Homework 3

Naga Kartheek Peddisetty, 50538422

04/05/2024

Question 1) The “chorSub” data from the “cluster” package contains measurements of 10 chemicals in 61 geological samples from the Kola Peninsula. Cluster the data using k-means and hierarchical clustering. What is a good choice of “k” for each of these methods? Justify your selection.

```
library(cluster)
library(GGally)
```

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

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(ggplot2)
library(bootcluster)
```

```
## Warning: package 'bootcluster' was built under R version 4.3.3
```

```
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.3.3
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
set.seed(123)

data(chorSub)
class(chorSub)
```

```
## [1] "matrix" "array"
```

```
dim(chorSub)
```

```
## [1] 61 10
```

```
paste("Number of rows = ", dim(chorSub)[1])
```

```
## [1] "Number of rows =  61"
```

```
paste("Number of columns = ", dim(chorSub)[2])
```

```
## [1] "Number of columns =  10"
```

```
dats <- data.frame(chorSub)
class(dats)
```

```
## [1] "data.frame"
```

```
summary(dats)
```

```
##        Al                  Ca                  Fe                 K          
##  Min.   :-201.000   Min.   :-178.000   Min.   :-200.00   Min.   :-133.000  
##  1st Qu.: -49.000   1st Qu.: -59.000   1st Qu.: -75.00   1st Qu.: -74.000  
##  Median :  -1.000   Median :   2.000   Median : -37.00   Median : -17.000  
##  Mean   :   0.541   Mean   :  -2.066   Mean   : -14.64   Mean   :  -4.295  
##  3rd Qu.:  47.000   3rd Qu.:  59.000   3rd Qu.:  48.00   3rd Qu.:  51.000  
##  Max.   : 285.000   Max.   : 211.000   Max.   : 162.00   Max.   : 248.000  
##        Mg                 Mn                 Na                 P          
##  Min.   :-155.00   Min.   :-139.00   Min.   :-242.00   Min.   :-102.000  
##  1st Qu.: -75.00   1st Qu.: -66.00   1st Qu.: -25.00   1st Qu.: -61.000  
##  Median : -30.00   Median : -41.00   Median :  47.00   Median : -36.000  
##  Mean   : -13.05   Mean   : -13.92   Mean   :  17.97   Mean   :  -6.623  
##  3rd Qu.:  31.00   3rd Qu.:   7.00   3rd Qu.:  81.00   3rd Qu.:   8.000  
##  Max.   : 254.00   Max.   : 354.00   Max.   : 187.00   Max.   : 406.000  
##        Si                  Ti          
##  Min.   :-223.000   Min.   :-190.000  
##  1st Qu.: -46.000   1st Qu.: -73.000  
##  Median :  22.000   Median : -24.000  
##  Mean   :   8.328   Mean   :  -7.361  
##  3rd Qu.:  60.000   3rd Qu.:  55.000  
##  Max.   : 177.000   Max.   : 351.000
```

```
str(dats)
```

```
## 'data.frame':    61 obs. of  10 variables:
##  $ Al: int  101 50 5 -40 -13 -49 44 285 4 -48 ...
##  $ Ca: int  11 129 65 -16 30 -43 -109 183 -83 102 ...
##  $ Fe: int  -22 23 -22 -158 -82 31 -40 133 -66 105 ...
##  $ K : int  -17 -82 -96 -70 -113 -74 118 -6 24 -92 ...
##  $ Mg: int  -34 47 -33 -104 26 -2 -98 31 -44 146 ...
##  $ Mn: int  -41 33 7 -114 -41 33 33 -15 -66 83 ...
##  $ Na: int  27 61 47 53 65 -72 -100 121 -3 30 ...
##  $ P : int  -36 90 49 -61 -90 -36 103 -49 -90 -36 ...
##  $ Si: int  -58 -24 30 103 43 78 10 -223 22 -61 ...
##  $ Ti: int  -28 9 -39 -160 -130 -51 9 115 -47 118 ...
```

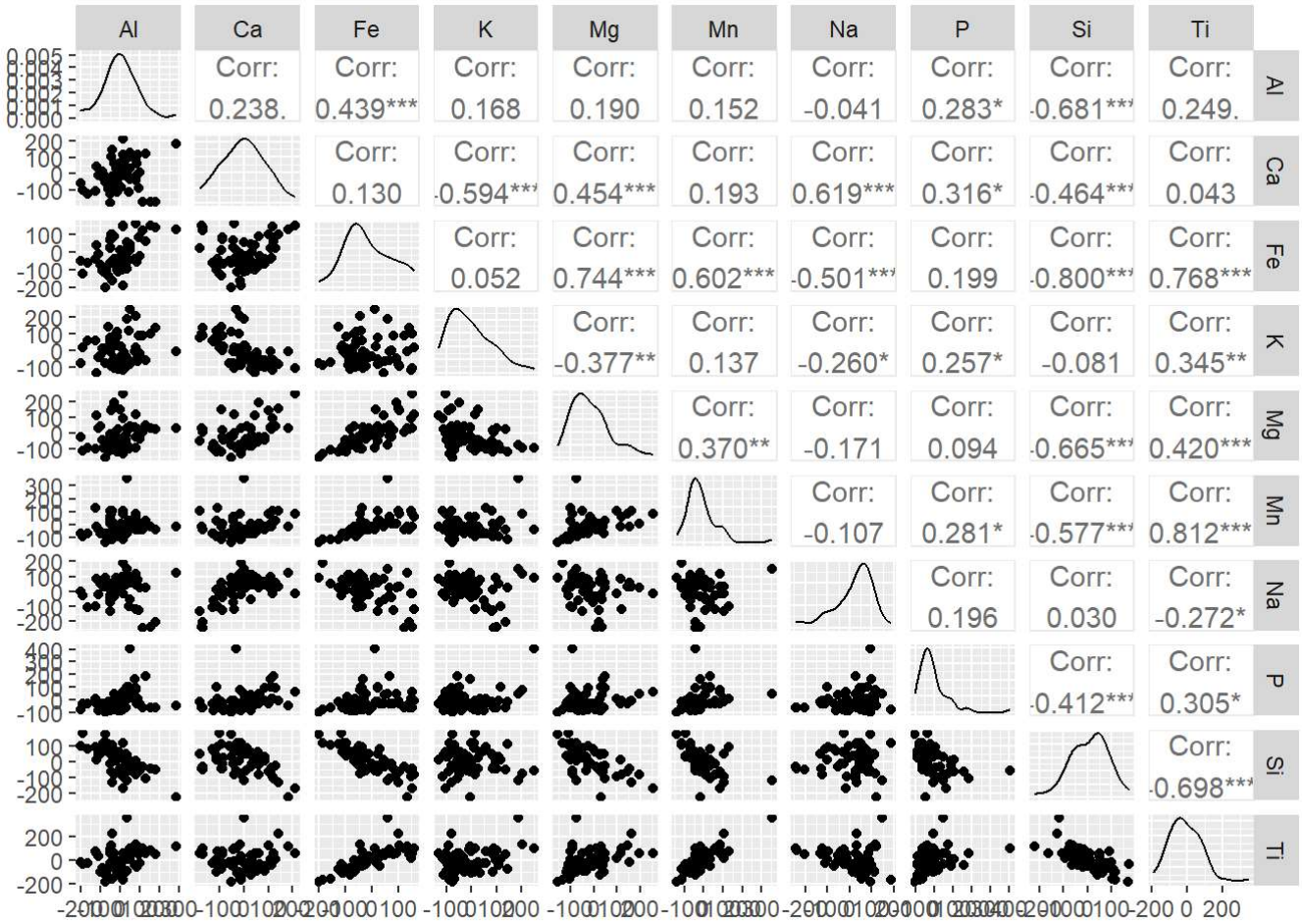
```
colSums(is.na(dats))
```

```
## Al Ca Fe  K Mg Mn Na  P Si Ti
##  0  0  0  0  0  0  0  0  0  0
```

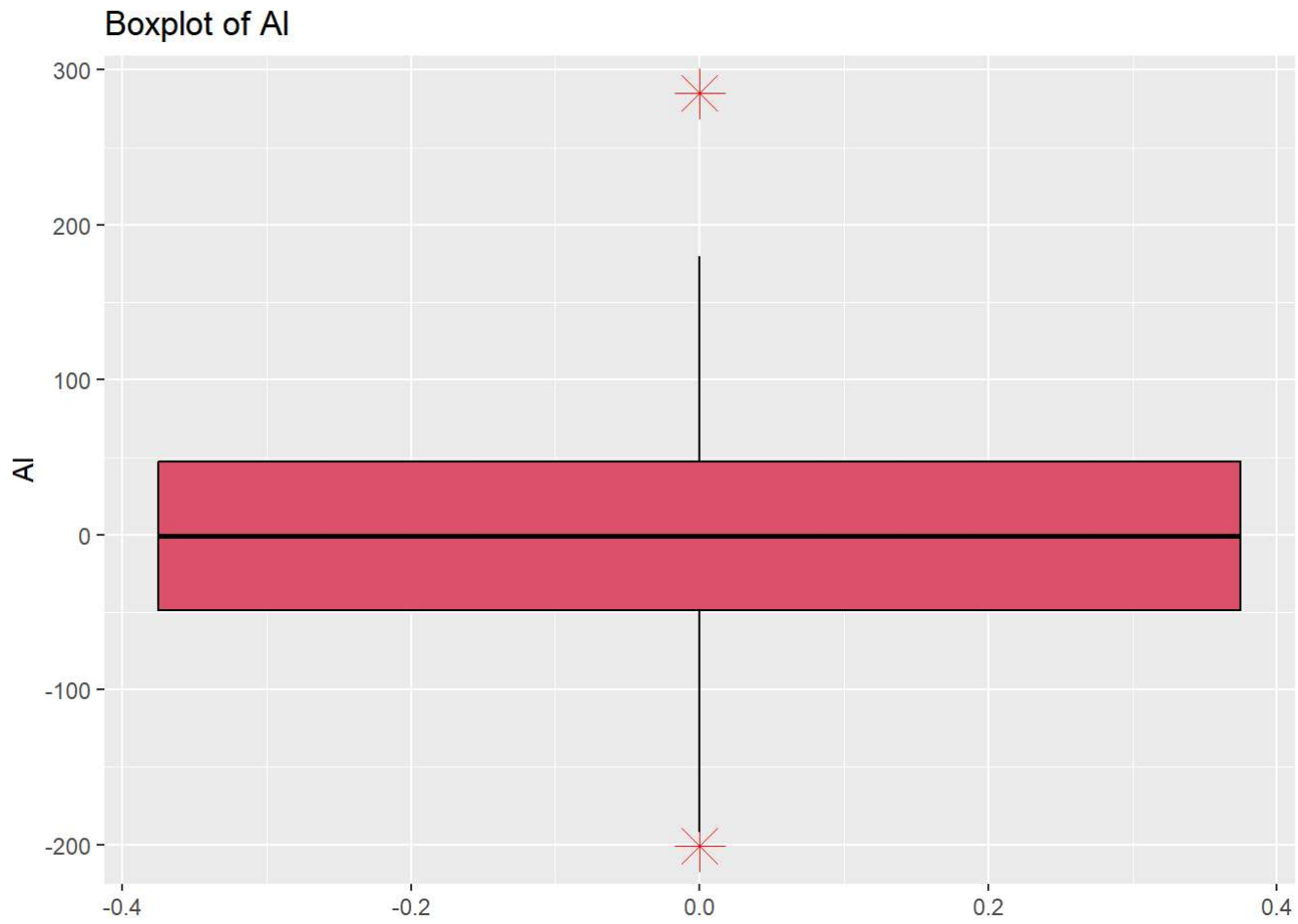
```
head(dats)
```

```
##      Al  Ca   Fe    K   Mg   Mn  Na   P  Si   Ti
## 190 101  11  -22  -17  -34  -41  27 -36 -58  -28
## 191  50 129   23  -82   47   33  61  90 -24    9
## 192   5  65  -22  -96  -33    7  47  49  30  -39
## 193 -40 -16 -158  -70 -104 -114  53 -61 103 -160
## 194 -13  30  -82 -113   26  -41  65 -90  43 -130
## 195 -49 -43   31  -74   -2   33 -72 -36  78  -51
```

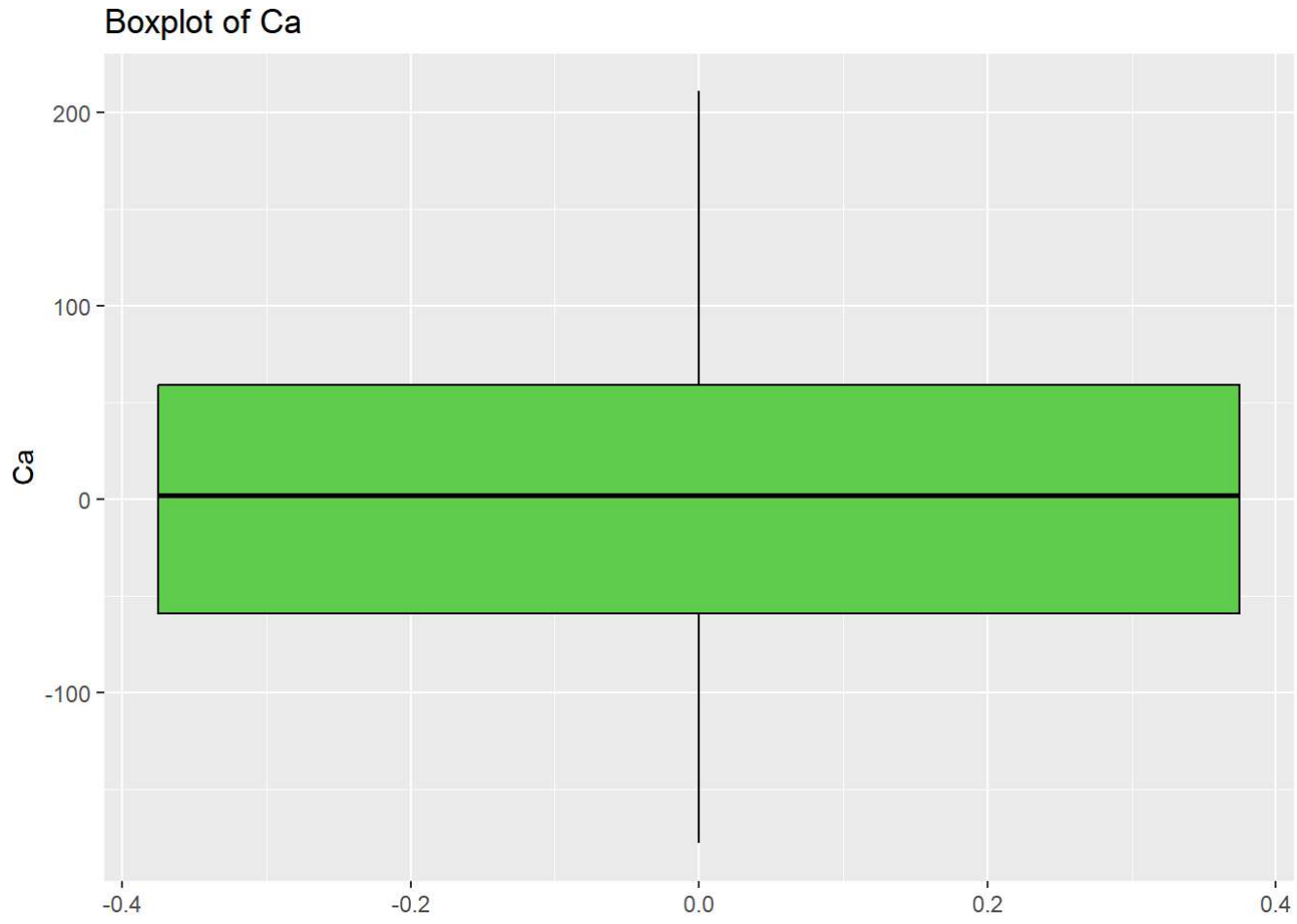
```
ggpairs(dats)
```



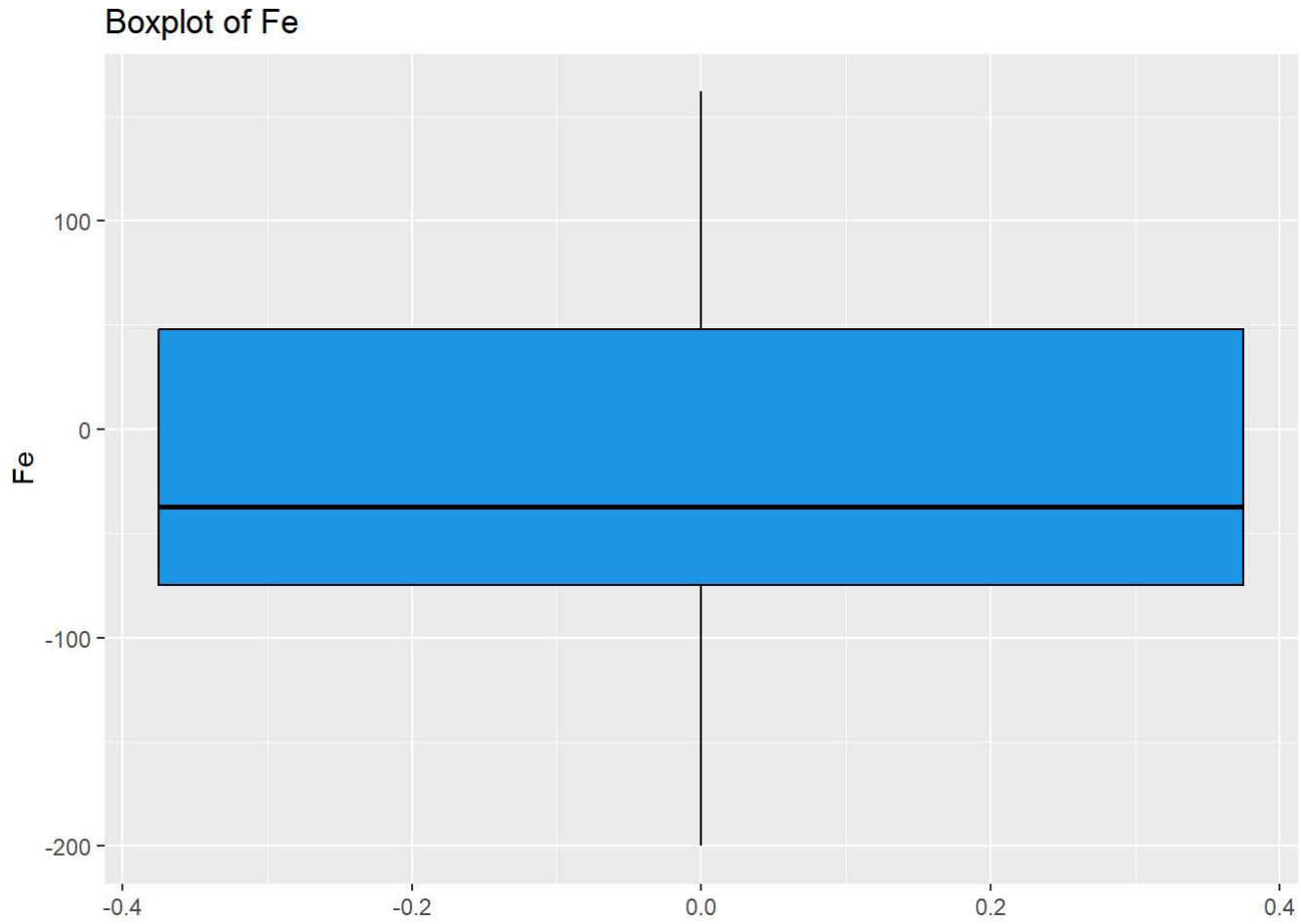

```
# Get the box plot of each feature
ggplot(dats, aes(y=dats$Al, fill=dats$Al)) +
geom_boxplot(color = "black",
fill = 2,
outlier.colour ="red",
outlier.shape = 8,
outlier.size = 6) +
ggtitle("Boxplot of Al") +
labs(y = "Al")
```



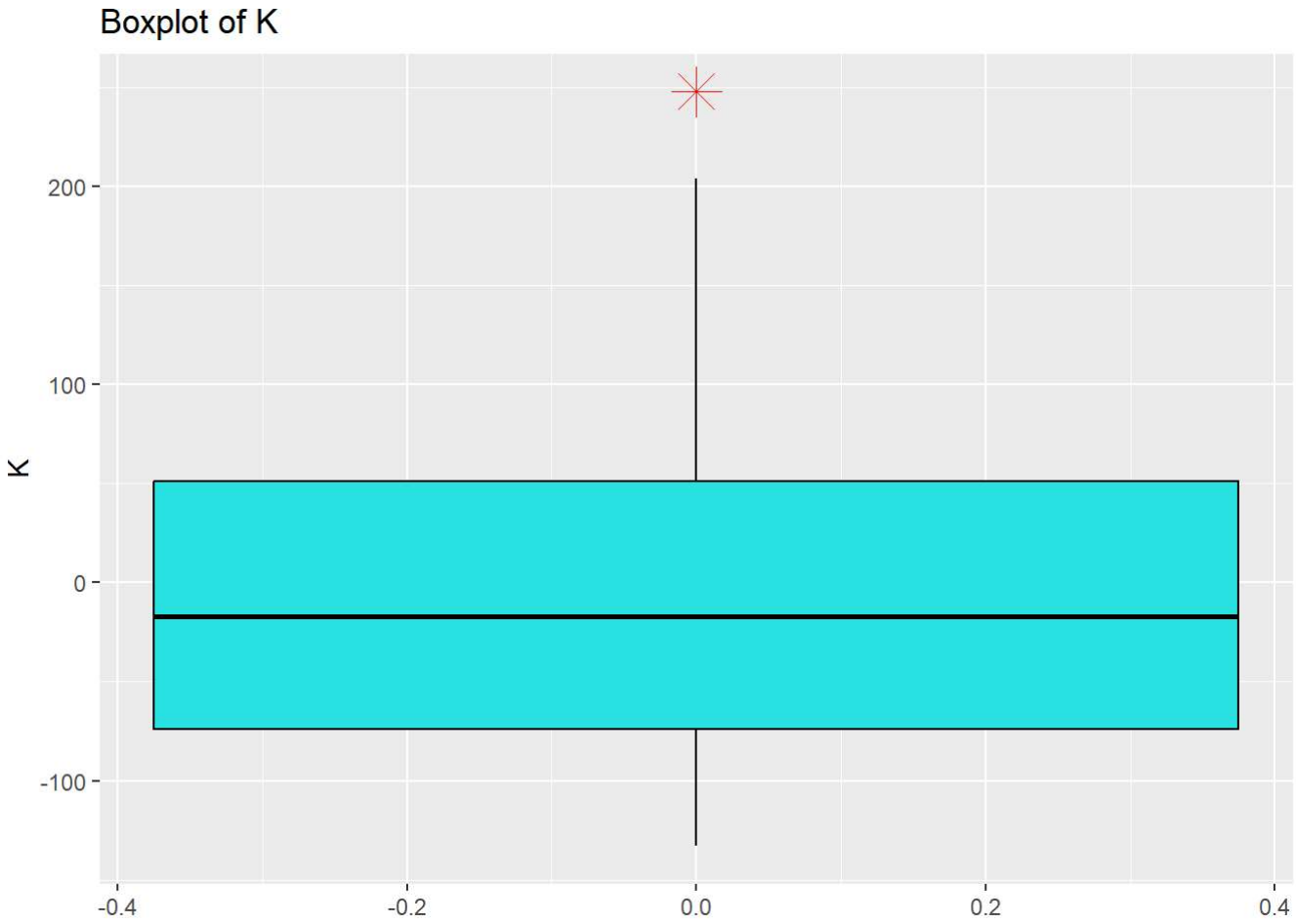

```
ggplot(dats, aes(y=dats$Ca, fill=dats$Ca)) +
geom_boxplot(color = "black",
fill = 3,
outlier.colour ="red",
outlier.shape = 8,
outlier.size = 6) +
ggtitle("Boxplot of Ca") +
labs(y = "Ca")
```



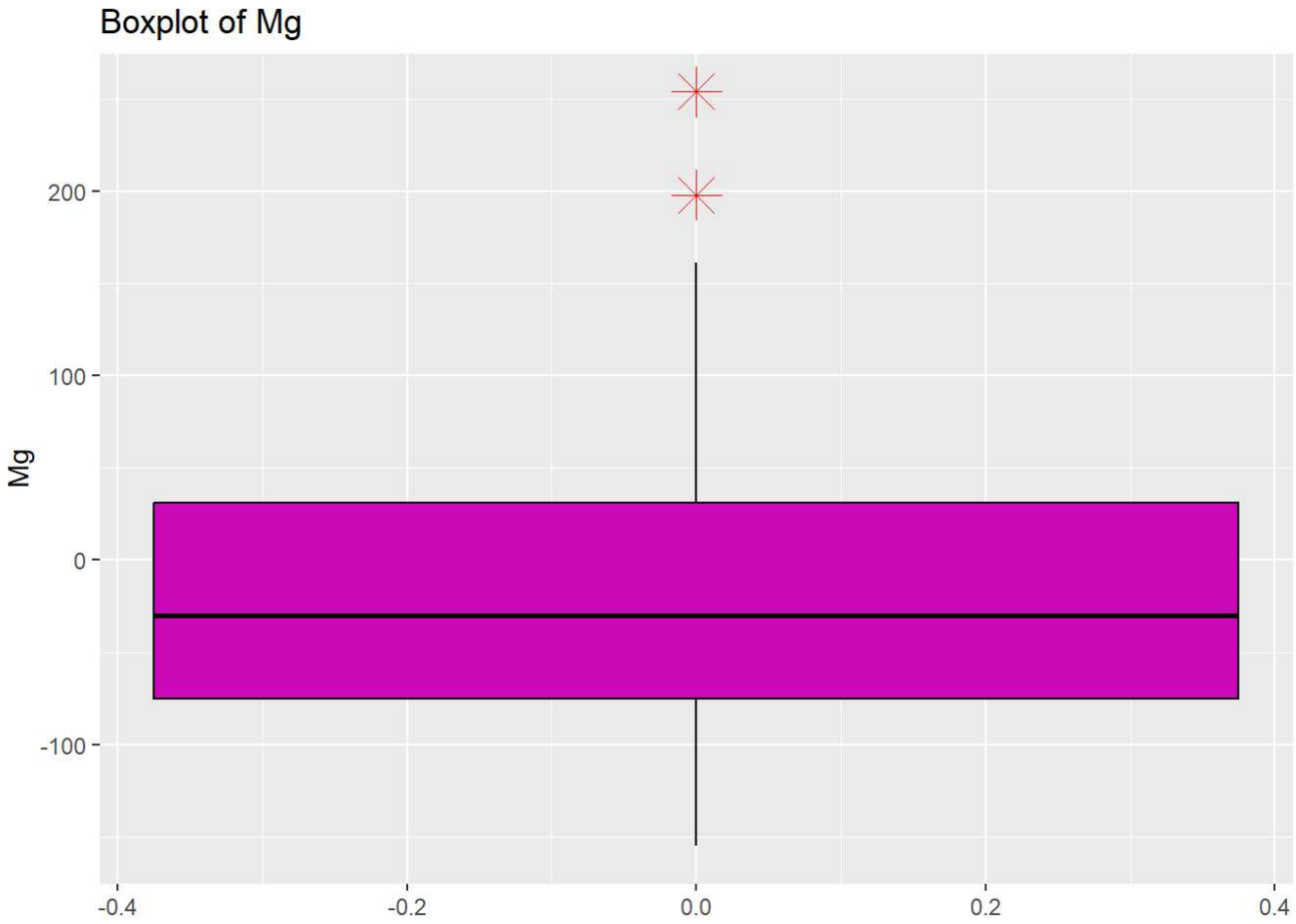

```
ggplot(dats, aes(y=dats$Fe, fill=dats$Fe)) +
geom_boxplot(color = "black",
fill = 4,
outlier.colour ="red",
outlier.shape = 8,
outlier.size = 6) +
ggtitle("Boxplot of Fe") +
labs(y = "Fe")
```



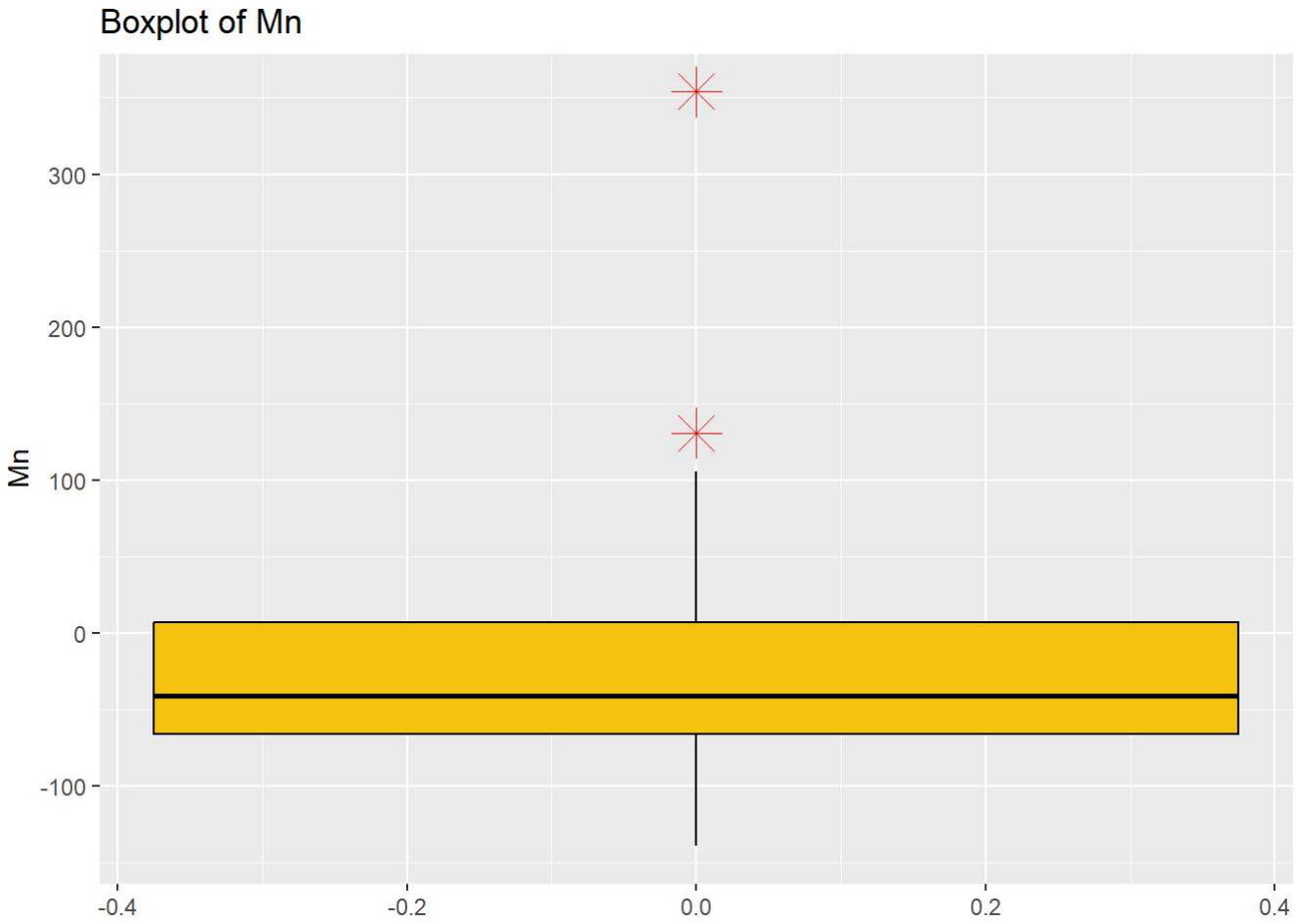

```
ggplot(dats, aes(y=dats$K, fill=dats$K)) +
geom_boxplot(color = "black",
fill = 5,
outlier.colour ="red",
outlier.shape = 8,
outlier.size = 6) +
ggtitle("Boxplot of K") +
labs(y = "K")
```



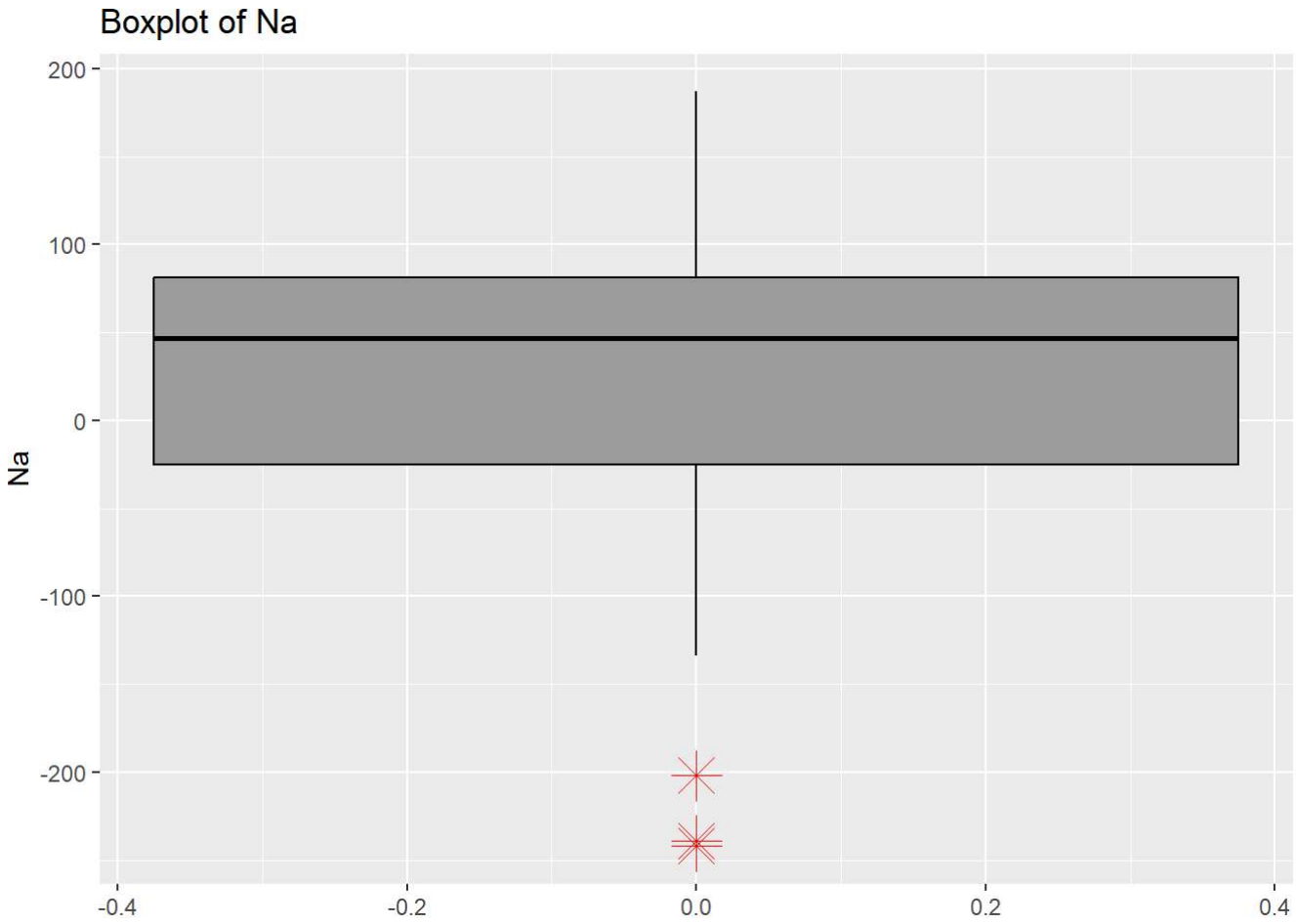

```
ggplot(dats, aes(y=dats$Mg, fill=dats$Mg)) +
geom_boxplot(color = "black",
fill = 6,
outlier.colour ="red",
outlier.shape = 8,
outlier.size = 6) +
ggtitle("Boxplot of Mg") +
labs(y = "Mg")
```



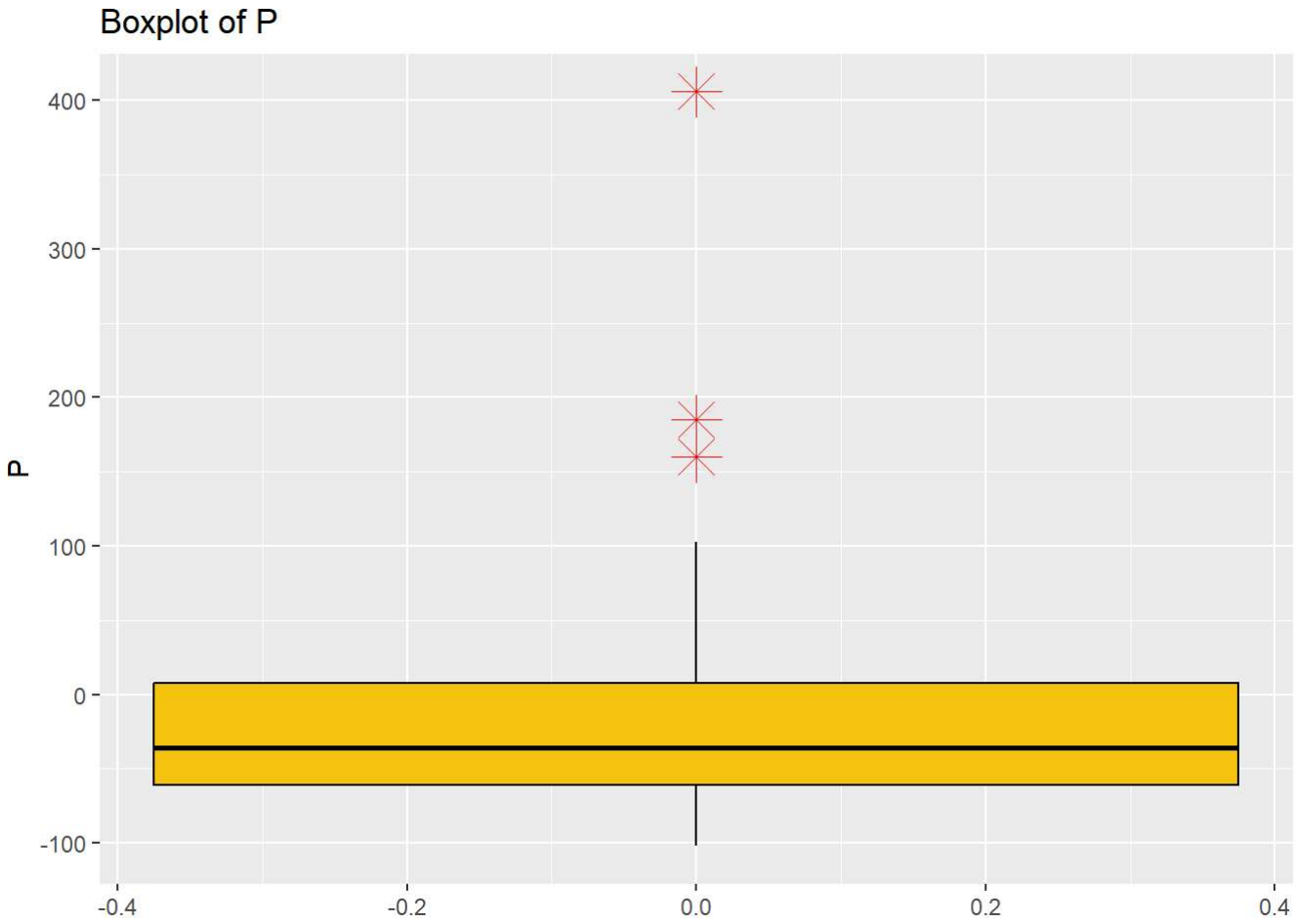

```
ggplot(dats, aes(y=dats$Mn, fill=dats$Mn)) +
geom_boxplot(color = "black",
fill = 7,
outlier.colour ="red",
outlier.shape = 8,
outlier.size = 6) +
ggtitle("Boxplot of Mn") +
labs(y = "Mn")
```



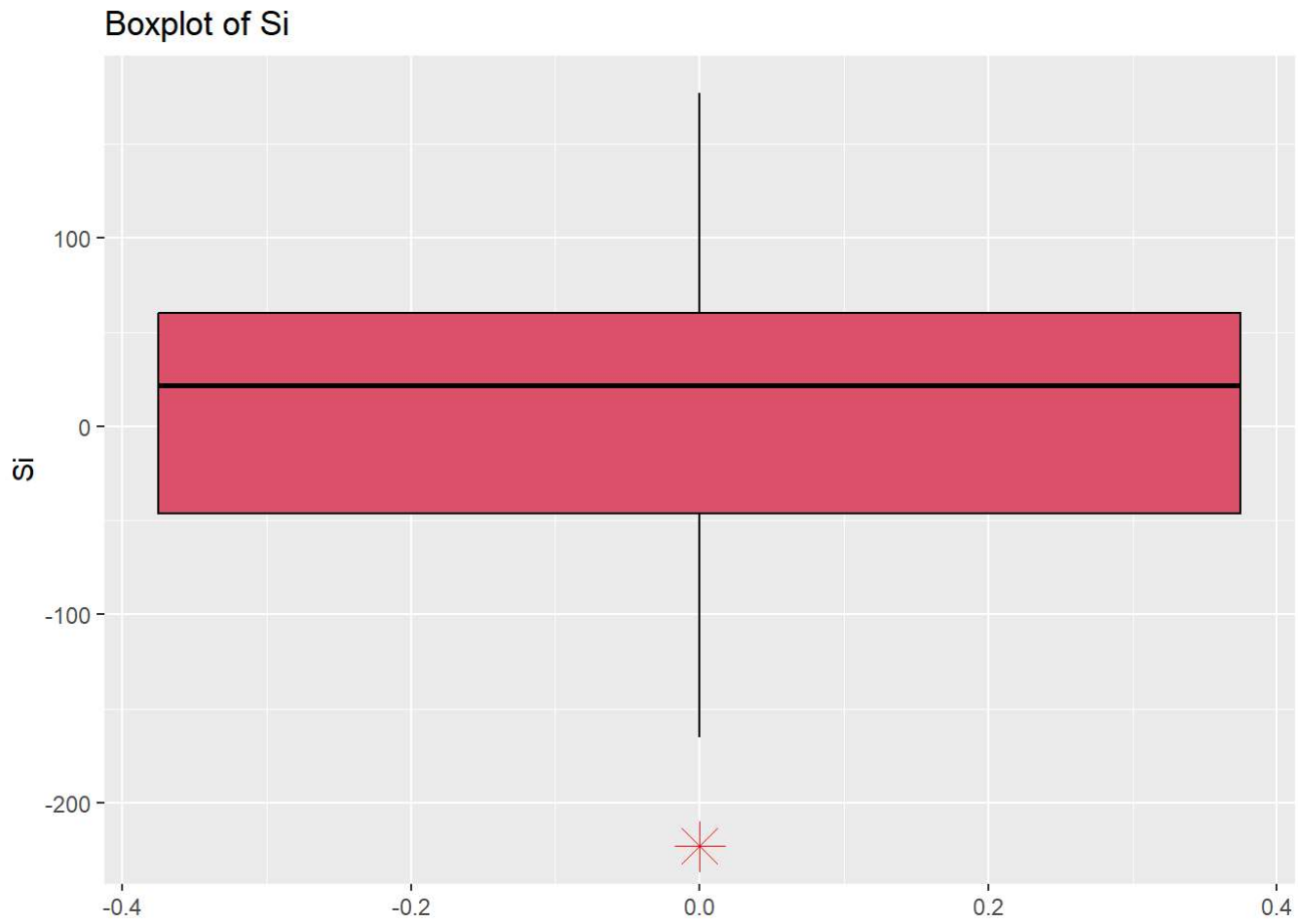

```
ggplot(dats, aes(y=dats$Na, fill=dats$Na)) +
geom_boxplot(color = "black",
fill = 8,
outlier.colour ="red",
outlier.shape = 8,
outlier.size = 6) +
ggtitle("Boxplot of Na") +
labs(y = "Na")
```



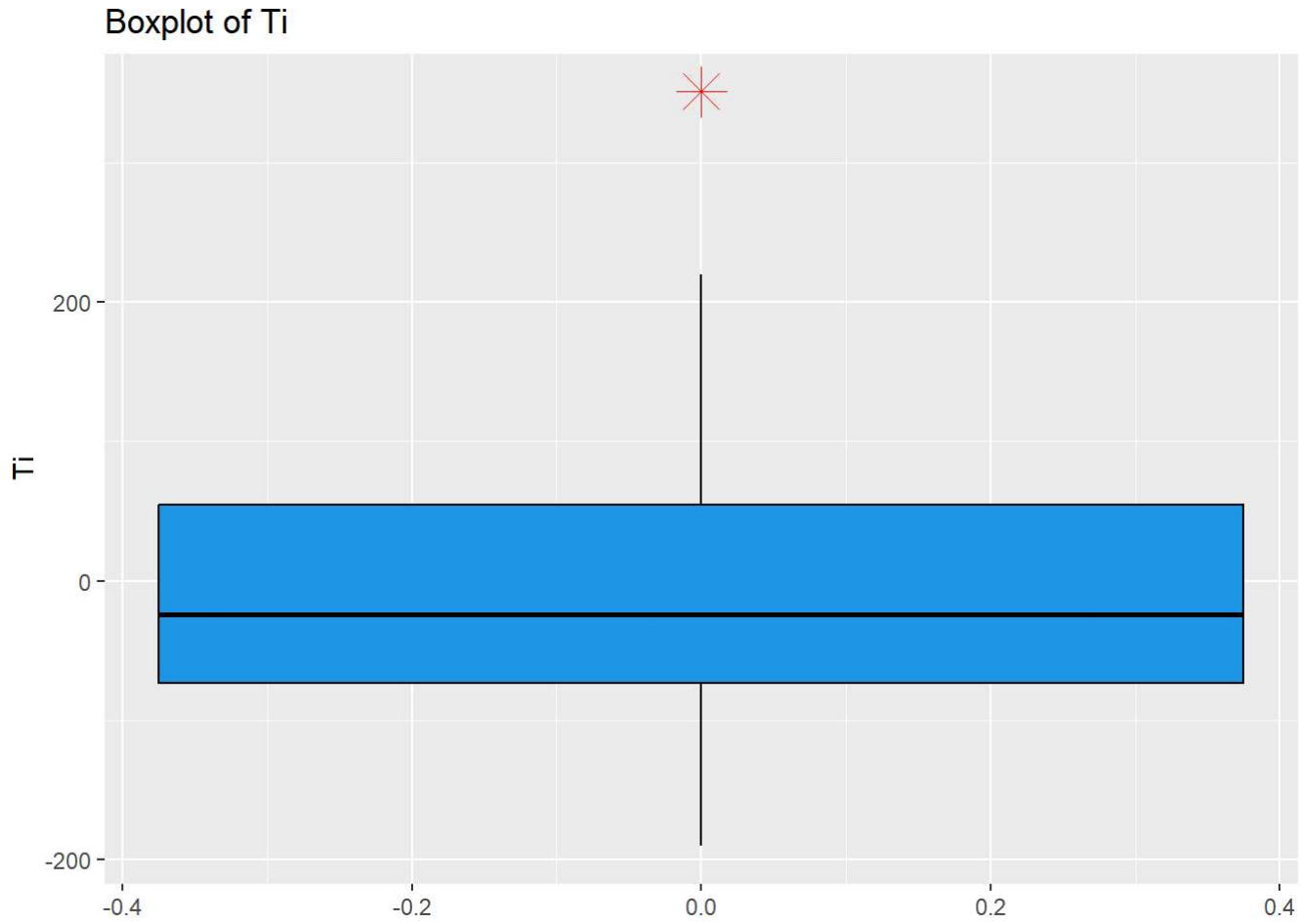

```
ggplot(dats, aes(y=dats$P, fill=dats$P)) +
geom_boxplot(color = "black",
fill = 15,
outlier.colour ="red",
outlier.shape = 8,
outlier.size = 6) +
ggtitle("Boxplot of P") +
labs(y = "P")
```




```
ggplot(dats, aes(y=dats$Si, fill=dats$Si)) +
geom_boxplot(color = "black",
fill = 10,
outlier.colour ="red",
outlier.shape = 8,
outlier.size = 6) +
ggtitle("Boxplot of Si") +
labs(y = "Si")
```




```
ggplot(dats, aes(y=dats$Ti, fill=dats$Ti)) +
geom_boxplot(color = "black",
fill = 12,
outlier.colour ="red",
outlier.shape = 8,
outlier.size = 6) +
ggtitle("Boxplot of Ti") +
labs(y = "Ti")
```



• From the boxplots of each feature and summary(dats):

• We can see that each feature is on different scale.

• For effective clustering analysis, it's important to scale variables to the same magnitude. As each feature in the data is on a different scale, it's better to standardize or normalize the data before applying any clustering techniques. This ensures equal weight and influence of each variable, resulting in more accurate and reliable clustering results.

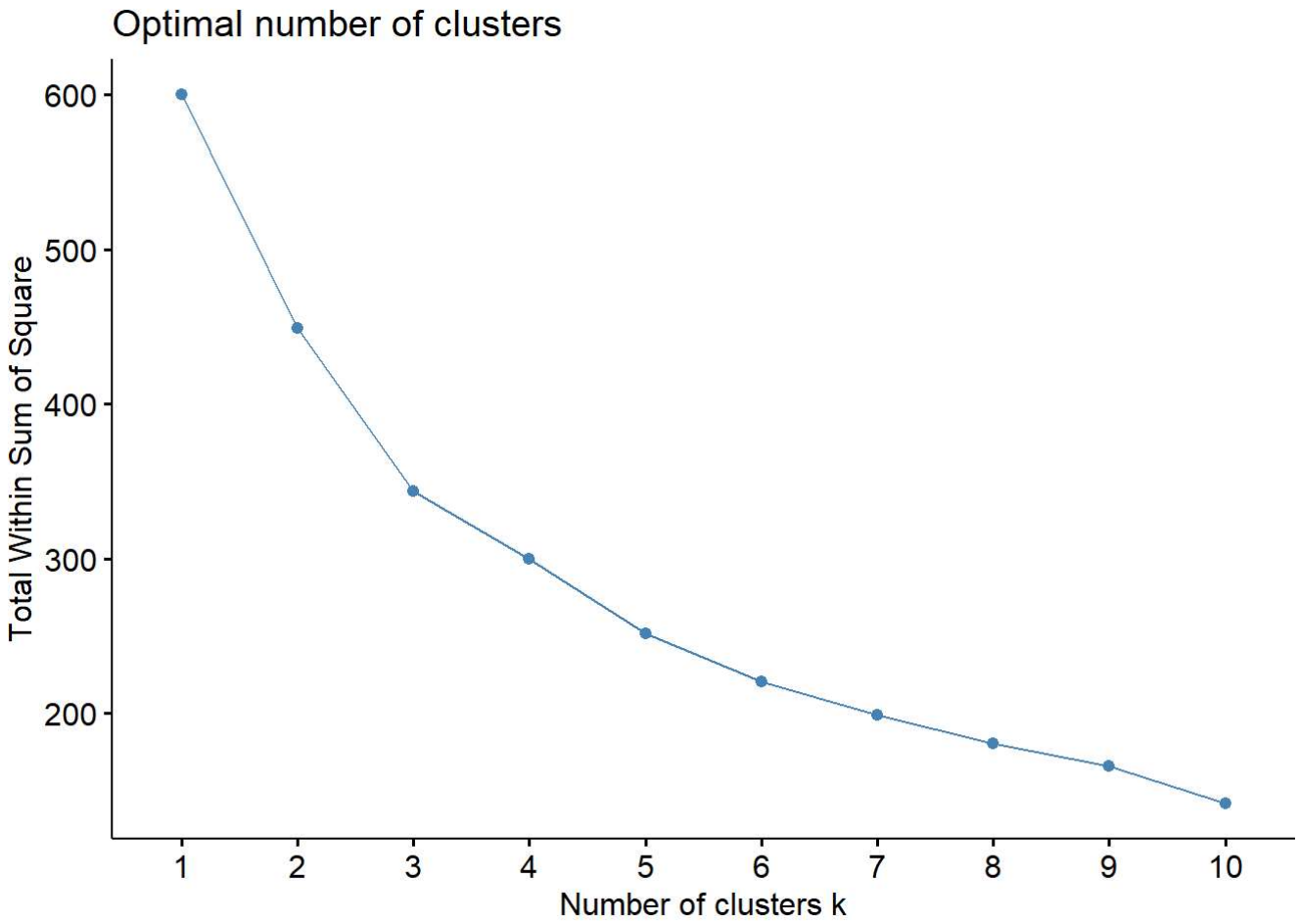
```
set.seed(123)

dats <- data.frame(scale(dats))

summary(dats)
```

```
##        Al                  Ca                  Fe                K          
##  Min.   :-2.32177   Min.   :-1.95876   Min.   :-2.0859   Min.   :-1.4906  
##  1st Qu.:-0.57071   1st Qu.:-0.63388   1st Qu.:-0.6792   1st Qu.:-0.8073  
##  Median :-0.01775   Median : 0.04526   Median :-0.2516   Median :-0.1471  
##  Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 0.53521   3rd Qu.: 0.67987   3rd Qu.: 0.7049   3rd Qu.: 0.6404  
##  Max.   : 3.27699   Max.   : 2.37215   Max.   : 1.9877   Max.   : 2.9220  
##        Mg                 Mn                 Na                 P          
##  Min.   :-1.6740   Min.   :-1.6266   Min.   :-2.8910   Min.   :-1.1735  
##  1st Qu.:-0.7306   1st Qu.:-0.6773   1st Qu.:-0.4778   1st Qu.:-0.6690  
##  Median :-0.1999   Median :-0.3522   Median : 0.3229   Median :-0.3614  
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 0.5195   3rd Qu.: 0.2720   3rd Qu.: 0.7010   3rd Qu.: 0.1799  
##  Max.   : 3.1493   Max.   : 4.7844   Max.   : 1.8798   Max.   : 5.0767  
##        Si                 Ti          
##  Min.   :-2.9207   Min.   :-1.9238  
##  1st Qu.:-0.6859   1st Qu.:-0.6914  
##  Median : 0.1726   Median :-0.1753  
##  Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 0.6524   3rd Qu.: 0.6569  
##  Max.   : 2.1296   Max.   : 3.7747
```

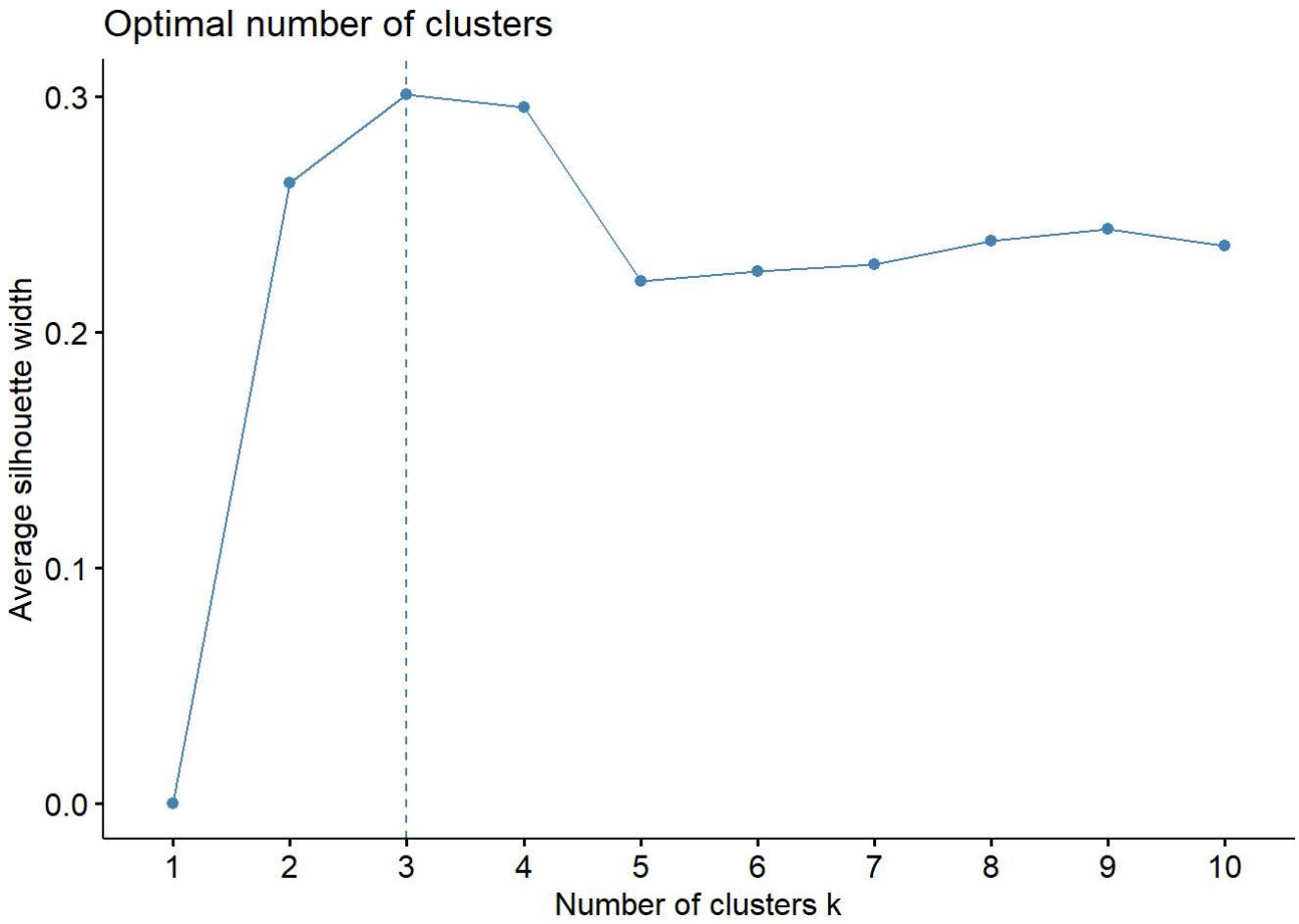
```
head(dats)
```

```
##              Al         Ca          Fe          K         Mg         Mn
## 190  1.15729442  0.1454649 -0.08283009 -0.1471431 -0.2470717 -0.3521752
## 191  0.56977110  1.4592119  0.42355879 -0.8999463  0.7081563  0.6101243
## 192  0.05136816  0.7466712 -0.08283009 -1.0620885 -0.2352788  0.2720191
## 193 -0.46703478 -0.1551382 -1.61324984 -0.7609672 -1.0725774 -1.3014707
## 194 -0.15599301  0.3570004 -0.75801527 -1.2589755  0.4605046 -0.3521752
## 195 -0.57071536 -0.4557414  0.51358348 -0.8072936  0.1303023  0.6101243
##             Na          P         Si         Ti
## 190  0.1004512 -0.3614384 -0.8374475 -0.2173996
## 191  0.4785559  1.1887936 -0.4081677  0.1723311
## 192  0.3228657  0.6843530  0.2736296 -0.3332655
## 193  0.3895900 -0.6690241  1.1953186 -1.6077902
## 194  0.5230388 -1.0258236  0.4377660 -1.2917923
## 195 -1.0005008 -0.3614384  0.8796717 -0.4596646
```

```
# elbow method
fviz_nbclust(dats, kmeans, method = "wss")
```




```
# silhouette method
fviz_nbclust(dats, kmeans, method = "silhouette")
```

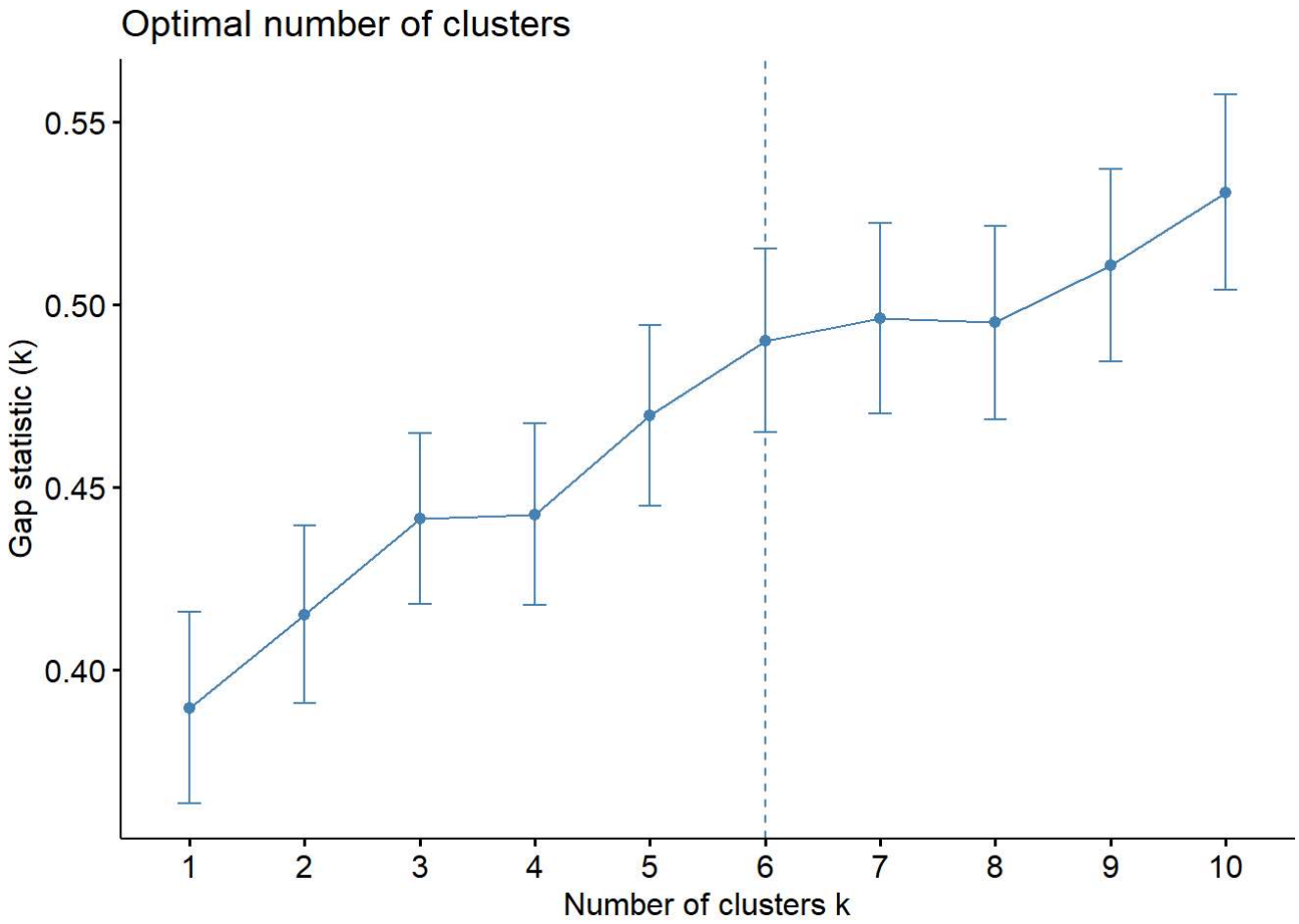


• From the elbow method, we can see the elbow at k=3.

• From the silhouette method, The average silhouette width is maximum at k = 3.

• Therefore, k = 3 is the best choice.

```
# gap statistic
gap_kmeans <- clusGap(dats, kmeans, nstart = 20, K.max = 10, B = 100)
fviz_gap_stat(gap_kmeans)
```



Justification for k=3

• According to elbow method, we can see the elbow at k = 3.

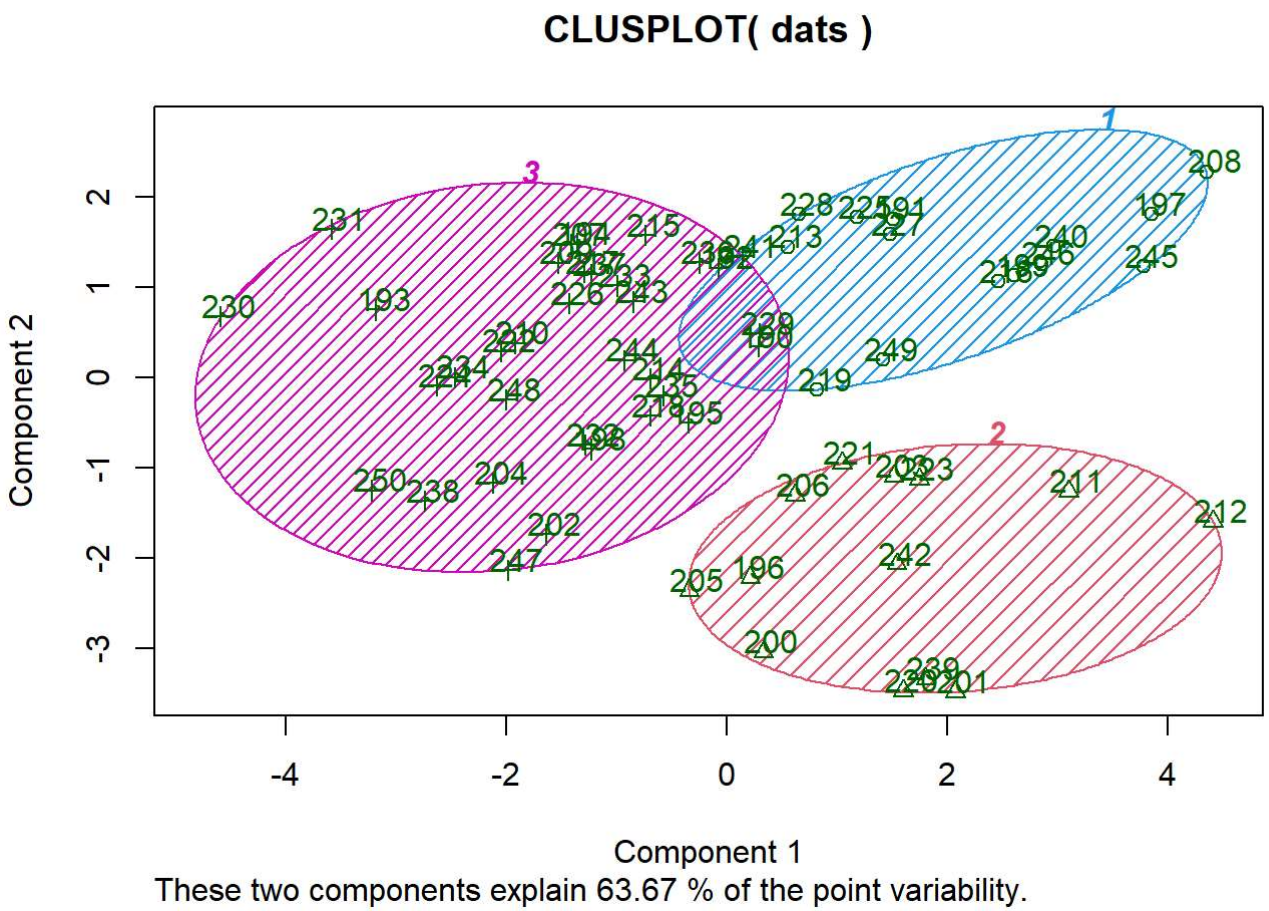
• According to the silhouette method, the optimal number of clusters is 3.

• According to the gap statistic method, the optimal number of clusters is 6, but due to the nature of the data and the size of the data, it is possible that this is not the best number of clusters to use. So I'm choosing k=3.

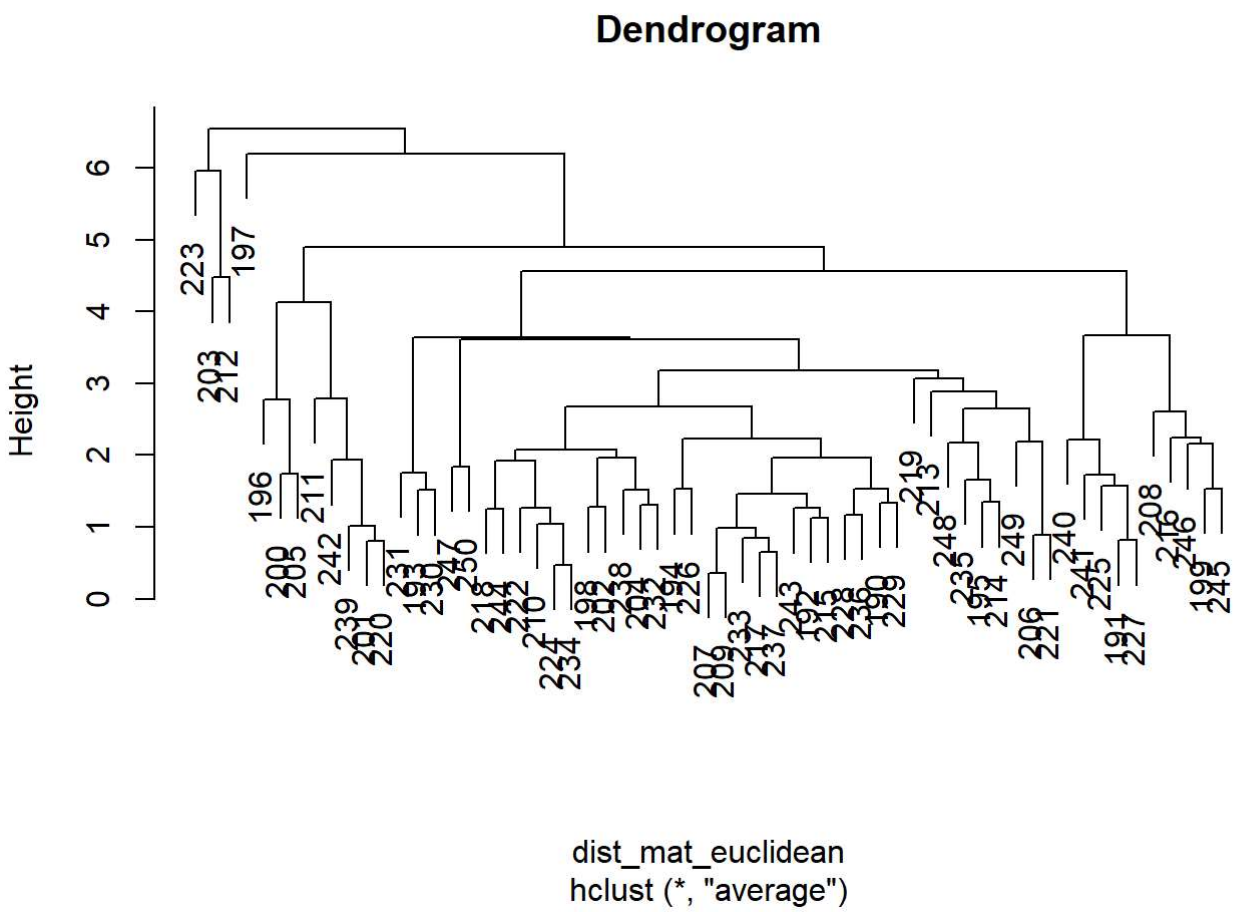
```
# apply k-means
km <- kmeans(dats, centers = 3, iter.max = 85, nstart = 20)
km
```

```
## K-means clustering with 3 clusters of sizes 14, 13, 34
##
## Cluster means:
##           Al          Ca         Fe          K         Mg         Mn         Na
## 1  0.3986158  1.12520844  0.8785050 -0.6774143  1.3230738  0.5432464  0.2895035
## 2  0.4820414 -1.00128037  0.8433855  1.2159206 -0.1490996  0.8802084 -1.0338630
## 3 -0.3484459 -0.08047863 -0.6842083 -0.1859755 -0.4877864 -0.5602400  0.2760932
##            P         Si         Ti
## 1  0.5657128 -1.1034927  0.5552904
## 2  0.3871779 -0.4644985  1.0838633
## 3 -0.3809792  0.6319817 -0.6430673
##
## Clustering vector:
## 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209
##   3   1   3   3   3   3   2   1   3   1   2   2   3   2   3   2   2   3   1   3
## 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229
##   3   2   2   1   3   3   1   3   3   1   2   2   3   2   3   1   3   1   1   3
## 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249
##   3   3   3   3   3   3   3   3   3   2   1   3   2   3   3   1   1   3   3   1
## 250
##   3
##
## Within cluster sum of squares by cluster:
## [1]  77.89491 136.16604 128.93087
##  (between_SS / total_SS =  42.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

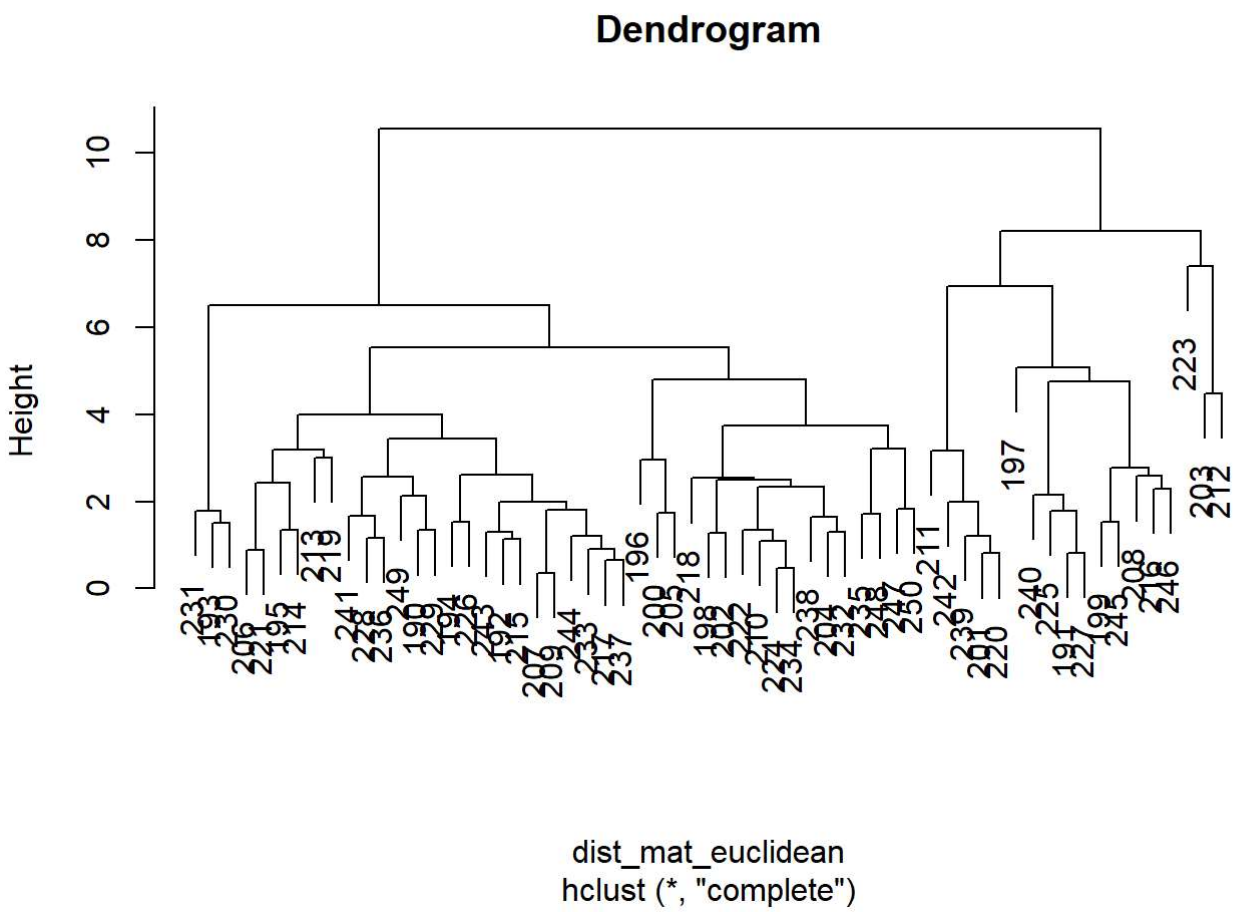
```
clusplot(dats, km$cluster, color=TRUE, shade=TRUE,labels=2,lines=0)
```




```
### Hierarchical clustering

# using method = 'euclidean'
dist_mat_euclidean <- dist(dats, method = 'euclidean')
# using method = 'manhattan'
dist_mat_manhattan <- dist(dats, method = 'manhattan')
# using method = 'max'
dist_mat_max <- dist(dats, method = 'max')

h_clust_avg <- hclust(dist_mat_euclidean, method = 'average')
plot(h_clust_avg, main='Dendrogram')
```




```
h_clust_com <- hclust(dist_mat_euclidean, method = 'complete')
plot(h_clust_com, main='Dendrogram')
```



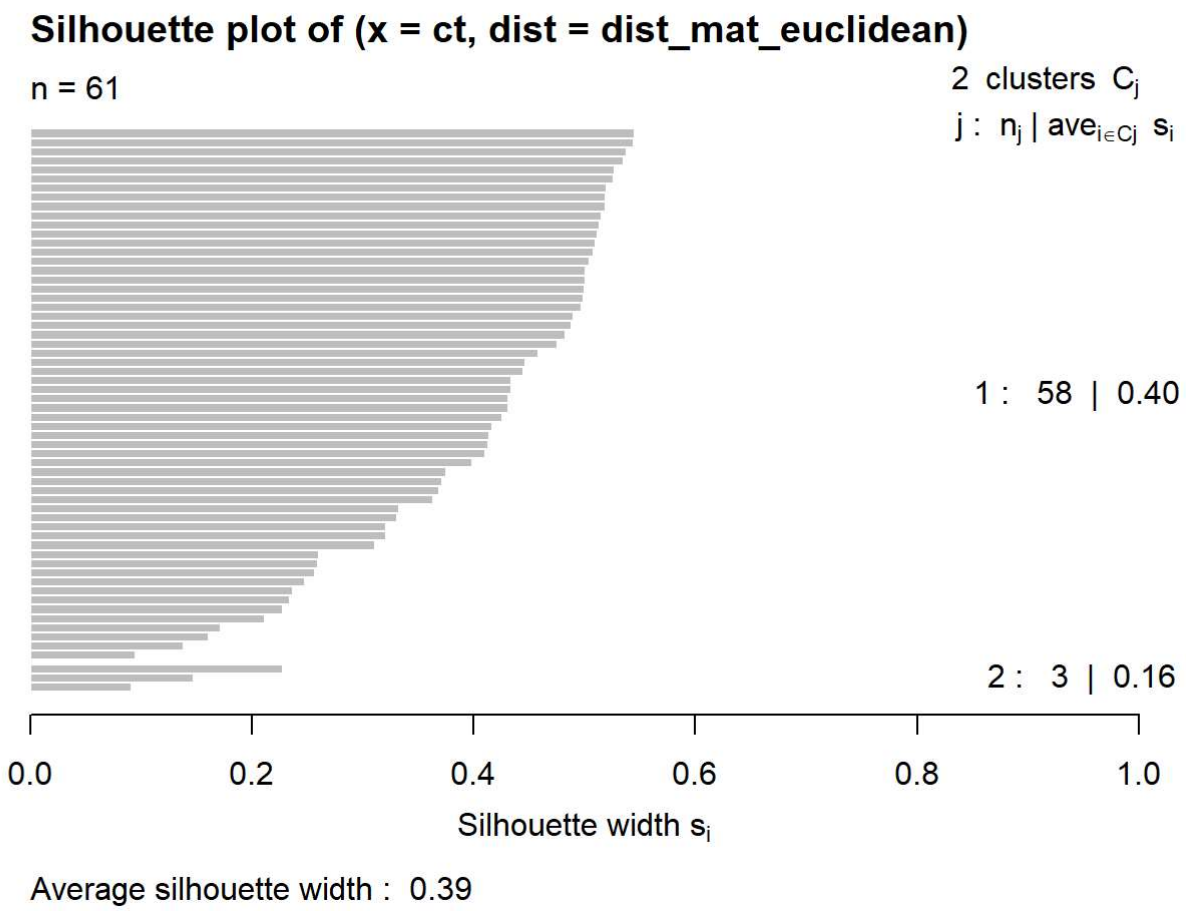

```
# Silhouette method

store <- c()
for (i in 2:6){
    ct <- cutree(h_clust_avg, k=i)
    si <- silhouette(ct, dist = dist_mat_euclidean)
    avg_width <- summary(si)$avg.width
    store <- c(store, avg_width)
}
store
```

```
## [1] 0.3918592 0.3288184 0.3211590 0.2663813 0.2850395
```

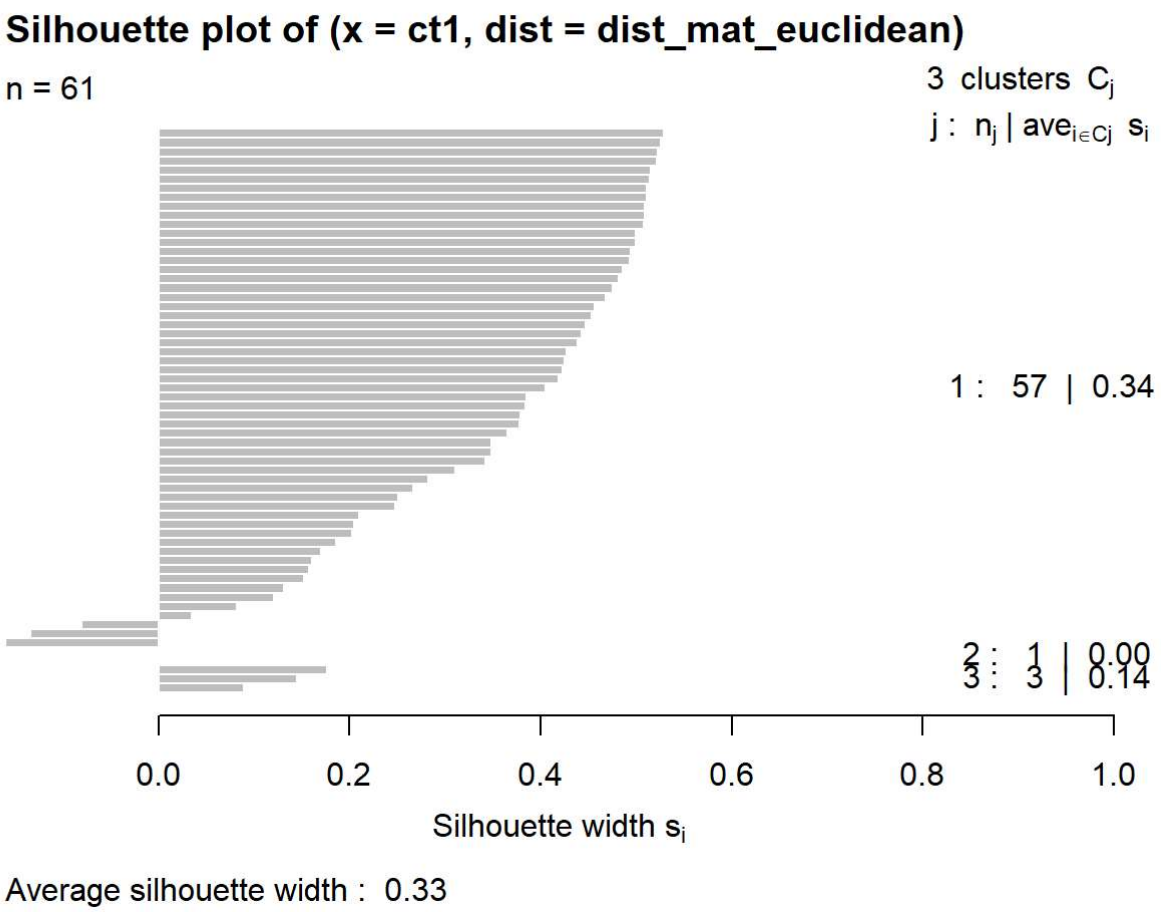
```
paste("The average silhouette width for k =", 2:6, " is", round(store,4))
```

```
## [1] "The average silhouette width for k = 2  is 0.3919"
## [2] "The average silhouette width for k = 3  is 0.3288"
## [3] "The average silhouette width for k = 4  is 0.3212"
## [4] "The average silhouette width for k = 5  is 0.2664"
## [5] "The average silhouette width for k = 6  is 0.285"
```

```
ct <- cutree(h_clust_avg, k = 2)
si <- silhouette(ct, dist = dist_mat_euclidean)
plot(si)
```




```
ct1 <- cutree(h_clust_avg, k = 3)
si1 <- silhouette(ct1, dist = dist_mat_euclidean)
plot(si1)
```



For the value k = 2 we are getting better average silhouette width. But for 'k=3' the silhouette plot visualization looks good and the distribution of silhouette values is more balanced compared to k=2. Therefore, the optimal k value for hierarchical clustering is 'k=3'. Because (k=3), it provides a better balance between the average silhouette width and the overall visualization of the silhouette plot.

Question 2) Consider the "diamonds" data from ggplot2. Use principal components on the variables {carat, x, y, z, depth, table}, and answer the following questions.

```
library(ggplot2)
library(factoextra)
data(diamonds)

dim(diamonds)
```

```
## [1] 53940    10
```

```
str(diamonds)
```

```
## tibble [53,940 × 10] (S3: tbl_df/tbl/data.frame)
##  $ carat  : num [1:53940] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
##  $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
##  $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
##  $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
##  $ depth  : num [1:53940] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
##  $ table  : num [1:53940] 55 61 65 58 58 57 57 55 61 61 ...
##  $ price  : int [1:53940] 326 326 327 334 335 336 336 337 337 338 ...
##  $ x      : num [1:53940] 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ y      : num [1:53940] 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
##  $ z      : num [1:53940] 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

```
summary(diamonds)
```

```
##      carat               cut        color        clarity          depth      
##  Min.   :0.2000   Fair      : 1610   D: 6775   SI1    :13065   Min.   :43.00  
##  1st Qu.:0.4000   Good      : 4906   E: 9797   VS2    :12258   1st Qu.:61.00  
##  Median :0.7000   Very Good:12082   F: 9542   SI2    : 9194   Median :61.80  
##  Mean   :0.7979   Premium   :13791   G:11292   VS1    : 8171   Mean   :61.75  
##  3rd Qu.:1.0400   Ideal     :21551   H: 8304   VVS2   : 5066   3rd Qu.:62.50  
##  Max.   :5.0100                     I: 5422   VVS1   : 3655   Max.   :79.00  
##                                     J: 2808   (Other): 2531                  
##      table           price             x                y         
##  Min.   :43.00   Min.   :  326   Min.   : 0.000   Min.   : 0.000  
##  1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710   1st Qu.: 4.720  
##  Median :57.00   Median : 2401   Median : 5.700   Median : 5.710  
##  Mean   :57.46   Mean   : 3933   Mean   : 5.731   Mean   : 5.735  
##  3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540   3rd Qu.: 6.540  
##  Max.   :95.00   Max.   :18823   Max.   :10.740   Max.   :58.900  
##                                                                   
##        z         
##  Min.   : 0.000  
##  1st Qu.: 2.910  
##  Median : 3.530  
##  Mean   : 3.539  
##  3rd Qu.: 4.040  
##  Max.   :31.800  
##
```

```
head(diamonds)
```

```
## # A tibble: 6 × 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
## 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
## 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
## 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
## 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
## 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

```
carat <- diamonds[, 1]
x <- diamonds[, 8]
y <- diamonds[, 9]
z <- diamonds[, 10]
depth <- diamonds[, 5]
table <- diamonds[, 6]
diam <- cbind(carat,x,y,z,depth,table)
dim(diam)
```

```
## [1] 53940     6
```

```
str(diam)
```

```
## 'data.frame':    53940 obs. of  6 variables:
##  $ carat: num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
##  $ x    : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ y    : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
##  $ z    : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
##  $ depth: num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
##  $ table: num  55 61 65 58 58 57 57 55 61 61 ...
```

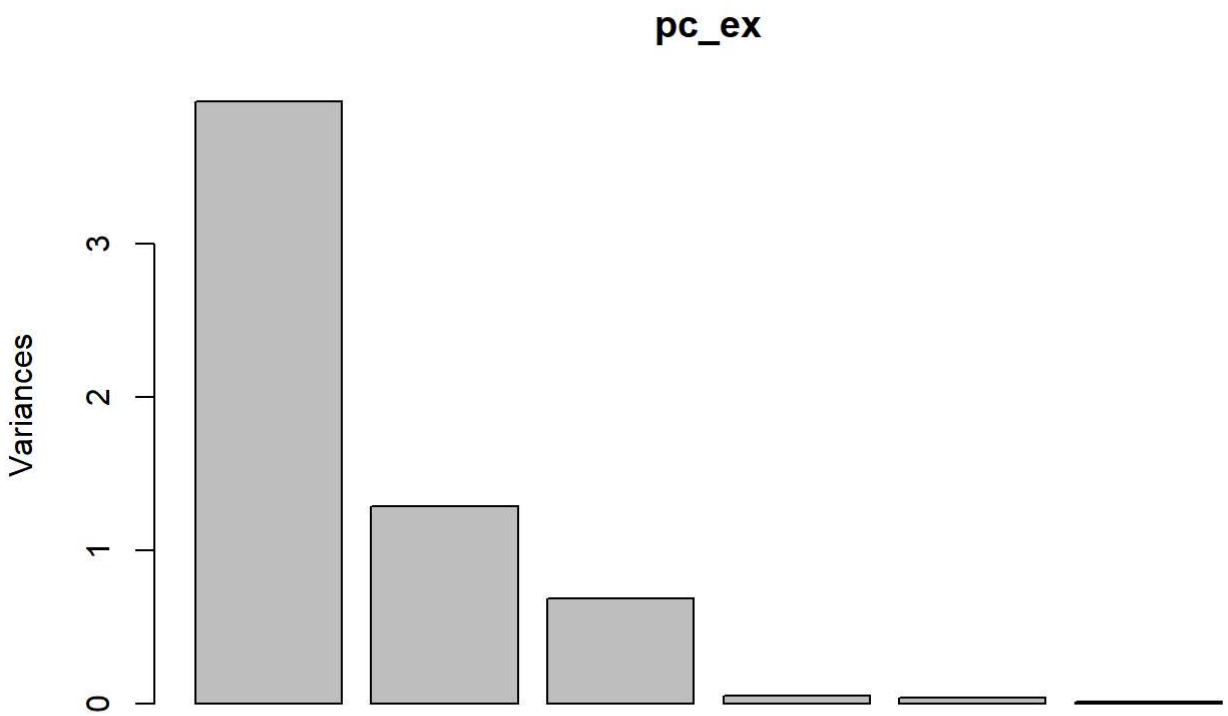
```
summary(diam)
```

```
##      carat              x                y                z         
##  Min.   :0.2000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000  
##  1st Qu.:0.4000   1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910  
##  Median :0.7000   Median : 5.700   Median : 5.710   Median : 3.530  
##  Mean   :0.7979   Mean   : 5.731   Mean   : 5.735   Mean   : 3.539  
##  3rd Qu.:1.0400   3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040  
##  Max.   :5.0100   Max.   :10.740   Max.   :58.900   Max.   :31.800  
##      depth           table      
##  Min.   :43.00   Min.   :43.00  
##  1st Qu.:61.00   1st Qu.:56.00  
##  Median :61.80   Median :57.00  
##  Mean   :61.75   Mean   :57.46  
##  3rd Qu.:62.50   3rd Qu.:59.00  
##  Max.   :79.00   Max.   :95.00
```

```
head(diam)
```

```
##   carat    x    y    z depth table
## 1  0.23 3.95 3.98 2.43  61.5    55
## 2  0.21 3.89 3.84 2.31  59.8    61
## 3  0.23 4.05 4.07 2.31  56.9    65
## 4  0.29 4.20 4.23 2.63  62.4    58
## 5  0.31 4.34 4.35 2.75  63.3    58
## 6  0.24 3.94 3.96 2.48  62.8    57
```

```
pc_ex <- prcomp(scale(diam), center = FALSE, scale = FALSE)
plot(pc_ex)
```




```
names(pc_ex)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

```
summary(pc_ex)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5    PC6
## Standard deviation     1.9830 1.1332 0.8267 0.2177 0.19950 0.1149
## Proportion of Variance 0.6554 0.2140 0.1139 0.0079 0.00663 0.0022
## Cumulative Proportion  0.6554 0.8694 0.9833 0.9912 0.99780 1.0000
```

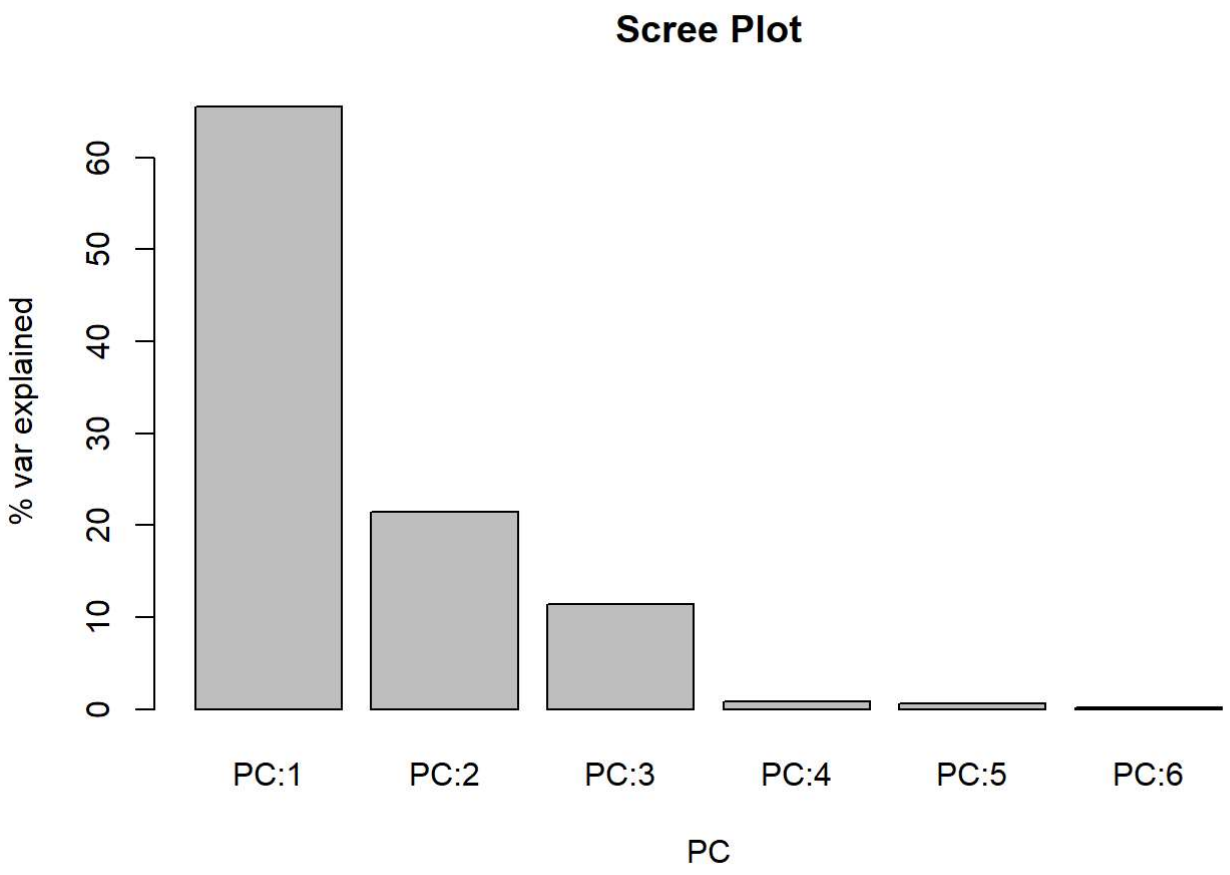
```
pc_ex$rotation # loadings
```

```
##                 PC1          PC2          PC3         PC4          PC5
## carat  0.4953672774  0.045129669  0.027908324  0.78996536  0.160214816
## x      0.5009101500  0.008203685  0.069979239  0.04075903 -0.048631847
## y      0.4952175606  0.009657367  0.086227223 -0.53762228  0.635139814
## z      0.4938820845  0.101283089 -0.007508905 -0.29133794 -0.748830865
## depth -0.0006822439  0.734082087 -0.671000723 -0.01402986  0.088358027
## table  0.1205813877 -0.669826823 -0.732523408 -0.01345637  0.002961066
##                PC6
## carat -0.319502171
## x      0.860289644
## y     -0.234074490
## z     -0.316449645
## depth  0.053638379
## table -0.003430812
```

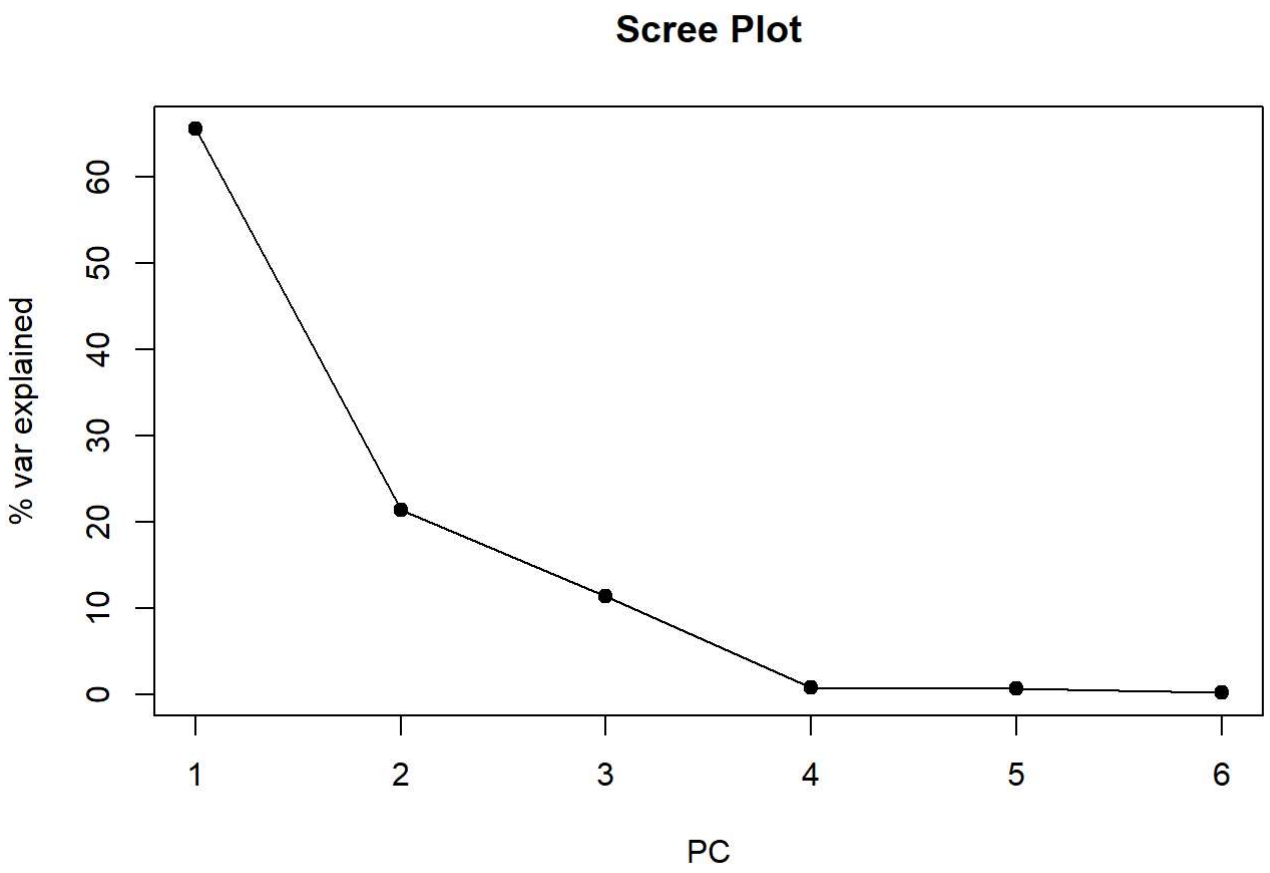
```
scores <- pc_ex$x #scores
head(scores)
```

```
##            PC1         PC2        PC3        PC4         PC5         PC6
## [1,] -3.058053  0.36772547  0.6571260 0.28963012  0.06742709 -0.13197943
## [2,] -2.925839 -2.32271485 -0.5278078 0.35007528  0.01585099 -0.15487212
## [3,] -2.516531 -5.00273834 -0.4523066 0.28526662 -0.02998189 -0.20750423
## [4,] -2.473886 -0.03205256 -0.7120120 0.17158042  0.06315317 -0.09252400
## [5,] -2.254885  0.45027707 -1.1158535 0.09515818  0.05874940 -0.04334452
## [6,] -2.918439  0.44220342 -0.6094869 0.26992953  0.08989147 -0.11910888
```

```
per_var_expl <- 100*((pc_ex$sdev)^2)/(sum(((pc_ex$sdev)^2)))

names(per_var_expl) <- paste("PC", 1:6, sep = ":")

barplot(per_var_expl, xlab = "PC", ylab = "% var explained", main = "Scree Plot")
```




```
plot(per_var_expl, xlab = "PC", ylab = "% var explained", main = "Scree Plot", type = 'o', pch = 19)
```



a) How much of the total variance does the first principal component account for? How many components are needed to account for at least 90% of the total variance?

```
per_var_expl
```

```
##       PC:1       PC:2       PC:3       PC:4       PC:5       PC:6
## 65.5356438 21.4008672 11.3899728  0.7900005  0.6633357  0.2201800
```

```
paste("Total variance of the first principal component is:",per_var_expl[1])
```

```
## [1] "Total variance of the first principal component is: 65.5356437602766"
```

```
paste("3 components are needed to account for at least 90% of the total variance")
```

```
## [1] "3 components are needed to account for at least 90% of the total variance"
```

b) Judging by the loadings, what do the first two principal components measure?

```
pc_ex$rotation
```

```
##                 PC1          PC2          PC3         PC4          PC5
## carat  0.4953672774  0.045129669  0.027908324  0.78996536  0.160214816
## x      0.5009101500  0.008203685  0.069979239  0.04075903 -0.048631847
## y      0.4952175606  0.009657367  0.086227223 -0.53762228  0.635139814
## z      0.4938820845  0.101283089 -0.007508905 -0.29133794 -0.748830865
## depth -0.0006822439  0.734082087 -0.671000723 -0.01402986  0.088358027
## table  0.1205813877 -0.669826823 -0.732523408 -0.01345637  0.002961066
##                PC6
## carat -0.319502171
## x      0.860289644
## y     -0.234074490
## z     -0.316449645
## depth  0.053638379
## table -0.003430812
```

For PC1:

• The variables with the highest loadings are carat, x, y, and z, all with positive loadings.

• These loadings suggest that PC1 primarily measures the size or dimensions of the diamonds, with higher weights on carat and the physical dimensions (x, y, z).

For PC2:

• The variables with the highest loadings are depth and table, with a high positive loading for depth and a high negative loading for table.

• This indicates that PC2 primarily measures the proportions or shape of the diamonds, with higher weights on depth and table.

• Therefore, based on the loadings:

• PC1 appears to measure the size or dimensions of the diamonds.

• PC2 appears to measure the proportions or shape of the diamonds.

c) What is the correlation between the first principal component and price?

```
correlation <- cor(scores[,1], diamonds$price)
paste("The correlation between the first principal component and price is:",correlation)
```

```
## [1] "The correlation between the first principal component and price is: 0.892005591182812"
```

```
### A strong positive correlation is there in between PC1 and diamond prices, indicating that as
the scores on PC1 increase, the prices of the diamonds tend to increase as well.
```

d) Can the first two principal components be used to distinguish between diamonds with different cuts?

Yes, the first two principal components be used to distinguish between diamonds with different cuts. Because, the total variance for first two principal components is approximately 87%.

```
c <- as.factor(diamonds$cut)
temp <- data.frame(scores, c)
names(temp)[7] <- "cuts"

temp <- temp[1:500, ]  ### For visualization purpose using 500 rows
dim(temp)
```

```
## [1] 500   7
```

```
colSums(is.na(temp))
```

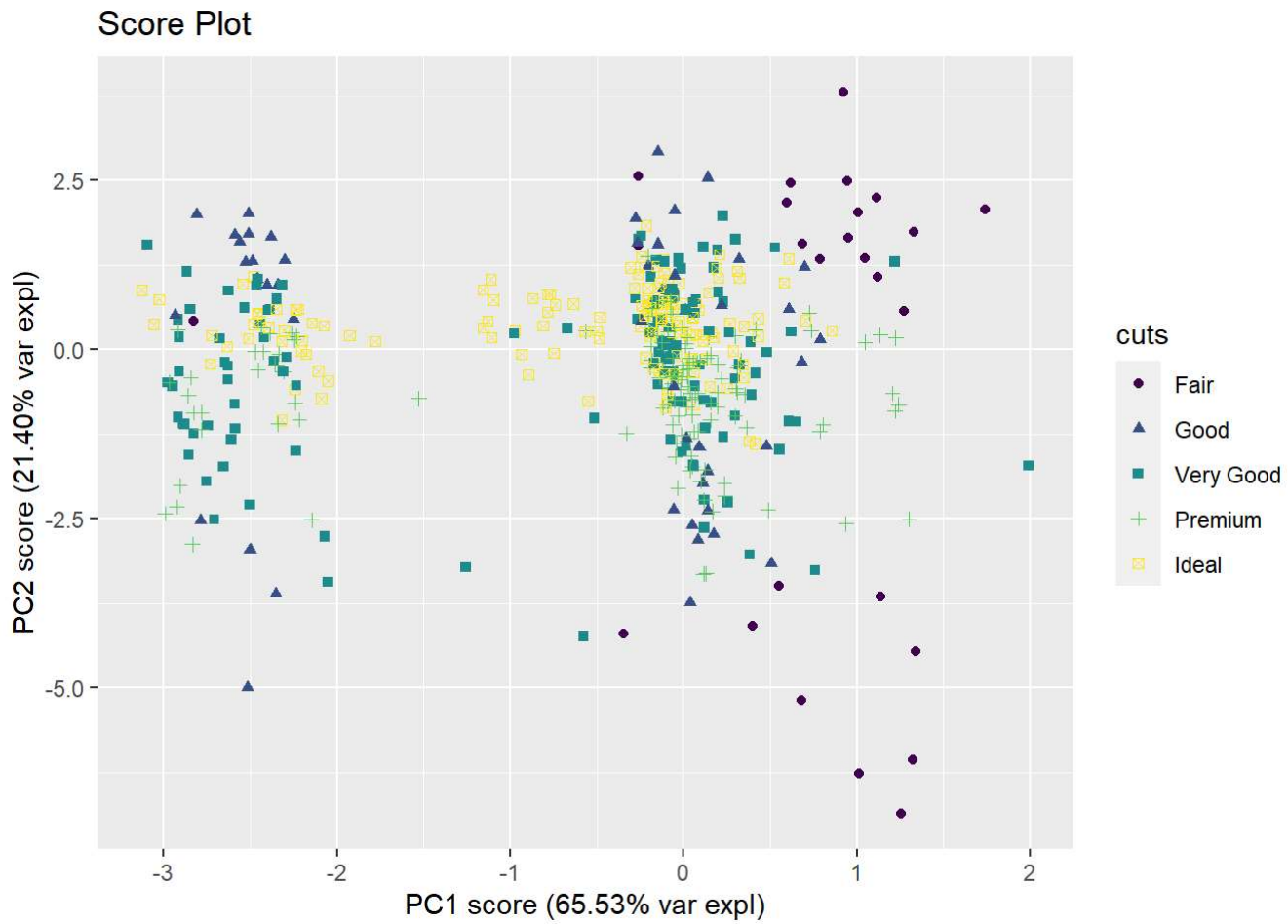
```
##  PC1  PC2  PC3  PC4  PC5  PC6 cuts
##    0    0    0    0    0    0    0
```

```
unique(temp$cuts)
```

```
## [1] Ideal     Premium   Good      Very Good Fair
## Levels: Fair < Good < Very Good < Premium < Ideal
```

```
p <- ggplot(temp, aes(PC1, PC2, color = cuts, shape = cuts)) + geom_point()
p + labs(title = "Score Plot", x = "PC1 score (65.53% var expl)", y = "PC2 score (21.40% var exp
l)")
```

```
## Warning: Using shapes for an ordinal variable is not advised
```



Question 3) Consider the Iris data(iris)

```
library(ggplot2)
library(factoextra)
library(cluster)
library(fossil)
```

```
## Warning: package 'fossil' was built under R version 4.3.3
```

```
## Loading required package: sp
```

```
## Warning: package 'sp' was built under R version 4.3.3
```

```
## Loading required package: maps
```

```
## Warning: package 'maps' was built under R version 4.3.3
```

```
##
## Attaching package: 'maps'
```

```
## The following object is masked from 'package:cluster':
##
##     votes.repub
```

```
## Loading required package: shapefiles
```

```
## Loading required package: foreign
```

```
##
## Attaching package: 'shapefiles'
```

```
## The following objects are masked from 'package:foreign':
##
##     read.dbf, write.dbf
```

```
data(iris)
dim(iris)
```

```
## [1] 150   5
```

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

```
colSums(is.na(iris))
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
##            0            0            0            0            0
```

```
ir <- iris[, 1:4]
dim(ir)
```

```
## [1] 150   4
```

```
head(ir)
```

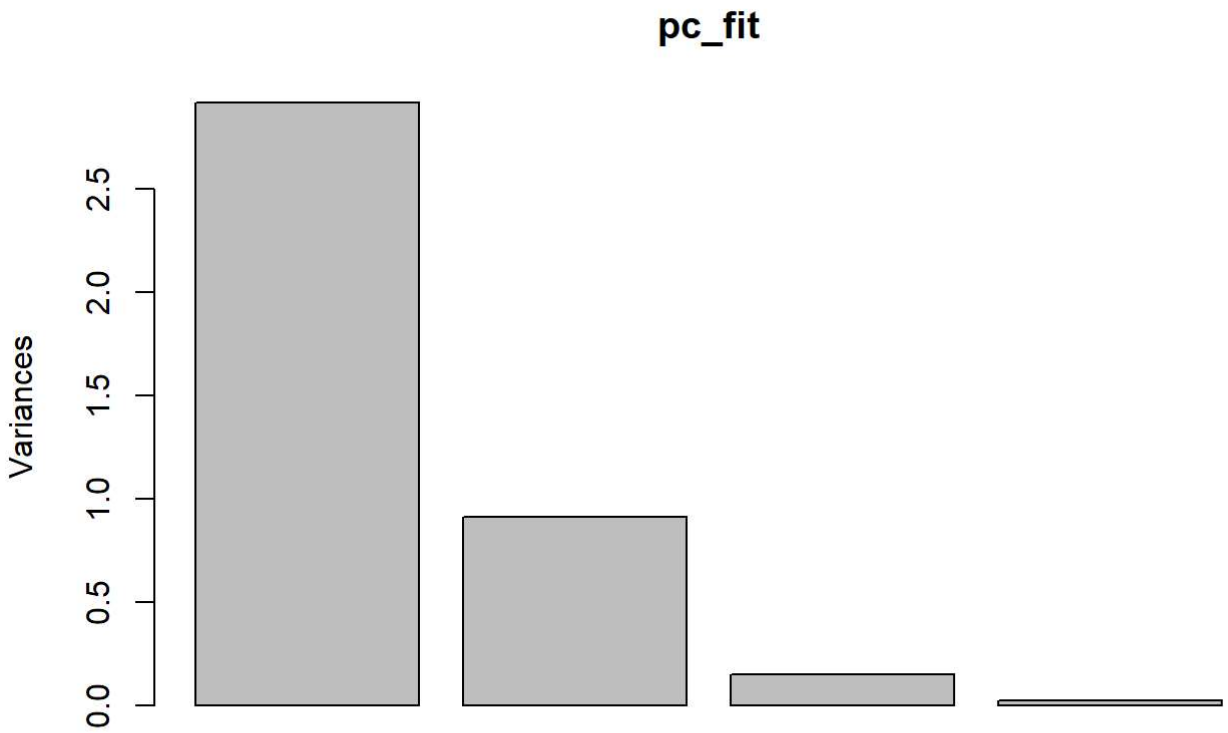
```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1          5.1         3.5          1.4         0.2
## 2          4.9         3.0          1.4         0.2
## 3          4.7         3.2          1.3         0.2
## 4          4.6         3.1          1.5         0.2
## 5          5.0         3.6          1.4         0.2
## 6          5.4         3.9          1.7         0.4
```

a) Create a plot using the first two principal components, and color the iris species by class.

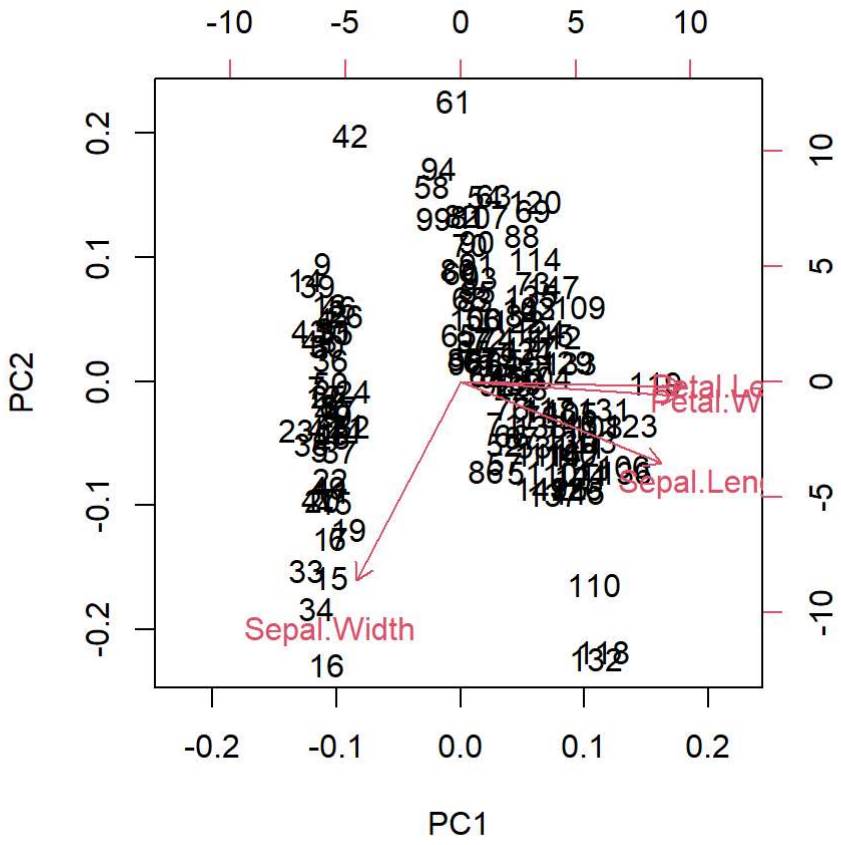
```
pc_fit <- prcomp(ir, center=TRUE, scale=TRUE)
pc_fit
```

```
## Standard deviations (1, .., p=4):
## [1] 1.7083611 0.9560494 0.3830886 0.1439265
##
## Rotation (n x k) = (4 x 4):
##                    PC1         PC2        PC3        PC4
## Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
## Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
## Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
## Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
```

```
plot(pc_fit)
```



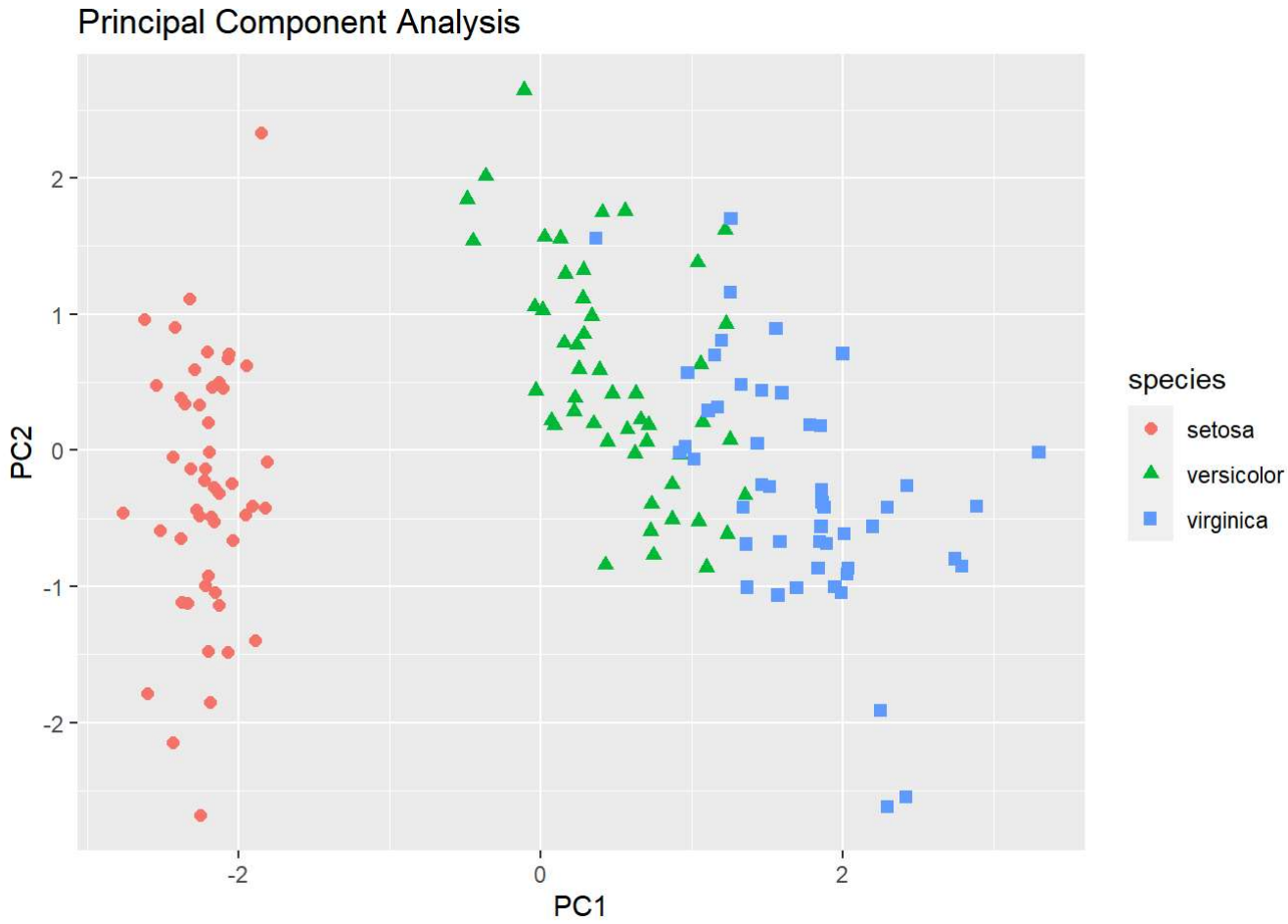

```
biplot(pc_fit)
```




```
pc_1 <- pc_fit$x[, 1]
pc_2 <- pc_fit$x[, 2]

pc_comb <- cbind(pc_1, pc_2)
head(pc_comb)
```

```
##           pc_1       pc_2
## [1,] -2.257141 -0.4784238
## [2,] -2.074013  0.6718827
## [3,] -2.356335  0.3407664
## [4,] -2.291707  0.5953999
## [5,] -2.381863 -0.6446757
## [6,] -2.068701 -1.4842053
```

```
species <- iris$Species

ggplot(data.frame(pc_comb), aes(x = pc_1, y = pc_2)) +
geom_point(aes(color = species, shape = species), size = 2) +
ggtitle("Principal Component Analysis") +
labs(x = "PC1", y = "PC2")
```



b) Perform k-means clustering on the first two principal components of the iris data. Plot the clusters different colors, and the specify different symbols to depict the species labels.

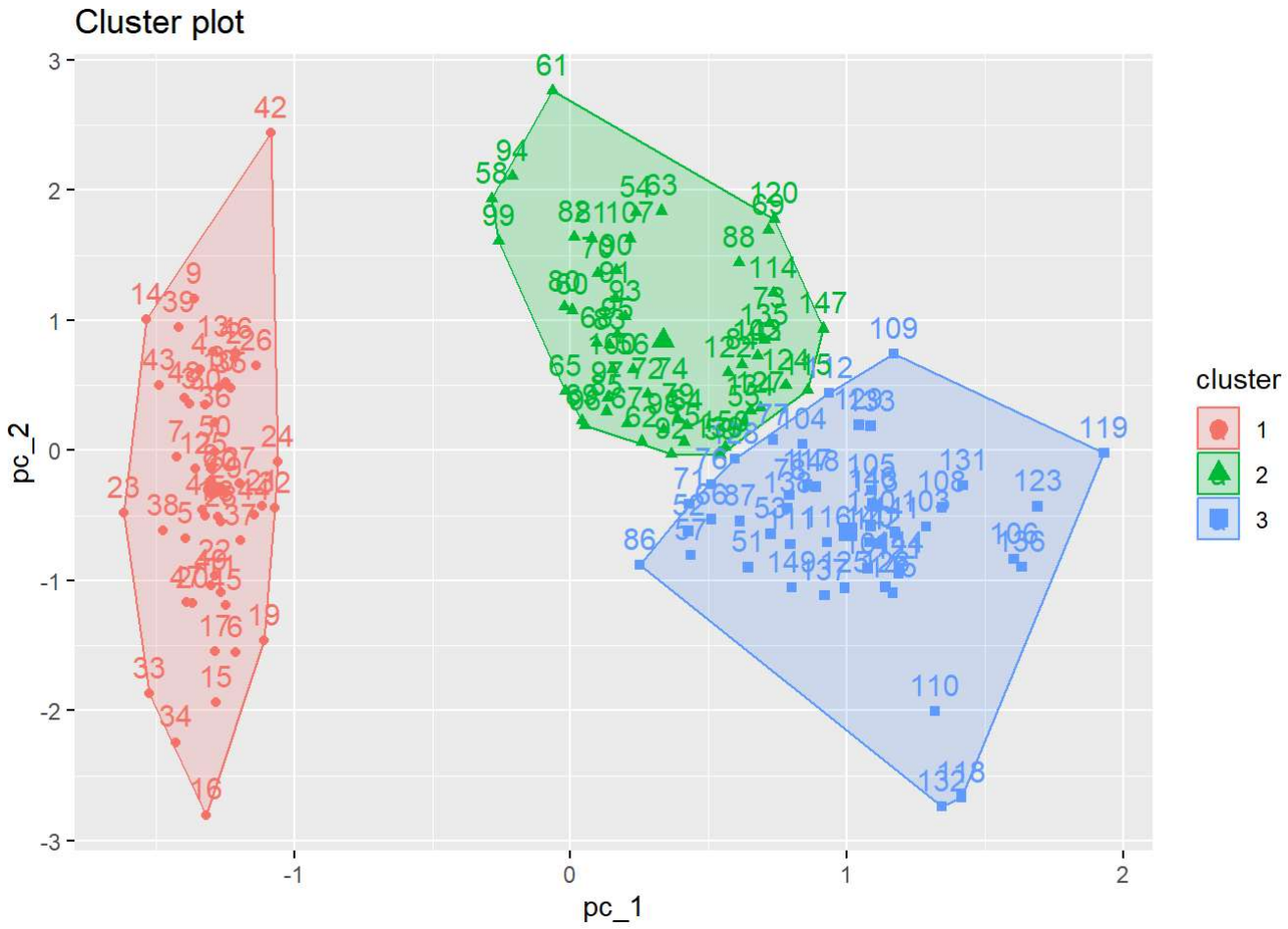
```
set.seed(123)
 # Perform K-Means k=3

k_means_model <- kmeans(pc_comb, centers = 3, nstart = 10)

k_means_model
```

```
## K-means clustering with 3 clusters of sizes 50, 53, 47
##
## Cluster means:
##         pc_1       pc_2
## 1 -2.2173249 -0.2879627
## 2  0.5707095  0.8045137
## 3  1.7152903 -0.6008742
##
## Clustering vector:
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 2 2 2 3 2 2 2 2 2 2 2 3 2 2 2 2 3 2 2 2
##  [75] 2 3 3 3 2 2 2 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3
## [112] 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 3 3 3 3 3 3 2 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [149] 3 2
##
## Within cluster sum of squares by cluster:
## [1] 44.58037 34.85204 34.82154
##  (between_SS / total_SS =  80.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
# how well does the clustering match the labels
table(k_means_model$cluster, species)
```

```
##    species
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         39        14
##   3      0         11        36
```

```
fviz_cluster(k_means_model, pc_comb)
```



c) Use rand index and adjusted rand index to assess how well the cluster assignments capture the species labels.

• The rand index measures the agreement between the two clusters, with a value of 1 indicating perfect agreement and a value of 0 indicating no agreement.

• The adjusted rand index is a corrected-for-chance version of the rand index and has a range of -1 to 1, with values close to 1 indicating high agreement, values close to 0 indicating random agreement and values close to -1 indicating disagreement.

```
rand.index(k_means_model$cluster, as.numeric(iris$Species))
```

```
## [1] 0.8322148
```

```
adj.rand.index(k_means_model$cluster, as.numeric(iris$Species))
```

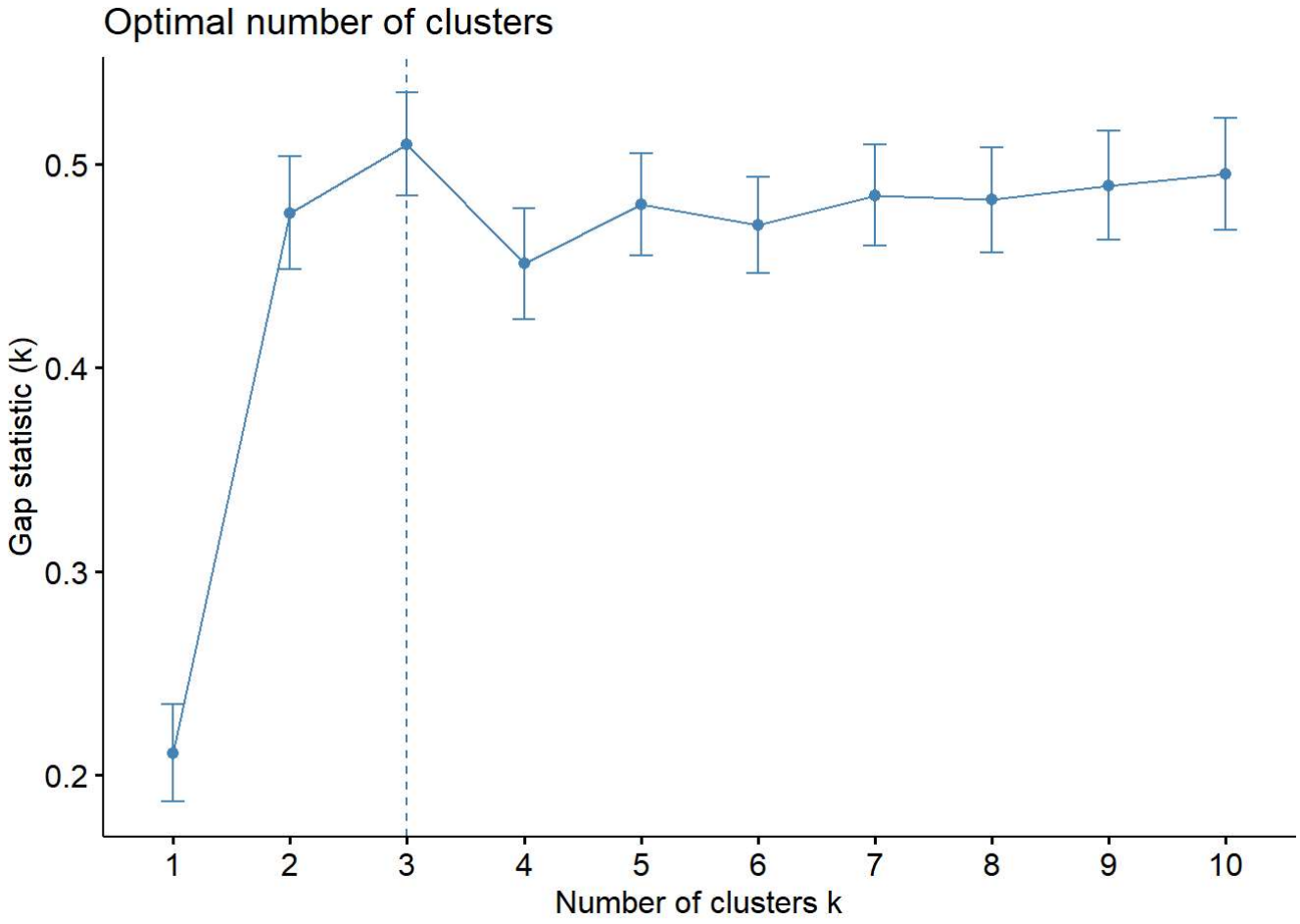
```
## [1] 0.6201352
```

d) Use the gap statistic and silhouette plots to determine the number of clusters.

• The gap statistic is a measure used to estimate the optimal number of clusters in a data set.

• The gap statistic measures the difference between within-cluster dispersion data to a null reference distribution, and a larger gap statistic indicates a better separation between clusters.

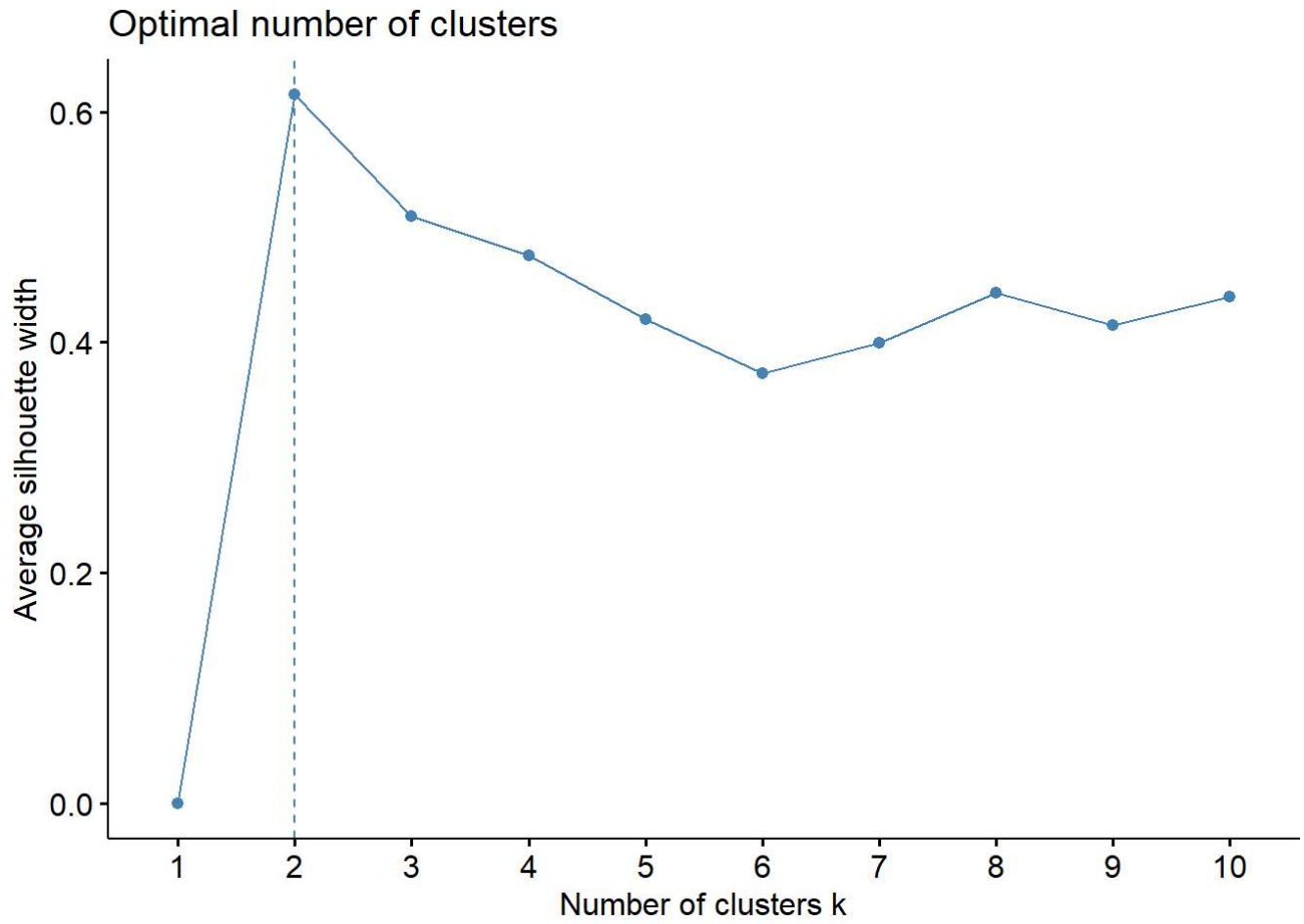
```
gap_statistic <- clusGap(pc_comb, kmeans, nstart = 20, K.max = 10, B = 50)
fviz_gap_stat(gap_statistic)
```



• From the above gap statistic plot k=3.

• The silhouette value measures how similar an object is to its own cluster compared to other clusters, and a high silhouette value indicates that the object is well-matched to its own cluster.

```
fviz_nbclust(pc_comb, kmeans, method = "silhouette")
```



• From above plot the average silhouette width is maximum at k=2.

e) Reflect on the results, especially c-d. What does this tell us about the clustering?

1. The value of rand.index = 0.832
2. The value of adjusted.rand.index = 0.620

Based on the results obtained from the iris scores data using the gap statistic, silhouette plots, and the Rand index measures, we can draw the following conclusions:

• The gap statistic plot suggests that the optimal number of clusters for this dataset is 3.

• The silhouette plot suggests that the optimal number of clusters for this dataset is 2.

• Since we already have the information about the class labels, we choose k=3 as the best k.

• The Rand index shows how well the clustering method classifies the iris dataset's true species labels. The clustering method worked well according to the Rand index of 0.832 and adjusted Rand index of 0.602. Even though, the adjusted Rand index is lower than the Rand index, indicating a chance match between expected cluster assignments and genuine class labels. The clustering approach performs well on iris dataset, which has just three classes. But, real-world datasets may have more complicated structures and dimensions, making the clustering technique even more challenging.

• Overall, the kmeans clustering algorithm did a good job of capturing the true species labels of the iris dataset.

Question 4) Consider the wine quality data (https://archive.ics.uci.edu/dataset/186/wine+quality (https://archive.ics.uci.edu/dataset/186/wine+quality))

a) Perform exploratory data analysis on the data. Summarize the data quality and characteristics. Discuss any apparent outliers and associations.

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ lubridate 1.9.3     ✓ tibble    3.2.1
## ✓ purrr     1.0.2     ✓ tidyr     1.3.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ purrr::map()    masks maps::map()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

```
library(ggplot2)
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.3.3
```

```
## corrplot 0.92 loaded
```

```
library(GGally)
library(factoextra)

# read winequality-red.csv
df_red <- read.table("./winequality-red.csv", header=TRUE, sep=";")

# read winequality-white.csv
df_white <- read.table("./winequality-white.csv", header=TRUE, sep=";")

dim(df_red)
```

```
## [1] 1599   12
```

```
dim(df_white)
```

```
## [1] 4898   12
```

```
colnames(df_red)
```

```
##  [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"
##  [4] "residual.sugar"       "chlorides"            "free.sulfur.dioxide"
##  [7] "total.sulfur.dioxide" "density"              "pH"
## [10] "sulphates"            "alcohol"              "quality"
```

```
colnames(df_white)
```

```
##  [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"
##  [4] "residual.sugar"       "chlorides"            "free.sulfur.dioxide"
##  [7] "total.sulfur.dioxide" "density"              "pH"
## [10] "sulphates"            "alcohol"              "quality"
```

```
colSums(is.na(df_red))
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##                    0                    0                    0
##       residual.sugar            chlorides  free.sulfur.dioxide
##                    0                    0                    0
## total.sulfur.dioxide              density                   pH
##                    0                    0                    0
##            sulphates              alcohol              quality
##                    0                    0                    0
```

```
colSums(is.na(df_white))
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##                    0                    0                    0
##       residual.sugar            chlorides  free.sulfur.dioxide
##                    0                    0                    0
## total.sulfur.dioxide              density                   pH
##                    0                    0                    0
##            sulphates              alcohol              quality
##                    0                    0                    0
```

```
str(df_red)
```

```
## 'data.frame':    1599 obs. of  12 variables:
##  $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
##  $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
##  $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
##  $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
##  $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
##  $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
##  $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
##  $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
##  $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
##  $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
##  $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
##  $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
```

```
str(df_white)
```

```
## '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 ...
```

```
# get the unique values of quality
unique(df_red$quality)
```

```
## [1] 5 6 7 4 8 3
```

```
unique(df_white$quality)
```

```
## [1] 6 5 7 8 4 3 9
```

```
# count of quality red-wine
table(df_red$quality)
```

```
##
##   3   4   5   6   7   8
##  10  53 681 638 199  18
```

```
# count of quality white-wine
table(df_white$quality)
```

```
##
##    3    4    5    6    7    8    9
##   20  163 1457 2198  880  175    5
```

```
# get the correlation values
cor(df_red)
```

```
##                       fixed.acidity volatile.acidity citric.acid residual.sugar
## fixed.acidity           1.00000000     -0.256130895  0.67170343    0.114776724
## volatile.acidity       -0.25613089      1.000000000 -0.55249568    0.001917882
## citric.acid             0.67170343     -0.552495685  1.00000000    0.143577162
## residual.sugar          0.11477672      0.001917882  0.14357716    1.000000000
## chlorides               0.09370519      0.061297772  0.20382291    0.055609535
## free.sulfur.dioxide    -0.15379419     -0.010503827 -0.06097813    0.187048995
## total.sulfur.dioxide   -0.11318144      0.076470005  0.03553302    0.203027882
## density                 0.66804729      0.022026232  0.36494718    0.355283371
## pH                     -0.68297819      0.234937294 -0.54190414   -0.085652422
## sulphates               0.18300566     -0.260986685  0.31277004    0.005527121
## alcohol                -0.06166827     -0.202288027  0.10990325    0.042075437
## quality                 0.12405165     -0.390557780  0.22637251    0.013731637
##                       chlorides free.sulfur.dioxide total.sulfur.dioxide
## fixed.acidity         0.093705186        -0.153794193          -0.11318144
## volatile.acidity      0.061297772        -0.010503827           0.07647000
## citric.acid           0.203822914        -0.060978129           0.03553302
## residual.sugar        0.055609535         0.187048995           0.20302788
## chlorides             1.000000000         0.005562147           0.04740047
## free.sulfur.dioxide   0.005562147         1.000000000           0.66766645
## total.sulfur.dioxide  0.047400468         0.667666450           1.00000000
## density               0.200632327        -0.021945831           0.07126948
## pH                   -0.265026131         0.070377499          -0.06649456
## sulphates             0.371260481         0.051657572           0.04294684
## alcohol              -0.221140545        -0.069408354          -0.20565394
## quality              -0.128906560        -0.050656057          -0.18510029
##                          density          pH    sulphates     alcohol
## fixed.acidity         0.66804729 -0.68297819  0.183005664 -0.06166827
## volatile.acidity      0.02202623  0.23493729 -0.260986685 -0.20228803
## citric.acid           0.36494718 -0.54190414  0.312770044  0.10990325
## residual.sugar        0.35528337 -0.08565242  0.005527121  0.04207544
## chlorides             0.20063233 -0.26502613  0.371260481 -0.22114054
## free.sulfur.dioxide  -0.02194583  0.07037750  0.051657572 -0.06940835
## total.sulfur.dioxide  0.07126948 -0.06649456  0.042946836 -0.20565394
## density               1.00000000 -0.34169933  0.148506412 -0.49617977
## pH                   -0.34169933  1.00000000 -0.196647602  0.20563251
## sulphates             0.14850641 -0.19664760  1.000000000  0.09359475
## alcohol              -0.49617977  0.20563251  0.093594750  1.00000000
## quality              -0.17491923 -0.05773139  0.251397079  0.47616632
##                          quality
## fixed.acidity         0.12405165
## volatile.acidity     -0.39055778
## citric.acid           0.22637251
## residual.sugar        0.01373164
## chlorides            -0.12890656
## free.sulfur.dioxide  -0.05065606
## total.sulfur.dioxide -0.18510029
## density              -0.17491923
## pH                   -0.05773139
## sulphates             0.25139708
```

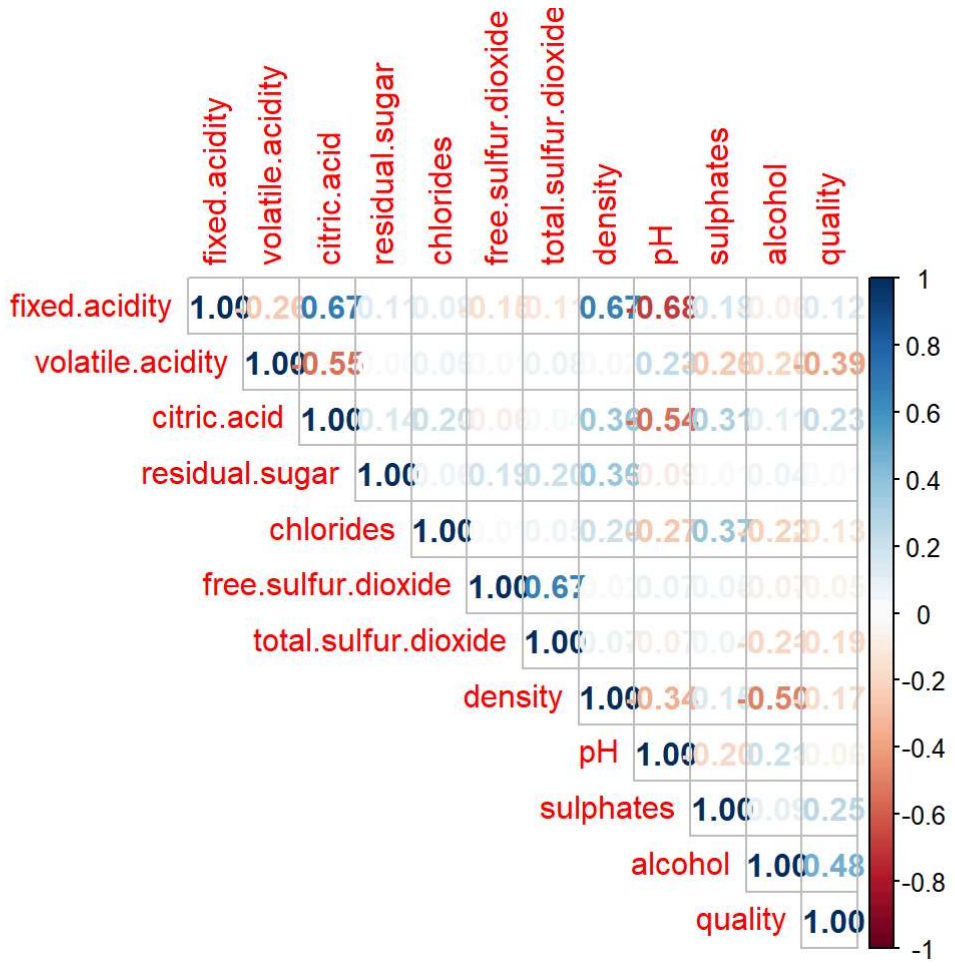
```
## alcohol              0.47616632
## quality              1.00000000
```

```
cor(df_white)
```

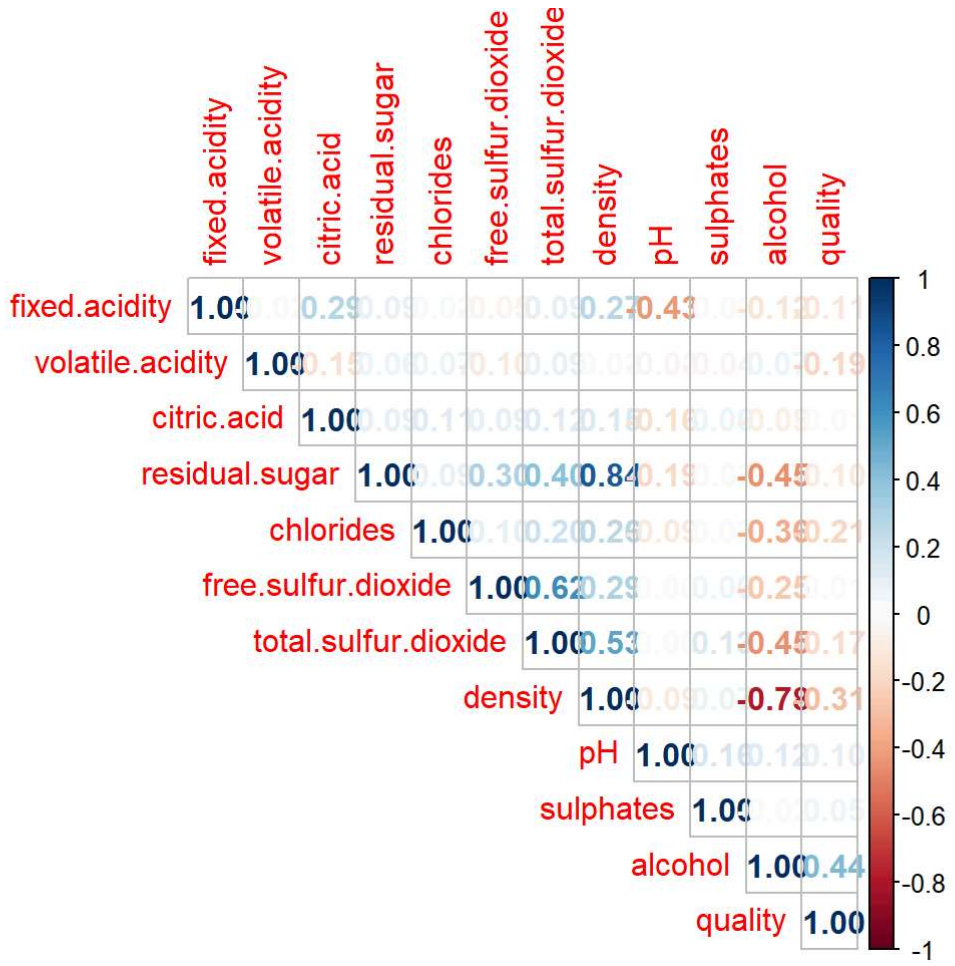
```
##                       fixed.acidity volatile.acidity  citric.acid residual.sugar
## fixed.acidity           1.00000000      -0.02269729  0.289180698     0.08902070
## volatile.acidity       -0.02269729       1.00000000 -0.149471811     0.06428606
## citric.acid             0.28918070      -0.14947181  1.000000000     0.09421162
## residual.sugar          0.08902070       0.06428606  0.094211624     1.00000000
## chlorides               0.02308564       0.07051157  0.114364448     0.08868454
## free.sulfur.dioxide    -0.04939586      -0.09701194  0.094077221     0.29909835
## total.sulfur.dioxide    0.09106976       0.08926050  0.121130798     0.40143931
## density                 0.26533101       0.02711385  0.149502571     0.83896645
## pH                     -0.42585829      -0.03191537 -0.163748211    -0.19413345
## sulphates              -0.01714299      -0.03572815  0.062330940    -0.02666437
## alcohol                -0.12088112       0.06771794 -0.075728730    -0.45063122
## quality                -0.11366283      -0.19472297 -0.009209091    -0.09757683
##                        chlorides free.sulfur.dioxide total.sulfur.dioxide
## fixed.acidity         0.02308564       -0.0493958591          0.091069756
## volatile.acidity      0.07051157       -0.0970119393          0.089260504
## citric.acid           0.11436445        0.0940772210          0.121130798
## residual.sugar        0.08868454        0.2990983537          0.401439311
## chlorides             1.00000000        0.1013923521          0.198910300
## free.sulfur.dioxide   0.10139235        1.0000000000          0.615500965
## total.sulfur.dioxide  0.19891030        0.6155009650          1.000000000
## density               0.25721132        0.2942104109          0.529881324
## pH                   -0.09043946       -0.0006177961          0.002320972
## sulphates             0.01676288        0.0592172458          0.134562367
## alcohol              -0.36018871       -0.2501039415         -0.448892102
## quality              -0.20993441        0.0081580671         -0.174737218
##                          density            pH   sulphates     alcohol
## fixed.acidity         0.26533101 -0.4258582910 -0.01714299 -0.12088112
## volatile.acidity      0.02711385 -0.0319153683 -0.03572815  0.06771794
## citric.acid           0.14950257 -0.1637482114  0.06233094 -0.07572873
## residual.sugar        0.83896645 -0.1941334540 -0.02666437 -0.45063122
## chlorides             0.25721132 -0.0904394560  0.01676288 -0.36018871
## free.sulfur.dioxide   0.29421041 -0.0006177961  0.05921725 -0.25010394
## total.sulfur.dioxide  0.52988132  0.0023209718  0.13456237 -0.44889210
## density               1.00000000 -0.0935914935  0.07449315 -0.78013762
## pH                   -0.09359149  1.0000000000  0.15595150  0.12143210
## sulphates             0.07449315  0.1559514973  1.00000000 -0.01743277
## alcohol              -0.78013762  0.1214320987 -0.01743277  1.00000000
## quality              -0.30712331  0.0994272457  0.05367788  0.43557472
##                           quality
## fixed.acidity        -0.113662831
## volatile.acidity     -0.194722969
## citric.acid          -0.009209091
## residual.sugar       -0.097576829
## chlorides            -0.209934411
## free.sulfur.dioxide   0.008158067
## total.sulfur.dioxide -0.174737218
## density              -0.307123313
## pH                    0.099427246
## sulphates             0.053677877
```

```
## alcohol              0.435574715
## quality              1.000000000
```

```
# correlation plots of red & white wine
corrplot(cor(df_red), method="number", type="upper")
```




```
corrplot(cor(df_white), method="number", type="upper")
```



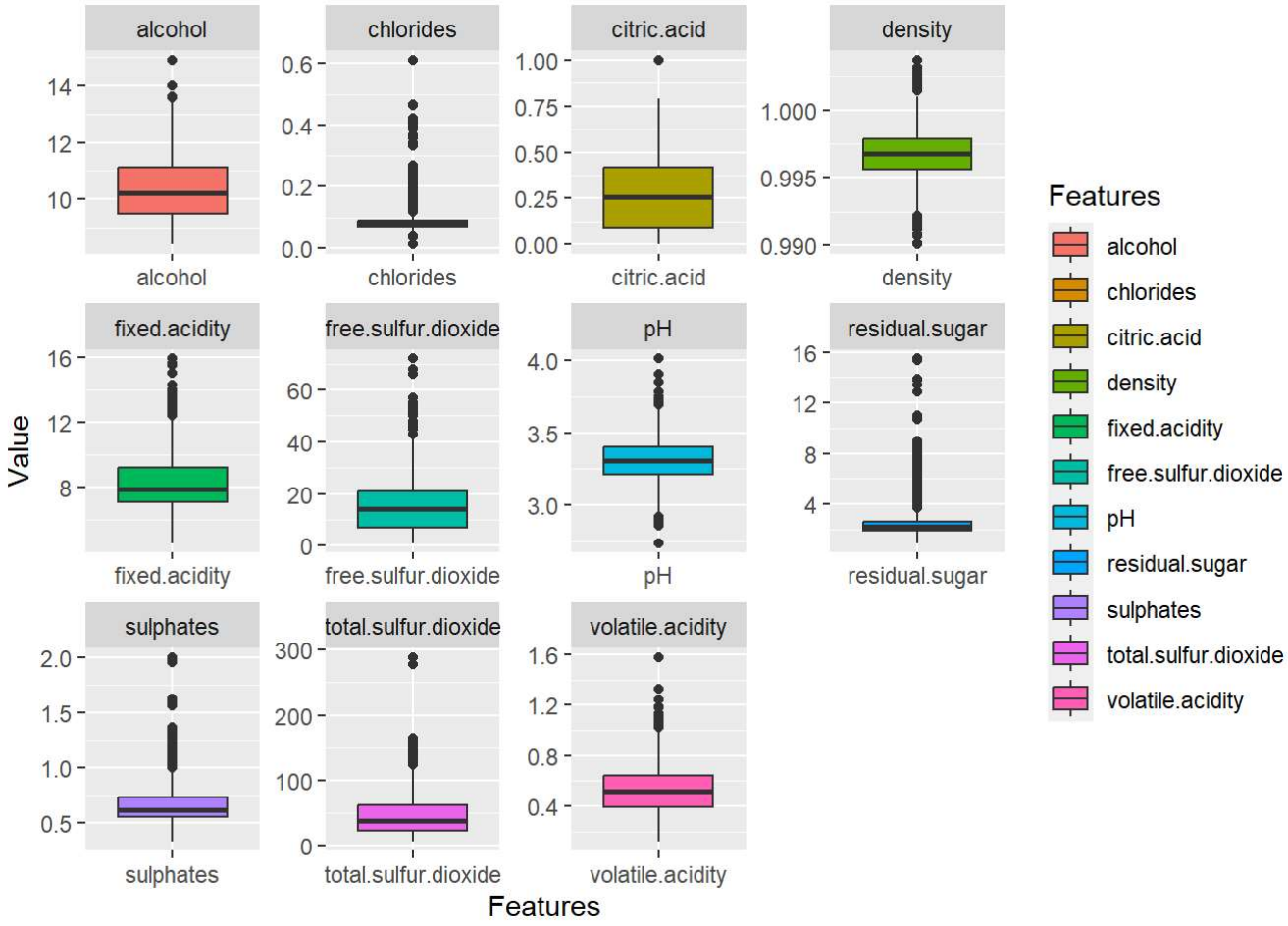

```
# Create a function to calculate the number of outliers
count_outliers <- function(f) {
  outliers <- lapply(f, function(x) {
    Q1 <- quantile(x, 0.25)
    Q3 <- quantile(x, 0.75)
    IQR <- Q3 - Q1
    lower_bound <- Q1 - 1.5 * IQR
    upper_bound <- Q3 + 1.5 * IQR
    sum(x < lower_bound | x > upper_bound)
})
return(outliers)
}

# print the number of outliers of df_red
count_outliers(df_red)
```

```
## $fixed.acidity
## [1] 49
##
## $volatile.acidity
## [1] 19
##
## $citric.acid
## [1] 1
##
## $residual.sugar
## [1] 155
##
## $chlorides
## [1] 112
##
## $free.sulfur.dioxide
## [1] 30
##
## $total.sulfur.dioxide
## [1] 55
##
## $density
## [1] 45
##
## $pH
## [1] 35
##
## $sulphates
## [1] 59
##
## $alcohol
## [1] 13
##
## $quality
## [1] 28
```

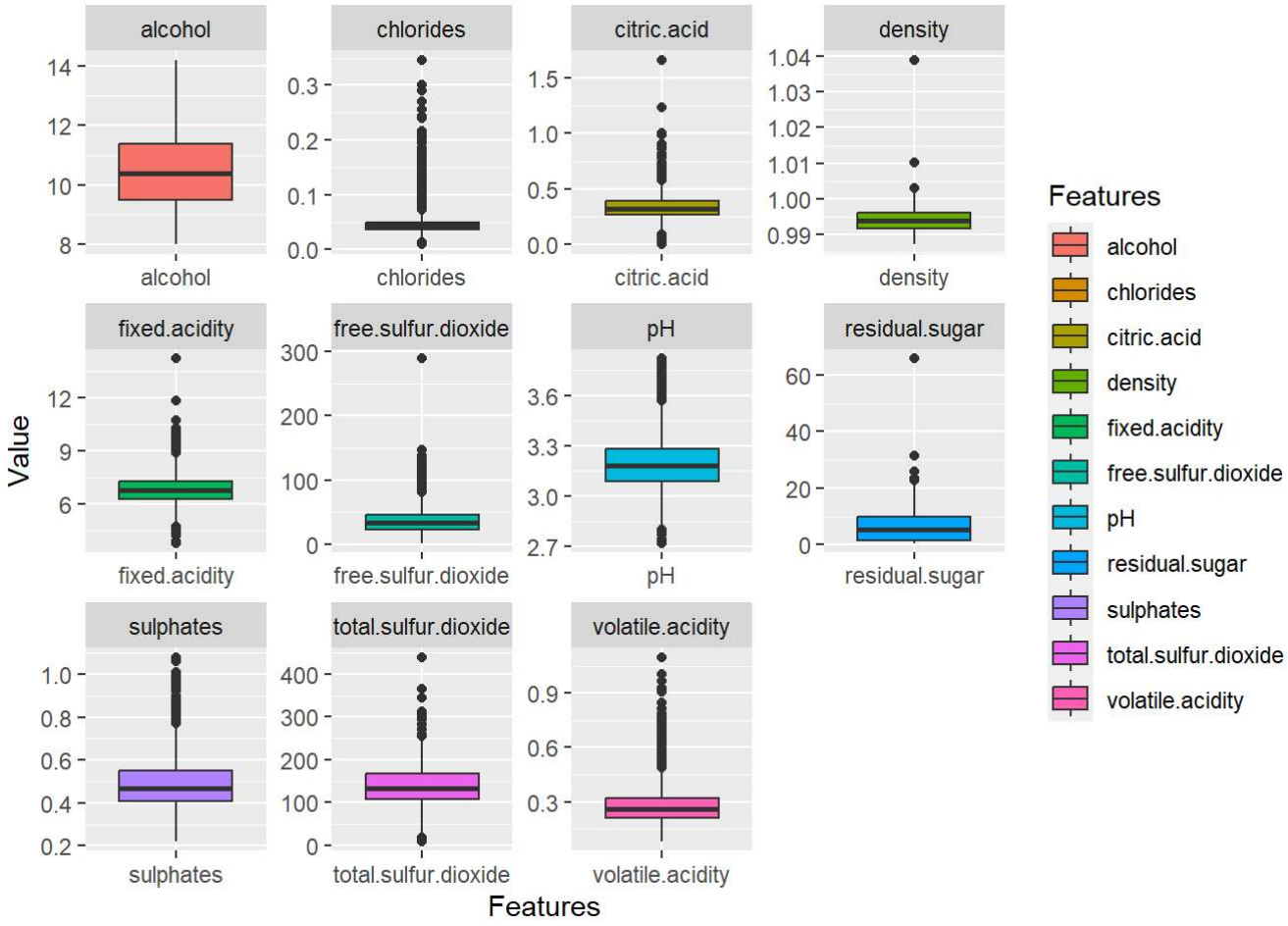
```
# print the number of outliers of df_White
count_outliers(df_white)
```

```
## $fixed.acidity
## [1] 119
##
## $volatile.acidity
## [1] 186
##
## $citric.acid
## [1] 270
##
## $residual.sugar
## [1] 7
##
## $chlorides
## [1] 208
##
## $free.sulfur.dioxide
## [1] 50
##
## $total.sulfur.dioxide
## [1] 19
##
## $density
## [1] 5
##
## $pH
## [1] 75
##
## $sulphates
## [1] 124
##
## $alcohol
## [1] 0
##
## $quality
## [1] 200
```

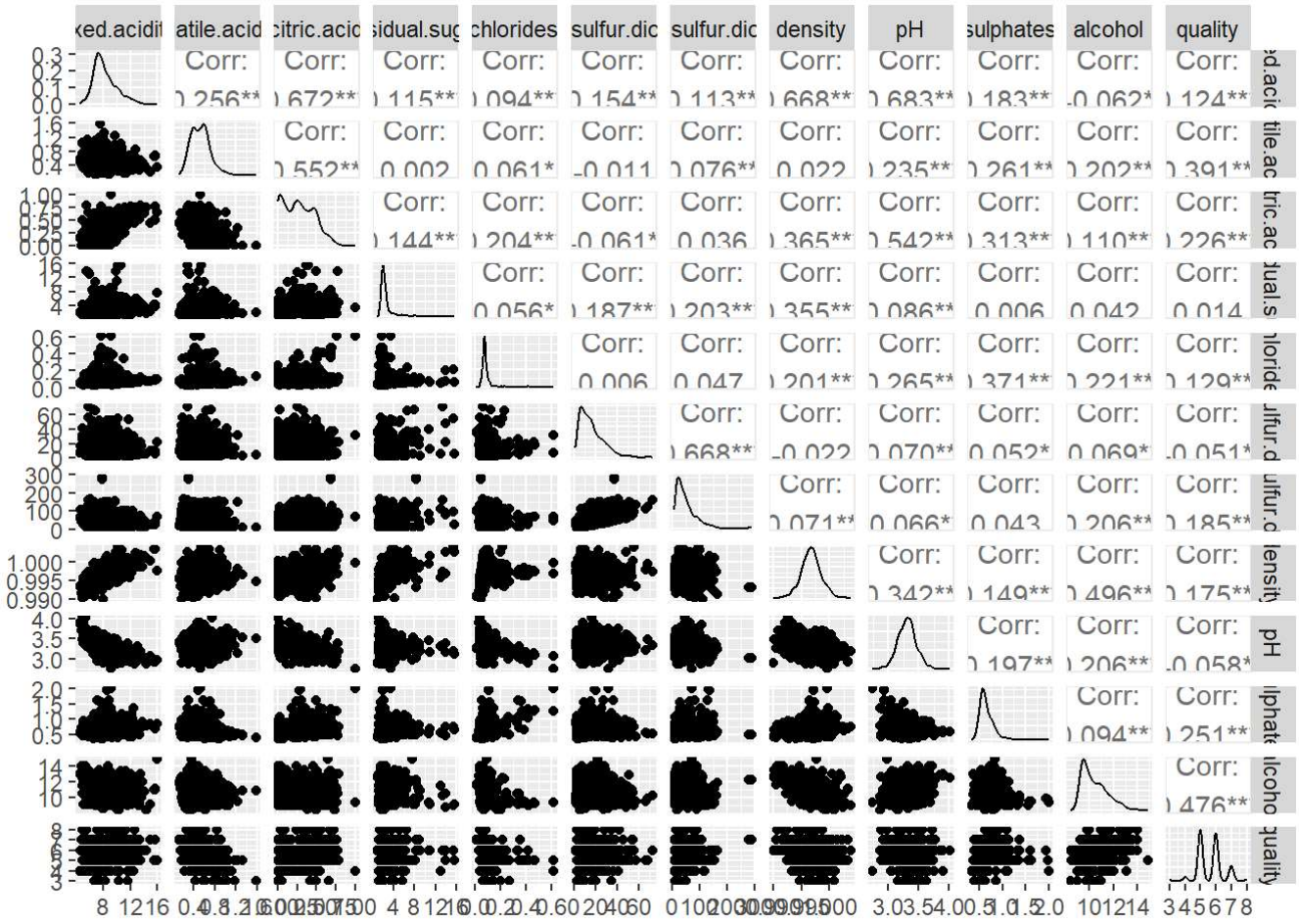
```
# plotting the box plots for each variable of red wine
df_red %>%
  gather(key = "Features", value = "Value", -quality) %>%
  ggplot(aes(x = Features, y = Value, fill = Features)) +
  geom_boxplot() +
  facet_wrap(~Features, scales = "free")
```



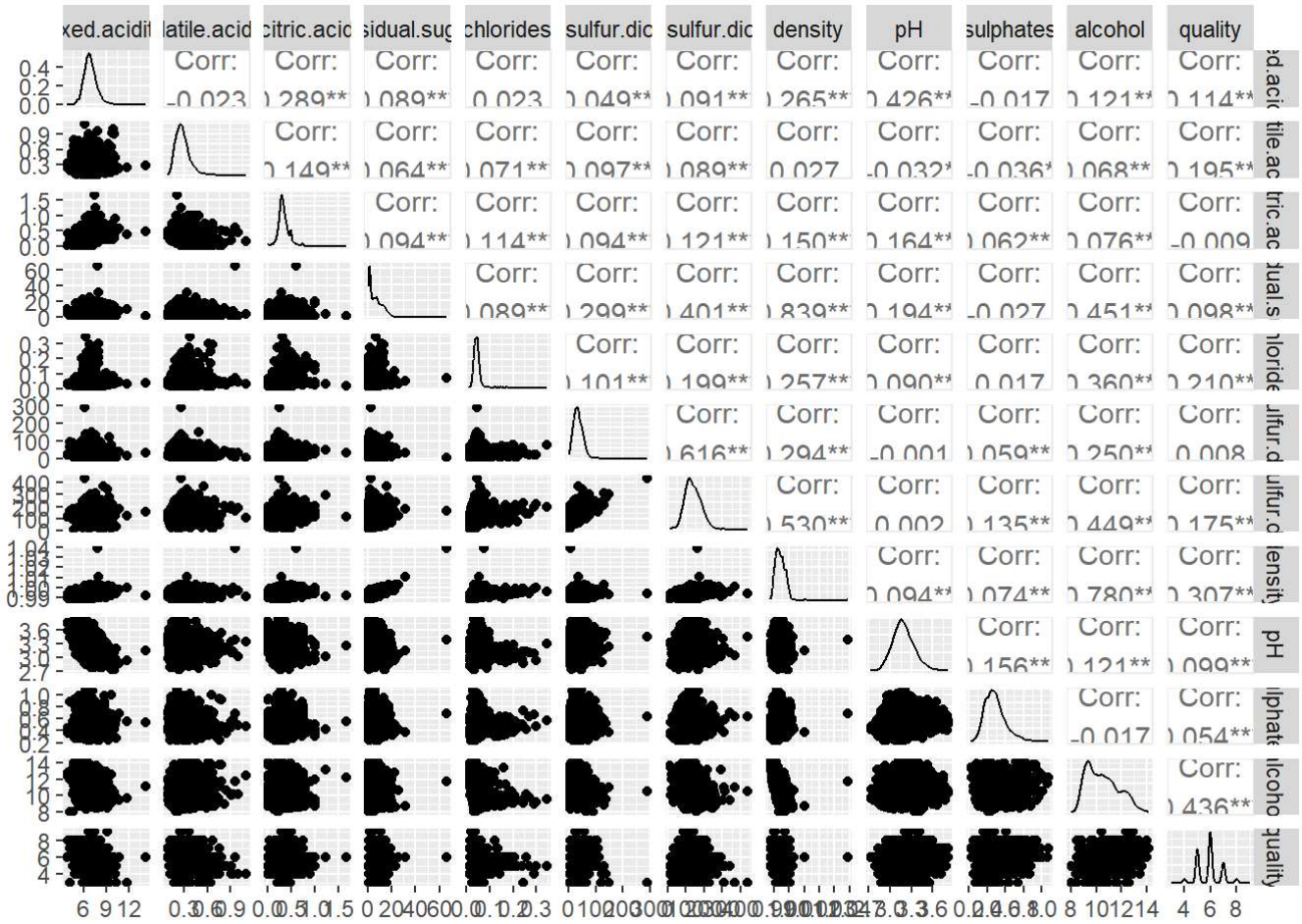

```
# plotting the box plots for each variable of white wine
df_white %>%
  gather(key = "Features", value = "Value", -quality) %>%
  ggplot(aes(x = Features, y = Value, fill = Features)) +
  geom_boxplot() +
  facet_wrap(~Features, scales = "free")
```




```
# Scatter plots
ggpairs(df_red)
```




```
ggpairs(df_white)
```



Mostly all the features in both red and white wine there are some outliers in the data.

b) Perform k-means using Principal Components of the wine data. Justify your choice of “k”. Visualize the result using a biplot and color the points (samples) according to “wine color”

```
set.seed(123)

# creating new column named color and assigning 2 to red color and 1 for white color.
df_w <- df_white
df_r <- df_red

df_w$color <- 1
df_r$color <- 2

wine_data <- rbind(df_r,df_w)
dim(wine_data)
```

```
## [1] 6497   13
```

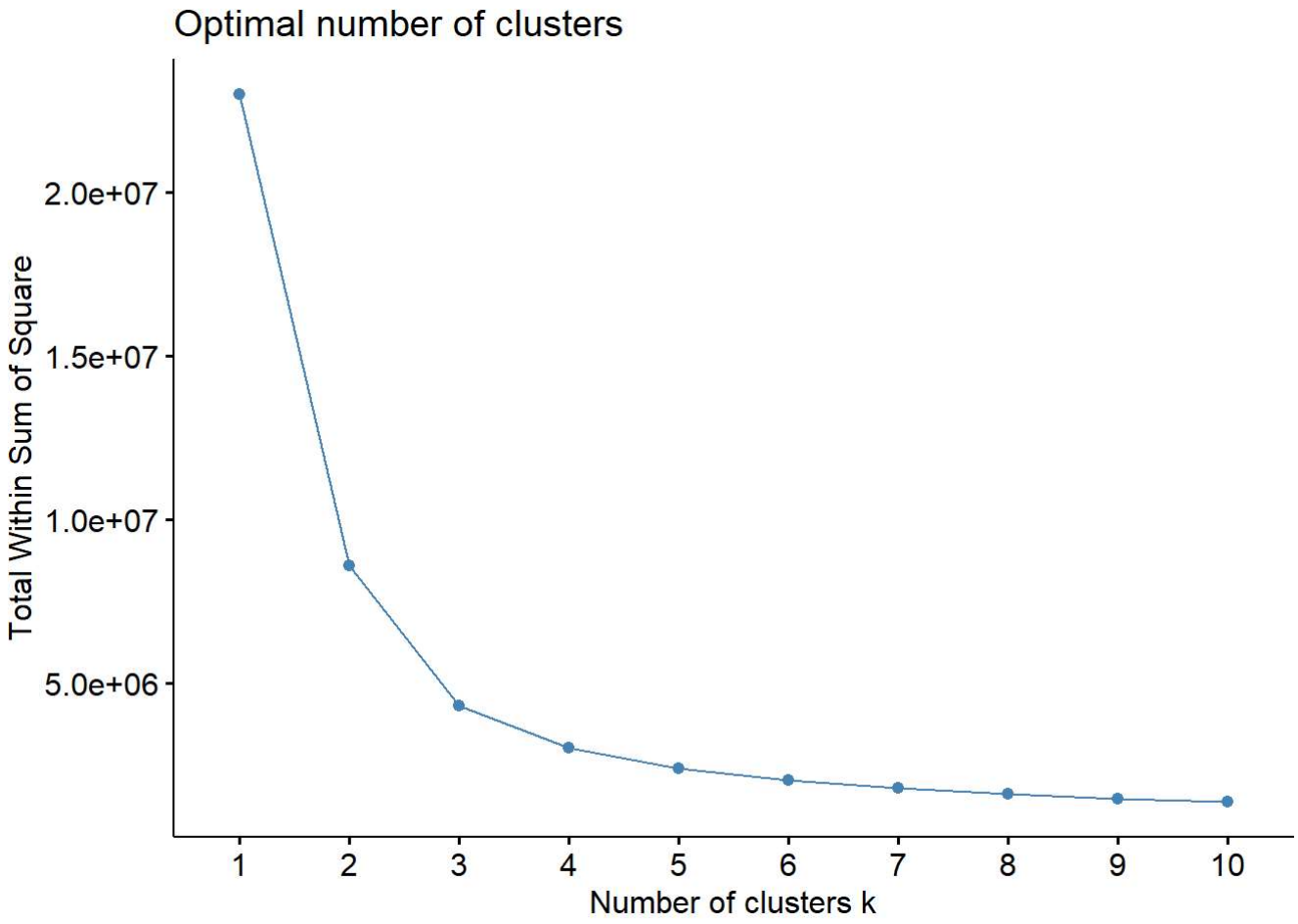
```
unique(wine_data$color)
```

```
## [1] 2 1
```

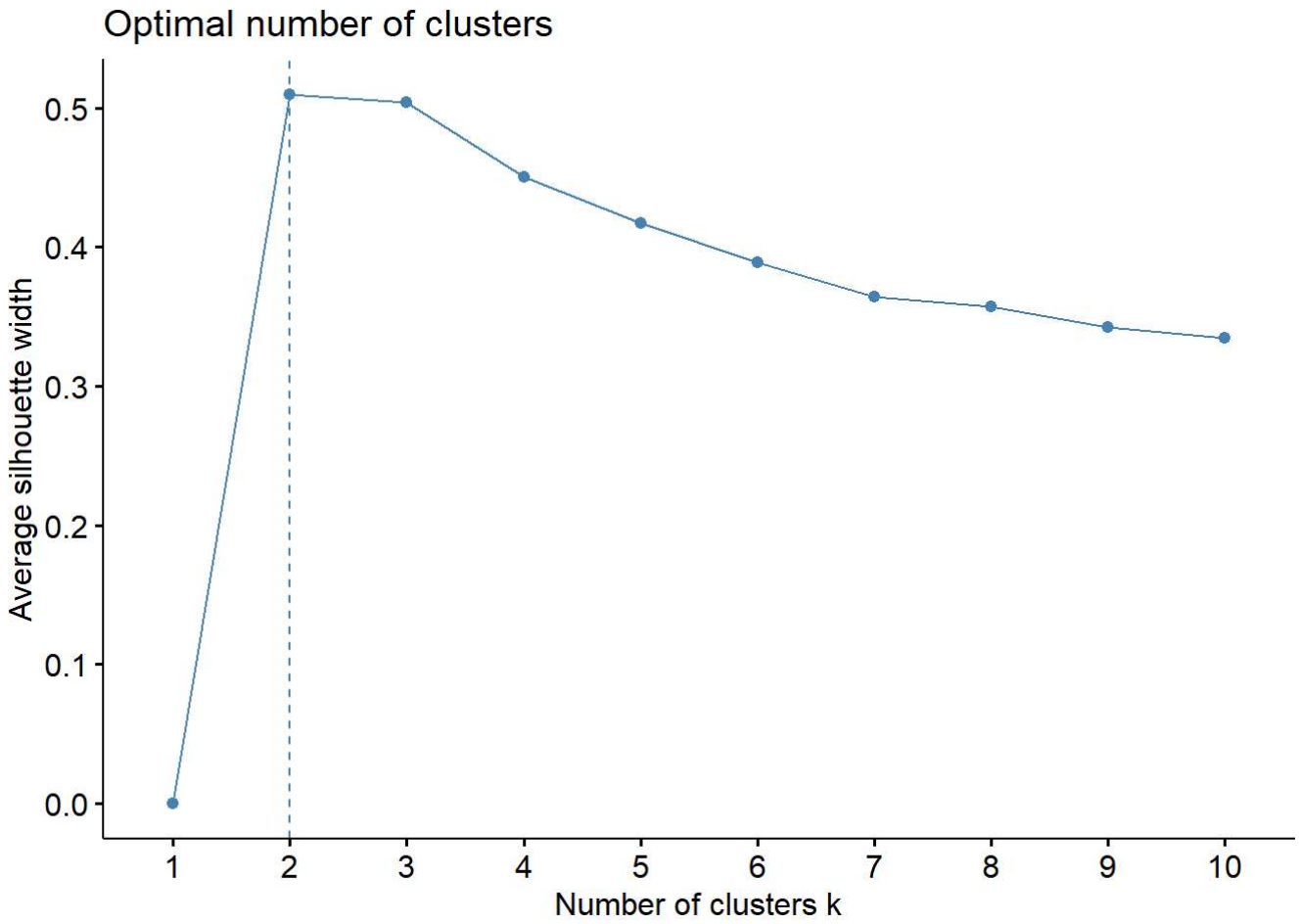
```
table(wine_data$color)
```

```
##
##    1    2
## 4898 1599
```

```
# elbow method
fviz_nbclust(wine_data[, -13], kmeans, method = "wss")
```



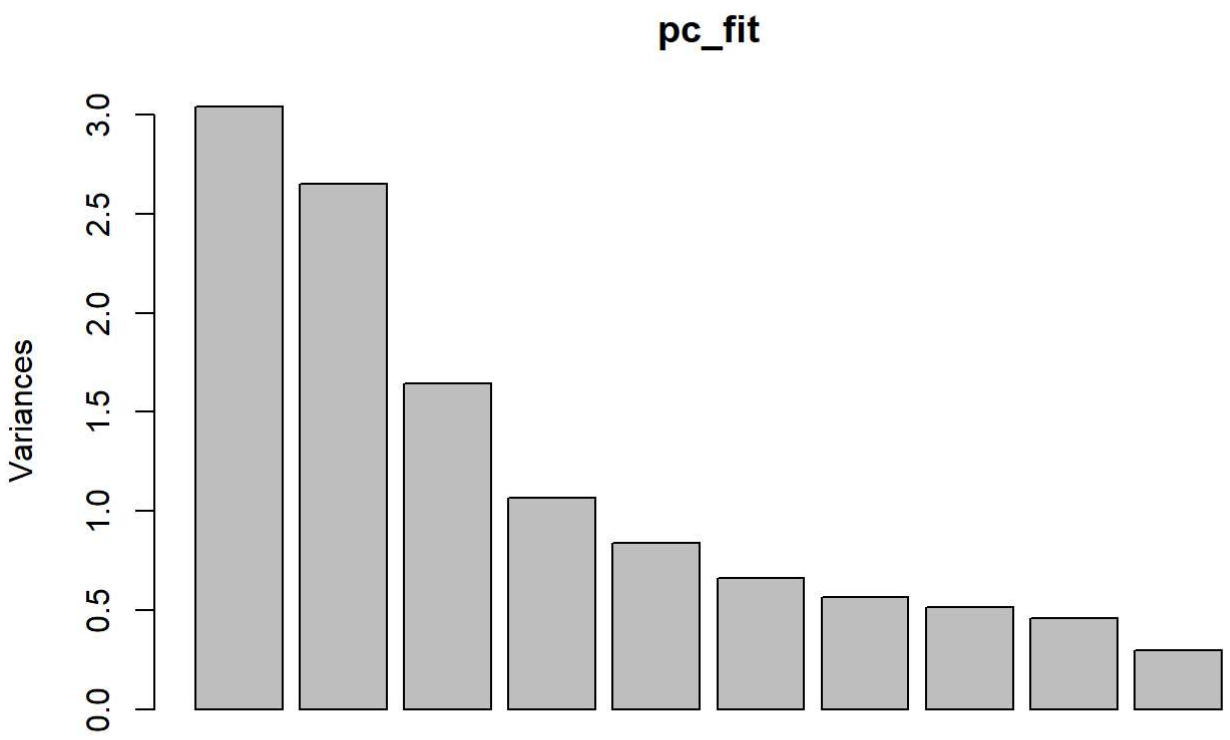

```
# silhouette method
fviz_nbclust(wine_data[, -13], kmeans, method = "silhouette")
```




```
scaled_data <- scale(wine_data[, -13])

pc_fit <- prcomp(scaled_data)
pc_fit
```

```
## Standard deviations (1, .., p=12):
##  [1] 1.7440032 1.6278372 1.2812130 1.0337433 0.9167881 0.8126495 0.7508838
##  [8] 0.7183195 0.6770320 0.5468207 0.4770613 0.1810667
##
## Rotation (n x k) = (12 x 12):
##                              PC1        PC2         PC3         PC4
## fixed.acidity        -0.25692873 -0.2618431 -0.46748619 -0.14396377
## volatile.acidity     -0.39493118 -0.1051983  0.27968932 -0.08005785
## citric.acid           0.14646061 -0.1440935 -0.58807557  0.05551036
## residual.sugar        0.31890519 -0.3425850  0.07550170  0.11245623
## chlorides            -0.31344994 -0.2697701 -0.04676921  0.16529004
## free.sulfur.dioxide   0.42269137 -0.1111788  0.09899801  0.30330631
## total.sulfur.dioxide  0.47441968 -0.1439475  0.10128143  0.13223199
## density              -0.09243753 -0.5549205  0.05156338  0.15057853
## pH                   -0.20806957  0.1529219  0.40678741  0.47147768
## sulphates            -0.29985192 -0.1196342 -0.16869128  0.58801992
## alcohol              -0.05892408  0.4927275 -0.21293142  0.08003179
## quality               0.08747571  0.2966009 -0.29583773  0.47243936
##                              PC5         PC6         PC7          PC8
## fixed.acidity         0.165362611  0.03003708 -0.39343530  0.001155415
## volatile.acidity      0.147774077 -0.38266373 -0.44511080  0.310077574
## citric.acid          -0.234621394  0.36224839 -0.04769762  0.444962196
## residual.sugar        0.507921181 -0.06331719  0.09576310  0.081944829
## chlorides            -0.393896604 -0.42544212  0.47329609  0.375531558
## free.sulfur.dioxide  -0.248451958 -0.28318017 -0.36271398  0.120097626
## total.sulfur.dioxide -0.223966811 -0.10676882 -0.23481304  0.011279269
## density               0.330357303  0.15455292 -0.01328572  0.042943625
## pH                   -0.001457502  0.56089714 -0.07932113  0.362281828
## sulphates            -0.193245549 -0.02014082 -0.17023612 -0.592220645
## alcohol               0.116023190 -0.16947538 -0.33890566  0.226040866
## quality               0.459129140 -0.27788835  0.27317740  0.093046206
##                              PC9        PC10         PC11          PC12
## fixed.acidity        -0.42416913 -0.27243245  0.276932032  0.3350925313
## volatile.acidity      0.12323319  0.49394624 -0.140799120  0.0824207437
## citric.acid           0.24623265  0.33035538 -0.229276179 -0.0013466535
## residual.sugar        0.48802361 -0.20717448 -0.005144195  0.4512148285
## chlorides             0.04404975 -0.23887258  0.193398397  0.0432778637
## free.sulfur.dioxide  -0.30139683 -0.30344826 -0.486158400  0.0009050256
## total.sulfur.dioxide -0.00181386  0.29478016  0.720162498 -0.0640632122
## density              -0.07108094 -0.07681483  0.003324491 -0.7156665472
## pH                   -0.13666158 -0.11240888  0.139083153  0.2067626762
## sulphates             0.29740108  0.08546912 -0.047219109  0.0781995574
## alcohol               0.41706026 -0.41605905  0.191289020 -0.3320120790
## quality              -0.35664731  0.30782956  0.018082206 -0.0082880344
```

```
plot(pc_fit)
```




```
pc_fit$rotation # loadings
```

```
##                              PC1        PC2         PC3         PC4
## fixed.acidity        -0.25692873 -0.2618431 -0.46748619 -0.14396377
## volatile.acidity     -0.39493118 -0.1051983  0.27968932 -0.08005785
## citric.acid           0.14646061 -0.1440935 -0.58807557  0.05551036
## residual.sugar        0.31890519 -0.3425850  0.07550170  0.11245623
## chlorides            -0.31344994 -0.2697701 -0.04676921  0.16529004
## free.sulfur.dioxide   0.42269137 -0.1111788  0.09899801  0.30330631
## total.sulfur.dioxide  0.47441968 -0.1439475  0.10128143  0.13223199
## density              -0.09243753 -0.5549205  0.05156338  0.15057853
## pH                   -0.20806957  0.1529219  0.40678741  0.47147768
## sulphates            -0.29985192 -0.1196342 -0.16869128  0.58801992
## alcohol              -0.05892408  0.4927275 -0.21293142  0.08003179
## quality               0.08747571  0.2966009 -0.29583773  0.47243936
##                              PC5         PC6         PC7          PC8
## fixed.acidity         0.165362611  0.03003708 -0.39343530  0.001155415
## volatile.acidity      0.147774077 -0.38266373 -0.44511080  0.310077574
## citric.acid          -0.234621394  0.36224839 -0.04769762  0.444962196
## residual.sugar        0.507921181 -0.06331719  0.09576310  0.081944829
## chlorides            -0.393896604 -0.42544212  0.47329609  0.375531558
## free.sulfur.dioxide  -0.248451958 -0.28318017 -0.36271398  0.120097626
## total.sulfur.dioxide -0.223966811 -0.10676882 -0.23481304  0.011279269
## density               0.330357303  0.15455292 -0.01328572  0.042943625
## pH                   -0.001457502  0.56089714 -0.07932113  0.362281828
## sulphates            -0.193245549 -0.02014082 -0.17023612 -0.592220645
## alcohol               0.116023190 -0.16947538 -0.33890566  0.226040866
## quality               0.459129140 -0.27788835  0.27317740  0.093046206
##                              PC9        PC10         PC11          PC12
## fixed.acidity        -0.42416913 -0.27243245  0.276932032  0.3350925313
## volatile.acidity      0.12323319  0.49394624 -0.140799120  0.0824207437
## citric.acid           0.24623265  0.33035538 -0.229276179 -0.0013466535
## residual.sugar        0.48802361 -0.20717448 -0.005144195  0.4512148285
## chlorides             0.04404975 -0.23887258  0.193398397  0.0432778637
## free.sulfur.dioxide  -0.30139683 -0.30344826 -0.486158400  0.0009050256
## total.sulfur.dioxide -0.00181386  0.29478016  0.720162498 -0.0640632122
## density              -0.07108094 -0.07681483  0.003324491 -0.7156665472
## pH                   -0.13666158 -0.11240888  0.139083153  0.2067626762
## sulphates             0.29740108  0.08546912 -0.047219109  0.0781995574
## alcohol               0.41706026 -0.41605905  0.191289020 -0.3320120790
## quality              -0.35664731  0.30782956  0.018082206 -0.0082880344
```

```
# pc_fit$x # for scores

per_var_expl <- 100*((pc_fit$sdev)^2)/(sum(((pc_fit$sdev)^2)))
per_var_expl
```

```
##  [1] 25.3462261 22.0821166 13.6792235  8.9052105  7.0041705  5.5033265
##  [7]  4.6985537  4.2998570  3.8197690  2.4917742  1.8965627  0.2732097
```

```
PC1 <- pc_fit$x[,1]
PC2 <- pc_fit$x[,2]
PC3 <- pc_fit$x[,3]
PC_dats <- cbind(PC1, PC2, PC3)

## As per the elbow method and silhouette method k = 2.

km <- kmeans(PC_dats, centers = 2, nstart = 20)
km
```

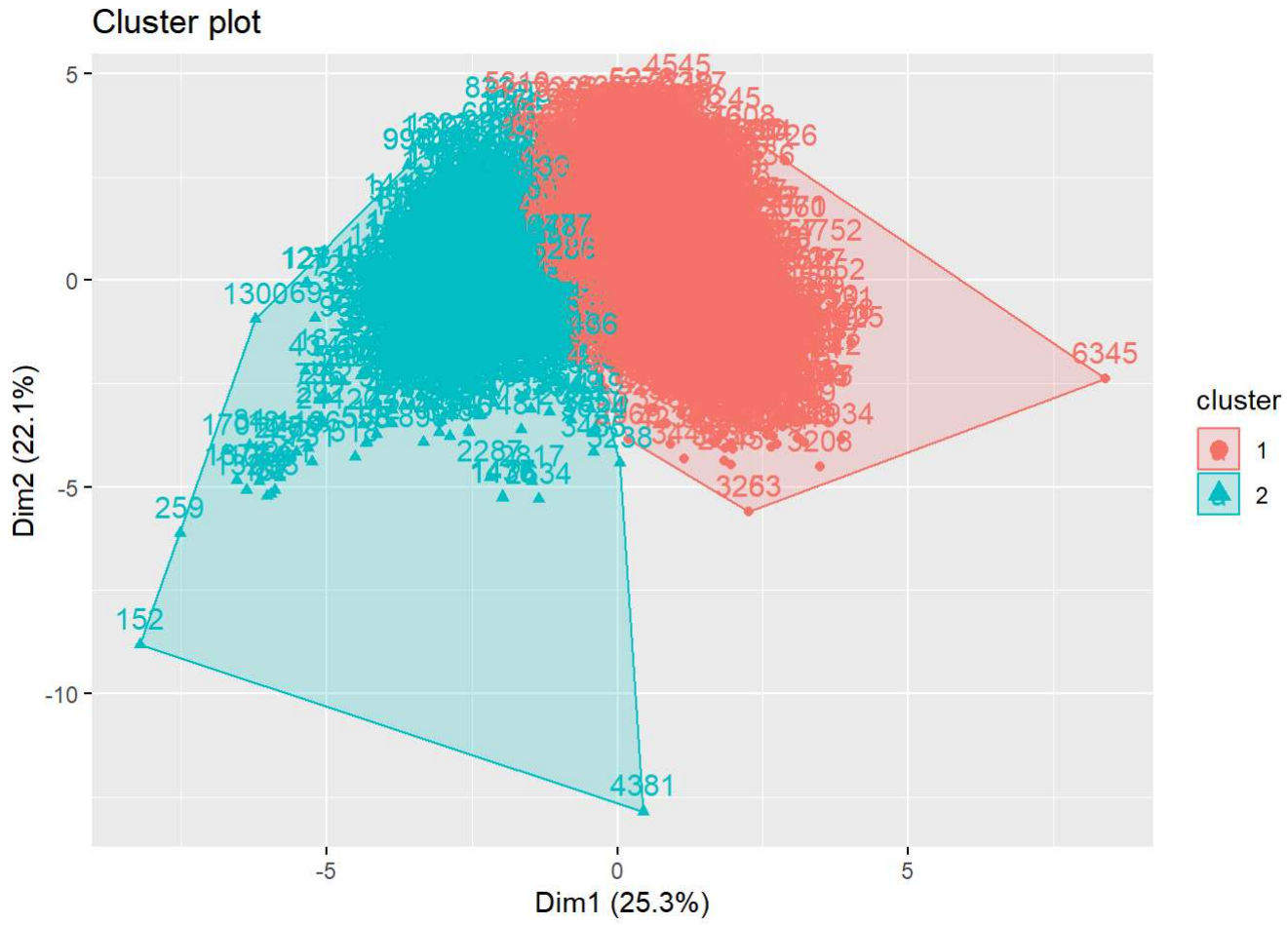
```
## K-means clustering with 2 clusters of sizes 4786, 1711
##
## Cluster means:
##          PC1        PC2         PC3
## 1  0.8942524  0.2054713 -0.02723989
## 2 -2.5013980 -0.5747433  0.07619527
##
## Clustering vector:
##    [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##   [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##   [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [149] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [186] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [223] 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [260] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [297] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [334] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2
##  [371] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [408] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [445] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [482] 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [519] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [556] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1
##  [593] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [630] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [667] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [704] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [741] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [778] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [815] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2
##  [852] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [889] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [926] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [963] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1000] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1037] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1074] 2 2 2 2 2 2 1 2 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1111] 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1148] 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1185] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1222] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2
## [1259] 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2
## [1296] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1333] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1370] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1407] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1444] 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1481] 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1518] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1555] 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1592] 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1
```

```
## [1629] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1666] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1703] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1740] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1777] 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
## [1814] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1851] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1888] 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1925] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1962] 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1999] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1
## [2036] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2073] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
## [2110] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2147] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2184] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
## [2221] 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2258] 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1
## [2295] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
## [2332] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2369] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2406] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1
## [2443] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1
## [2480] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1
## [2517] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1
## [2554] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
## [2591] 2 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
## [2628] 1 1 1 1 1 1 2 1 2 2 1 1 2 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
## [2665] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2702] 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2739] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2776] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2813] 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2850] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2887] 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2924] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2961] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
## [2998] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3035] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3072] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3109] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3146] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1
## [3183] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3220] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3257] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
## [3294] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3331] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3368] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3405] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1
## [3442] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3479] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3516] 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1
```

```
## [3553] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3590] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2
## [3627] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
## [3664] 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1
## [3701] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3738] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1
## [3775] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3812] 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
## [3849] 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
## [3886] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1
## [3923] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2
## [3960] 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3997] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1
## [4034] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4071] 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1
## [4108] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4145] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4182] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4219] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
## [4256] 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4293] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4330] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4367] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4404] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4441] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4478] 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4515] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4552] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4589] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
## [4626] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4663] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4700] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4737] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4774] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4811] 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4848] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4885] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4922] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4959] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4996] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5033] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5070] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5107] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5144] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1
## [5181] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5218] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5255] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5292] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5329] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5366] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1
## [5403] 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5440] 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
## [5477] 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1
## [5514] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5551] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5588] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1
## [5625] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5662] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5699] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1
## [5736] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5773] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5810] 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5847] 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5884] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
## [5921] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5958] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5995] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6032] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6069] 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6106] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6143] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6180] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1
## [6217] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1
## [6254] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1
## [6291] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6328] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6365] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1
## [6402] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1
## [6439] 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6476] 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## Within cluster sum of squares by cluster:
## [1] 21354.51 10966.31
##  (between_SS / total_SS =  32.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
fviz_cluster(km, data = wine_data[,-13])
```




```
names(km)
```

```
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

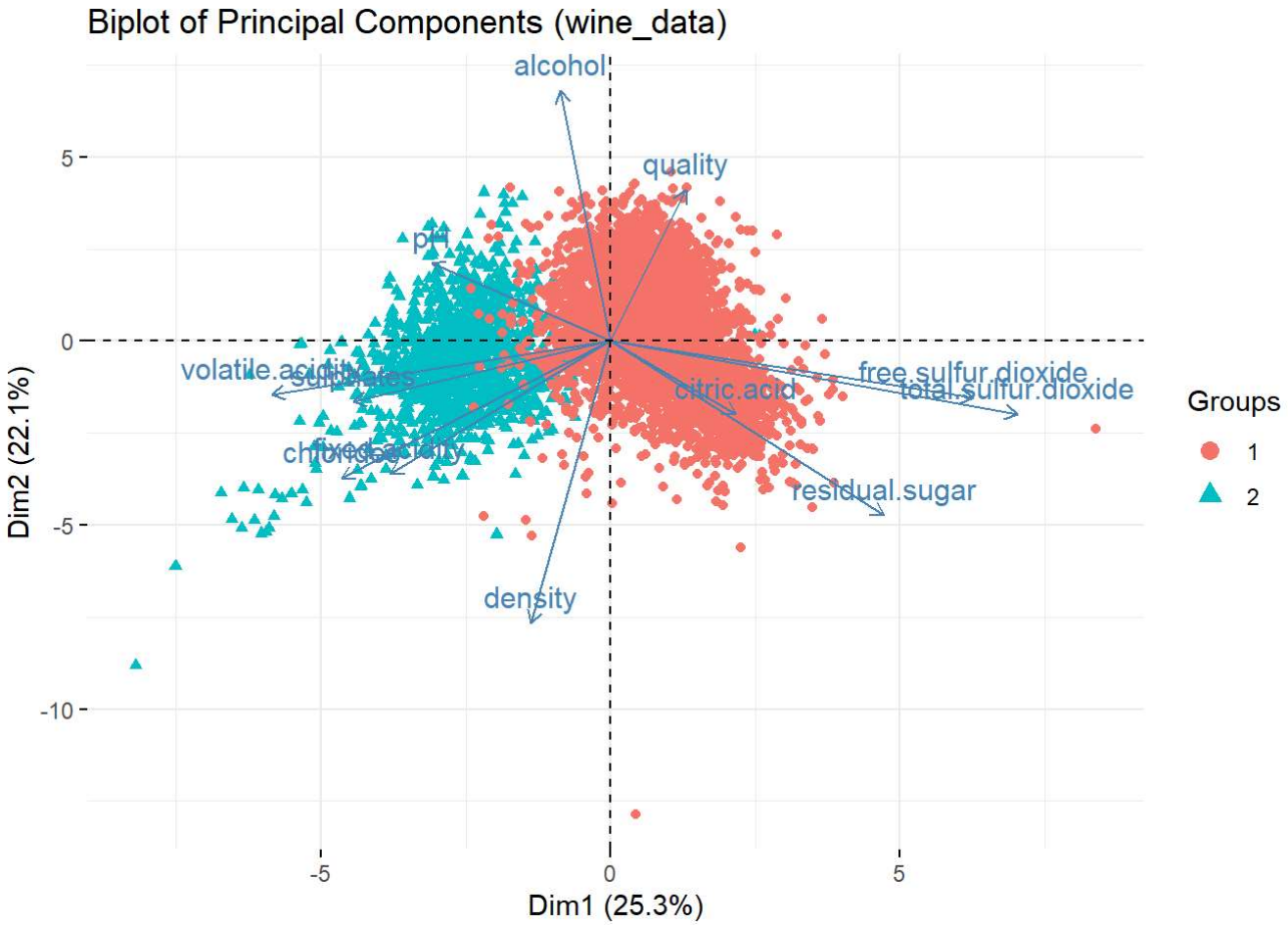
```
cm <- table(km$cluster, wine_data$color)
cm
```

```
##
##        1    2
##   1 4763   23
##   2  135 1576
```

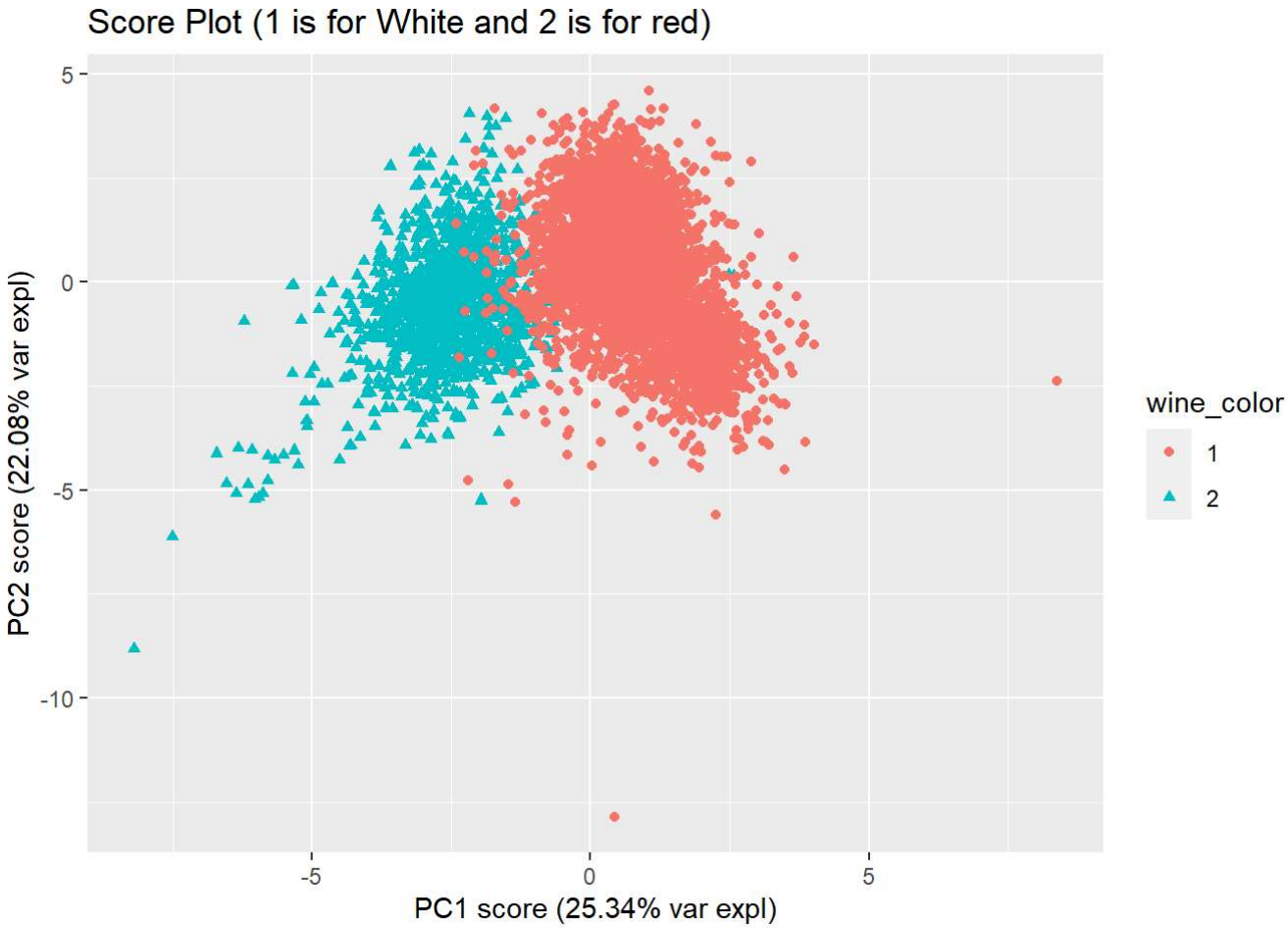
```
accuracy <- ((cm[1,1] + cm[2,2]) / (cm[1,1] + cm[1,2] + cm[2,1] + cm[2,2])) * 100
accuracy
```

```
## [1] 97.56811
```

```
# In the below plot 1 is for white and 2 is for red color.
# plotting the biplot of PCA of wine_data
fviz_pca_biplot(pc_fit,label ="var",habillage = wine_data$color,title = 'Biplot of Principal Com
ponents (wine_data)')
```




```
# Score plot in ggplot (works with data frame)
# first have to create a data frame with everything we want to use
temp <- data.frame(pc_fit$x, wine_color = wine_data$color)
names(temp)[13] <- "wine_color"

temp$wine_color <- as.factor(temp$wine_color)

ggplot(temp, aes(PC1, PC2, color = wine_color)) +
  geom_point(aes(shape = wine_color)) +
  labs(title = "Score Plot (1 is for White and 2 is for red)", x = "PC1 score (25.34% var exp
l)", y = "PC2 score (22.08% var expl)")
```



c) Fit an SOM and color the samples according to wine color. Cluster the codebook vectors of the prototypes using hclust.

```
library(kohonen)
```

```
## Warning: package 'kohonen' was built under R version 4.3.3
```

```
##
## Attaching package: 'kohonen'
```

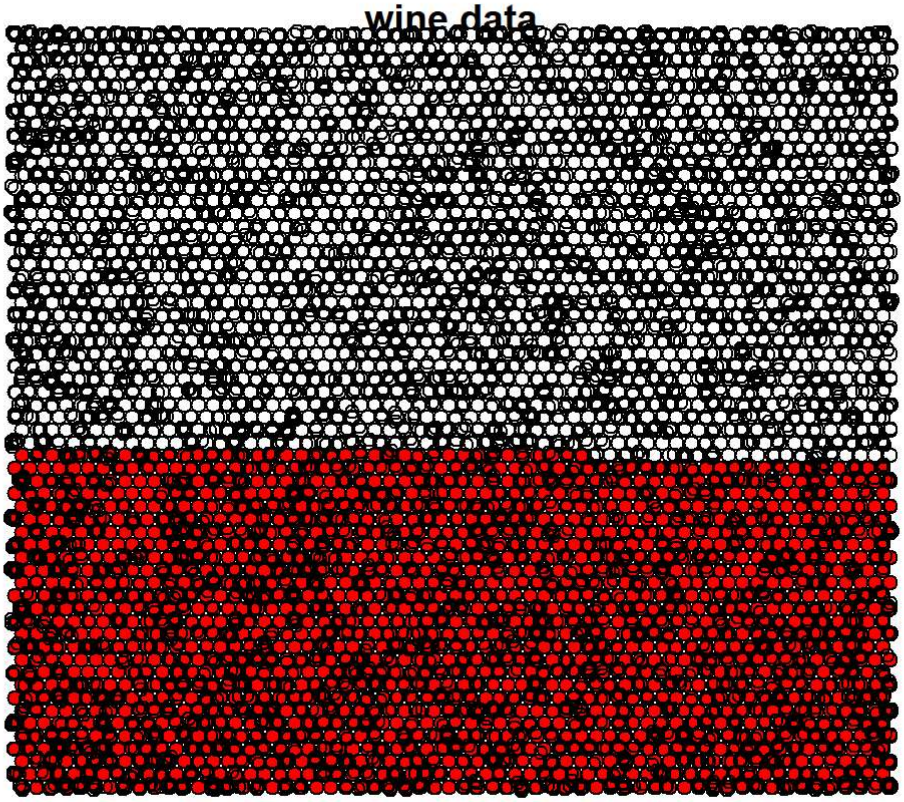
```
## The following object is masked from 'package:purrr':
##
##     map
```

```
## The following object is masked from 'package:maps':
##
##     map
```

```
red_wine <- df_red
white_wine <- df_white

wine_comb <- rbind(red_wine,white_wine)

colors <- c(rep("red", nrow(red_wine)), rep("white", nrow(white_wine)))

dim(wine_comb)
```

```
## [1] 6497   12
```

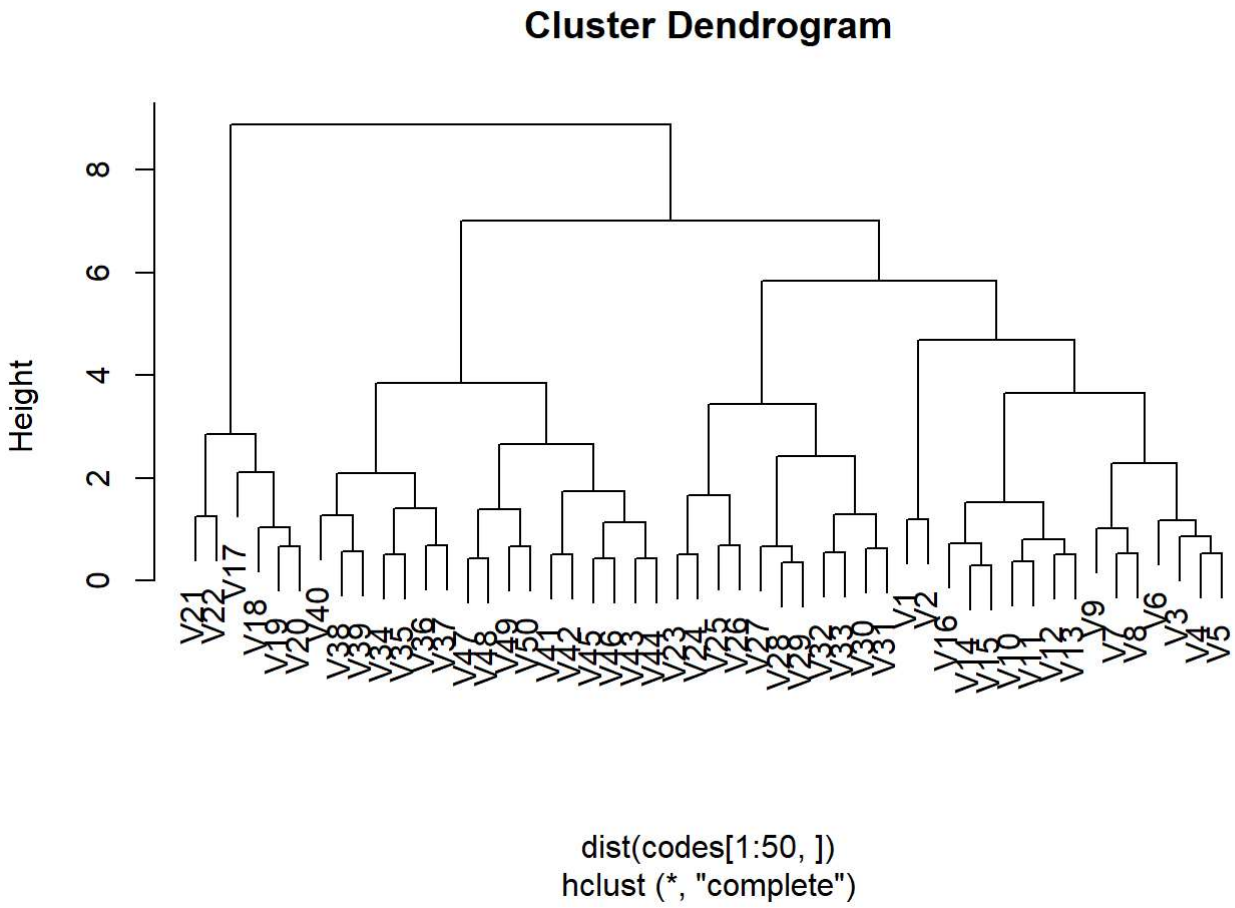
```
s_data <- scale(wine_comb)

# setting the SOM grid dimensions
grid_size <- somgrid(xdim = 60, ydim = 60, topo = "hexagonal")
# Train the SOM
som_model <- som(s_data, grid = grid_size, rlen = 100)
# Plot the SOM
plot(som_model, type = "mapping", bgcol = colors, main = "wine data")
```



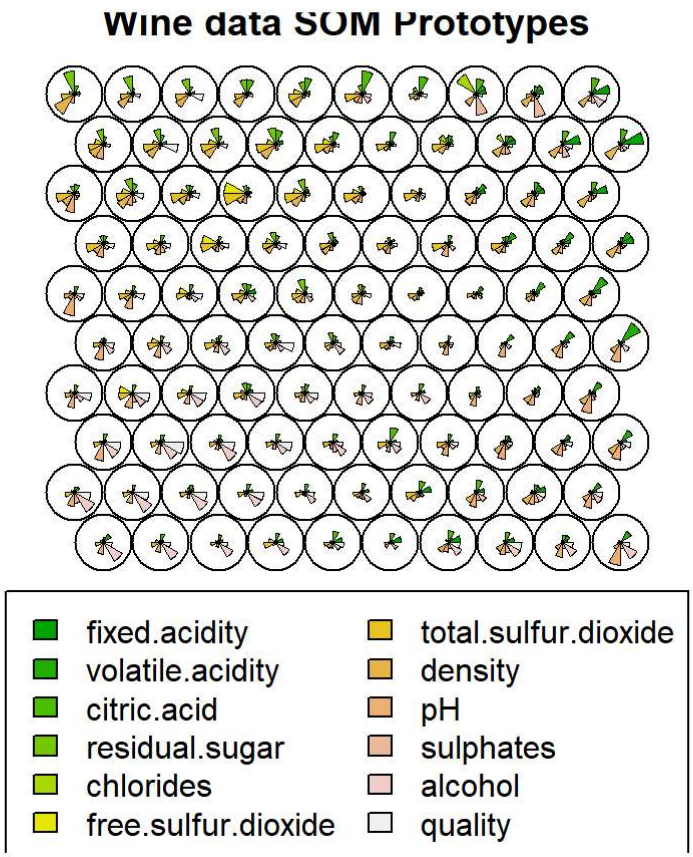

```
som_model
```

```
## SOM of size 60x60 with a hexagonal topology.
## Training data included.
```

```
codes <- som_model$codes[[1]]

# For visualization purpose considering few codebook vectors

hc <- hclust(dist(codes[1:50, ]))
plot(hc)
```




```
# setting the SOM grid dimensions
grid_size <- somgrid(xdim = 10, ydim = 10, topo = "hexagonal")
# Train the SOM
som_model <- som(s_data, grid = grid_size, rlen = 100)
# Plotting the prototypes
plot(som_model, type = "codes", main = "Wine data SOM Prototypes")
```



d) Construct phase-plots (aka component planes) for some of the variables in the dataset.

```
names(som_model)
```

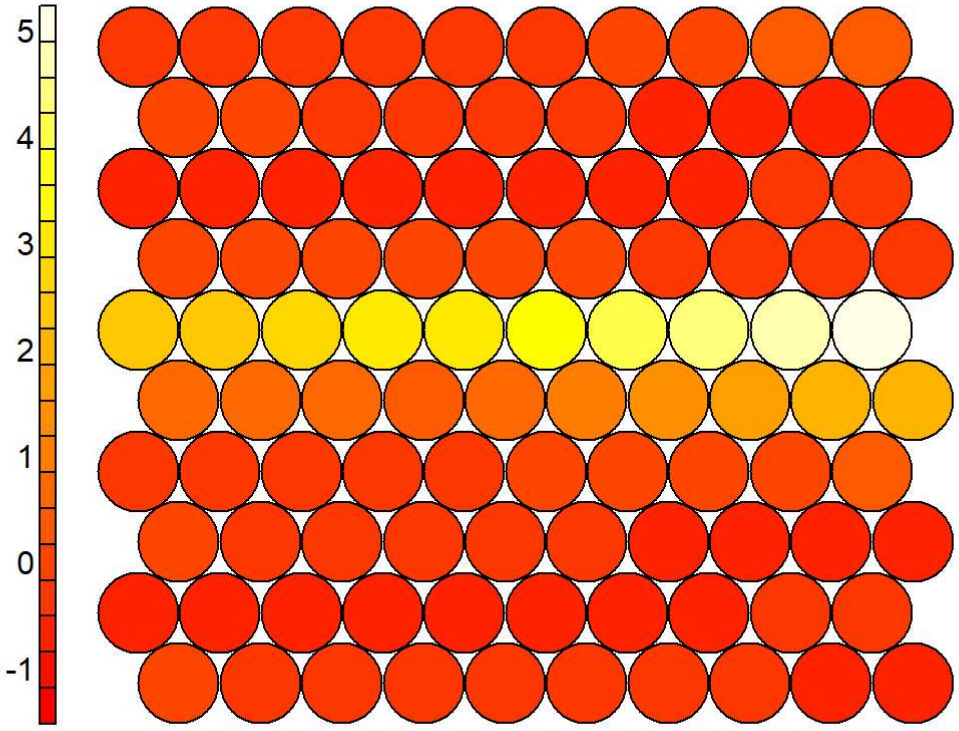
```
##  [1] "data"            "unit.classif"    "distances"       "grid"
##  [5] "codes"           "changes"         "alpha"           "radius"
##  [9] "na.rows"         "user.weights"    "distance.weights" "whatmap"
## [13] "maxNA.fraction"  "dist.fcts"
```

```
head(codes)
```

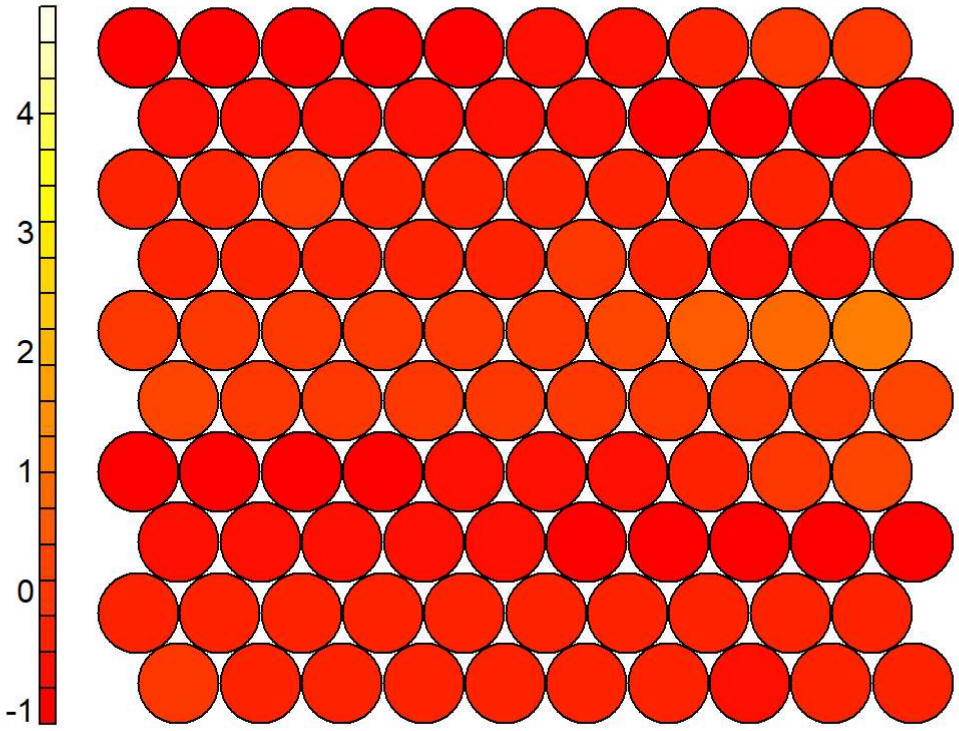
```
##    fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
## V1    -0.1371414       -0.1669339    4.086906       1.516808 -0.4526692
## V2    -0.2230146       -0.3064257    3.262887       1.455703 -0.3719233
## V3    -0.2857109       -0.4042784    2.474562       1.594279 -0.3203321
## V4    -0.2698292       -0.5102869    2.040348       1.763845 -0.2841697
## V5    -0.2540497       -0.3382549    1.696867       1.655322 -0.2614144
## V6    -0.2648464       -0.2692166    1.461132       1.386459 -0.2098688
##    free.sulfur.dioxide total.sulfur.dioxide   density         pH  sulphates
## V1           0.4247877            0.8404623 0.4629391 -0.3083396 -0.5234720
## V2           0.4133393            0.7278511 0.6720522 -0.6432947 -0.5538839
## V3           0.2784315            0.5179366 0.9181502 -1.0581398 -0.5268833
## V4           0.2036618            0.4033813 1.0153733 -1.4034981 -0.4603518
## V5           0.3949701            0.3815304 0.9809173 -1.1661977 -0.3274451
## V6           0.6615708            0.4877148 0.8795665 -1.0794574 -0.4490075
##       alcohol     quality
## V1  0.1665643  0.03999726
## V2 -0.5467526 -0.11213018
## V3 -1.1256000 -0.26645029
## V4 -1.1822874 -0.39654445
## V5 -1.1636009 -0.42404482
## V6 -1.2086809 -0.61198762
```

```
# phase or component plane plots for different variables
for (i in 1:ncol(codes)) {
  plot(som_model, type = "property", property=codes[, i], main = colnames(codes)[i])
}
```

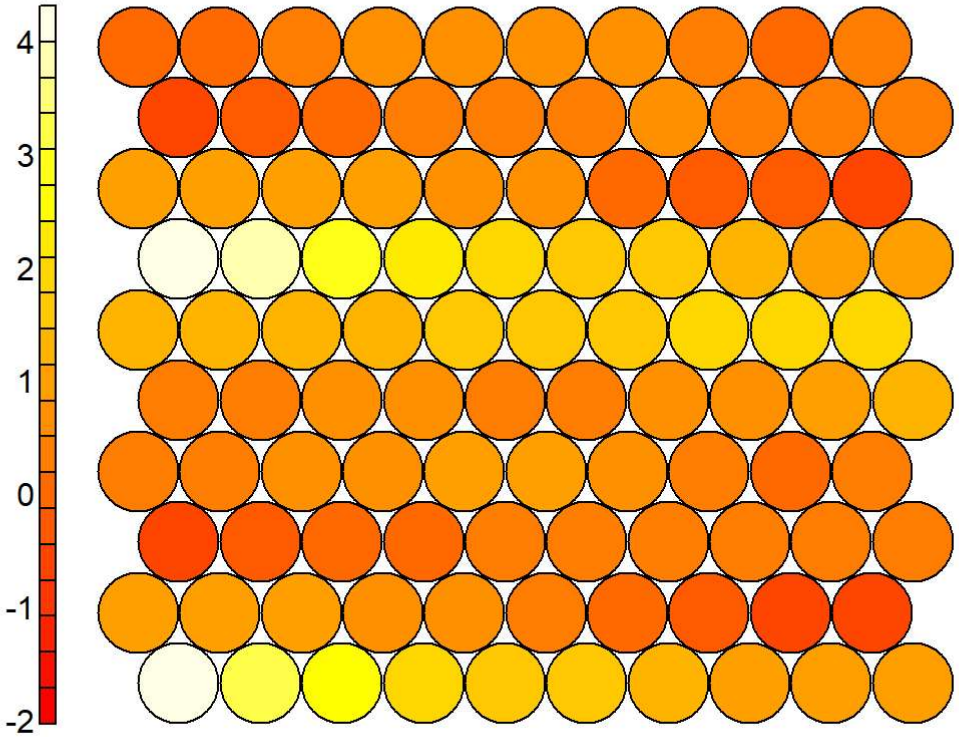
fixed.acidity



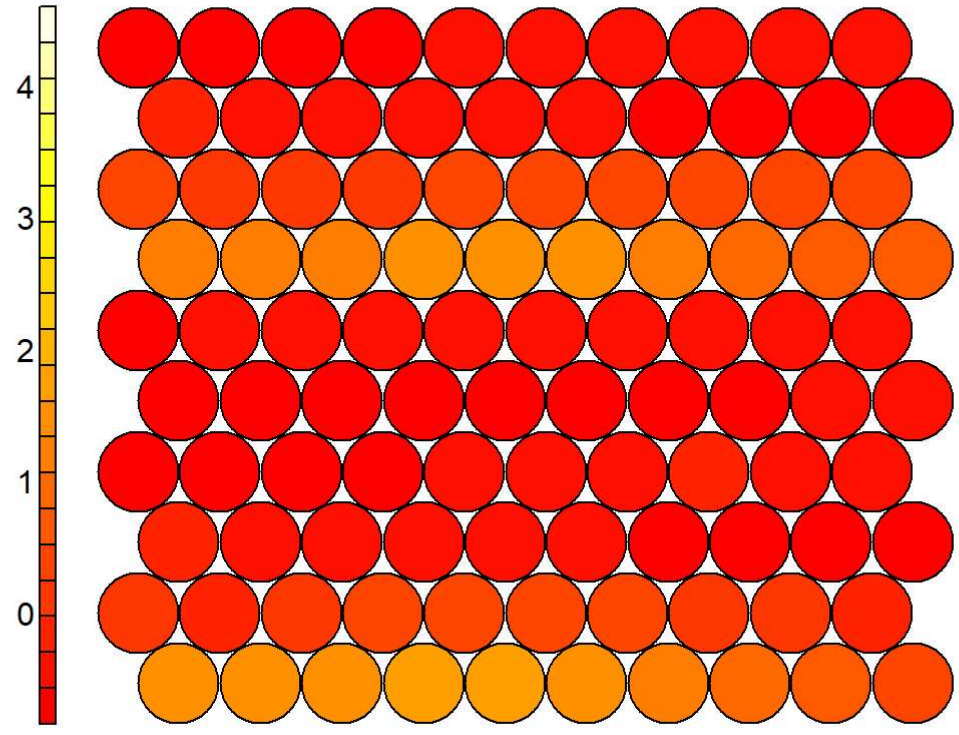
volatile.acidity



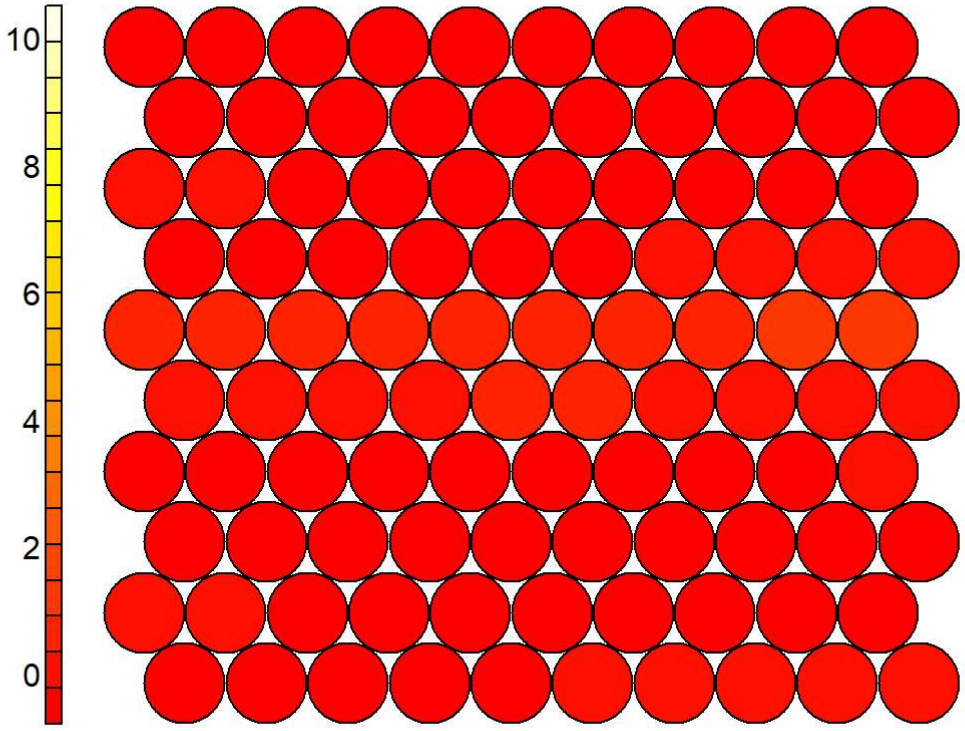
citric.acid



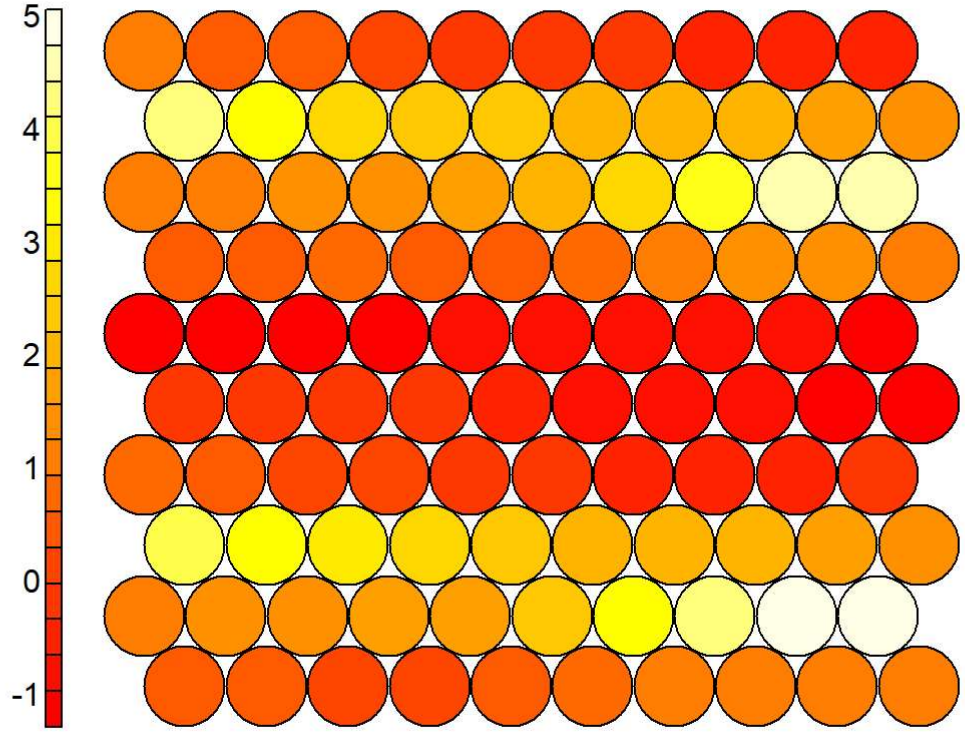
residual.sugar



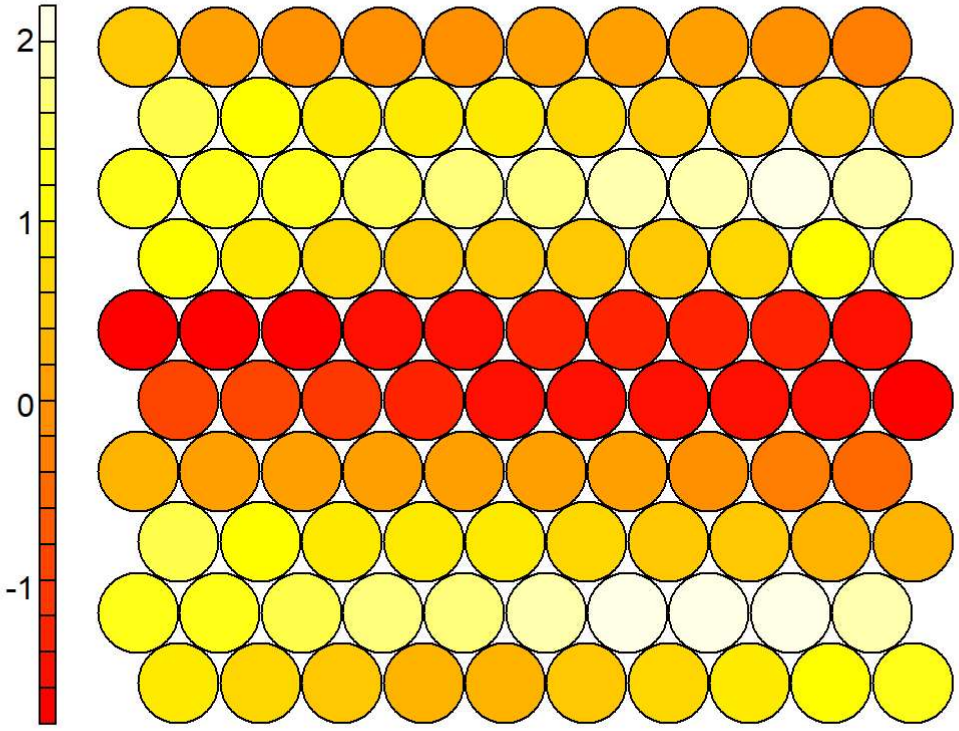
chlorides



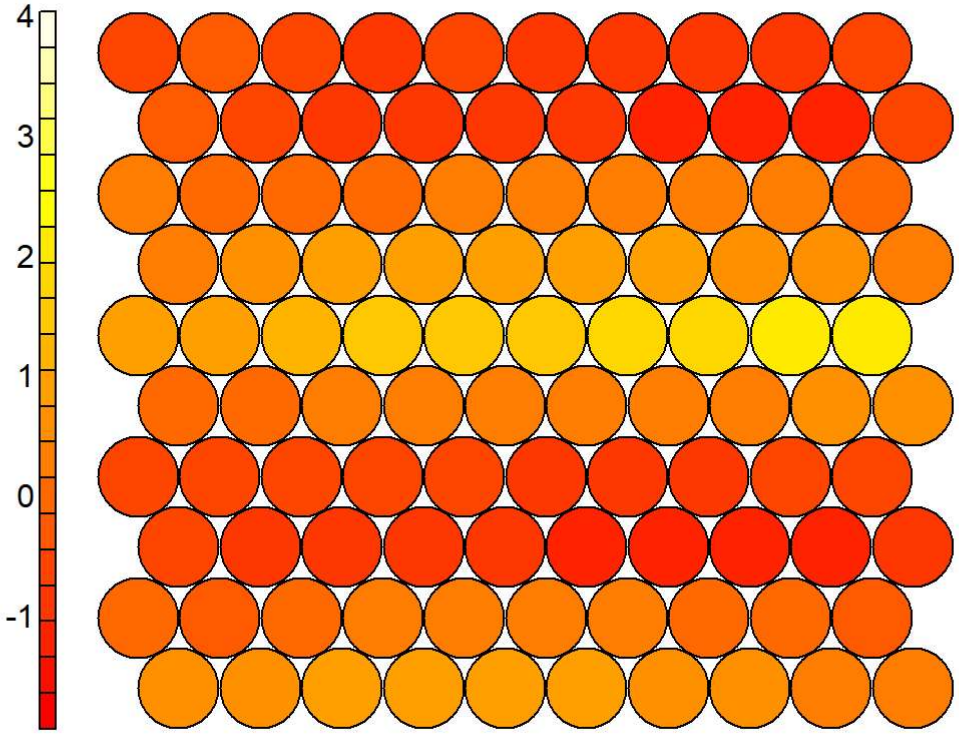
free.sulfur.dioxide



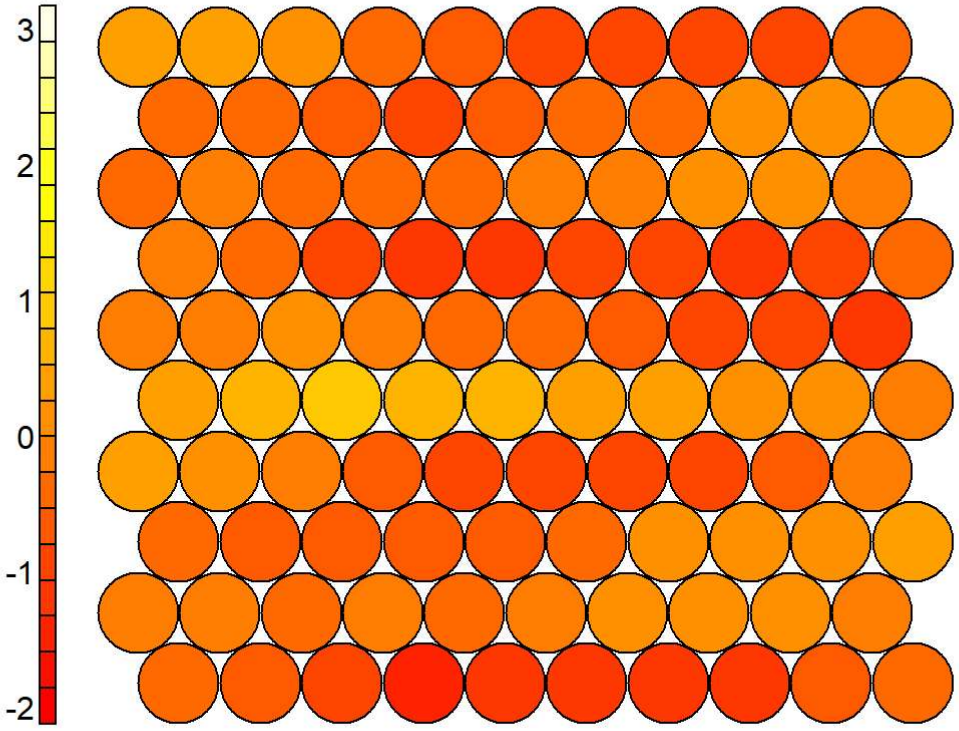
total.sulfur.dioxide



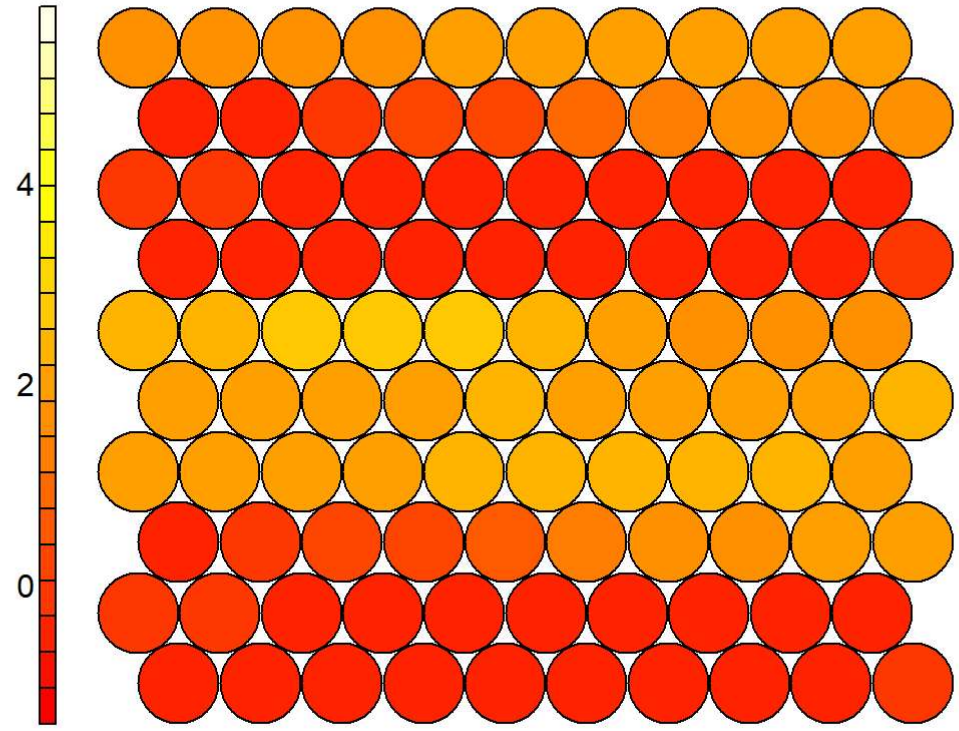
density



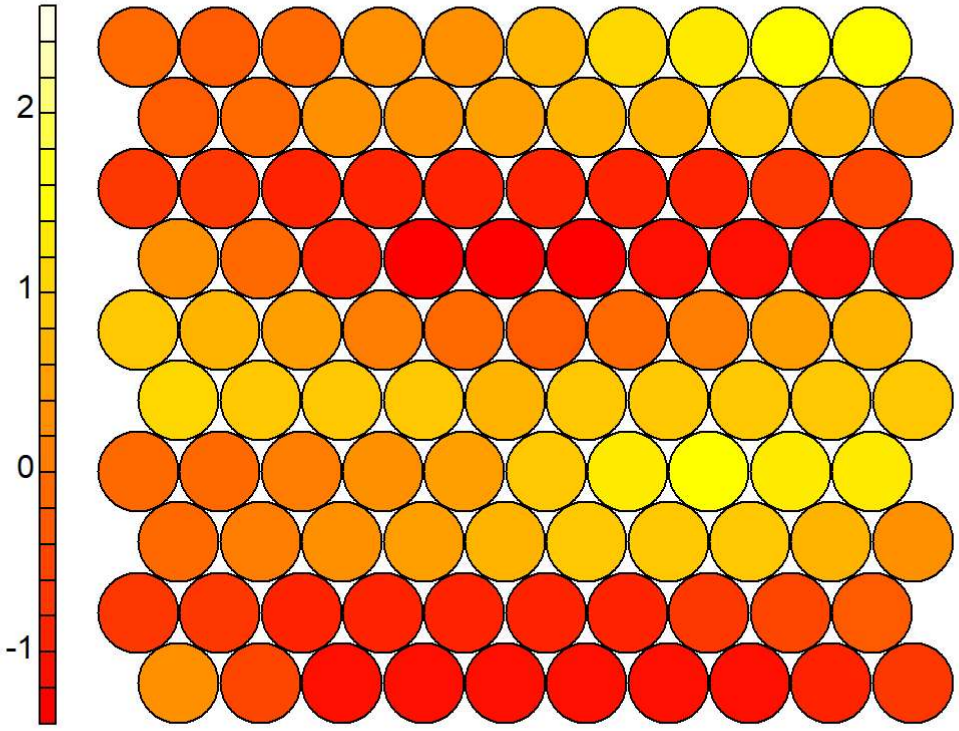
pH



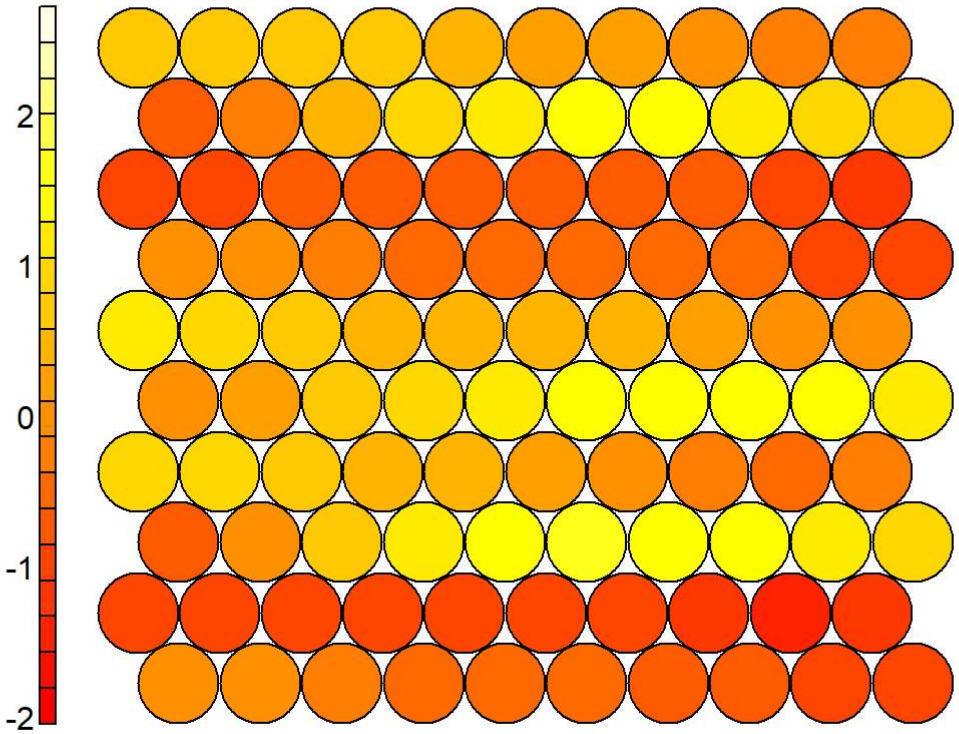
sulphates



alcohol



quality



e) Comment on the differences between b and c.

1. Dimensionality Reduction:

• PCA is a dimensionality reduction technique that projects the original high-dimensional data onto a lower-dimensional subspace while preserving as much of the variance in the data as possible.

• SOM, on the other hand, performs non-linear dimensionality reduction by mapping the high-dimensional data onto a lower-dimensional (usually 2D) grid while preserving the topological structure of the data.

2. Clustering Approach:

• k-means using PCA first reduces the dimensionality of the data using PCA, and then performs k-means clustering on the few top principal components.

• SOM performs both dimensionality reduction and clustering simultaneously.

3. Cluster Shapes and Assumptions:

• k-means clustering assumes that the clusters are spherical and have equal variance, which may not always be the case in real-world data.

• SOM does not make any assumptions about the shape or distribution of the clusters, and it can capture non-linear relationships and complex cluster structures in the data.