840008        0.6239707      0.600442
## K-Mean                 2.755253        0.6753959      0.7451774
##                        within.cluster.ss
## Group Average          79.33622
## Ward's Method          79.33622
## DBSCAN with Outlier    72.78451
## DBSCAN without Outlier 62.15977
## K-Mean                 79.2834
```

A better cluster generally is compact with low variance in the group and the distance between each cluster is high relative to the variance.

Evaluation measures:

- average.between: average distance between clusters
- average.within: average distance within the clusters
- avg.silwidth: Average silhouette width
- within.cluster.ss: Within cluster's sum of squares

Looking at the values the best method is DBSCAN even with the outliers included. The measures seems to show that DBSCAN is the most suitable method for this problem. Since it creates the most compact cluster that is well seperated.