FDA_Assignment_4

Sanket Praveen Patil

2023-11-06

Problem 1:

Loading required libraries

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Question 1:

Load red and white wine datasets

```
red_wine <- read.csv("winequality-red.csv", header = T,sep = ";")
white_wine <- read.csv("winequality-white.csv",header = T,sep = ";")
```

Add a type column to each dataset

```
red_wine$type <- "red"
white_wine$type <- "white"

head(red_wine)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4             0.70        0.00            1.9     0.076
## 2           7.8             0.88        0.00            2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
## 4          11.2             0.28        0.56            1.9     0.075
## 5           7.4             0.70        0.00            1.9     0.076
## 6           7.4             0.66        0.00            1.8     0.075
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                11                34  0.9978 3.51      0.56     9.4
## 2                25                67  0.9968 3.20      0.68     9.8
## 3                15                54  0.9970 3.26      0.65     9.8
## 4                17                60  0.9980 3.16      0.58     9.8
## 5                11                34  0.9978 3.51      0.56     9.4
## 6                13                40  0.9978 3.51      0.56     9.4
##   quality type
## 1       5  red
## 2       5  red
## 3       5  red
## 4       6  red
## 5       5  red
## 6       5  red
```

```
head(white_wine)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.0             0.27        0.36           20.7     0.045
## 2           6.3             0.30        0.34            1.6     0.049
## 3           8.1             0.28        0.40            6.9     0.050
## 4           7.2             0.23        0.32            8.5     0.058
## 5           7.2             0.23        0.32            8.5     0.058
## 6           8.1             0.28        0.40            6.9     0.050
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  45                  170  1.0010 3.00      0.45     8.8
## 2                  14                  132  0.9940 3.30      0.49     9.5
## 3                  30                   97  0.9951 3.26      0.44    10.1
## 4                  47                  186  0.9956 3.19      0.40     9.9
## 5                  47                  186  0.9956 3.19      0.40     9.9
## 6                  30                   97  0.9951 3.26      0.44    10.1
##   quality  type
## 1       6 white
## 2       6 white
## 3       6 white
## 4       6 white
## 5       6 white
## 6       6 white
```

Checking class of all variables

```
sapply(red_wine, class)
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##            "numeric"            "numeric"            "numeric"
##       residual.sugar            chlorides  free.sulfur.dioxide
##            "numeric"            "numeric"            "numeric"
## total.sulfur.dioxide              density                   pH
##            "numeric"            "numeric"            "numeric"
##            sulphates              alcohol              quality
##            "numeric"            "numeric"            "integer"
##                 type
##          "character"
```

```
sapply(white_wine, class)
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##            "numeric"            "numeric"            "numeric"
##       residual.sugar            chlorides  free.sulfur.dioxide
##            "numeric"            "numeric"            "numeric"
## total.sulfur.dioxide              density                   pH
##            "numeric"            "numeric"            "numeric"
##            sulphates              alcohol              quality
##            "numeric"            "numeric"            "integer"
##                 type
##          "character"
```

Merge the two datasets

```
df <- full_join(red_wine, white_wine)
```

```
## Joining with `by = join_by(fixed.acidity, volatile.acidity, citric.acid,
## residual.sugar, chlorides, free.sulfur.dioxide, total.sulfur.dioxide, density,
## pH, sulphates, alcohol, quality, type)`
```

Question 2:

```
library(ggplot2)
library(caret)
```

```
## Loading required package: lattice
```

Separate dependent and independent variables

```
x <- df[, -ncol(df)]
y <- df$type
```

Scale the data

```
scaled_x <- scale(x)
```

Perform PCA

```
df_pca <- prcomp(scaled_x, center = TRUE, scale. = TRUE)
```

Extract the first two principal components to create a projection of the data to 2D

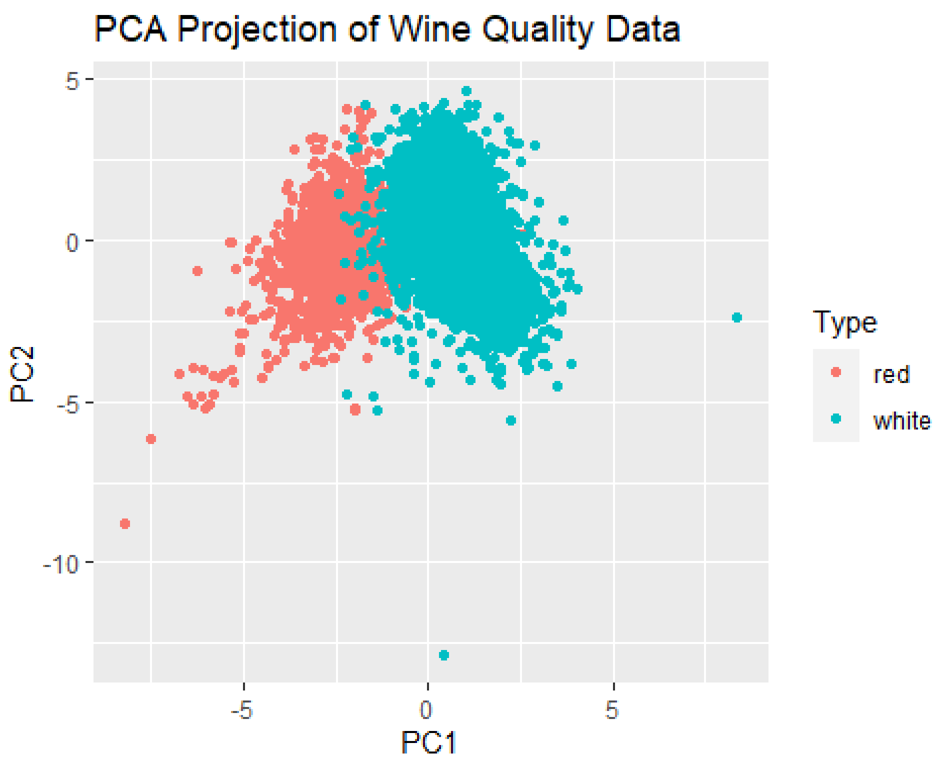
```
pc1 <- df_pca$x[, 1]
pc2 <- df_pca$x[, 2]
```

Create a new dataframe for plotting

```
df_scatterplot <- data.frame(PC1 = pc1, PC2 = pc2, Type = y)
```

Creating scatterplot

```
ggplot(df_scatterplot, aes(x = PC1, y = PC2, color = Type)) + geom_point() + labs(title = "PCA
Projection of Wine Quality Data")
```



Question 3:

By seeing the scatterplot in question b, we can see the data is well separated by type hence we will perform KNN as it might be the best choice.

Question 4:

```
library(class)
library(e1071)
library(rpart)
library(caret)
```

Convert type to factor

```
df$type <- as.factor(df$type)
```

Split the data into training and testing sets

```
set.seed(123)
train_indices <- createDataPartition(df$type, p = 0.8, list = FALSE)
df_train <- df[train_indices, ]
df_test <- df[-train_indices, ]
```

kNN

```
k_values <- 1:10  # Add more values as needed
tune_grid <- expand.grid(k = k_values)

knn_model_tuned <- train(
  type ~ .,
  data = df_train,
  method = "knn",
  preProcess = c("center", "scale"),
  tuneGrid = tune_grid,  # Specify the tuning grid
  trControl = trainControl(method = "cv", number = 5)  # 5-fold cross-validation
)
print(knn_model_tuned)

## k-Nearest Neighbors
##
## 5199 samples
##   12 predictor
##    2 classes: 'red', 'white'
##
## Pre-processing: centered (12), scaled (12)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 4160, 4159, 4159, 4159, 4159
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.9936522  0.9829264
##    2  0.9901895  0.9735941
##    3  0.9911514  0.9761821
##    4  0.9915364  0.9771940
##    5  0.9926903  0.9803142
##    6  0.9924980  0.9797993
##    7  0.9921133  0.9787670
##    8  0.9913441  0.9767065
##    9  0.9917287  0.9777128
##   10  0.9919209  0.9782479
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 1.

predictions_knn <- predict(knn_model_tuned, newdata = df_test)
knn_accuracy <- sum(predictions_knn == df_test$type) / length(df_test$type)
```

Decision Trees

```
colnames(df) <- make.names(colnames(df))
tree_model <- rpart(type ~ ., data = df_train, method = "class")
tree_predictions <- predict(tree_model, df_test, type = "class")
tree_accuracy <- sum(tree_predictions == df_test$type) / length(df_test$type)
```

SVM

```
df_train$type <- as.factor(df_train$type)
df_test$type <- as.factor(df_test$type)
svm_model <- svm(type ~ ., data = df_test, kernel = "linear", cost = 1)
svm_predictions <- predict(svm_model, df_test)
svm_accuracy <- sum(svm_predictions == df_test$type) / length(df_test$type)
```

Compare accuracies

```
accuracy_df <- data.frame(Classifier = c("kNN", "Decision Tree", "SVM"),
                          Accuracy = c(knn_accuracy, tree_accuracy, svm_accuracy))
print(accuracy_df)
```

```
##      Classifier  Accuracy
## 1           kNN 0.9946071
## 2 Decision Tree 0.9799692
## 3           SVM 0.9938367
```

kNN classifier performed the best based in terms of accuracy, as it achieved the highest accuracy of 99.46%.

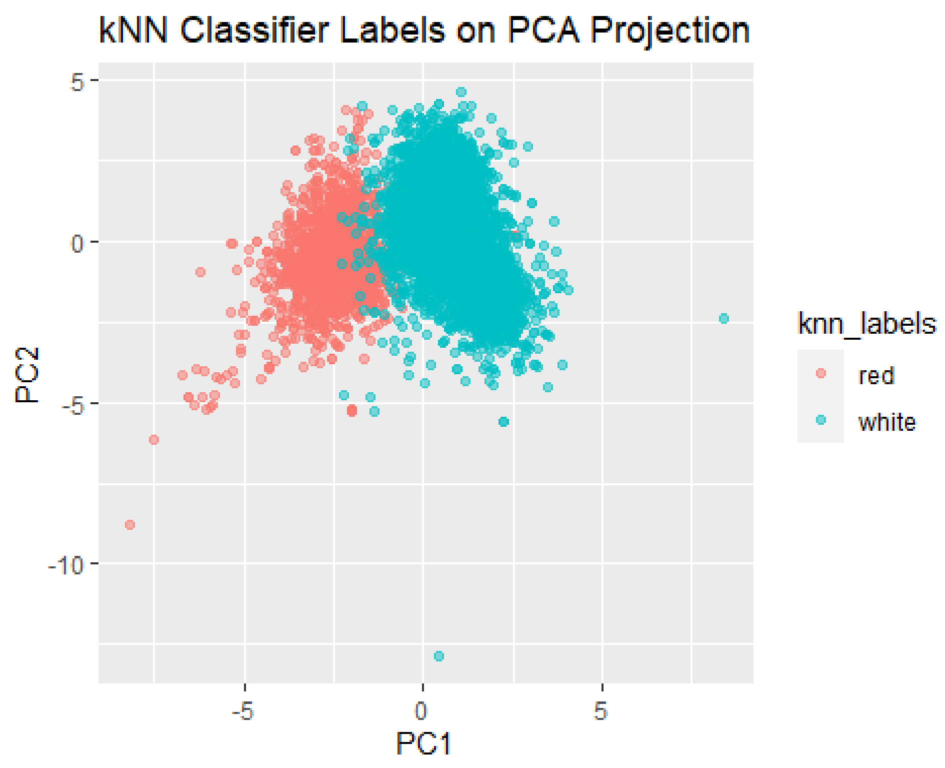
Question 5:

Add predicted labels to the original dataframe

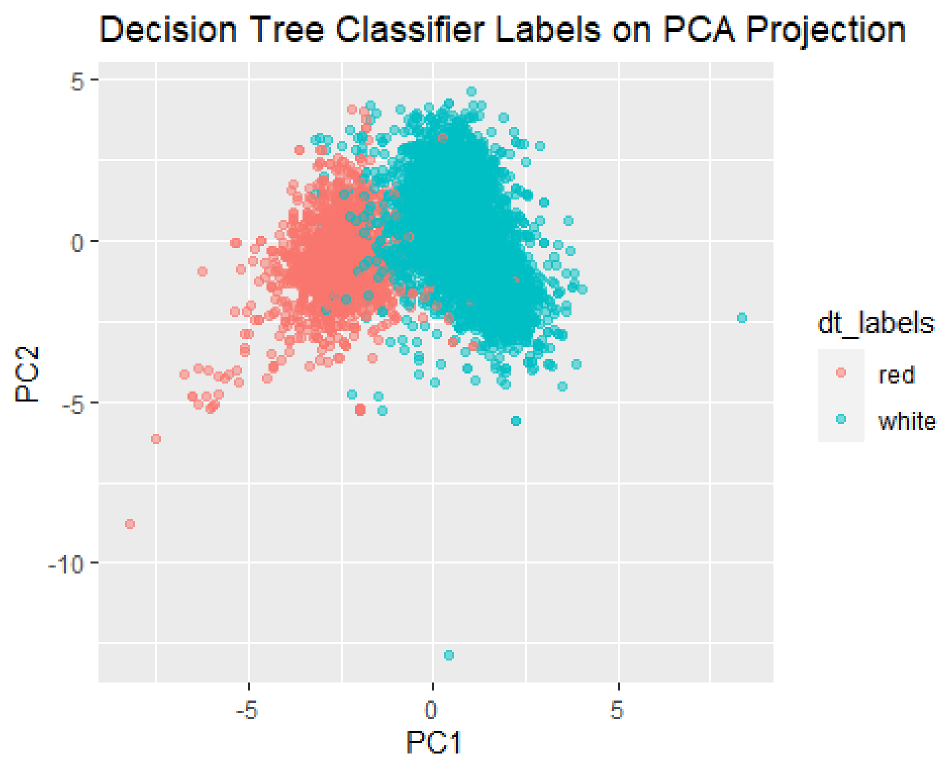
```
df_scatterplot$knn_labels <- predict(knn_model_tuned, newdata = df)
df_scatterplot$dt_labels <- predict(tree_model, df, type = "class")
df_scatterplot$svm_labels <- predict(svm_model, df)
```

Scatterplot with classifier labels

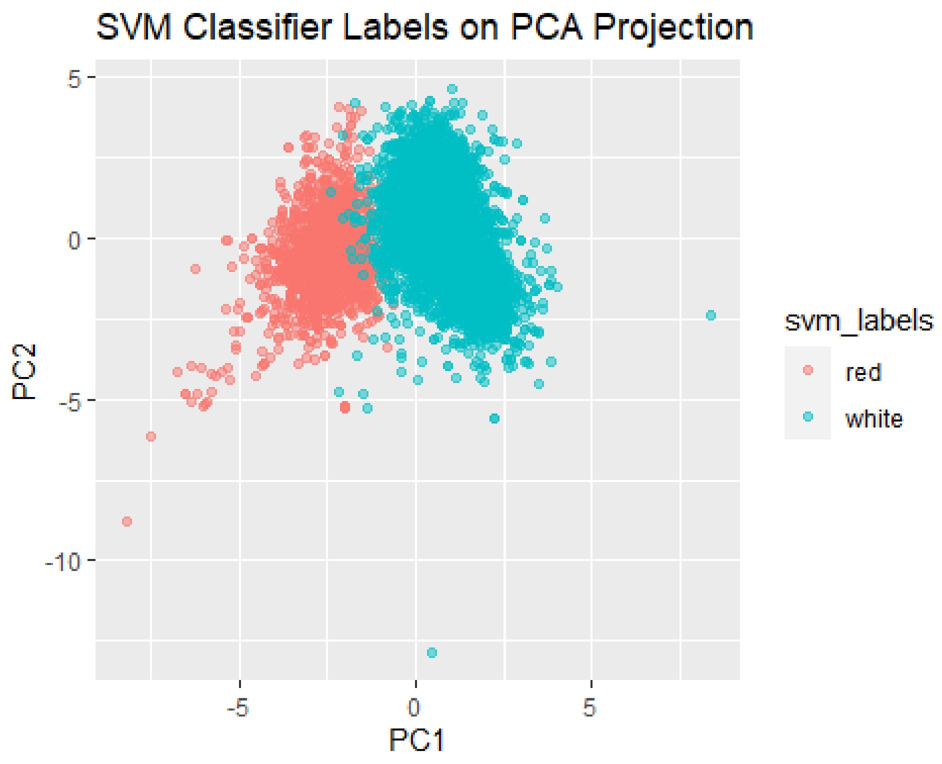
```
ggplot(df_scatterplot, aes(x = PC1, y = PC2)) +
  geom_point(aes(color = knn_labels), alpha = 0.5) +
  labs(title = "kNN Classifier Labels on PCA Projection")
```



```
ggplot(df_scatterplot, aes(x = PC1, y = PC2)) +
  geom_point(aes(color = dt_labels), alpha = 0.5) +
  labs(title = "Decision Tree Classifier Labels on PCA Projection")
```



```r
ggplot(df_scatterplot, aes(x = PC1, y = PC2)) +
  geom_point(aes(color = svm_labels), alpha = 0.5) +
  labs(title = "SVM Classifier Labels on PCA Projection")
```



By seeing the above plots, we can see all 3 methods are performing similar.

Problem 2 :

```r
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ lubridate 1.9.3     ✔ tibble    3.2.1
## ✔ purrr     1.0.2     ✔ tidyr     1.3.0
## ✔ readr     2.1.4
## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ purrr::lift()   masks caret::lift()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(class)
```

Question 1 :

```
data("Sacramento")

head(Sacramento)

##         city    zip beds baths sqft        type price latitude longitude
## 1 SACRAMENTO z95838    2     1  836 Residential 59222 38.63191 -121.4349
## 2 SACRAMENTO z95823    3     1 1167 Residential 68212 38.47890 -121.4310
## 3 SACRAMENTO z95815    2     1  796 Residential 68880 38.61830 -121.4438
## 4 SACRAMENTO z95815    2     1  852 Residential 69307 38.61684 -121.4391
## 5 SACRAMENTO z95824    2     1  797 Residential 81900 38.51947 -121.4358
## 6 SACRAMENTO z95841    3     1 1122       Condo 89921 38.66260 -121.3278

dim(Sacramento)

## [1] 932   9
```

Check for unique values

```
sapply(Sacramento, function(x) n_distinct(x))

##      city       zip      beds     baths      sqft      type     price  latitude 
##        37        68         7         9       687         3       602       920 
## longitude 
##       920
```

Separating target variable from data

```
target_variable <- data.frame(Sacramento$type)
```

Drop type variable

```
Sacramento <- Sacramento[, !colnames(Sacramento) %in% "type"]
```

Converting to dummies

```
Sacramento_dummies <- Sacramento %>%
  select(-zip) %>%  # We can exclude zip varible as there is no dependency
  model.matrix(~ . - 1, data = .) %>%  # Create dummy variables for all variables
  as.data.frame()

Sacramento_dummies$type <- target_variable$Sacramento.type

head(Sacramento_dummies)

##   cityANTELOPE cityAUBURN cityCAMERON_PARK cityCARMICHAEL
cityCITRUS_HEIGHTS
```

```
## 1           0         0              0            0                0
## 2           0         0              0            0                0
## 3           0         0              0            0                0
## 4           0         0              0            0                0
## 5           0         0              0            0                0
## 6           0         0              0            0                0
##   cityCOOL cityDIAMOND_SPRINGS cityEL_DORADO cityEL_DORADO_HILLS
cityELK_GROVE
## 1       0                0           0                0           0
## 2       0                0           0                0           0
## 3       0                0           0                0           0
## 4       0                0           0                0           0
## 5       0                0           0                0           0
## 6       0                0           0                0           0
##   cityELVERTA cityFAIR_OAKS cityFOLSOM cityFORESTHILL cityGALT
## 1          0           0         0            0       0
## 2          0           0         0            0       0
## 3          0           0         0            0       0
## 4          0           0         0            0       0
## 5          0           0         0            0       0
## 6          0           0         0            0       0
##   cityGARDEN_VALLEY cityGOLD_RIVER cityGRANITE_BAY cityGREENWOOD
cityLINCOLN
## 1              0            0            0           0         0
## 2              0            0            0           0         0
## 3              0            0            0           0         0
## 4              0            0            0           0         0
## 5              0            0            0           0         0
## 6              0            0            0           0         0
##   cityLOOMIS cityMATHER cityMEADOW_VISTA cityNORTH_HIGHLANDS
cityORANGEVALE
## 1         0         0             0               0            0
## 2         0         0             0               0            0
## 3         0         0             0               0            0
## 4         0         0             0               0            0
## 5         0         0             0               0            0
## 6         0         0             0               0            0
##   cityPENRYN cityPLACERVILLE cityPOLLOCK_PINES cityRANCHO_CORDOVA
## 1         0             0              0               0
## 2         0             0              0               0
## 3         0             0              0               0
## 4         0             0              0               0
## 5         0             0              0               0
## 6         0             0              0               0
##   cityRANCHO_MURIETA cityRIO_LINDA cityROCKLIN cityROSEVILLE citySACRAMENTO
## 1                0            0          0           0            1
## 2                0            0          0           0            1
## 3                0            0          0           0            1
## 4                0            0          0           0            1
## 5                0            0          0           0            1
## 6                0            0          0           0            1
##   cityWALNUT_GROVE cityWEST_SACRAMENTO cityWILTON beds baths sqft price
## 1                0                   0          0    2     1  836 59222
## 2                0                   0          0    3     1 1167 68212
## 3                0                   0          0    2     1  796 68880
## 4                0                   0          0    2     1  852 69307
## 5                0                   0          0    2     1  797 81900
## 6                0                   0          0    3     1 1122 89921
##   latitude longitude        type
## 1 38.63191 -121.4349 Residential
## 2 38.47890 -121.4310 Residential
## 3 38.61830 -121.4438 Residential
## 4 38.61684 -121.4391 Residential
## 5 38.51947 -121.4358 Residential
## 6 38.66260 -121.3278       Condo
```

```
dim(Sacramento_dummies)
```

```
## [1] 932  44
```

Question 2:

Choosing distance function for kNN
For high dimensionality, Manhattan distance (p = 1) might be a good choice.

Question 3:

```
library(kknn)
```

```
## Warning: package 'kknn' was built under R version 4.3.2
```

```
##
## Attaching package: 'kknn'
```

```
## The following object is masked from 'package:caret':
##
##     contr.dummy
```

setup a tuneGrid with the tuning parameters

```
tuneGrid <- expand.grid(kmax = 3:7,                        # test a range of k values 3 to 7
                        kernel = c("rectangular", "cos"),  # regular and cosine-based distance functions
                        distance = 1:3)
suppressWarnings({
kknn_fit <- train(type ~ .,
                  data = Sacramento_dummies,
                  method = 'kknn',
                  #trControl = ctrl,
                  preProcess = c('center', 'scale'),
                  tuneGrid = tuneGrid)})
kknn_fit
```

```
## k-Nearest Neighbors
##
## 932 samples
##  43 predictor
##   3 classes: 'Condo', 'Multi_Family', 'Residential'
##
## Pre-processing: centered (43), scaled (43)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 932, 932, 932, 932, 932, 932, ...
## Resampling results across tuning parameters:
##
##   kmax  kernel       distance  Accuracy   Kappa
##   3     rectangular  1         0.9270967  0.4561882
##   3     rectangular  2         0.9307722  0.4608153
##   3     rectangular  3         0.9307784  0.4601172
##   3     cos          1         0.9282342  0.4592521
##   3     cos          2         0.9314449  0.4623316
##   3     cos          3         0.9316790  0.4627645
##   4     rectangular  1         0.9270967  0.4561882
##   4     rectangular  2         0.9307722  0.4608153
##   4     rectangular  3         0.9307784  0.4601172
##   4     cos          1         0.9282342  0.4592521
##   4     cos          2         0.9314449  0.4623316
##   4     cos          3         0.9316790  0.4627645
##   5     rectangular  1         0.9270967  0.4561882
##   5     rectangular  2         0.9307722  0.4608153
##   5     rectangular  3         0.9307784  0.4601172
##   5     cos          1         0.9282342  0.4592521
##   5     cos          2         0.9314449  0.4623316
##   5     cos          3         0.9316790  0.4627645
##   6     rectangular  1         0.9270967  0.4561882
##   6     rectangular  2         0.9307722  0.4608153
##   6     rectangular  3         0.9307784  0.4601172
##   6     cos          1         0.9282342  0.4592521
##   6     cos          2         0.9314449  0.4623316
##   6     cos          3         0.9316790  0.4627645
##   7     rectangular  1         0.9270967  0.4561882
##   7     rectangular  2         0.9307722  0.4608153
##   7     rectangular  3         0.9307784  0.4601172
##   7     cos          1         0.9282342  0.4592521
##   7     cos          2         0.9314449  0.4623316
##   7     cos          3         0.9316790  0.4627645
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were kmax = 7, distance = 3 and kernel
##  = cos.
```

Predicting the type

```
pred_knn <- predict(kknn_fit, Sacramento_dummies)
```

Generate confusion matrix

```
confusionMatrix(Sacramento_dummies$type, pred_knn)

## Confusion Matrix and Statistics
##
##              Reference
## Prediction    Condo Multi_Family Residential
##   Condo          36            0          17
##   Multi_Family    0            6           7
##   Residential     4            0         862
##
## Overall Statistics
##
##                Accuracy : 0.97
##                  95% CI : (0.9569, 0.9799)
##     No Information Rate : 0.9506
##     P-Value [Acc > NIR] : 0.002451
##
##                   Kappa : 0.7368
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Condo Class: Multi_Family Class: Residential
## Sensitivity               0.90000            1.000000             0.9729
## Specificity               0.98094            0.992441             0.9130
## Pos Pred Value            0.67925            0.461538             0.9954
## Neg Pred Value            0.99545            1.000000             0.6364
## Prevalence                0.04292            0.006438             0.9506
## Detection Rate            0.03863            0.006438             0.9249
## Detection Prevalence      0.05687            0.013948             0.9292
## Balanced Accuracy         0.94047            0.996220             0.9430

table(Sacramento_dummies$type, pred_knn)

##               pred_knn
##                Condo Multi_Family Residential
##   Condo           36            0          17
##   Multi_Family     0            6           7
##   Residential      4            0         862

knn_results = kknn_fit$results # gives just the table of results by parameter
head(knn_results)

##   kmax      kernel distance  Accuracy     Kappa AccuracySD    KappaSD
## 1    3 rectangular        1 0.9270967 0.4561882 0.01164946 0.06183649
## 4    3         cos        1 0.9282342 0.4592521 0.01125368 0.05908934
## 2    3 rectangular        2 0.9307722 0.4608153 0.01105597 0.06210100
## 5    3         cos        2 0.9314449 0.4623316 0.01152306 0.06278057
## 3    3 rectangular        3 0.9307784 0.4601172 0.01088815 0.06721108
## 6    3         cos        3 0.9316790 0.4627645 0.01036985 0.06568797
```

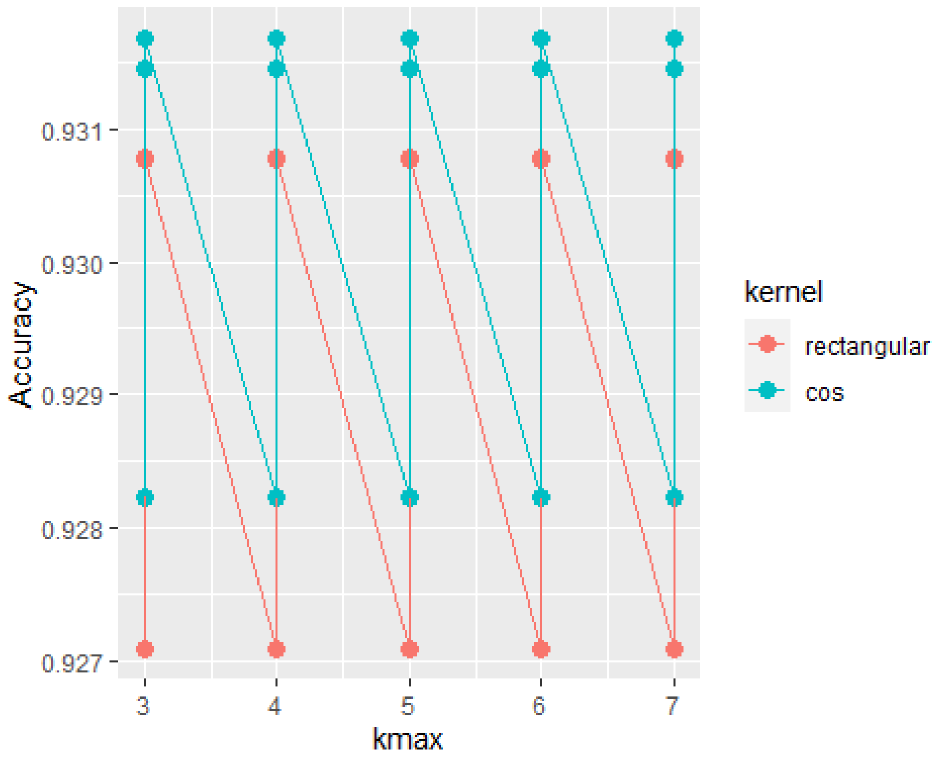
group by k and distance function, create an aggregation by averaging

```
knn_results <- knn_results %>%
  group_by(kmax, kernel) %>%
  mutate(avgacc = mean(Accuracy))
head(knn_results)
```

```
## # A tibble: 6 × 8
## # Groups:   kmax, kernel [2]
##    kmax kernel      distance Accuracy Kappa AccuracySD KappaSD avgacc
##   <int> <fct>          <int>    <dbl> <dbl>      <dbl>   <dbl>  <dbl>
## 1     3 rectangular        1    0.927 0.456     0.0116  0.0618  0.930
## 2     3 cos                1    0.928 0.459     0.0113  0.0591  0.930
## 3     3 rectangular        2    0.931 0.461     0.0111  0.0621  0.930
## 4     3 cos                2    0.931 0.462     0.0115  0.0628  0.930
## 5     3 rectangular        3    0.931 0.460     0.0109  0.0672  0.930
## 6     3 cos                3    0.932 0.463     0.0104  0.0657  0.930
```

plot aggregated (over Minkowski power) accuracy per k, split by distance function

```
ggplot(knn_results, aes(x=kmax, y=Accuracy, color=kernel)) +
  geom_point(size=3) + geom_line()
```



The final values used for the model were kmax = 7, distance = 1(Manhattan) and kernel = cos.

Problem 3 :

Question 1:

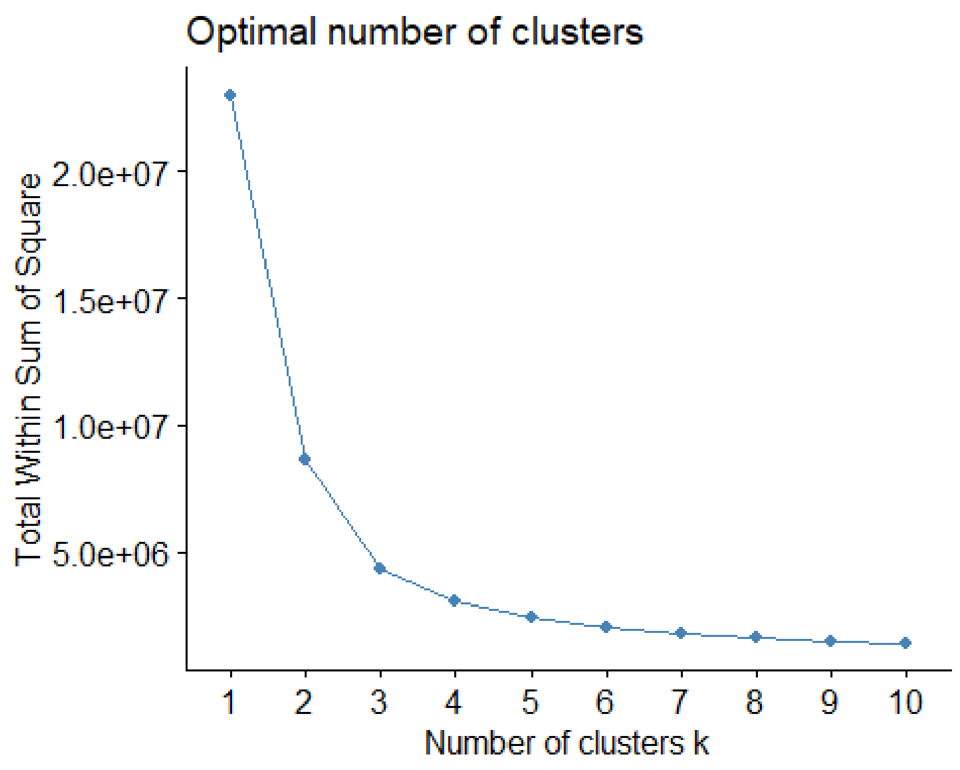
```
library(cluster)

## Warning: package 'cluster' was built under R version 4.3.2

library(stats)
library(factoextra)

## Warning: package 'factoextra' was built under R version 4.3.2

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

library(ggplot2)
library(tidyverse)
library(caret)

head(df)

##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4             0.70        0.00            1.9     0.076
## 2           7.8             0.88        0.00            2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
## 4          11.2             0.28        0.56            1.9     0.075
## 5           7.4             0.70        0.00            1.9     0.076
## 6           7.4             0.66        0.00            1.8     0.075
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  11                   34  0.9978 3.51      0.56     9.4
## 2                  25                   67  0.9968 3.20      0.68     9.8
## 3                  15                   54  0.9970 3.26      0.65     9.8
## 4                  17                   60  0.9980 3.16      0.58     9.8
## 5                  11                   34  0.9978 3.51      0.56     9.4
## 6                  13                   40  0.9978 3.51      0.56     9.4
##   quality type
## 1       5  red
## 2       5  red
## 3       5  red
## 4       6  red
## 5       5  red
## 6       5  red
```

Select columns for clustering

```
df_wine <- select(df, -type)
```

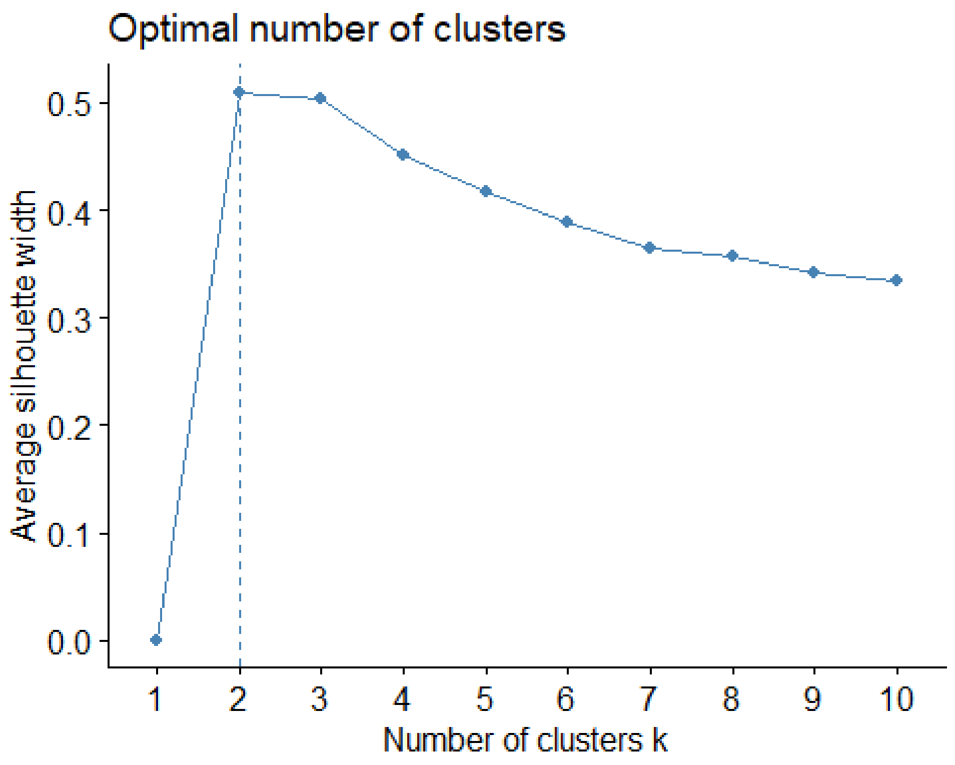
Find the knee

```
fviz_nbclust(df_wine, kmeans, method = "wss")
```



# Comparing the average silhoutte scores

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



# By seeing at both the knee plot, we cannot see a clear elbow but with silhouette method, we can see model will perform better at k=4. Hence we will choose k=4 to fit the model.

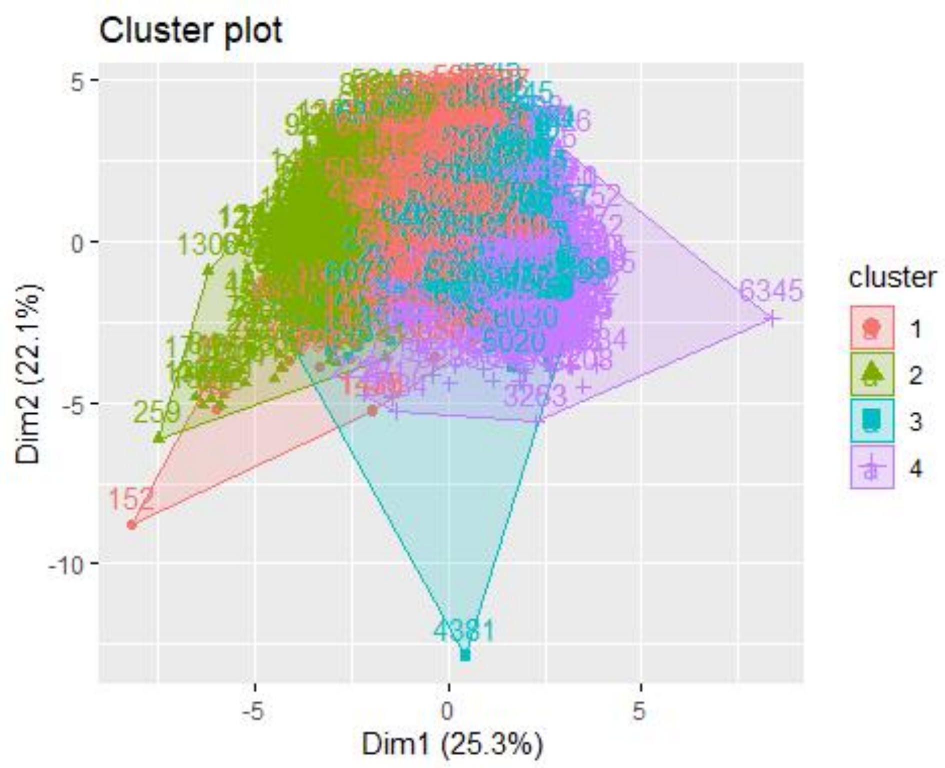
Fit the data using KMeans

```
fit <- kmeans(df_wine, centers = 4, nstart = 25)
fit

## K-means clustering with 4 clusters of sizes 2049, 1377, 1971, 1100
##
## Cluster means:
##   fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
## 1      6.955881        0.3121352   0.3134407       4.108394 0.04833626
## 2      8.258606        0.4983805   0.2711038       2.468046 0.08185911
## 3      6.892922        0.2826814   0.3375850       6.741020 0.04831253
## 4      6.970182        0.2943727   0.3538455       9.328682 0.05187909
##   free.sulfur.dioxide total.sulfur.dioxide   density       pH sulphates
## 1            25.25061             98.85652 0.9930695 3.206725 0.5065105
## 2            12.68155             33.91649 0.9962079 3.299005 0.6374074
## 3            37.13825            144.70472 0.9944472 3.195043 0.4920294
## 4            50.83864            197.74500 0.9962827 3.181691 0.5148273
##     alcohol  quality
## 1 10.959655 5.961445
## 2 10.570044 5.673929
## 3 10.385843 5.912735
## 4  9.712227 5.563636
##
```

```
## Clustering vector:
##    [1] 2 1 2 2 2 2 2 2 2 1 2 1 2 2 3 3 1 2 2 2 2 1 2 2 2 2 2 2 2 2 1 2 1 1 2 2 2
##   [38] 2 2 1 1 2 2 2 2 2 1 2 2 1 2 2 2 1 1 2 2 1 2 2 2 1 2 2 2 2 2 2 1 2 2 1 1 2
##   [75] 1 2 2 2 1 1 2 1 1 2 2 2 3 2 3 2 3 3 3 2 1 1 2 2 2 2 2 2 2 2 2 2 2 1 2 1 3 2
##  [112] 1 1 2 2 2 2 2 2 1 1 2 2 2 1 1 2 2 2 2 3 1 1 2 2 2 2 2 1 1 2 2 2 2 2 3 1 1
##  [149] 2 2 2 1 1 1 3 3 3 3 2 1 2 2 2 3 3 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2
##  [186] 1 2 2 3 3 3 2 3 2 2 1 2 2 1 2 2 3 2 2 2 2 2 1 1 2 2 2 2 1 2 1 2 2 2 3 2 1
##  [223] 2 2 2 2 1 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 1 2 2 2 2 2 1 2 1 2 2 2
##  [260] 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 1 1 2 2 2 1 2 2 2 2 2 2
##  [297] 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 1 1 3 1 2 1 1 2 1 2 1 2 1 2 2 2 2 2 2 2 2 1
##  [334] 2 2 2 2 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 1 3 2 2 2 2 2 1 2 2 2 2 2 2 2 1 2
##  [371] 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1 2 2 3 2 2 2 3 2 2 2 2 2 2
##  [408] 2 2 2 1 1 1 2 1 3 2 3 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
##  [445] 2 2 1 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 2 3 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2
##  [482] 2 2 2 2 2 2 2 1 2 2 2 2 1 1 2 2 1 2 1 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2
##  [519] 2 1 2 2 3 3 1 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 1 2
##  [556] 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1 1 2 2 2 2 2 1 2 2 1 2 2 1 3
##  [593] 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 2 2 2 1 1 2 2 2 2 2 2 2
##  [630] 1 2 2 2 1 1 2 3 3 2 2 2 1 2 1 2 2 2 2 2 3 2 3 1 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [667] 2 2 2 2 2 2 3 2 2 2 2 2 1 2 2 2 2 2 3 2 2 2 2 2 2 1 2 1 3 2 2 2 1 2 1 2 2
##  [704] 1 2 2 2 2 2 2 1 1 2 2 1 2 2 2 2 2 2 1 2 3 2 2 2 2 2 2 2 2 2 1 2 2 2 2 1 2
##  [741] 2 3 2 1 1 2 2 1 2 2 2 2 1 2 2 2 2 2 2 1 1 2 2 2 2 2 1 1 1 2 1 3 3 2 2 2 2
##  [778] 2 2 1 2 2 1 2 1 2 2 2 2 1 1 3 1 2 2 2 2 2 2 2 1 2 1 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 2 2 1 1 2 2 2 2 1 1 2 2 2 2 2 2 2
##  [852] 2 1 2 2 2 2 2 2 2 1 1 2 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 1 1 2 2 2
##  [889] 1 1 2 1 2 1 1 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1 2 2 2
##  [926] 1 1 1 2 2 2 2 1 2 2 1 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
##  [963] 2 2 2 2 2 1 2 2 2 2 2 2 2 1 1 1 2 2 2 2 1 2 2 2 2 1 2 2 2 1 2 1 1 2 2 2 2
## [1000] 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 1 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 1 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1
## [1074] 2 1 1 2 2 2 4 2 4 1 2 1 1 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1
## [1111] 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 3 2 2 2 2 2 2 1 1 1 1 2 2 1 2 2
## [1148] 2 2 2 2 1 2 2 1 2 1 1 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
## [1185] 1 2 2 2 1 2 2 2 2 2 2 2 1 2 1 1 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2
## [1222] 2 1 2 2 1 2 2 1 2 2 1 2 2 2 1 2 2 2 2 2 1 2 1 3 2 2 2 2 2 2 2 2 2 2 2 2 1
## [1259] 2 2 2 2 1 2 2 2 2 2 2 1 2 2 2 2 2 1 2 2 1 2 2 2 2 1 2 2 2 2 1 1 2 2 2 2 2
## [1296] 1 1 2 2 2 2 2 2 1 1 1 1 2 1 2 1 2 2 2 1 1 2 2 1 2 1 2 2 2 2 2 2 2 2 1 1 1
## [1333] 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 1 1 2 2 2 2 2 2 2 2 1 1
## [1370] 2 2 2 2 1 2 1 2 1 2 2 2 2 1 1 1 1 2 2 2 1 2 2 2 2 1 2 2 1 2 2 3 3 2 2 2 2
## [1407] 2 2 2 2 2 2 2 1 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 1 2 1 2 2 1 1 1 2 2 1 2 1 2
## [1444] 2 1 1 2 2 2 2 2 2 2 1 2 2 1 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 2 2
## [1481] 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 3 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 1 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 3 3 3 3 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2 2 2 2 2 1 2 2 2 2 1 1 2
## [1592] 2 2 2 2 2 2 2 2 3 3 1 4 4 1 3 3 3 3 2 1 1 3 4 1 1 1 3 3 1 1 3 3 3 4 3 3 3
## [1629] 1 3 2 1 1 4 1 3 3 3 3 3 3 3 3 3 3 4 4 3 3 3 3 1 1 3 4 4 3 1 1 3 3 3 1 1 1
## [1666] 3 4 1 1 4 4 4 1 1 1 1 1 1 1 1 3 3 4 3 3 3 4 3 3 3 4 3 3 3 4 3 1 1 3 4 3 3
## [1703] 3 4 1 3 4 4 4 3 4 4 3 3 1 3 1 4 4 1 3 3 3 3 3 3 4 4 4 2 4 4 4 4 4 3 3 1 1
## [1740] 1 3 1 1 1 1 3 1 1 1 3 3 1 1 1 4 4 3 3 3 3 3 1 4 4 4 4 1 3 1 3 1 1 3 3 3 2
## [1777] 4 1 4 4 4 3 4 4 4 3 1 1 4 4 3 3 3 4 4 4 4 4 4 4 4 4 3 1 3 3 1 1 3 1 1 3 3
## [1814] 1 3 3 3 4 3 3 3 1 3 3 3 4 4 4 3 3 4 4 4 4 4 4 4 3 3 4 1 1 4 3 4 3 1 1 1 4
## [1851] 4 3 1 3 3 1 1 1 1 1 3 1 4 3 3 4 3 3 3 4 3 3 3 4 3 3 2 2 1 1 1 4 4 4 4 4 4
## [1888] 4 4 4 3 4 3 4 4 3 4 3 1 1 1 1 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 1 2 3 3 3 4
## [1925] 4 4 3 4 1 1 3 1 3 1 1 1 4 3 3 3 3 3 3 3 3 1 3 1 3 3 3 3 3 4 4 4 3 3 3 3 1
## [1962] 3 4 1 1 3 3 4 1 3 4 4 3 1 1 1 1 3 1 1 4 3 3 3 1 1 4 3 4 4 2 3 1 3 4 1 1 3
## [1999] 1 1 3 1 4 3 4 1 1 1 1 3 3 1 1 3 3 1 4 1 3 3 4 4 4 3 4 4 4 1 4 4 1 4 3 1 1
## [2036] 4 4 4 1 1 3 3 4 3 2 3 1 1 3 3 3 3 1 3 1 1 1 4 4 1 3 3 1 4 3 3 1 4 4 3 4 1
## [2073] 3 1 4 1 1 3 3 3 1 3 3 4 1 1 1 4 4 1 2 4 3 1 3 4 3 3 4 4 3 4 4 3 3 3 3 3 3
## [2110] 3 3 3 2 1 3 3 3 1 2 3 3 1 1 1 3 2 1 1 1 1 3 4 4 4 4 4 4 2 4 3 4 3 3 3 3 3
## [2147] 1 3 4 3 1 1 3 1 1 3 3 3 3 3 3 4 3 3 1 2 3 3 4 4 1 4 3 3 4 4 3 1 3 4 1 3 1
## [2184] 3 1 3 3 3 3 3 3 3 3 3 3 3 3 1 2 3 1 3 3 3 3 3 3 3 1 1 1 3 3 3 3 1 4 4 3 4
## [2221] 4 3 2 3 3 4 4 4 1 3 3 3 4 3 3 3 3 4 4 3 4 4 3 3 3 3 3 4 4 4 4 4 3 3 1 1 3
## [2258] 4 4 1 3 4 1 4 3 4 4 3 4 4 1 3 3 4 4 4 1 1 1 3 3 3 4 4 4 1 4 4 3 3 4 4 4 4
## [2295] 4 1 4 4 4 4 3 1 1 1 2 4 3 3 1 4 3 3 4 4 3 4 3 3 3 4 4 3 1 1 3 1 1 3 4 3 4
## [2332] 2 4 4 3 4 4 3 3 2 2 3 3 3 1 4 4 4 4 4 4 3 4 3 1 4 4 3 3 4 4 4 3 3 3 4 2 1
## [2369] 1 1 3 3 4 3 1 1 4 4 3 2 4 4 3 4 1 1 1 1 1 1 1 3 1 3 3 4 3 1 1 3 4 4 3 1 3
## [2406] 4 4 4 4 4 1 3 3 4 3 1 3 3 3 1 4 1 1 1 3 3 3 1 1 1 3 1 1 1 1 4 4 4 1 1 3 3
## [2443] 1 1 1 3 1 3 1 1 3 1 3 1 1 3 4 3 3 1 4 3 1 3 1 1 3 4 1 3 3 3 2 2 3 3 1 1 1
## [2480] 3 1 3 3 4 1 4 1 4 1 3 1 1 3 1 1 4 1 2 4 3 1 4 4 3 1 1 3 3 4 3 3 3 2 1 2 3
## [2517] 1 1 1 3 3 3 4 3 1 1 4 4 1 1 4 4 4 4 4 2 1 4 4 4 4 1 3 3 1 4 3 2 1 3 3 1 1
## [2554] 3 3 3 1 1 4 4 1 4 1 3 1 4 3 1 1 1 1 3 1 1 1 3 4 3 2 2 3 1 1 3 1 3 3 3 1 3
## [2591] 1 4 2 4 3 2 3 4 3 1 4 4 3 3 1 3 1 4 3 1 4 1 4 3 1 3 1 3 4 3 3 4 4 4 3 3 2
## [2628] 3 4 3 4 4 4 4 3 1 1 1 1 3 1 1 4 1 1 1 2 1 1 3 3 1 1 2 1 1 4 1 3 1 4 4 4 3
## [2665] 3 4 3 1 3 3 3 1 4 4 3 4 3 4 4 1 3 3 4 3 3 3 3 4 3 4 4 3 3 4 1 3 3 1 1 3 1
## [2702] 3 1 3 4 3 1 1 3 1 1 4 1 2 1 2 4 2 3 1 1 1 1 3 4 4 1 1 1 3 3 3 1 3 3 4 4 3
## [2739] 1 2 3 3 3 3 4 3 4 1 4 4 4 1 1 3 3 3 3 4 3 3 3 3 4 1 3 1 1 1 3 1 1 1 1 4 4
## [2776] 4 4 3 1 1 3 1 3 3 4 4 1 3 2 1 3 1 3 4 3 3 3 1 1 1 1 4 1 1 4 4 4 3 3 2 4 3
## [2813] 1 2 3 1 4 3 1 4 3 1 3 1 1 1 1 1 1 4 1 1 3 3 4 1 1 3 3 4 4 3 3 3 4 3 1 1 4
## [2850] 3 3 3 3 3 1 3 4 4 4 4 3 3 4 1 4 1 3 3 4 3 3 3 3 4 3 3 3 3 3 3 3 3 3 3 3 3
## [2887] 1 1 1 1 4 1 1 1 4 3 1 3 3 3 4 4 3 4 1 1 3 3 1 3 1 3 4 3 3 4 3 1 4 1 1 3 3
## [2924] 4 1 3 3 1 1 1 1 4 4 1 4 3 4 4 1 3 3 1 4 3 1 3 3 3 2 1 3 4 4 4 3 4 3 1 3 3
## [2961] 4 1 1 3 3 1 1 4 4 1 3 4 4 1 3 3 3 3 1 3 1 1 1 3 3 1 1 3 4 3 1 1 1 1 1 3 4
## [2998] 4 1 4 4 3 3 1 1 1 1 4 1 1 1 3 1 3 2 3 4 3 1 3 1 1 4 1 3 3 3 1 1 1 1 1 1 3
## [3035] 1 4 1 4 4 3 1 1 3 3 3 1 3 3 4 3 3 3 3 4 3 1 3 3 3 3 3 3 3 3 3 4 1 2 3 3 3
## [3072] 2 3 1 3 4 1 3 1 3 3 4 1 3 3 2 3 4 4 1 1 4 4 4 4 2 3 1 3 3 3 1 3 3 3 4 3 4
## [3109] 3 3 1 3 3 3 3 3 3 1 3 3 1 3 4 3 3 3 4 3 3 3 3 4 1 4 1 2 3 1 1 4 1 1 1 2 3
## [3146] 3 3 1 1 3 3 3 3 3 3 4 3 1 2 3 1 3 3 1 1 3 1 4 3 3 4 3 1 1 4 1 3 3 4 1 1 1
## [3183] 3 4 3 3 4 3 1 3 3 1 2 3 4 3 4 1 1 3 4 1 2 3 1 1 1 4 4 1 1 1 3 3 3 4 3 4 3
## [3220] 1 3 3 1 3 3 4 3 1 1 1 1 1 1 4 3 3 3 4 1 3 3 3 3 3 3 3 1 1 3 3 4 1 4 4 3 1
## [3257] 3 4 4 4 4 1 4 4 1 1 3 1 1 3 1 3 3 4 4 1 3 3 1 4 4 4 4 4 4 3 4 4 3 3 4 4 4
## [3294] 3 3 4 4 4 4 1 4 1 3 3 3 3 4 3 4 2 1 4 3 1 3 1 3 4 1 4 4 3 3 3 3 1 4 1 3 1
## [3331] 4 1 3 4 4 1 4 1 2 3 3 3 3 3 4 1 3 1 3 4 3 4 3 1 3 4 4 2 4 4 3 1 1 4 4 4 4
## [3368] 3 1 4 3 4 4 1 3 3 3 3 3 3 4 3 1 3 3 1 3 3 4 3 1 3 4 1 3 3 3 4 3 4 2 4 3 4
## [3405] 1 1 4 1 4 4 1 1 1 1 3 3 1 1 3 3 1 1 4 3 1 3 4 4 3 3 3 3 3 3 4 1 1 4 1 3 1
## [3442] 4 3 1 3 4 3 4 4 4 1 3 1 4 4 4 3 3 4 3 4 1 4 1 3 4 3 3 3 3 3 3 3 3 3 3 1 4
## [3479] 1 4 1 4 4 3 1 1 4 4 1 4 4 3 3 3 4 3 3 1 1 3 1 4 1 4 3 1 3 1 3 4 1 1 3 3 1
## [3516] 4 1 4 4 3 3 3 2 1 1 1 3 4 4 4 4 1 4 1 4 4 3 1 3 4 3 4 4 4 3 3 4 3 3 4 4 3
## [3553] 4 4 4 2 1 4 1 1 1 1 4 4 3 1 4 3 3 1 3 3 3 3 4 4 3 1 4 4 4 4 4 4 4 3 4 4 1
## [3590] 1 4 1 4 3 4 3 4 4 4 3 3 3 4 4 3 4 1 3 1 1 3 3 3 1 1 1 1 3 3 3 3 4 3 4 3 2
## [3627] 4 3 4 3 3 1 4 1 1 3 3 1 1 4 1 1 1 1 1 3 3 3 3 1 3 3 1 1 3 3 3 4 3 4 1 3 3
## [3664] 4 3 3 3 1 3 3 3 3 4 4 3 1 4 1 1 1 1 1 1 3 3 1 3 3 3 4 4 1 3 3 3 3 3 3 3 3
## [3701] 1 4 3 3 3 4 3 3 4 3 3 3 1 4 3 1 1 3 1 3 1 4 3 1 3 3 4 3 1 3 3 3 3 1 4 3 4
## [3738] 1 1 1 1 4 1 3 3 1 3 1 1 1 1 1 1 4 4 1 1 1 2 3 1 1 1 3 3 3 4 4 4 4 4 1 3 4
## [3775] 4 1 3 3 1 3 1 3 3 1 1 1 3 1 3 1 3 3 3 1 3 1 1 4 4 3 3 4 3 3 4 3 3 3 3 3 3
## [3812] 3 1 3 3 1 3 3 3 3 3 3 3 3 3 4 4 3 3 3 3 1 3 1 3 4 3 3 3 4 4 3 4 4 3 3 3 1
## [3849] 4 4 1 4 4 4 3 3 3 4 3 4 3 2 3 3 3 3 3 3 3 1 3 1 1 1 4 4 1 4 3 1 2 4 4 4 4
## [3886] 3 3 4 1 1 3 4 1 1 3 3 4 1 1 3 3 4 3 3 3 3 3 3 1 3 3 3 1 3 3 1 3 3 1 3 3 1
## [3923] 3 4 1 1 1 1 3 4 3 4 1 4 3 4 4 3 1 3 3 1 3 1 4 4 1 3 4 4 4 3 1 1 3 3 1 4 3
## [3960] 1 1 3 4 4 3 4 4 4 3 1 4 1 2 4 3 4 1 4 4 4 3 1 1 1 3 4 4 1 1 3 3 3 3 4 4 4
## [3997] 1 1 1 1 1 4 1 3 4 1 1 4 1 4 4 4 1 4 1 4 4 1 4 1 4 4 1 4 1 3 4 3 4 4 4 4 4
## [4034] 4 3 4 3 4 3 3 3 4 4 4 4 4 3 1 4 3 3 3 3 4 4 3 3 4 3 3 1 1 4 3 3 3 3 1 1 4
## [4071] 3 1 3 1 1 3 1 4 3 1 4 4 4 4 4 3 1 1 4 1 4 4 3 3 3 1 3 3 4 3 4 1 1 4 4 4 3
## [4108] 4 3 4 4 1 3 3 2 3 4 2 4 3 3 3 4 3 3 3 1 1 3 3 3 3 3 1 3 1 3 3 1 4 3 3 1 1
## [4145] 3 3 3 4 4 3 4 1 3 3 3 4 3 3 3 1 3 1 3 3 4 1 3 4 3 3 1 2 4 3 4 4 4 1 1 4 3
## [4182] 3 3 3 3 3 1 3 3 3 3 3 3 1 3 4 1 4 4 3 4 1 1 1 2 3 4 4 2 4 4 1 3 1 3 3 3 3
## [4219] 3 3 3 3 3 1 3 3 3 4 4 1 1 4 4 4 1 4 4 1 1 1 3 1 3 4 1 1 3 3 4 3 3 1 3 4 4
## [4256] 4 3 3 1 3 4 1 1 1 1 3 3 1 4 3 3 3 2 1 1 1 3 3 1 3 3 3 3 3 1 3 4 3 1 3 1 3
## [4293] 3 1 1 4 1 3 3 1 3 3 3 4 4 4 3 4 4 4 3 4 4 4 4 4 3 1 3 1 3 1 3 3 3 1 1 4 1
## [4330] 4 3 3 1 3 4 1 1 3 1 3 3 3 2 2 1 3 3 4 3 4 1 3 1 2 4 4 1 1 3 3 3 3 1 3 1 1
## [4367] 3 1 3 4 3 3 1 3 3 3 1 1 1 1 3 4 4 4 3 1 3 4 4 4 4 4 1 3 1 1 3 1 3 4 4 1 1
## [4404] 1 3 3 3 4 3 1 3 1 3 1 1 3 1 3 3 3 4 3 2 4 3 4 3 3 3 3 4 1 1 3 3 4 3 1 1 1
## [4441] 1 1 1 1 1 1 1 3 4 3 1 3 1 1 3 3 1 3 1 3 1 1 1 1 1 3 1 3 1 1 1 4 1 3 1 3 3
## [4478] 3 1 2 1 3 1 1 1 1 1 2 3 3 3 4 3 1 4 4 4 1 3 1 1 3 1 1 3 4 1 1 1 4 3 1 4 1
## [4515] 1 3 1 2 1 1 4 3 3 1 4 2 3 3 1 3 2 4 3 2 1 1 4 1 1 3 1 4 1 3 3 1 1 4 3 1 1
## [4552] 1 1 1 3 1 1 1 1 1 1 1 1 3 1 1 1 3 1 3 1 3 1 1 3 3 3 3 3 1 1 4 4 2 1 1 1 1
## [4589] 4 4 3 3 1 1 1 1 3 1 1 1 3 3 1 1 3 4 4 4 4 4 1 1 1 2 1 1 3 1 1 2 1 3 2 1 3
## [4626] 1 3 3 3 3 3 1 4 3 1 4 4 1 4 3 4 4 1 3 1 1 3 3 4 4 4 4 1 1 1 2 1 4 2 4 4 1
## [4663] 4 4 3 4 3 1 1 1 1 4 3 3 1 1 4 1 1 2 4 1 1 1 1 1 4 1 1 1 4 3 1 2 2 1 3 3 1
## [4700] 3 1 1 1 3 3 3 3 3 4 1 1 3 3 1 1 1 1 3 2 4 3 1 1 3 3 3 1 1 3 3 4 1 3 1 4 1
## [4737] 3 1 3 4 2 3 1 3 3 3 3 3 3 1 3 4 1 1 3 3 3 1 3 1 3 3 1 4 3 3 1 1 1 3 1 1 3
## [4774] 3 3 1 3 1 1 1 1 3 1 1 1 1 3 3 3 1 3 1 3 1 3 3 3 3 3 3 3 1 3 4 3 1 3 3 3 1
## [4811] 3 3 3 1 2 1 1 1 1 1 1 1 3 3 1 3 4 4 1 3 1 1 1 3 3 3 3 1 1 1 1 1 1 1 1 1 3
## [4848] 3 3 3 4 1 1 4 4 4 4 4 4 4 1 4 1 4 3 1 3 3 4 1 1 1 1 3 1 3 3 4 3 1 3 3 3 1
## [4885] 3 3 3 3 4 1 1 4 1 1 4 4 4 1 1 1 1 1 1 3 1 3 4 1 1 3 3 1 2 4 1 1 1 1 3 3 1
## [4922] 1 1 1 3 4 1 1 1 4 3 3 3 1 4 4 4 2 1 1 1 1 4 4 4 4 3 2 1 3 1 1 1 3 1 3 1 1
## [4959] 1 1 1 3 1 1 1 1 1 3 1 1 1 3 3 3 3 3 4 3 4 1 3 1 3 4 3 3 4 1 1 1 1 1 1 4 4
## [4996] 3 4 4 1 3 1 3 1 1 1 1 4 4 1 3 3 3 4 3 3 4 1 4 3 3 1 1 3 1 3 3 3 1 3 3 3 1
## [5033] 1 1 1 1 1 4 1 1 1 1 1 4 3 4 1 3 3 1 1 3 1 1 1 4 3 1 3 1 4 1 1 4 1 1 4 1 3
## [5070] 4 1 3 1 4 4 1 3 4 1 1 3 1 1 1 1 1 4 1 1 3 1 3 3 3 1 3 3 3 3 3 3 4 1 3 1 3
## [5107] 1 4 4 4 1 1 2 1 1 4 1 3 3 4 3 4 4 3 3 3 3 2 3 4 4 1 3 4 4 1 1 3 1 1 3 3 3
## [5144] 3 4 3 4 3 3 1 1 1 1 1 3 1 1 1 2 1 1 3 1 1 1 1 1 1 1 2 1 1 1 1 3 3 1 1 3
## [5181] 3 1 1 1 1 3 1 1 1 1 4 3 3 1 3 1 3 4 4 1 3 3 3 1 3 3 1 3 3 3 1 1 3 3 1 4 1
## [5218] 3 3 4 3 1 3 1 4 1 4 4 4 1 1 3 1 3 1 2 3 1 1 3 3 3 3 3 3 1 1 3 3 1 4 1 4 1
## [5255] 3 4 3 3 3 1 3 1 3 3 3 3 1 1 1 3 1 1 1 1 3 1 3 1 1 4 3 3 4 3 4 4 1 4 3 3 1
## [5292] 1 3 1 3 3 4 1 4 3 3 4 4 4 4 1 1 4 3 2 4 3 4 1 4 4 1 3 4 3 3 3 3 1 1 1 3 1
## [5329] 1 4 3 1 1 3 1 1 3 3 3 3 3 3 3 3 3 3 4 3 3 3 1 4 4 3 3 1 4 4 3 1 1 2 1 3 4
## [5366] 4 4 3 4 3 3 3 1 4 1 3 1 1 1 4 1 1 4 3 1 4 4 4 4 4 4 3 4 1 3 1 1 3 3 2 2 3
## [5403] 1 2 2 1 1 1 1 3 3 3 3 3 1 3 3 1 1 3 3 4 4 1 1 1 1 3 1 1 3 3 1 3 1 1 1 3 1
## [5440] 1 4 1 1 1 4 1 4 1 1 1 3 1 1 1 3 1 1 3 4 4 3 4 4 1 1 1 4 4 3 3 4 4 4 4 1 3
## [5477] 1 4 2 1 1 3 3 1 2 3 2 1 3 1 3 3 2 3 1 4 4 1 3 2 2 2 1 1 1 1 1 3 3 3 1 1 1
## [5514] 3 1 3 4 1 1 3 4 1 1 1 1 1 1 3 3 1 1 1 1 1 4 1 3 1 3 3 3 1 1 3 3 3 3 1 3 3
## [5551] 3 1 3 3 1 3 1 1 1 3 4 1 1 4 1 1 4 4 3 3 3 1 4 4 3 3 1 4 4 3 3 3 1 3 1 1 4
## [5588] 3 1 4 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 2 1 1 3 3 3 1 4 4 1 3 1 4 1 1 1 3 3 3
## [5625] 1 3 3 1 3 1 1 1 3 1 4 4 4 1 1 4 3 3 3 3 3 3 3 3 1 3 1 3 3 3 1 3 1 3 1 1 1
## [5662] 1 3 1 1 3 3 3 3 3 1 3 1 1 3 3 1 1 1 3 1 3 4 1 1 1 1 1 3 4 1 1 1 1 1 3 1 3
## [5699] 1 1 3 3 1 1 3 3 3 1 3 4 1 1 1 1 1 3 3 3 3 3 3 3 1 1 3 3 4 4 1 3 4 4 1 1 3
## [5736] 3 4 1 3 3 3 3 3 3 3 3 3 1 1 4 4 1 4 4 4 4 3 3 3 3 3 3 1 1 1 3 1 1 3 3 1 3
## [5773] 1 4 3 3 1 3 4 3 1 3 1 1 4 1 1 1 1 2 3 1 1 1 1 1 3 3 1 1 1 1 4 3 1 3 1 1 1
## [5810] 3 4 3 2 4 4 4 1 4 4 1 3 1 2 1 4 4 1 4 3 1 1 1 1 1 1 1 1 1 1 3 3 3 1 1 1 1
## [5847] 3 1 3 1 1 1 1 1 1 1 3 3 1 3 1 3 3 1 3 1 1 3 4 4 4 1 1 1 1 1 4 2 3 1 3 3 1
## [5884] 1 1 1 1 1 1 3 4 1 4 1 4 1 1 1 3 3 3 4 1 1 1 3 1 1 1 3 1 1 1 1 1 1 3 1 1 4
## [5921] 3 3 3 1 4 3 4 3 4 4 3 3 3 3 3 3 3 3 1 3 3 3 1 1 1 1 3 3 4 1 3 3 3 3 3 4 4
## [5958] 4 4 3 1 3 3 1 3 1 3 1 1 1 1 1 1 4 4 3 1 1 3 3 3 1 1 4 4 4 1 1 1 3 4 3 3 3
## [5995] 3 4 4 4 3 4 3 4 4 4 4 3 1 4 3 4 1 1 1 1 3 4 3 1 1 1 1 3 3 1 1 3 4 3 3 3 1
## [6032] 1 4 3 3 1 3 3 1 3 1 3 1 1 1 1 1 3 3 3 4 4 1 3 4 4 4 3 1 3 3 1 1 3 3 3 3 1
## [6069] 1 1 1 1 3 3 1 1 4 4 3 3 4 3 1 3 1 1 1 1 1 1 1 2 3 1 4 1 3 1 1 4 4 3 1 3 3
## [6106] 1 1 3 1 3 1 1 1 4 3 1 1 2 4 4 4 2 3 3 4 4 1 1 3 3 3 1 3 1 3 3 1 3 2 1 1 1
## [6143] 1 1 1 1 2 4 1 1 1 1 1 1 1 3 4 3 1 1 1 3 1 1 3 3 3 3 3 1 1 1 1 1 3 2 1 1 3
## [6180] 1 3 3 3 1 3 3 1 3 3 3 4 3 1 3 3 1 1 1 1 1 1 3 3 3 1 1 4 1 2 1 1 3 3 3 3 1
## [6217] 3 3 4 1 1 3 3 1 3 4 1 1 1 3 1 4 4 3 1 4 1 4 4 1 3 1 4 3 3 1 1 4 1 1 1 4 1
## [6254] 3 3 3 3 3 1 1 3 1 1 1 1 4 1 1 1 3 4 4 1 3 3 3 1 3 4 3 2 2 1 4 1 3 3 3 3 3
## [6291] 3 3 3 3 1 1 1 1 4 4 3 3 1 4 1 1 1 1 3 1 1 1 1 1 1 1 3 3 1 1 1 3 4 1 1 1 3
## [6328] 3 2 1 3 3 3 1 2 2 3 1 1 4 3 3 3 1 4 3 1 4 4 3 4 1 1 1 1 1 1 3 3 3 1 1 3 1
## [6365] 1 3 4 3 3 3 3 4 2 3 3 3 1 1 1 3 3 3 3 3 1 3 1 3 1 3 3 3 3 3 3 3 1 1 1 3 3
## [6402] 3 1 2 1 3 1 1 3 3 1 1 4 1 1 3 3 1 1 4 3 3 3 3 3 3 1 3 1 3 3 3 1 1 1 1 4 1
## [6439] 2 3 4 1 4 4 1 3 1 3 3 3 3 1 1 1 3 3 1 4 1 1 1 1 1 3 1 1 2 3 1 3 1 4 3 1 1
## [6476] 1 1 1 4 4 1 3 3 4 4 3 2 3 1 1 3 1 1 4 1 1 1
## 
## Within cluster sum of squares by cluster:
## [1]  726016.2  471359.5  835244.9 1010896.2
##  (between_SS / total_SS =  86.8 %)
## 
## Available components:
## 
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Displaying the cluster plot

```r
fviz_cluster(fit, data = df_wine)
```



# Question 2:

Defining distance and linkage functions

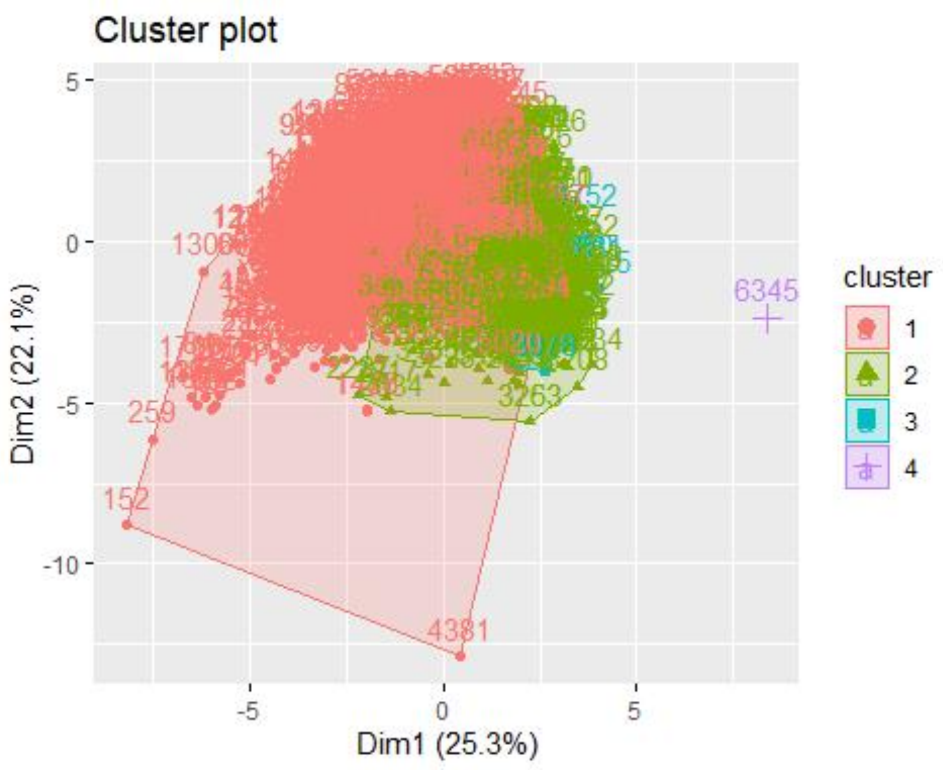
```
distance_functions <- c("euclidean", "manhattan")
linkage_functions <- c("complete", "ward.D2")
```

Using euclidean distances

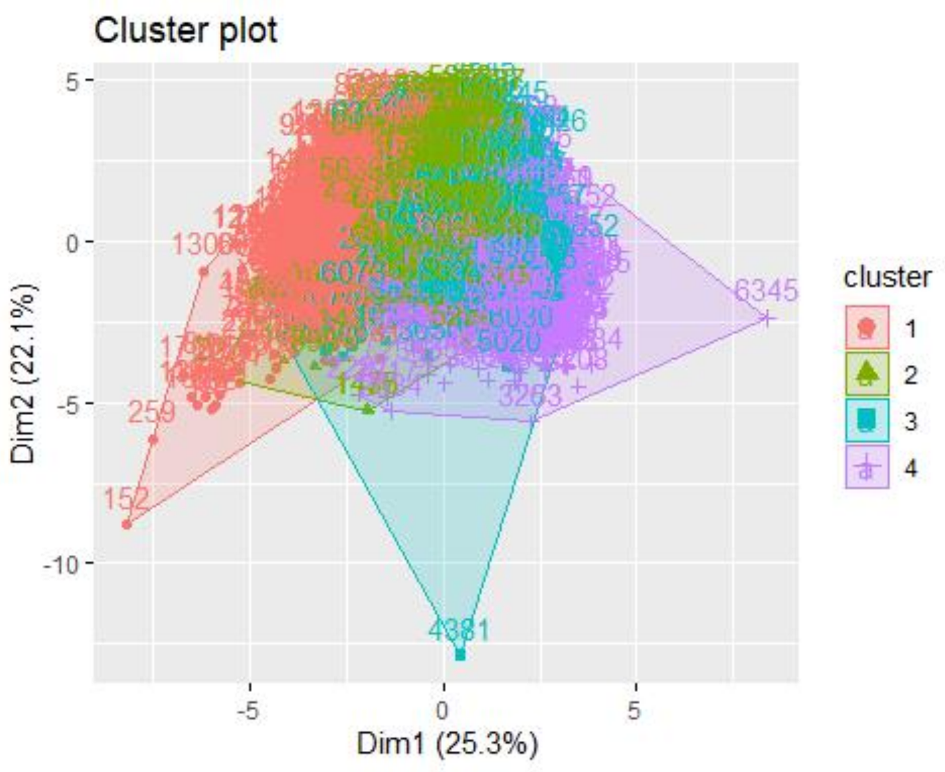
```
dist_mat_euclidean <- dist(df_wine, method = 'euclidean')
```

Euclidean and Complete

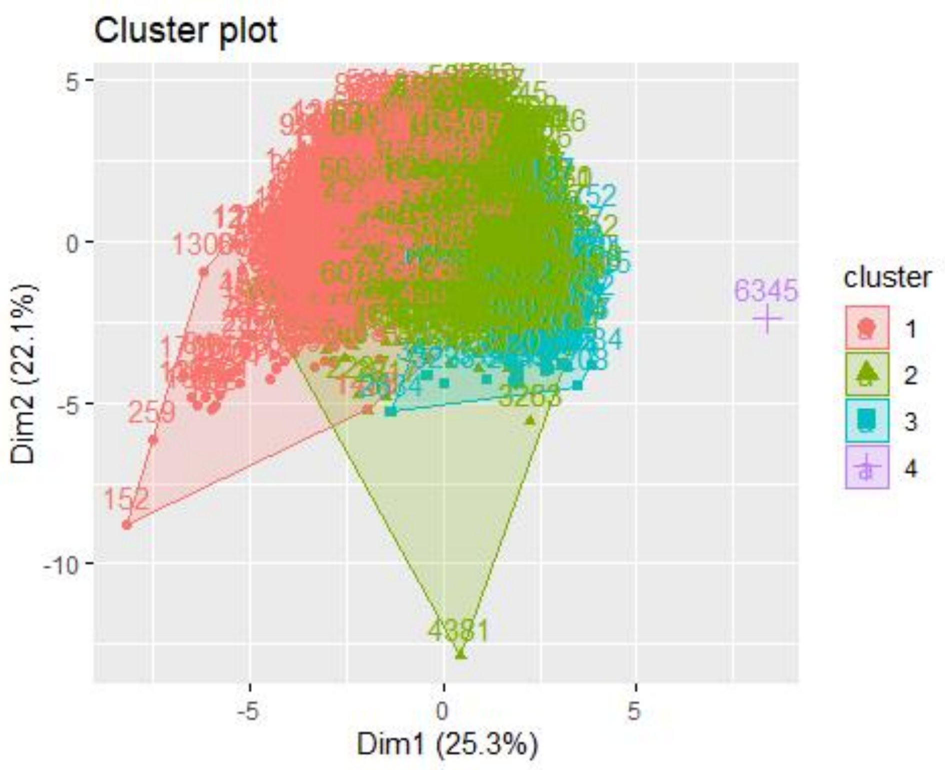
```
hfit_euclidean_complete <- hclust(dist_mat_euclidean, method = 'complete')
# Build the new model
hac_euclidean_complete <- cutree(hfit_euclidean_complete, k=4)
# Displaying the cluster plot
fviz_cluster(list(data = df_wine, cluster = hac_euclidean_complete))
```



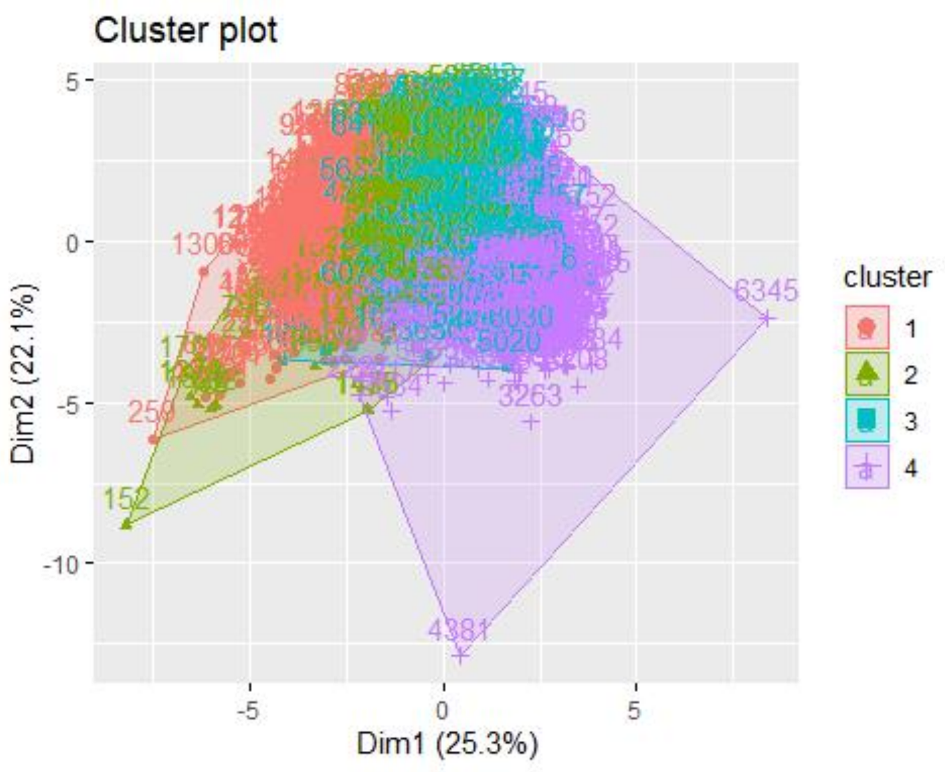
```
# Euclidean and Ward.D2
hfit_euclidean_ward <- hclust(dist_mat_euclidean, method = 'ward.D2')
# Build the new model
hac_euclidean_ward <- cutree(hfit_euclidean_ward, k=4)
# Displaying the cluster plot
fviz_cluster(list(data = df_wine, cluster = hac_euclidean_ward))
```



```
# Using manhattan distances
dist_mat_manhattan <- dist(df_wine, method = 'manhattan')

# Manhattan and Complete
hfit_manhattan_complete <- hclust(dist_mat_manhattan, method = 'complete')
# Build the new model
hac_manhattan_complete <- cutree(hfit_manhattan_complete, k=4)
# Displaying the cluster plot
fviz_cluster(list(data = df_wine, cluster = hac_manhattan_complete))
```



```
# Manhattan and Ward.D2
hfit_manhattan_ward <- hclust(dist_mat_manhattan, method = 'ward.D2')
# Build the new model
hac_manhattan_ward <- cutree(hfit_manhattan_ward, k=4)
# Displaying the cluster plot
fviz_cluster(list(data = df_wine, cluster = hac_manhattan_ward))
```



# Question 3:

Create a dataframe

```
result_1 <- data.frame(Type = df$type, Kmeans = fit$cluster, hac_euclidean_complete = 
hac_euclidean_complete, hac_euclidean_ward = hac_euclidean_ward, hac_manhattan_complete = 
hac_manhattan_complete, hac_manhattan_ward = hac_manhattan_ward)
```

Crosstab for K Means

```
result_1 %>% group_by(Kmeans) %>% select(Kmeans, Type) %>% table()
```

```
##       Type
## Kmeans  red white
##      1  301  1748
##      2 1240   137
##      3   56  1915
##      4    2  1098
```

Crosstabulation for hac_euclidean_complete

```
result_1 %>% group_by(hac_euclidean_complete) %>% select(hac_euclidean_complete, Type) %>% 
table()
```

```
##                       Type
## hac_euclidean_complete  red white
##                      1 1594  3259
##                      2    3  1629
##                    3    2     9
##                    4    0     1
```

Crosstabulation for hac_euclidean_ward

```
result_1 %>% group_by(hac_euclidean_ward) %>% select(hac_euclidean_ward, Type) %>% table()
```

```
##                   Type
## hac_euclidean_ward  red white
##                  1 1331   286
##                  2  215  1762
##                  3   50  1542
##                  4    3  1308
```

Crosstabulation for hac_manhattan_complete

```
result_1 %>% group_by(hac_manhattan_complete) %>% select(hac_manhattan_complete, Type)
%>% table()
```

```
##                       Type
## hac_manhattan_complete  red white
##                      1 1463   857
##                      2  134  3702
##                      3    2   338
##                      4    0     1
```

Crosstabulation for hac_manhattan_ward

```
result_1 %>% group_by(hac_manhattan_ward) %>% select(hac_manhattan_ward, Type) %>%
table()
```

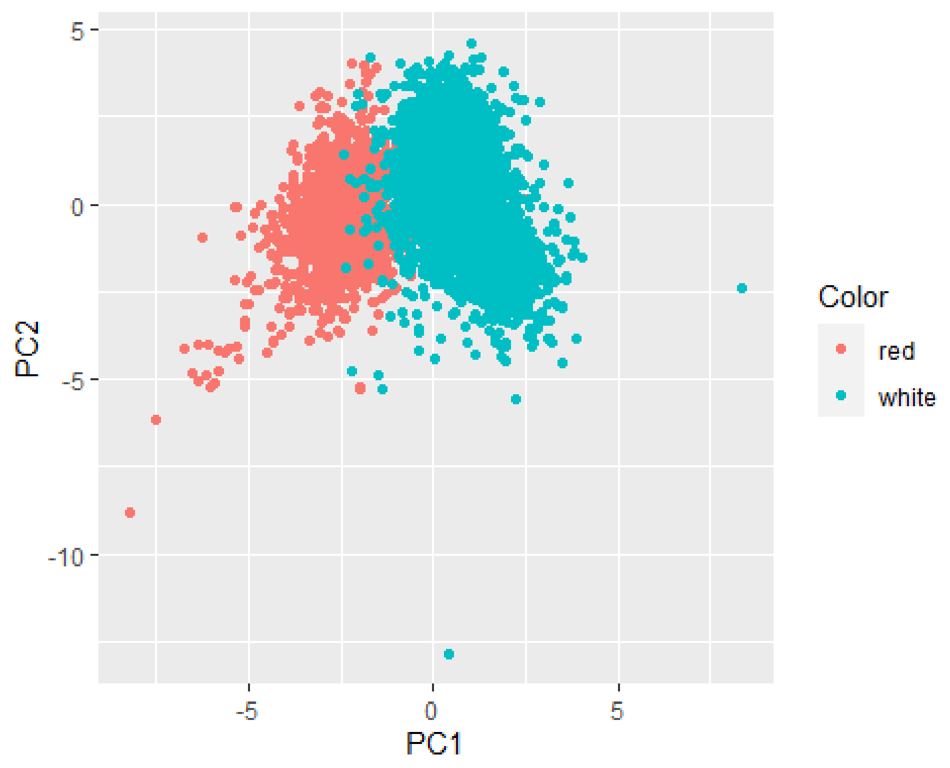
```
##                   Type
## hac_manhattan_ward  red white
##                  1 1193    67
##                  2  290   873
##                  3  105  2174
##                  4   11  1784
```

Question 4:

```
rotated_data = data.frame(PC1=pc1, PC2=pc2)
rotated_data$Color <- df$type
```

PCA plot to show type lable

```
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Color)) + geom_point()
```

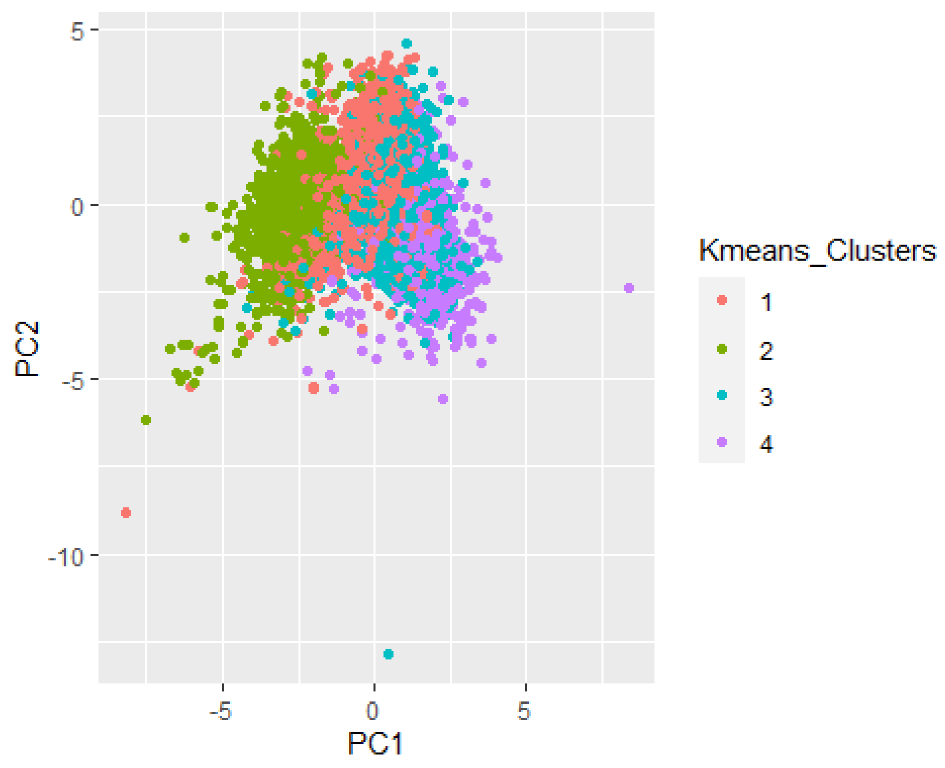


# PCA plot to show Kmeans cluster lable

```
rotated_data$Kmeans_Clusters = as.factor(fit$cluster)
```

Plot and color by labels

```
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Kmeans_Clusters)) + geom_point()
```

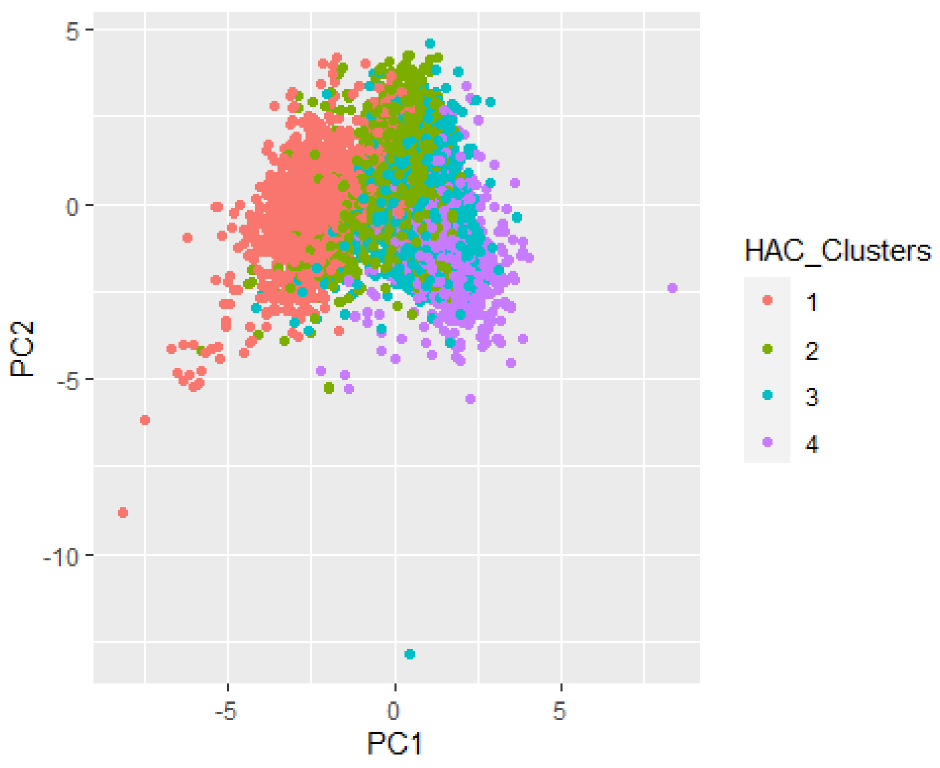


PCA plot to show HAC cluster lable

```
rotated_data$HAC_Clusters = as.factor(hac_euclidean_ward)
```

Plot and color by labels

```
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = HAC_Clusters)) + geom_point()
```



Question 5:

- By looking at the cluster plots, we can see that, k-means shows better separation of clusters than HAC. But we can see a clear separation between two classes if we plot the data by type column in original dataset.
- By looking at the cross tabulation,
- K-Means and HAC (Euclidean, Ward Linkage) seem to perform well, aligning closely with the original labels.
- HAC with Euclidean distance and Complete linkage formed one large cluster (Cluster 1).
- HAC with Manhattan distance and both linkages shows misclassification in Cluster 4.

Problem 4 :

```
library(dplyr)
```

Read Star Wars data

```
starwars_data <- dplyr::starwars
```

Removing variables (name, films, vehicles, starships)

```
starwars_data <- select(starwars_data, -c(name, films, vehicles, starships))

sapply(starwars, class)
```

```
##        name      height        mass  hair_color  skin_color   eye_color 
## "character"   "integer"   "numeric" "character" "character" "character" 
##  birth_year         sex      gender   homeworld     species       films 
##   "numeric" "character" "character" "character" "character"      "list" 
##    vehicles   starships 
##      "list"      "list"
```

```
starwars_data <- na.omit(starwars_data)

head(starwars_data)
```

```
## # A tibble: 6 × 10
##   height  mass hair_color skin_color eye_color birth_year sex   gender homeworld
##    <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>  <chr>    
## 1    172    77 blond      fair       blue            19   male  mascu… Tatooine 
## 2    202   136 none       white      yellow          41.9 male  mascu… Tatooine 
## 3    150    49 brown      light      brown           19   fema… femin… Alderaan 
## 4    178   120 brown, gr… light      blue            52   male  mascu… Tatooine 
## 5    165    75 brown      light      blue            47   fema… femin… Tatooine 
## 6    183    84 black      light      brown           24   male  mascu… Tatooine 
## # ℹ 1 more variable: species <chr>
```

Converting categorical variables to factors

```
starwars_data <- starwars_data %>%
  mutate_if(is.character,as.factor)

sapply(starwars_data, class)
```

```
##     height       mass hair_color skin_color  eye_color birth_year        sex 
##  "integer"  "numeric"   "factor"   "factor"   "factor"  "numeric"   "factor" 
##     gender  homeworld    species 
##   "factor"   "factor"   "factor"
```

Question 1:

```
library(cluster)
library(dendextend)
```

```
## Warning: package 'dendextend' was built under R version 4.3.2

## 
## ---------------------
## Welcome to dendextend version 1.17.1
## Type citation('dendextend') for how to cite the package.
## 
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
## 
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## You may ask questions at stackoverflow, use the r and dendextend tags:
```

```
##   https://stackoverflow.com/questions/tagged/dendextend
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
##
## Attaching package: 'dendextend'
## The following object is masked from 'package:rpart':
##
##     prune
## The following object is masked from 'package:stats':
##
##     cutree
library(cluster)
```

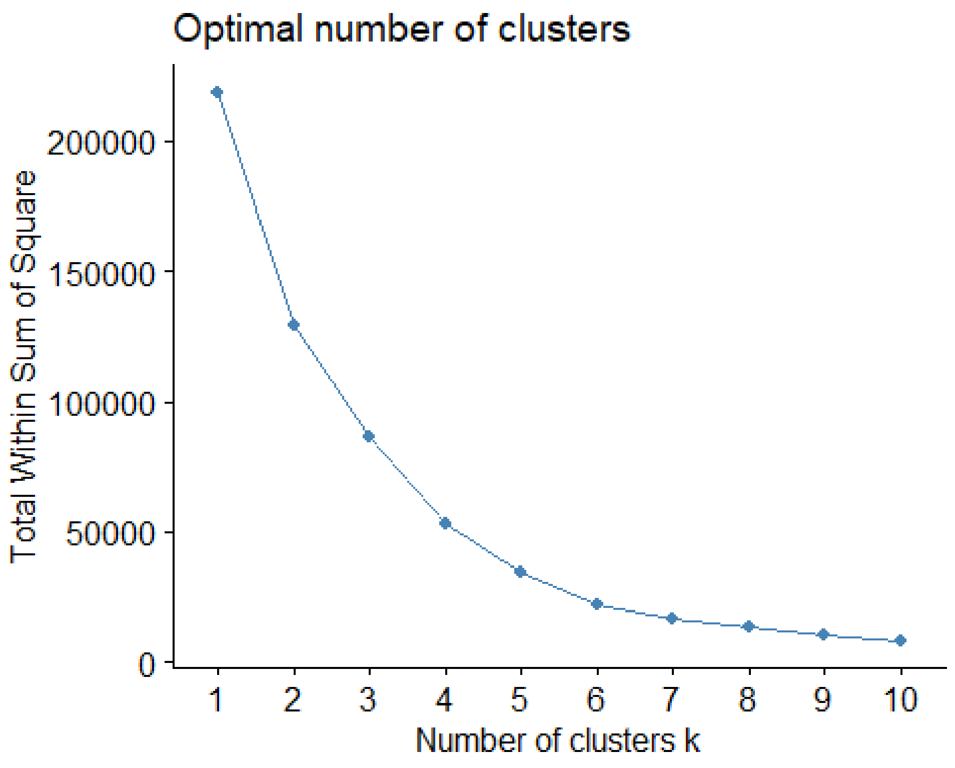
finding distances with the help of gower metric as we have categorical and numeric data

```
gower_dist <- daisy(starwars_data, metric = "gower")

summary(gower_dist)
## 406 dissimilarities, summarized :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.01758 0.42973 0.53501 0.53062 0.64062 0.86402
## Metric :  mixed ;  Types = I, I, N, N, N, I, N, N, N, N
## Number of objects : 29
```

Finding the optimum number of clusters with the help of below plots

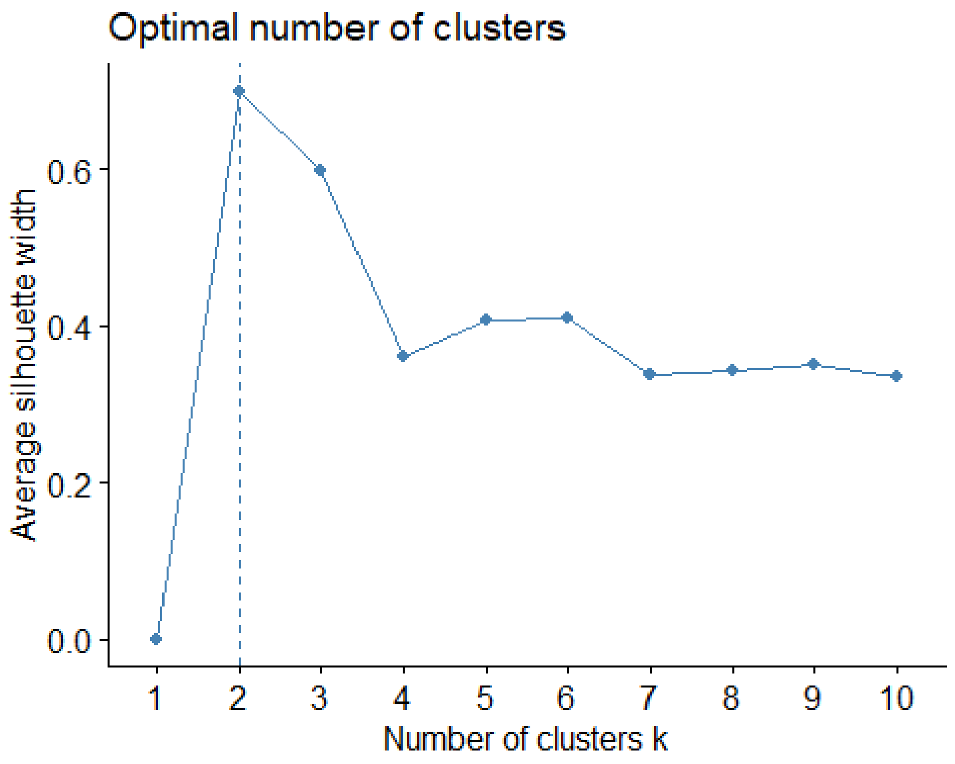
Knee plot

```
fviz_nbclust(starwars_data, FUN = hcut, method = "wss")
## Warning in stats::dist(x): NAs introduced by coercion
## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion
## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion
```



# silhouette method

```
fviz_nbclust(starwars_data, FUN = hcut, method = "silhouette")
## Warning in stats::dist(x): NAs introduced by coercion
## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion
## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion
```



# With the help of above plots, we will choose number of clusters = 6.

fit the data using average linkage method as we have more number of clusters

```
hfit <- hclust(gower_dist, method = 'average')
```

Build the new model

```
hac_gower_average <- cutree(hfit, k=6)
```

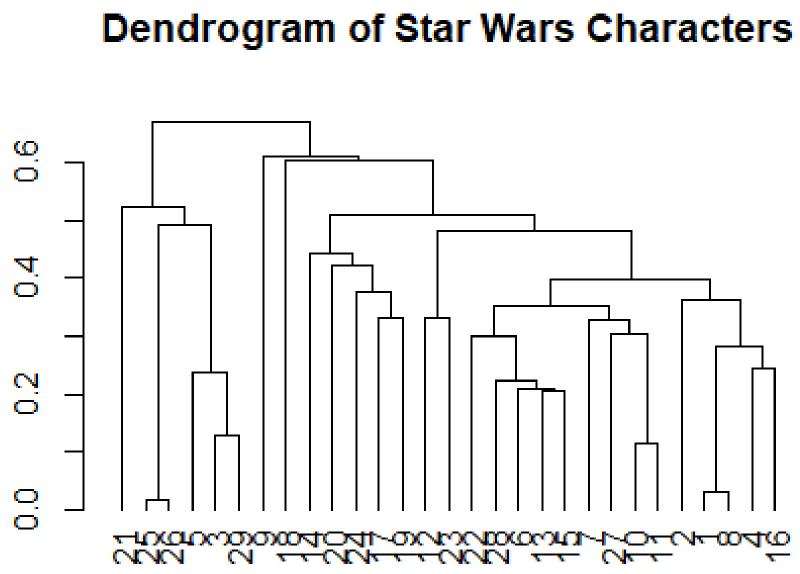
Question 2:

Build hierarchical agglomerative clustering using optimal_k_silhouette

```
hac_model <- hclust(gower_dist, method = "average")
dendrogram <- as.dendrogram(hac_model)
```

Plot dendrogram

```
plot(dendrogram, main = "Dendrogram of Star Wars Characters")
```



We can see that some of the characters does not seem to fit easily.

Advantages:
We can easily identify the patterns in the dendogram and we can decide the number of clusters by cutting the dendogram’s vertical lines.
Clustering algorithms can detect patterns and structures in data that are not visible when only looking at summary statistics.
Anomalies are identified based on deviations from the established clusters, providing for a more detailed interpretation of new data points.
Clustering methods are less sensitive to individual outliers compared to traditional methods.

Disadvantages:
The performance of clustering algorithms can be sensitive to the choice of parameters, such as the number of clusters (k) and the linkage method.
Noisy data may be treated as clusters which can lead to missclassifications.

Question 3:

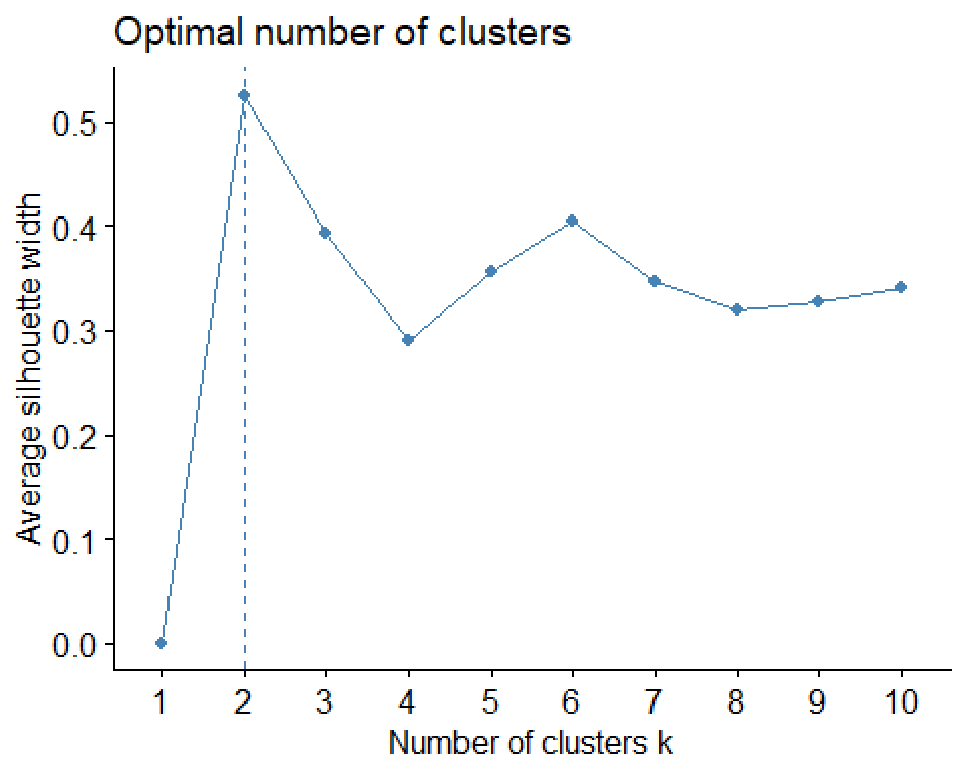
```
starwars_data

## # A tibble: 29 × 10
##    height  mass hair_color    skin_color eye_color birth_year sex    gender
##     <int> <dbl> <fct>         <fct>      <fct>          <dbl> <fct>  <fct>
##  1    172    77 blond         fair       blue            19   male   masculine
##  2    202   136 none          white      yellow          41.9 male   masculine
##  3    150    49 brown         light      brown           19   female feminine
##  4    178   120 brown, grey   light      blue            52   male   masculine
##  5    165    75 brown         light      blue            47   female feminine
##  6    183    84 black         light      brown           24   male   masculine
##  7    182    77 auburn, white fair       blue-gray       57   male   masculine
##  8    188    84 blond         fair       blue            41.9 male   masculine
##  9    228   112 brown         unknown    blue           200   male   masculine
## 10    180    80 brown         fair       brown           29   male   masculine
## # ℹ 19 more rows
## # ℹ 2 more variables: homeworld <fct>, species <fct>

sapply(starwars_data, class)

##     height       mass hair_color skin_color  eye_color birth_year        sex
##  "integer"  "numeric"   "factor"   "factor"   "factor"  "numeric"   "factor"
##     gender  homeworld    species
##   "factor"   "factor"   "factor"

sapply(starwars_data, function(x) n_distinct(x))

##     height       mass hair_color skin_color  eye_color birth_year        sex
##         19         19          8         14          8         26          2
##     gender  homeworld    species
##          2         20         11
```

Applying dummies

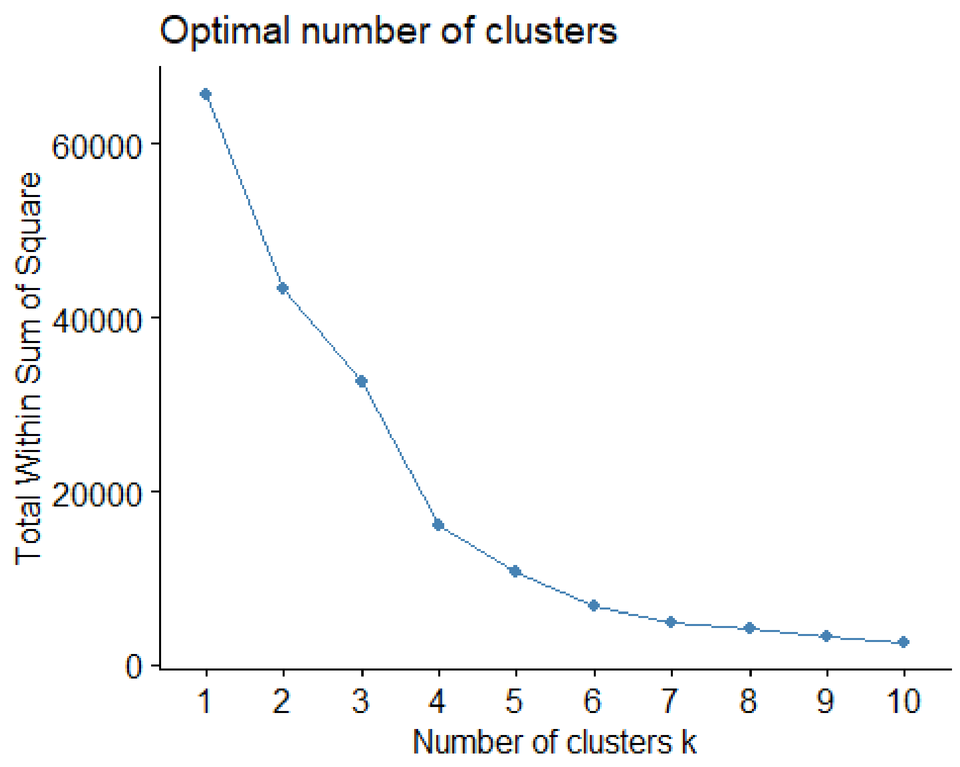
```
starwars_data_dummies <- starwars_data %>%
  model.matrix(~ . - 1, data = .) %>%  # Create dummy variables for all variables (excluding intercept)
  as.data.frame()
```

Finding the optimum number of clusters with the help of silhouette method

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



```
fviz_nbclust(starwars_data_dummies, kmeans, method = "wss")
```



# With the help of plots, we can take k=4 to fit the k means model.

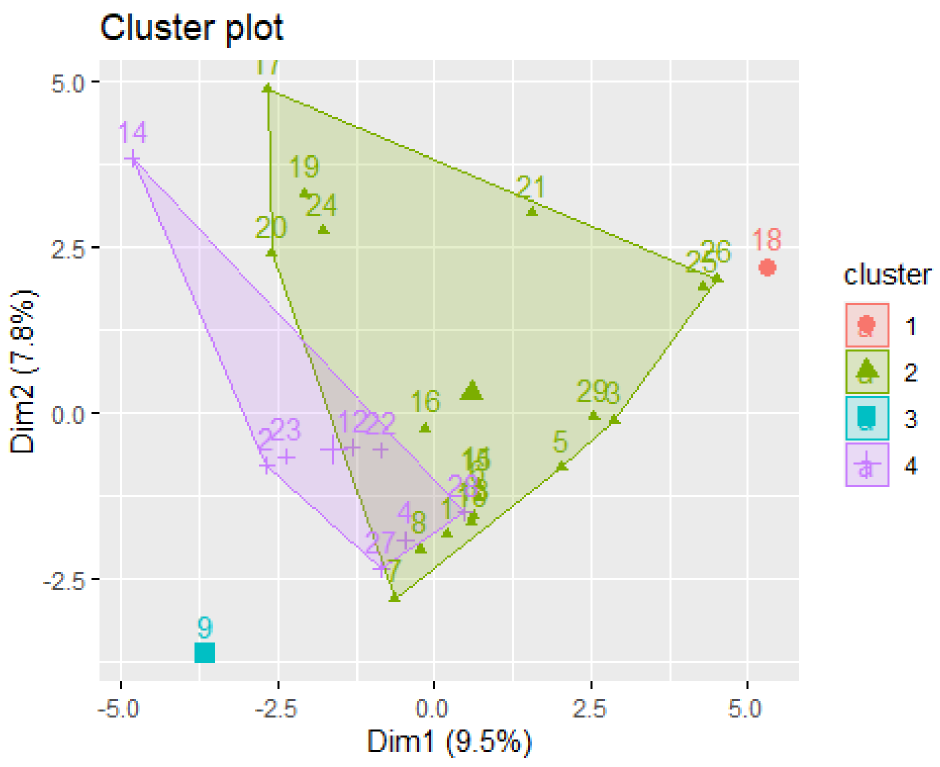
Fit the data with k-means

```
kmeans_4 <- kmeans(starwars_data_dummies, centers = 4, nstart = 25)
kmeans_4
```

```
## K-means clustering with 4 clusters of sizes 1, 19, 1, 8
##
## Cluster means:
##     height      mass hair_colorauburn, white hair_colorblack hair_colorblond
## 1  88.0000  20.00000              0.00000000       0.0000000       0.0000000
## 2 175.9474  71.28421              0.05263158       0.2631579       0.1052632
## 3 228.0000 112.00000              0.00000000       0.0000000       0.0000000
## 4 187.7500  96.12500              0.00000000       0.1250000       0.0000000
##   hair_colorbrown hair_colorbrown, grey hair_colorgrey hair_colornone
## 1       1.0000000                 0.000          0.000      0.0000000
## 2       0.2631579                 0.000          0.000      0.3157895
## 3       1.0000000                 0.000          0.000      0.0000000
## 4       0.0000000                 0.125          0.125      0.3750000
##   hair_colorwhite skin_colorbrown skin_colorbrown mottle skin_colordark
## 1            0.00               1             0.00000000     0.00000000
## 2            0.00               0             0.05263158     0.05263158
## 3            0.00               0             0.00000000     0.00000000
## 4            0.25               0             0.00000000     0.12500000
##   skin_colorfair skin_colorgreen skin_colorlight skin_colororange
## 1      0.0000000           0.000       0.0000000        0.0000000
## 2      0.3157895           0.000       0.2631579        0.1052632
## 3      0.0000000           0.000       0.0000000        0.0000000
## 4      0.1250000           0.125       0.1250000        0.0000000
##   skin_colorpale skin_colorred skin_colortan skin_colorunknown skin_colorwhite
## 1           0.00    0.00000000         0.000                 0           0.000
## 2           0.00    0.05263158         0.000                 0           0.000
## 3           0.00    0.00000000         0.000                 1           0.000
## 4           0.25    0.00000000         0.125                 0           0.125
##   skin_coloryellow eye_colorblue eye_colorblue-gray eye_colorbrown
## 1        0.0000000     0.0000000         0.00000000      1.0000000
## 2        0.1052632     0.3157895         0.05263158      0.3157895
## 3        0.0000000     1.0000000         0.00000000      0.0000000
## 4        0.0000000     0.1250000         0.00000000      0.3750000
##   eye_colorhazel eye_colororange eye_colorred eye_coloryellow birth_year
## 1      0.0000000       0.0000000        0.000      0.00000000    8.00000
## 2      0.1052632       0.1052632        0.000      0.05263158   37.81053
## 3      0.0000000       0.0000000        0.000      0.00000000  200.00000
## 4      0.0000000       0.0000000        0.125      0.37500000   70.11250
##     sexmale gendermasculine homeworldBespin homeworldCerea
## 1 1.0000000       1.0000000      0.00000000          0.000
## 2 0.6842105       0.6842105      0.05263158          0.000
## 3 1.0000000       1.0000000      0.00000000          0.000
## 4 1.0000000       1.0000000      0.00000000          0.125
##   homeworldConcord Dawn homeworldCorellia homeworldDathomir homeworldDorin
## 1                 0.000         0.0000000        0.00000000     0.00000000
## 2                 0.000         0.1052632        0.05263158     0.05263158
## 3               0.000        0.0000000       0.00000000     0.00000000
## 4               0.125        0.0000000       0.00000000     0.00000000
##   homeworldEndor homeworldHaruun Kal homeworldKamino homeworldKashyyyk
## 1              1               0.000      0.00000000                 0
## 2              0               0.000      0.05263158                 0
## 3              0               0.000      0.00000000                 1
## 4              0               0.125      0.00000000                 0
##   homeworldMirial homeworldMon Cala homeworldNaboo homeworldRyloth
## 1       0.0000000        0.00000000      0.0000000      0.00000000
## 2       0.1052632        0.05263158      0.1052632      0.05263158
## 3       0.0000000        0.00000000      0.0000000      0.00000000
## 4       0.0000000        0.00000000      0.1250000      0.00000000
##   homeworldSerenno homeworldSocorro homeworldStewjon homeworldTatooine
## 1            0.000       0.00000000       0.00000000         0.0000000
## 2            0.000       0.05263158       0.05263158         0.2105263
## 3            0.000       0.00000000       0.00000000         0.0000000
## 4            0.125       0.00000000       0.00000000         0.2500000
##   homeworldTrandosha speciesEwok speciesGungan speciesHuman speciesKel Dor
## 1              0.000           1    0.00000000    0.0000000     0.00000000
## 2              0.000           0    0.05263158    0.6315789     0.05263158
## 3              0.000           0    0.00000000    0.0000000     0.00000000
## 4              0.125           0    0.00000000    0.7500000     0.00000000
##   speciesMirialan speciesMon Calamari speciesTrandoshan speciesTwi'lek
## 1       0.0000000          0.00000000             0.000     0.00000000
## 2       0.1052632          0.05263158             0.000     0.05263158
## 3       0.0000000          0.00000000             0.000     0.00000000
## 4       0.0000000          0.00000000             0.125     0.00000000
##   speciesWookiee speciesZabrak
## 1              0    0.00000000
## 2              0    0.05263158
## 3              1    0.00000000
## 4              0    0.00000000
##
## Clustering vector:
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
##  2  4  2  4  2  2  2  2  3  2  2  4  2  4  2  2  2  1  2  2  2  4  4  2  2  2
## 27 28 29
##  4  4  2
##
## Within cluster sum of squares by cluster:
## [1]    0.000 8294.439    0.000 7666.259
##  (between_SS / total_SS =  75.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

display the cluster plot

```
fviz_cluster(kmeans_4, data = starwars_data_dummies)
```



Question 4:

Create a dataframe

```
result_4 <- data.frame(Species = starwars_data$species, HAC = hac_gower_average, Kmeans =
kmeans_4$cluster)
```

Crosstabulation for HAC

```
result_4 %>% group_by(HAC) %>% select(HAC, Species) %>% table()

##    Species
## HAC Cerean Ewok Gungan Human Kel Dor Mirialan Mon Calamari Trandoshan Twi'lek
##   1      1    0      0    15       0        0            0          0       0
##   2      0    0      0     3       0        2            0          0       0
##   3      0    0      0     0       0        0            0          0       0
##   4      0    0      1     0       1        0            1          1       0
##   5      0    1      0     0       0        0            0          0       0
##   6      0    0      0     0       0        0            0          0       1
##    Species
## HAC Wookiee Zabrak
##   1     0     0
##   2     0     0
##   3     1     0
##   4     0     1
##   5     0     0
##   6     0     0
```

Crosstabulation for K Means

```
result_4 %>% group_by(Kmeans) %>% select(Kmeans, Species) %>% table()

##       Species
## Kmeans Cerean Ewok Gungan Human Kel Dor Mirialan Mon Calamari Trandoshan
##      1      0    1      0     0       0        0            0          0
##      2      0    0      1    12       1        2            1          0
##      3      0    0      0     0       0        0            0          0
##      4      1    0      0     6       0        0            0          1
##       Species
## Kmeans Twi'lek Wookiee Zabrak
##      1       0       0      0
##      2       1       0      1
##      3       0       1      0
##      4       0       0      0
```

- By looking at the cross tabulation of K-means and HAC, we can say both the methods performed similar in terms of clustering the data.
- We can consider HAC performed slightly better as it has differentiated the data very well between all the classes.