Homework 2

Rachael Hageman Blair

2023-03-27

Loading Packages:

```
library(recommenderlab)
```

```
## Loading required package: Matrix
```

```
## Loading required package: arules
```

```
## Warning: package 'arules' was built under R version 4.2.3
```

```
##
## Attaching package: 'arules'
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
## Loading required package: proxy
```

```
##
## Attaching package: 'proxy'
```

```
## The following object is masked from 'package:Matrix':
##
##     as.matrix
```

```
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
```

```
## The following object is masked from 'package:base':
##
##     as.matrix
```

```
## Registered S3 methods overwritten by 'registry':
##   method               from
##   print.registry_field proxy
##   print.registry_entry proxy
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
library(datasets)
library("cluster")
library("multtest")
```

```
## Loading required package: BiocGenerics
```

```
##
## Attaching package: 'BiocGenerics'

## The following object is masked from 'package:recommenderlab':
##
##     normalize

## The following objects are masked from 'package:arules':
##
##     duplicated, intersect, match, setdiff, sort, union, unique

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which.max, which.min

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
library("fpc")

## Warning: package 'fpc' was built under R version 4.2.3
library("bootcluster")

## Warning: package 'bootcluster' was built under R version 4.2.3

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
library("fossil")

## Warning: package 'fossil' was built under R version 4.2.3

## Loading required package: sp

## Warning: package 'sp' was built under R version 4.2.3

## Loading required package: maps

## Warning: package 'maps' was built under R version 4.2.3

##
## 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
```

Question 1:

Consider the MovieLense data that is available in the recommenderlab package

>data(MovieLense)

>?MovieLense

The data was collected through the MovieLens web site during a seven-month, and contains

about 100,000 ratings (1-5) from 943 users on 1664 movies. See the help file on the data

To understand how to best manipulate the object. Design and evaluate a user-based recommender system. Create the system so that outputs a user's top ten recommendations. Demo it on 3 users. Read in the Movie Lense data

```
data(MovieLense)
Ratings <- MovieLense
head(Ratings) # 6 x 1664
```

```
## 6 x 1664 rating matrix of class 'realRatingMatrix' with 789 ratings.
```

```
dim(Ratings) # 943 1664
```

```
## [1]  943 1664
```

Look at the first few ratings of the first user

```
head(as(Ratings[1,], "list")[[1]])
```

```
##                                       Toy Story (1995)
##                                                      5
##                                       GoldenEye (1995)
##                                                      3
##                                      Four Rooms (1995)
##                                                      4
##                                      Get Shorty (1995)
##                                                      3
##                                         Copycat (1995)
##                                                      3
## Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)
##                                                      5
```

Create a "realRatingMatrix (not required as MovieLense is already an object of realRatingMatrix but doesn't make a difference)

```
R <- as(Ratings, "realRatingMatrix")
```

Get the Rating Matrix

```
dim(getRatingMatrix(R))
```

```
## [1]  943 1664
```

```
getRatingMatrix(R)[1:10, 1:10]
```

```
## 10 x 10 sparse Matrix of class "dgCMatrix"
##    [[ suppressing 10 column names 'Toy Story (1995)', 'GoldenEye (1995)', 'Four Rooms (1995)' ... ]]
```

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

Create a recommender system

```
recommenderRegistry$get_entries(dataType = "realRatingMatrix")
```

```
## $HYBRID_realRatingMatrix
## Recommender method: HYBRID for realRatingMatrix Description: Hybrid
##   recommender that aggegates several recommendation strategies using
##   weighted averages. Reference: NA
## Parameters:
##   recommenders weights aggregation_type
## 1         NULL    NULL            "sum"
##
## $ALS_realRatingMatrix
## Recommender method: ALS for realRatingMatrix Description: Recommender
##   for explicit ratings based on latent factors, calculated by
##   alternating least squares algorithm. Reference: Yunhong Zhou, Dennis
##   Wilkinson, Robert Schreiber, Rong Pan (2008). Large-Scale Parallel
##   Collaborative Filtering for the Netflix Prize, 4th Int'l Conf.
##   Algorithmic Aspects in Information and Management, LNCS 5034.
## Parameters:
##   normalize lambda n_factors n_iterations min_item_nr seed
## 1      NULL    0.1        10           10           1 NULL
##
## $ALS_implicit_realRatingMatrix
## Recommender method: ALS_implicit for realRatingMatrix Description:
##   Recommender for implicit data based on latent factors, calculated by
##   alternating least squares algorithm. Reference: Yifan Hu, Yehuda
##   Koren, Chris Volinsky (2008). Collaborative Filtering for Implicit
##   Feedback Datasets, ICDM '08 Proceedings of the 2008 Eighth IEEE
##   International Conference on Data Mining, pages 263-272.
## Parameters:
##   lambda alpha n_factors n_iterations min_item_nr seed
## 1    0.1    10        10           10           1 NULL
##
## $IBCF_realRatingMatrix
## Recommender method: IBCF for realRatingMatrix Description: Recommender
##   based on item-based collaborative filtering. Reference: NA
## Parameters:
##    k   method normalize normalize_sim_matrix alpha na_as_zero
## 1 30 "cosine"  "center"                FALSE   0.5      FALSE
##
## $LIBMF_realRatingMatrix
## Recommender method: LIBMF for realRatingMatrix Description: Matrix
```

```
##   factorization with LIBMF via package recosystem
##   (https://cran.r-project.org/web/packages/recosystem/vignettes/introduction.html).
##   Reference: NA
## Parameters:
##   dim costp_l2 costq_l2 nthread verbose
## 1  10     0.01     0.01       1   FALSE
##
## $POPULAR_realRatingMatrix
## Recommender method: POPULAR for realRatingMatrix Description:
##   Recommender based on item popularity. Reference: NA
## Parameters:
##   normalize
## 1  "center"
##                                                    aggregationRatings
## 1 new("standardGeneric", .Data = function (x, na.rm = FALSE, dims = 1,
##                                                  aggregationPopularity
## 1 new("standardGeneric", .Data = function (x, na.rm = FALSE, dims = 1,
##
## $RANDOM_realRatingMatrix
## Recommender method: RANDOM for realRatingMatrix Description: Produce
##   random recommendations (real ratings). Reference: NA
## Parameters: None
##
## $RERECOMMEND_realRatingMatrix
## Recommender method: RERECOMMEND for realRatingMatrix Description:
##   Re-recommends highly rated items (real ratings). Reference: NA
## Parameters:
##   randomize minRating
## 1         1        NA
##
## $SVD_realRatingMatrix
## Recommender method: SVD for realRatingMatrix Description: Recommender
##   based on SVD approximation with column-mean imputation. Reference: NA
## Parameters:
##    k maxiter normalize
## 1 10     100  "center"
##
## $SVDF_realRatingMatrix
## Recommender method: SVDF for realRatingMatrix Description: Recommender
##   based on Funk SVD with gradient descend
##   (https://sifter.org/~simon/journal/20061211.html). Reference: NA
## Parameters:
##    k gamma lambda min_epochs max_epochs min_improvement normalize verbose
## 1 10 0.015  0.001         50        200           1e-06  "center"   FALSE
##
## $UBCF_realRatingMatrix
## Recommender method: UBCF for realRatingMatrix Description: Recommender
##   based on user-based collaborative filtering. Reference: NA
## Parameters:
##     method nn sample weighted normalize min_matching_items min_predictive_items
## 1 "cosine" 25  FALSE     TRUE  "center"                  0                   0
```

```
recommender_popularity <- Recommender(R[943:1], method = "POPULAR")
names(getModel(recommender_popularity))
```

```
## [1] "topN"                  "ratings"               "normalize"
## [4] "aggregationRatings"    "aggregationPopularity" "verbose"
```

```
getModel(recommender_popularity)$topN
```

```
## Recommendations as 'topNList' with n = 1664 for 1 users.
```

Create top 10 recommendations for 3 users

```
recom <- predict(recommender_popularity, R[1:3], n=10)
recom
```

```
## Recommendations as 'topNList' with n = 10 for 3 users.
```

```
as(recom, "list")
```

```
## $`1`
##  [1] "Titanic (1997)"
##  [2] "Schindler's List (1993)"
##  [3] "L.A. Confidential (1997)"
##  [4] "Casablanca (1942)"
##  [5] "One Flew Over the Cuckoo's Nest (1975)"
##  [6] "Rear Window (1954)"
##  [7] "To Kill a Mockingbird (1962)"
##  [8] "Boot, Das (1981)"
##  [9] "Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1963)"
## [10] "North by Northwest (1959)"
##
## $`2`
##  [1] "Raiders of the Lost Ark (1981)"   "Silence of the Lambs, The (1991)"
##  [3] "Schindler's List (1993)"          "Shawshank Redemption, The (1994)"
##  [5] "Empire Strikes Back, The (1980)"  "Return of the Jedi (1983)"
##  [7] "Usual Suspects, The (1995)"       "Casablanca (1942)"
##  [9] "Pulp Fiction (1994)"              "Princess Bride, The (1987)"
##
## $`3`
##  [1] "Star Wars (1977)"                 "Godfather, The (1972)"
##  [3] "Fargo (1996)"                     "Raiders of the Lost Ark (1981)"
##  [5] "Silence of the Lambs, The (1991)" "Titanic (1997)"
##  [7] "Shawshank Redemption, The (1994)" "Empire Strikes Back, The (1980)"
##  [9] "Usual Suspects, The (1995)"       "Casablanca (1942)"
```

Evaluation splitting the data into train and test

```
eval<- evaluationScheme(R,method = "split", given = 15, train=0.5, goodRating=4)
```

```
## as(<dgCMatrix>, "dgTMatrix") is deprecated since Matrix 1.5-0; do as(., "TsparseMatrix") instead
```

```
eval
```

```
## Evaluation scheme with 15 items given
## Method: 'split' with 1 run(s).
## Training set proportion: 0.500
## Good ratings: >=4.000000
## Data set: 943 x 1664 rating matrix of class 'realRatingMatrix' with 99392 ratings.
```

Building a recommender model using user based collaborative filtering

```
userbased_model<- Recommender(getData(eval,"train"), "UBCF")
userbased_model
```

```
## Recommender of type 'UBCF' for 'realRatingMatrix'
## learned using 471 users.
```

Evaluation of top-N recommender algorithm using the Given-3 protocol i.e, for the test users all but 3 are withheld for evaluation.

```
scheme<- evaluationScheme(R, method="cross",k=4, given=3, goodRating=4)
scheme
```

```
## Evaluation scheme with 3 items given
## Method: 'cross-validation' with 4 run(s).
## Good ratings: >=4.000000
## Data set: 943 x 1664 rating matrix of class 'realRatingMatrix' with 99392 ratings.
```

Using the created evaluating scheme to evaluate the recommender method popular evaluating the top 1, 3, 5, 10,15,20 recommendation lists

```
results<- evaluate(scheme, method = "POPULAR", type="topNList", n=c(1,3,5,10,15,20))
```

```
## POPULAR run fold/sample [model time/prediction time]
##   1  [0sec/0.54sec]
##   2  [0sec/0.54sec]
##   3  [0sec/0.3sec]
##   4  [0sec/0.33sec]
```

```
results
```

```
## Evaluation results for 4 folds/samples using method 'POPULAR'.
```

```
getConfusionMatrix(results)[[1]]
```

```
##             TP         FP       FN       TN    N precision     recall
## [1,] 0.5630252  0.4369748 58.60084 1601.399 1661 0.5630252 0.01480784
## [2,] 1.3697479  1.6302521 57.79412 1600.206 1661 0.4565826 0.03247060
## [3,] 2.1092437  2.8907563 57.05462 1598.945 1661 0.4218487 0.04494890
## [4,] 3.7394958  6.2605042 55.42437 1595.576 1661 0.3739496 0.08120526
## [5,] 4.9747899 10.0252101 54.18908 1591.811 1661 0.3316527 0.10603617
## [6,] 6.1722689 13.8277311 52.99160 1588.008 1661 0.3086134 0.12651846
##             TPR          FPR  n
## [1,] 0.01480784 0.0002685241  1
## [2,] 0.03247060 0.0010061794  3
## [3,] 0.04494890 0.0017834832  5
## [4,] 0.08120526 0.0038690161 10
## [5,] 0.10603617 0.0062038919 15
## [6,] 0.12651846 0.0085599764 20
```

From the confusion matrix we can see each user recommendations has good amount of True postives and negatives which implies that model built has a good accuracy.

Question 2:

2) Data released from the US department of Commerce, Bureau of the Census is available in R (see, data(state)). Read the data

```
data(state)
dats <- state.x77
head(dats)
```

```
##            Population Income Illiteracy Life Exp Murder HS Grad Frost   Area
## Alabama          3615   3624        2.1    69.05   15.1    41.3    20  50708
## Alaska            365   6315        1.5    69.31   11.3    66.7   152 566432
## Arizona          2212   4530        1.8    70.55    7.8    58.1    15 113417
## Arkansas         2110   3378        1.9    70.66   10.1    39.9    65  51945
## California      21198   5114        1.1    71.71   10.3    62.6    20 156361
## Colorado         2541   4884        0.7    72.06    6.8    63.9   166 103766
```

scale the data

```
dats_scale <- scale(dats)
```

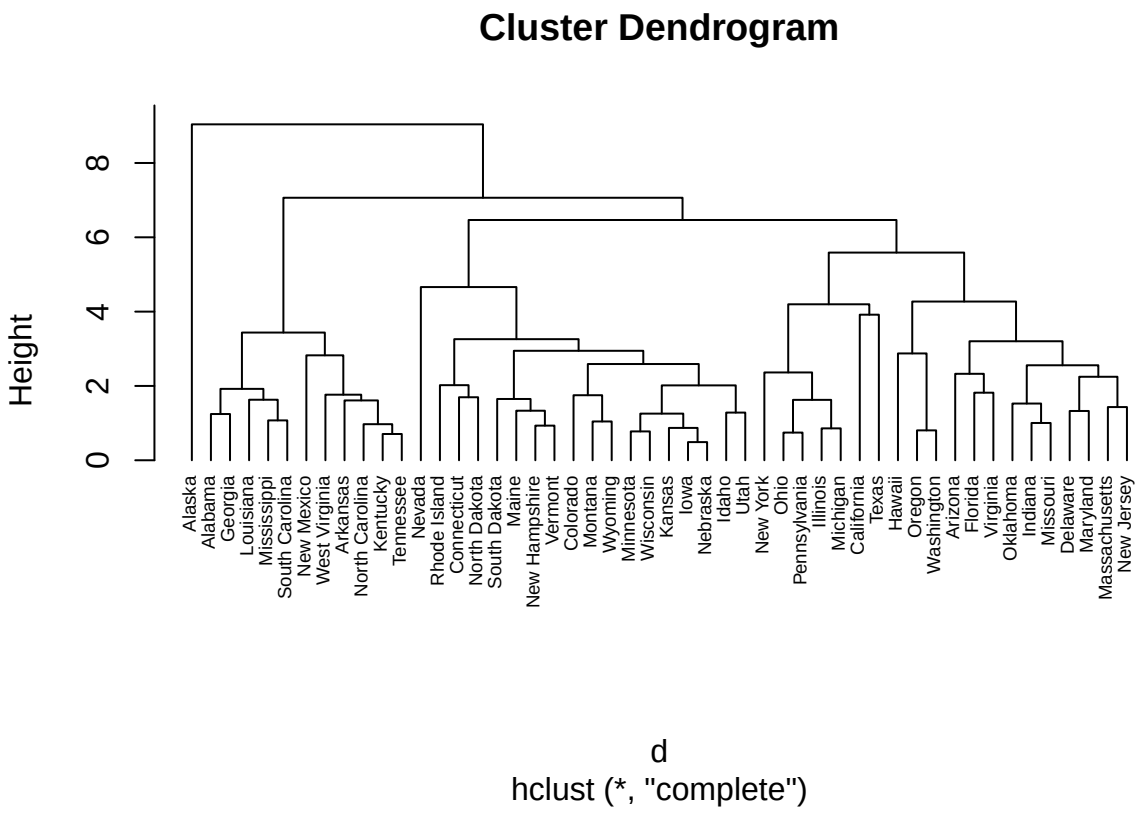
a) Focus on the data Population, Income, Illiteracy, Life Exp, Murder, HS Grad, Frost, Area

```
d <- dist(dats_scale)
dim(as.matrix(d))
```

Cluster this states using hierarchical clustering. Keep the class labels (region, or state name) in mind, but do not use them in the modeling.

```
## [1] 50 50
```

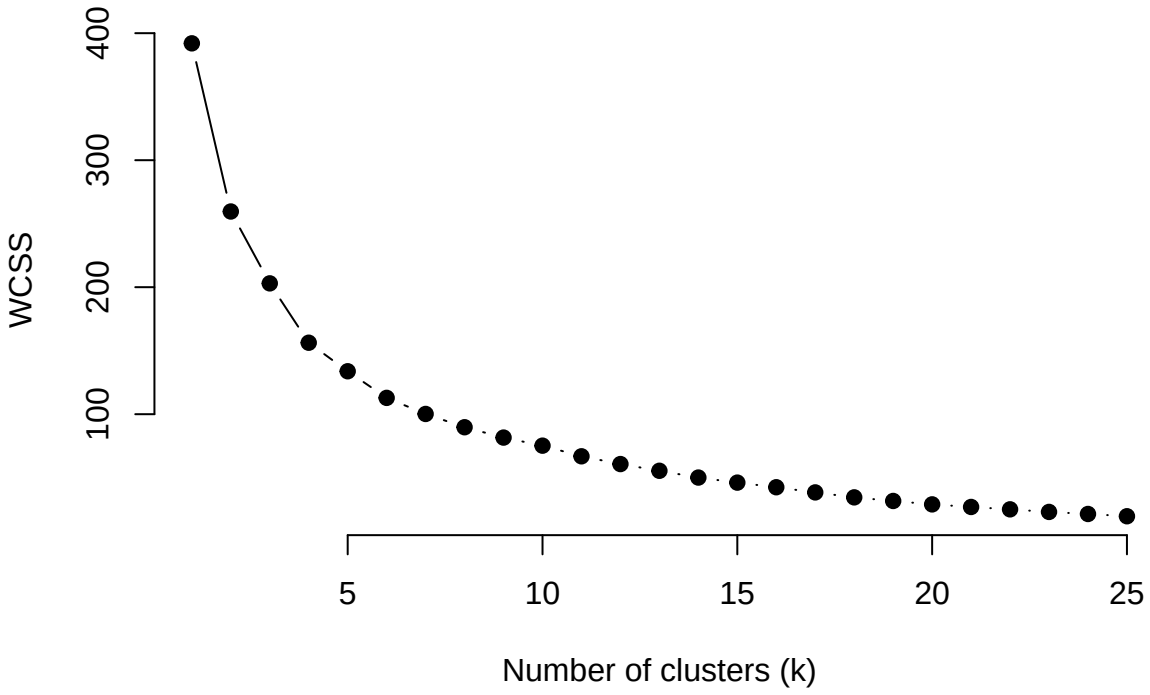
```
hc <- hclust(d,  method = "complete")
plot(hc, hang = -1,cex=0.6)
```



b) Cluster the states using k-means. Justify your choice of k. First we need to find the optimal number of clusters (K). In order to achieve this I have plotted an elbow plot.

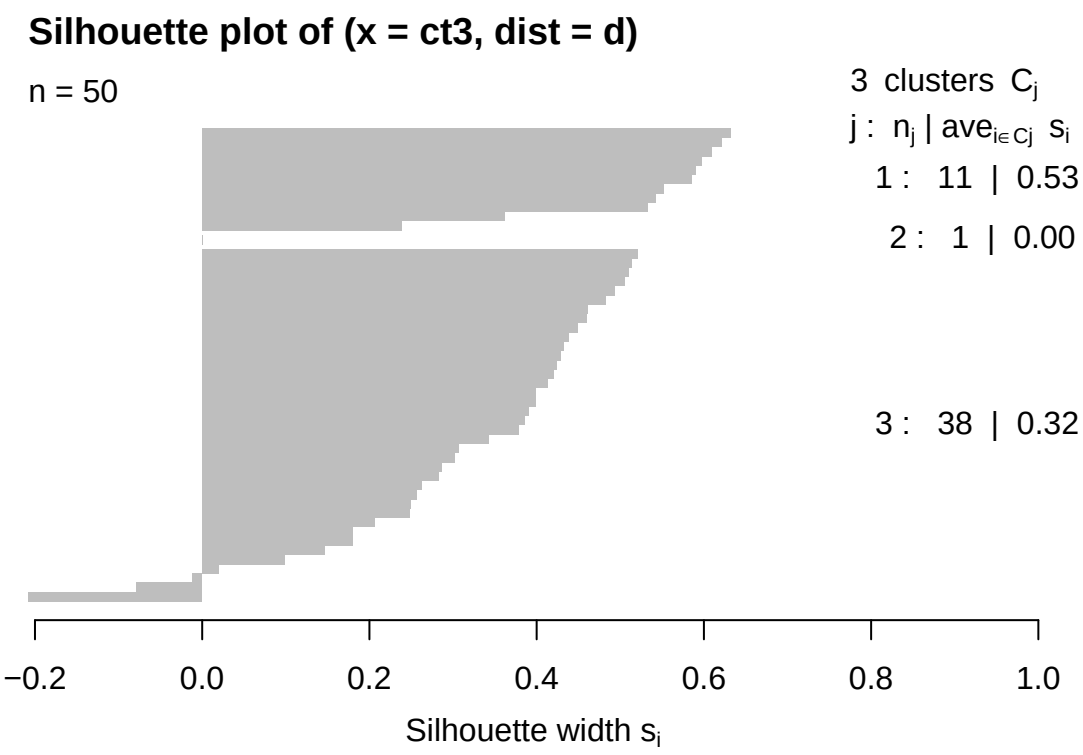
```
# calculate within-cluster sum of squares (WCSS) for different values of k
wcss <- c()
for (i in 1:25) {
  kmeans_fit <- kmeans(dats_scale, centers=i, nstart=25)
  wcss[i] <- kmeans_fit$tot.withinss
}

# plot elbow curve
#x11()
plot(1:25, wcss, type="b", pch=19, frame=FALSE,
     xlab="Number of clusters (k)", ylab="WCSS")
```



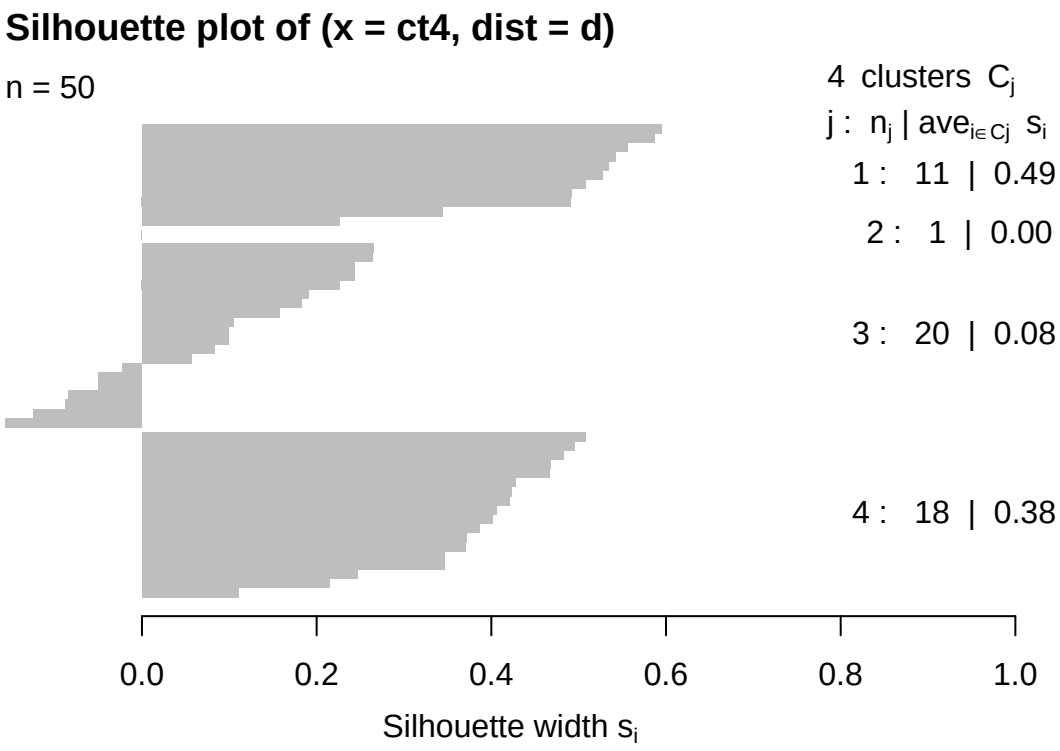
From the elbow plot the optimal cluster can be k = 3, 4, 5. So to further optimize I checked the silhouette plots of the cut trees.

```
ct3 <- cutree(hc, k = 3)
si3 <- silhouette(ct3, dist = d)
plot(si3)
```



Average silhouette width : 0.36

```
ct4 <- cutree(hc, k = 4)
si4 <- silhouette(ct4, dist = d)
plot(si4)
```



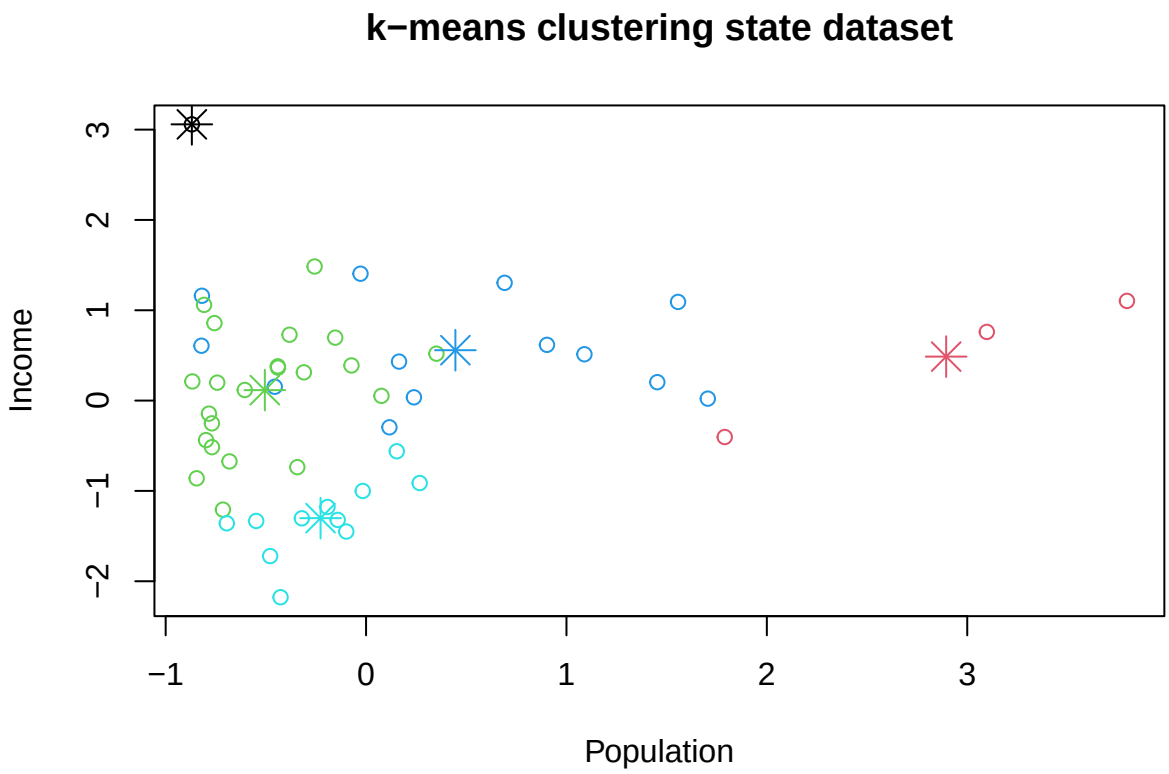
Average silhouette width : 0.28

```
ct5 <- cutree(hc, k = 5)
si5 <- silhouette(ct5, dist = d)
plot(si5)
```

From the silhouette plots we could observe that K = 5 has better clustering Because it has uniform distribution of datapoints between the clusters and has uniform pattern of clusters and also has less negative values.

```
#Run k-means on the scaled dataset
set.seed(123)
km3 <- kmeans(dats_scale, centers = 5, nstart = 10)

# plot the groups
#x11()
plot(dats_scale, col = km3$cluster, main = "k-means clustering state dataset" )
points(km3$centers, col = 1:5, pch = 8, cex= 2)
```



c) In your opinion, does k-means or hierarchical performs better. Justify your answer.

Hierarchical clustering may be more appropriate if the goal is to identify hierarchical relationships among the states

While k-means clustering may be more appropriate if the goal is to identify distinct groups or cluster of states based on their socioeconomic characteristics.

From the dendrogram we can clearly observe the similarities between different states and cluster formation hierarchy but it's hard to find the optimal number of clusters as there are so many clusters in the dendrogram and it becomes difficult to cluster the states and in K-means it is very easy to identify clusters plus it is computationally inexpensive. but in sillhoutte plots we can see that k-means getting effected with outliers. we can just avoid this my removing outliers while preprocessing the data. or choosing the right K value will help in clustering the outlier point in a seperate cluster.

So in total if our goal is to cluster the states on socioeconomic characteristics I feel k-means would be the better option to cluster and it is computationally inexpensive as well.

Question 3:

Consider the Iris data (>data(iris)). Read the data

```
data(iris)
dataset <- iris
head(dataset)
```

```
##   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
```

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

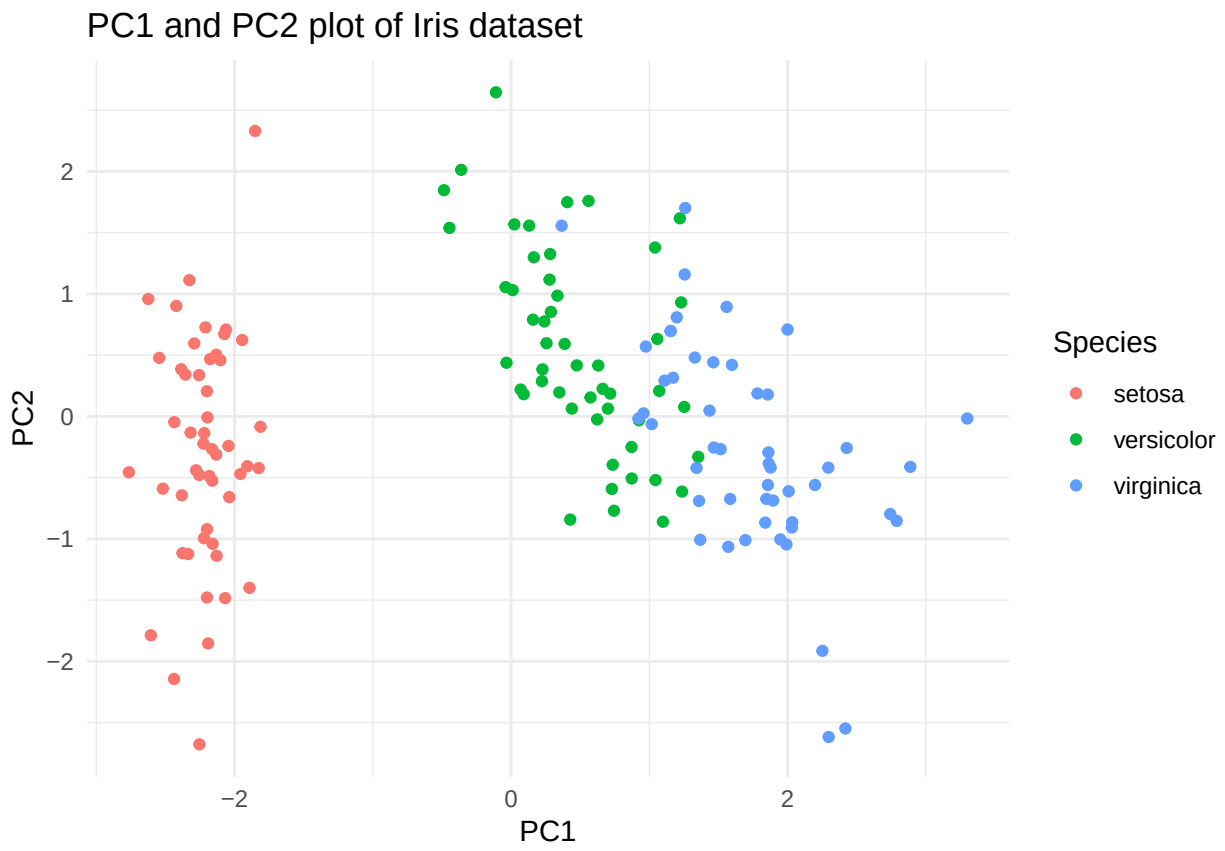
```
fit <- prcomp(dataset[,1:4], center=TRUE, scale=TRUE)
```

Extracting and combining first two principal components, PC1 and PC2 and combining with species columns.

```
PC1 <- fit$x[,1]
PC2 <- fit$x[,2]
PC_dats <- cbind(PC1, PC2)
PC_dats_df <- as.data.frame(PC_dats)
PC_dats_df$Species <- dataset$Species
```

Creating a plot with PC1 and PC2 and coloring the iris species by class.

```
#x11()
ggplot(PC_dats_df, aes(x = PC1, y = PC2, color = Species)) +
  geom_point() +
  labs(x = "PC1", y = "PC2", title = "PC1 and PC2 plot of Iris dataset") +
  theme_minimal()
```



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. Run k-means on the PC values

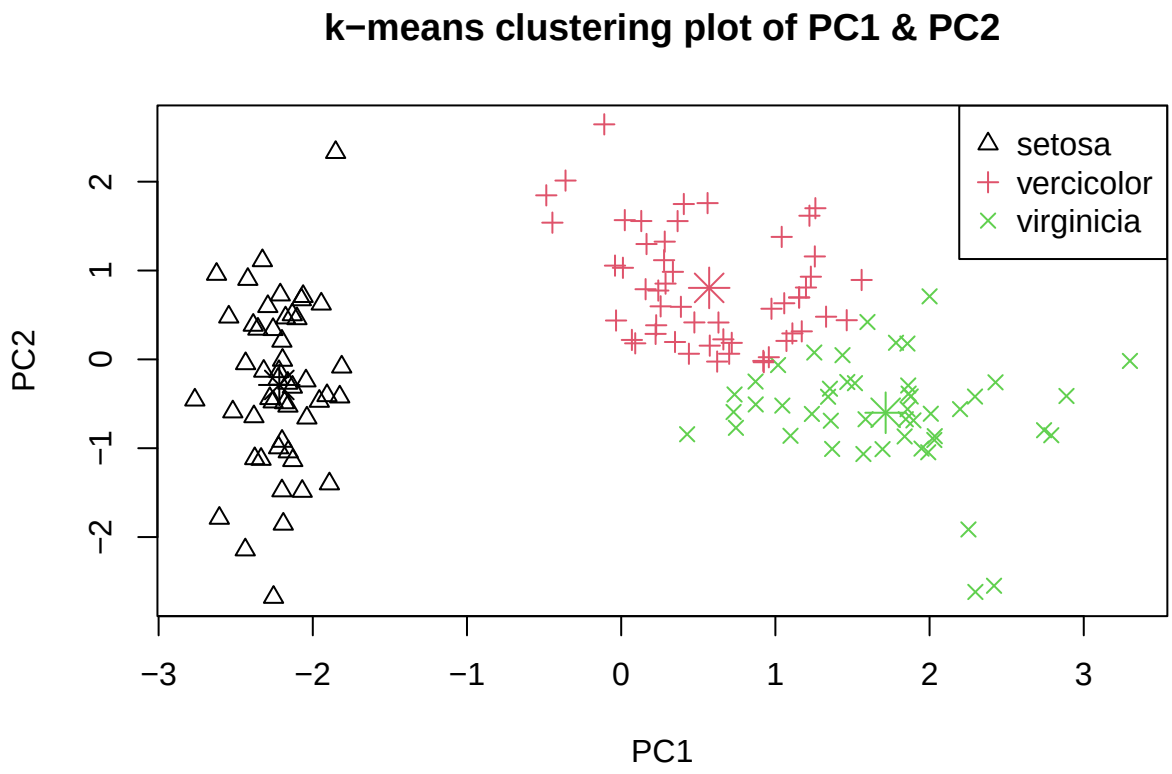
```
set.seed(123)
km2 <- kmeans(PC_dats, centers = 3, nstart = 10)
```

create a vector of symbols corresponding to each cluster

```
#x11()
#use different integers for each cluster
cluster_symbols <- c(2, 3, 4)
```

Plot the groups

```
plot(PC_dats, col = km2$cluster,pch = cluster_symbols[km2$cluster], main = "k-means clustering plot of I
points(km2$centers, col = 1:3, pch = 8, cex= 2)
# add legend for symbols
legend("topright", legend = c("setosa", "vercicolor", "virginicia"), pch = cluster_symbols, col = 1:3)
```




```
rand.index(km2$cluster, as.numeric(iris$Species))
```

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

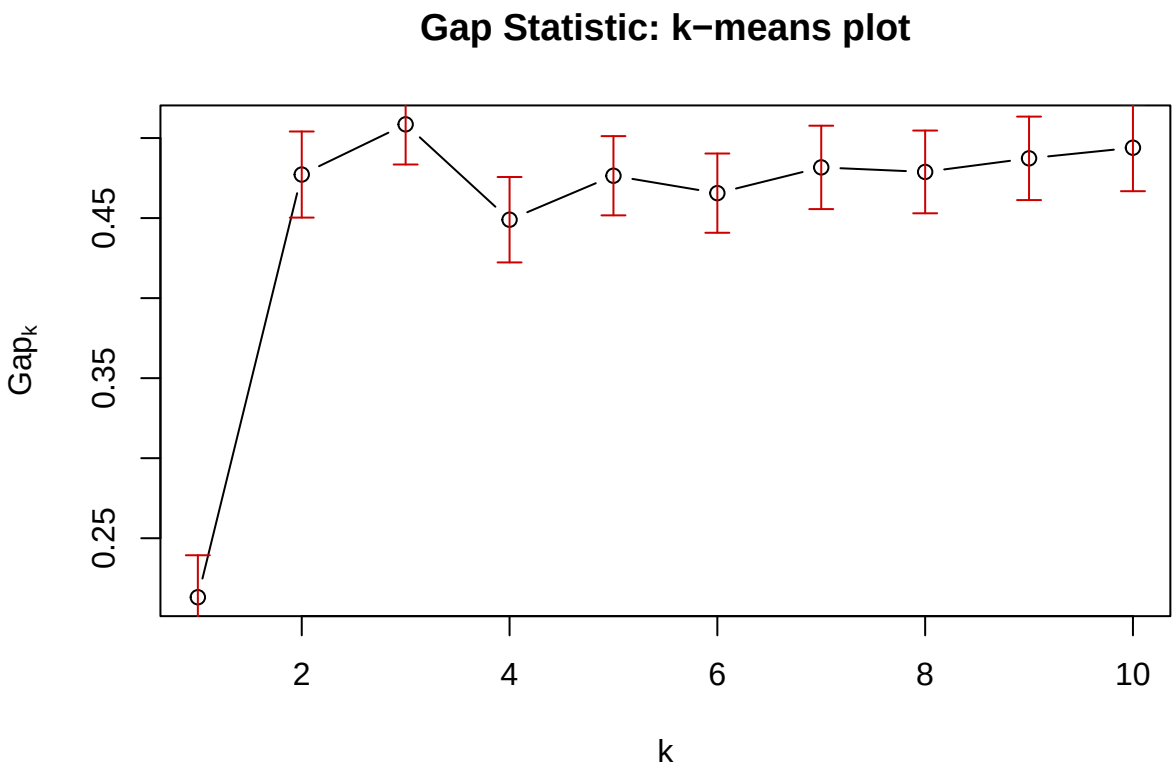
```
## [1] 0.8322148
```

```
adj.rand.index(km2$cluster, as.numeric(iris$Species))
```

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

d) Use the gap statistic and silhouette plots to determine the number of clusters. Gap statistics - K-means clustering

```
gap_kmeans <- clusGap(PC_dats, kmeans, nstart = 20, K.max = 10, B = 100)
#x11()
plot(gap_kmeans, main = "Gap Statistic: k-means plot")
```



Silhouette plot - K-means clustering.

```
sil_width <- silhouette(km2$cluster, dist(PC_dats))
#x11()
plot(sil_width, main = "Silhouette : k-means plot")
```

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

1.From the rand index and adjusted rand index, we can infer that the actual species labels (iris-Species) are only slightly different from the k-means clustering solution labels (km2-cluster).

2.Approximately 83.22% of data point pairs have the same cluster assignments in both the predicted and actual clusters, according to the Rand Index, which stands at 0.8322148.

3.The agreement between the predicted and actual clusters is better than would be predicted by chance, according to the Adjusted Rand Index of 0.6201352, but there is still space for growth.

4.From the "Gap Statistic: k-means plot" we could see that gap static reaches it's maximum at 3. So the optimal number of clusters for iris data set is 3.

5.From the "Silhouette : k-means plot" we can interpret that the cluster quality is good. For each cluster the bar chart is pretty balanced and similar and there no negative values, implying good quality clustering.

There is also nearly uniform no.of data points between the 3 clusters like 50, 53, 47 this implies that the clustering and structure is good for k=3.