Homework 4

Naga Kartheek Peddisetty, 50538422

04/29/2024

Question 1) Consider the “UKFaculty friends” network, which is available in the package “igraphdata”. library(igraphdata), data(UKfaculty) ?UKfaculty Using the hierarchical random graphs functions in “igraph” perform the following tasks:

```
library(igraph)
```

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

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
library(igraphdata)
```

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

```
data("UKfaculty")
```

a) Focus on the “UKFaculty friends” network. Transform this network into an undirected graph using the “igraph” package.

```
ukfaculty <- as.undirected(UKfaculty)
```

```
## This graph was created by an old(er) igraph version.
##   Call upgrade_graph() on it to use with the current igraph version
##   For now we convert it on the fly...
```

b) Create noisy datasets. Do this by deleting 8% of the edges randomly (track which ones they are). Perform MCMC for a random graph model (as in Clauset et al.) on this data followed by link-prediction. Are you able to predict the edges that you deleted?

```
set.seed(123)

cat("Number of vertices =", vcount(ukfaculty))
```

```
## Number of vertices = 81
```

```
V(ukfaculty)  # Names of the vertices
```

```
## + 81/81 vertices, from 14b75d7:
##  [1]  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] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
## [51] 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
## [76] 76 77 78 79 80 81
```

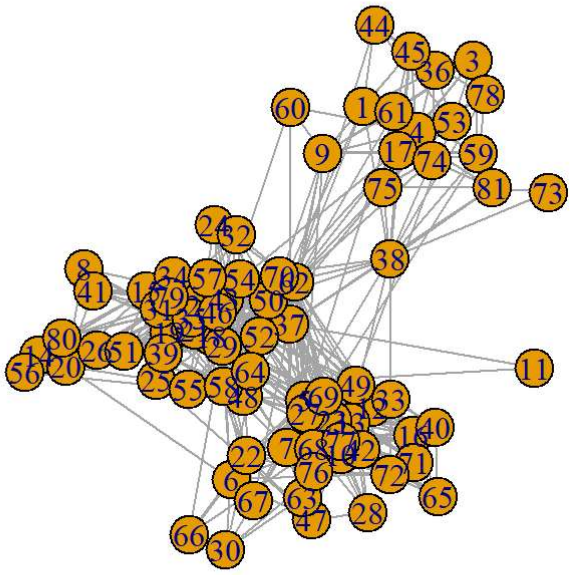
```
# count the number of edges in the graph
edge_count <- ecount(ukfaculty)
cat("Number of edges =", edge_count)
```

```
## Number of edges = 577
```

```
# Get the names of edges
E(ukfaculty)
```

```
## + 577/577 edges from 14b75d7:
##   [1]  1-- 4  3-- 4  5-- 6  5-- 7  6-- 7  3-- 9  4-- 9  5-- 9  5--10  7--10
##  [11]  5--12  7--12  5--13  7--13 10--13 12--13  2--15 14--15  5--16  7--16
##  [21] 12--16 13--16  3--17  4--17  9--17  2--18  7--18 15--18 15--19 18--19
##  [31]  2--20  6--20  8--20 14--20 15--20 18--20 19--20  2--21  8--21 10--21
##  [41] 13--21 15--21 19--21  5--22  7--22 10--22 12--22 21--22  5--23  7--23
##  [51] 10--23 12--23 13--23 22--23  2--25 15--25 20--25 21--25  2--26 14--26
##  [61] 20--26 21--26  5--27  7--27 10--27 13--27 16--27 19--27 22--27 23--27
##  [71]  5--28 10--28 13--28  2--29  4--29  6--29  7--29  8--29 14--29 15--29
##  [81] 18--29 19--29 20--29 21--29 22--29 23--29 24--29 26--29 27--29  6--30
##  [91]  7--30 10--30 22--30  2--31  8--31 15--31 19--31 20--31 21--31 25--31
## + ... omitted several edges
```

```
plot(ukfaculty)
```



```
# Get the number of edges to delete
delete_count <- as.integer(edge_count * 0.08)
delete_count
```

```
## [1] 46
```

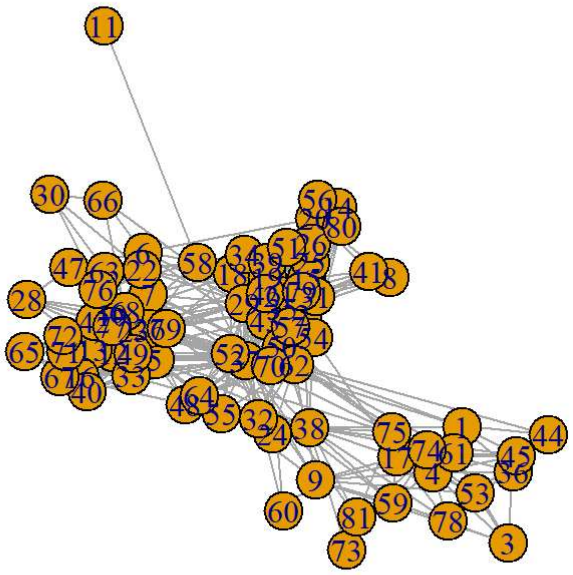
```
# Edges to delete
set.seed(123)
edges_to_delete <- sample(1:edge_count, delete_count, replace = FALSE)
edges_to_delete
```

```
##  [1] 415 463 179 526 195 118 299 229 244  14 374  91 348 355  26 519 426 211 555
## [20] 373 143 544 490  23 309 135 224 166 217 290  72 141 153 294 277 576  41 431
## [39]  90 316 223 528 116 456  39 159
```

```
# Delete the edges here
ukfaculty_noisy <- delete_edges(ukfaculty, edges_to_delete)
ukfaculty_noisy
```

```
## IGRAPH 14d6db4 U-W- 81 531 --
## + attr: Type (g/c), Date (g/c), Citation (g/c), Author (g/c), Group
## | (v/n), weight (e/n)
## + edges from 14d6db4:
##  [1]  1-- 4  3-- 4  5-- 6  5-- 7  6-- 7  3-- 9  4-- 9  5-- 9  5--10  7--10
## [11]  5--12  7--12  5--13 10--13 12--13  2--15 14--15  5--16  7--16 12--16
## [21] 13--16  4--17  9--17  7--18 15--18 15--19 18--19  2--20  6--20  8--20
## [31] 14--20 15--20 18--20 19--20  2--21 10--21 15--21 19--21  5--22  7--22
## [41] 10--22 12--22 21--22  5--23  7--23 10--23 12--23 13--23 22--23  2--25
## [51] 15--25 20--25 21--25  2--26 14--26 20--26 21--26  5--27  7--27 10--27
## [61] 13--27 16--27 19--27 22--27 23--27  5--28 13--28  2--29  4--29  6--29
## + ... omitted several edges
```

```
plot(ukfaculty_noisy)
```



```
# count the number of edges after deletion
# 577 - 46 = 531
ecount(ukfaculty_noisy)
```

```
## [1] 531
```

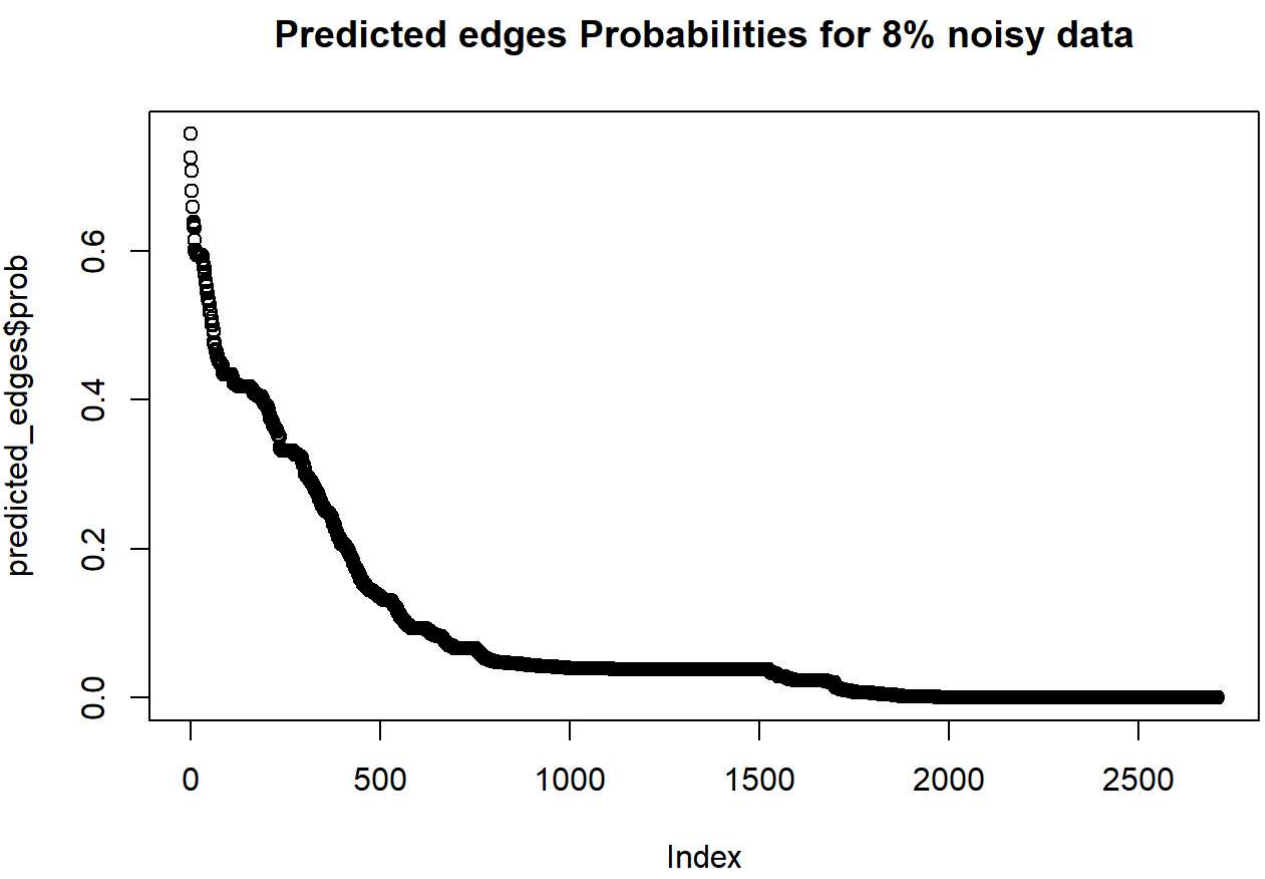
```
# Fit a random graph model to the noisy dataset
# Also mention the number MCMC (Markov Chain Monte Carlo) steps
ukfaculty_model <- fit_hrg(ukfaculty_noisy, steps = 100)
ukfaculty_model
```

```
## Hierarchical random graph, at level 3:
## g1        p= 0.16
## '- g2     p=0.065  71
##    '- g4  p=  0.2  1  67 17 56 21 59 68 16 43 37 61 54 10 25 23 48 8  45 44 51
##                    58 27 80 41 39 79 74 55 38 57 50
## '- g3     p=  0.1  63
##    '- g8  p= 0.15  78 18 2  70 31 30 81 75 19 32 4  53 36 9  64 62 77 72 66 11
##                    52 49 3  13 22 20 60 47 35 40 7  24 12 26 15 46 33 34 14 5
##                    65 6  73 42 28 69 29 76
```

```
# Predict the missing edges
predicted_edges <- predict_edges(ukfaculty_noisy)

# Get the predicted edges only
head(predicted_edges$edges)
```

```
##      [,1] [,2]
## [1,]    5   33
## [2,]   16   77
## [3,]   12   42
## [4,]   42   77
## [5,]   49   69
## [6,]    5   71
```

```
plot(predicted_edges$prob, main = "Predicted edges Probabilities for 8% noisy data")
```




```
# number of predicted edges
predicted_edges_count <- dim(predicted_edges$edges)[1]
```

The HRG model attempts to replicate the structure of the original graph by making a list of edges with their associated probabilities. This means that the number of predicted edges can be higher than the number of real deleted edges.

Due to imperfections in the model, false positive edges can be made, which makes the total number of predicted edges go up. It is important to check how accurate the model is by comparing its predictions to the actual deleted edges.

```
# Obtain true deleted edges as edge sequence
true_deleted_edges_seq <- E(ukfaculty)[edges_to_delete]
true_deleted_edges_seq
```

```
## + 46/577 edges from 14b75d7:
##  [1] 22--66 29--70 29--41 37--77 13--43 31--34 34--54  7--47 37--49  7--13
## [11] 18--62  7--30 50--58  4--60  2--18 16--77 10--68  9--45 39--79 17--62
## [21] 27--37 45--78 69--72  3--17 52--54 10--37 34--46  7--40 11--46 49--53
## [31] 10--28 21--37 10--38 19--54 33--52 74--81 13--21 27--68  6--30 14--56
## [41] 31--46 42--77 25--34 49--69  8--21 18--39
```

```r
# Convert edge sequence to matrix with numeric vertices
true_deleted_edges <- t(sapply(true_deleted_edges_seq, function(edge) ends(ukfaculty, edge)))
true_deleted_edges
```

```
##       [,1] [,2]
##  [1,]   22   66
##  [2,]   29   70
##  [3,]   29   41
##  [4,]   37   77
##  [5,]   13   43
##  [6,]   31   34
##  [7,]   34   54
##  [8,]    7   47
##  [9,]   37   49
## [10,]    7   13
## [11,]   18   62
## [12,]    7   30
## [13,]   50   58
## [14,]    4   60
## [15,]    2   18
## [16,]   16   77
## [17,]   10   68
## [18,]    9   45
## [19,]   39   79
## [20,]   17   62
## [21,]   27   37
## [22,]   45   78
## [23,]   69   72
## [24,]    3   17
## [25,]   52   54
## [26,]   10   37
## [27,]   34   46
## [28,]    7   40
## [29,]   11   46
## [30,]   49   53
## [31,]   10   28
## [32,]   21   37
## [33,]   10   38
## [34,]   19   54
## [35,]   33   52
## [36,]   74   81
## [37,]   13   21
## [38,]   27   68
## [39,]    6   30
## [40,]   14   56
## [41,]   31   46
## [42,]   42   77
## [43,]   25   34
## [44,]   49   69
## [45,]    8   21
## [46,]   18   39
```

```
# Link prediction (calculate all correct predictions)
correct_predictions <- sum(sapply(1:predicted_edges_count, function(i) {
  any(apply(true_deleted_edges, 1, function(x) all(x == predicted_edges$edges[i,])))
}))

cat("Count of correct predictions for 8% noisy data =",correct_predictions)
```

```
## Count of correct predictions for 8% noisy data = 46
```

```
# Calculate the accuracy
cat("Link prediction accuracy =", correct_predictions / delete_count * 100,"%")
```

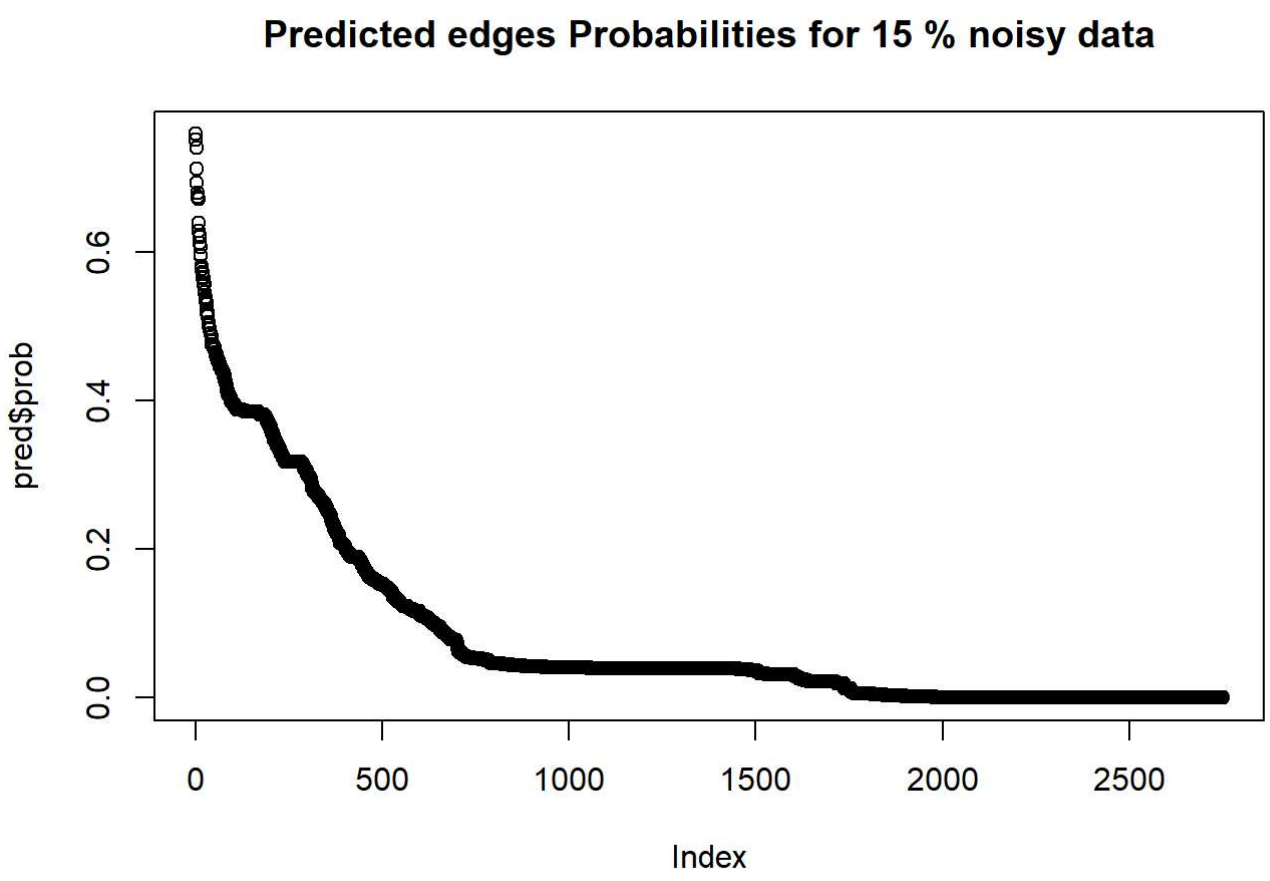
```
## Link prediction accuracy = 100 %
```

Yes,We are able to predict all the edges that were deleted with 100% accuracy.

c) Repeat the exercise in part (a) and (b) after deleting 15%, and 40% of the edges. Comment on your findings.

```
# Writing function

link_prediction_accuracy <- function(dataset, percent_to_delete) {

      # Load the data and convert to undirected graph
      g <- as.undirected(dataset)

      # Delete edges randomly
      set.seed(123)
      edge_count <- ecount(g)
      delete_count <- round(edge_count * (percent_to_delete / 100))
      edges_to_delete <- sample(1:edge_count, delete_count, replace = FALSE)
      g_noisy <- delete_edges(g, edges_to_delete)

      # Fit HRG model using MCMC
      model <- fit_hrg(g_noisy,steps = 100)

      # Perform link prediction
      pred <- predict_edges(g_noisy)

      plot(pred$prob, main = paste("Predicted edges Probabilities for", percent_to_delete,"% noi
sy data"))

      # Get true deleted edges
      true_deleted_edges_seq <- E(g)[edges_to_delete]
      true_deleted_edges <- t(sapply(true_deleted_edges_seq, function(edge) ends(g, edge)))

      # get the predicted edges and the count
      predicted_edges <- pred$edges
      pred_edge_count <- nrow(predicted_edges)

      # Link prediction (calculate all correct predictions)
correct_predictions <- sum(sapply(1:pred_edge_count, function(i) {
  any(apply(true_deleted_edges, 1, function(x) all(x == predicted_edges[i,])))
}))
      cat("Count of correct predictions for the data =", correct_predictions)
      # Calculate prediction accuracy
      accuracy <-(correct_predictions / delete_count)*100

      # Return a list with the accuracy, true deleted edges, and predicted edges
      result <- list("accuracy" = accuracy, "true_deleted_edges" = true_deleted_edges)
      return(result)
}
```

```
# deleting 15% of nodes

result <- link_prediction_accuracy(UKfaculty, 15)
```




```
## Count of correct predictions for the data = 87
```

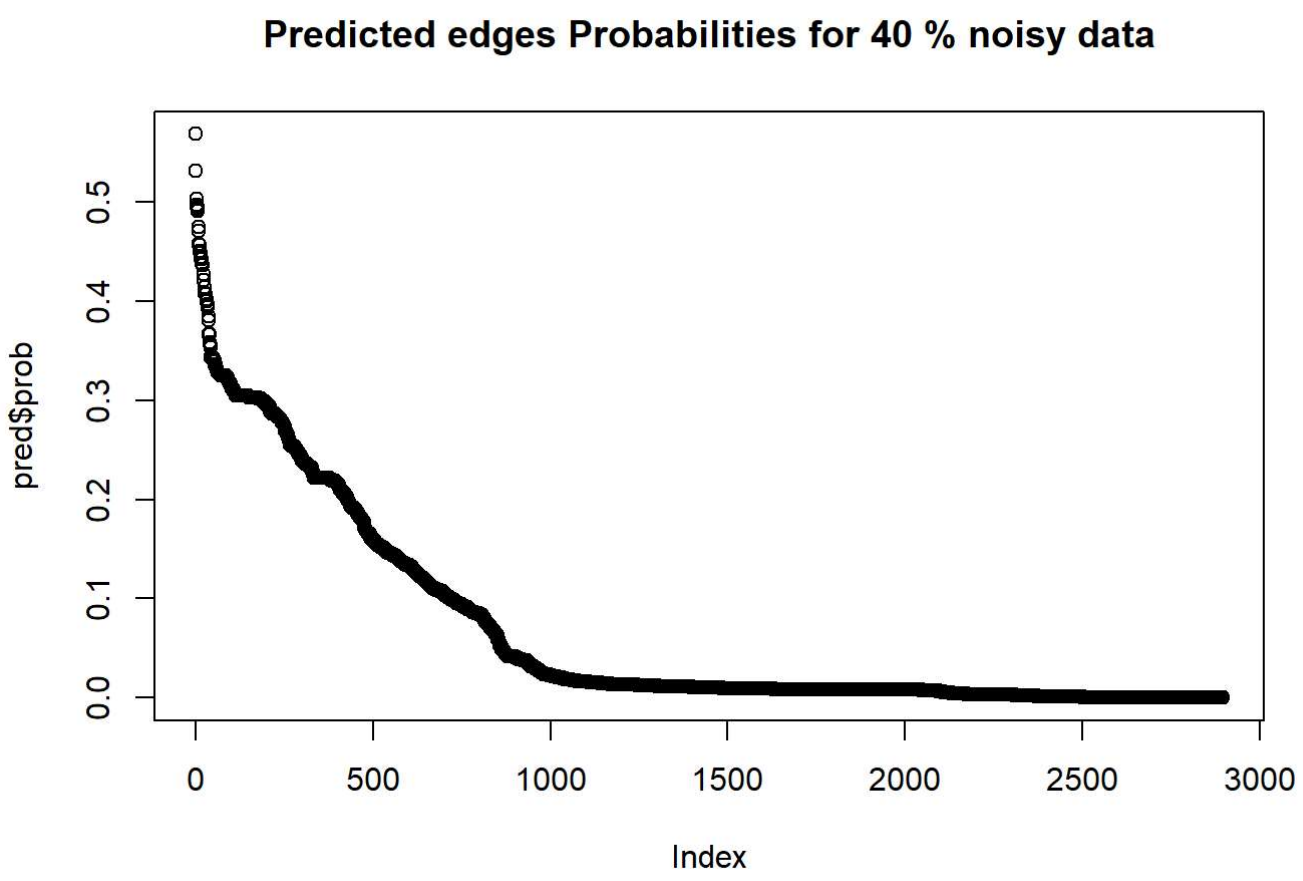
```
# count of 15% Edges that are deleted
true_deleted_edges <- result$true_deleted_edges
dim(true_deleted_edges)[1]
```

```
## [1] 87
```

```
link_pred_acc <- result$accuracy

cat("Prediction Accuracy:", link_pred_acc,"%","\n")
```

```
## Prediction Accuracy: 100 %
```

```
# deleting 40% of nodes

result1 <- link_prediction_accuracy(UKfaculty, 40)
```




```
## Count of correct predictions for the data = 231
```

```
# count of 40% Edges that are deleted
true_deleted_edges1 <- result1$true_deleted_edges
dim(true_deleted_edges1)[1]
```

```
## [1] 231
```

```
link_pred_acc1 <- result1$accuracy

cat("Prediction Accuracy:", link_pred_acc1,"%","\n")
```

```
## Prediction Accuracy: 100 %
```

We can observe that on deleting `15%` of edges and `40%` of edges from the UKfaculty dataset resulted in `100%` accurate predictions of all deleted edges. Specifically, the hierarchical random graph (HRG) model was used in combination with the Markov Chain Monte Carlo (MCMC) method to achieve this perfect prediction accuracy.

Question 2) Access the SwissBankNotes data (posted with assignment). The data consists of six variables measured on 200 old Swiss 1,000-franc bank notes. The first 100 are genuine and the second 100 are counterfeit. The six variables are length of the bank note, height of the bank note, measured on the left, height of the bank note

measured on the right, distance of the inner frame to the lower border, distance of inner frame to upper border, and length of the diagonal. Carry out a PCA of the 100 genuine bank notes, of the 100 counterfeit bank notes, and all of the 200 bank notes combined. Generate some biplots (use colors for the combined PCA). Do you notice any differences in the results? Show your work, and justify the selection of Principal Components, including diagnostic plots.

```
library(factoextra)
```

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

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

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

```
load("SwissBankNotes.RData")
sbn <- SwissBankNotes
dim(sbn)
```

```
## [1] 200   6
```

```
head(sbn)
```

```
##   length height.left height.right inner.lower inner.upper diagonal
## 1  214.8       131.0        131.1         9.0         9.7    141.0
## 2  214.6       129.7        129.7         8.1         9.5    141.7
## 3  214.8       129.7        129.7         8.7         9.6    142.2
## 4  214.8       129.7        129.6         7.5        10.4    142.0
## 5  215.0       129.6        129.7        10.4         7.7    141.8
## 6  215.7       130.8        130.5         9.0        10.1    141.4
```

```
str(sbn)
```

```
## 'data.frame':    200 obs. of  6 variables:
##  $ length      : num  215 215 215 215 215 ...
##  $ height.left : num  131 130 130 130 130 ...
##  $ height.right: num  131 130 130 130 130 ...
##  $ inner.lower : num  9 8.1 8.7 7.5 10.4 9 7.9 7.2 8.2 9.2 ...
##  $ inner.upper : num  9.7 9.5 9.6 10.4 7.7 10.1 9.6 10.7 11 10 ...
##  $ diagonal    : num  141 142 142 142 142 ...
```

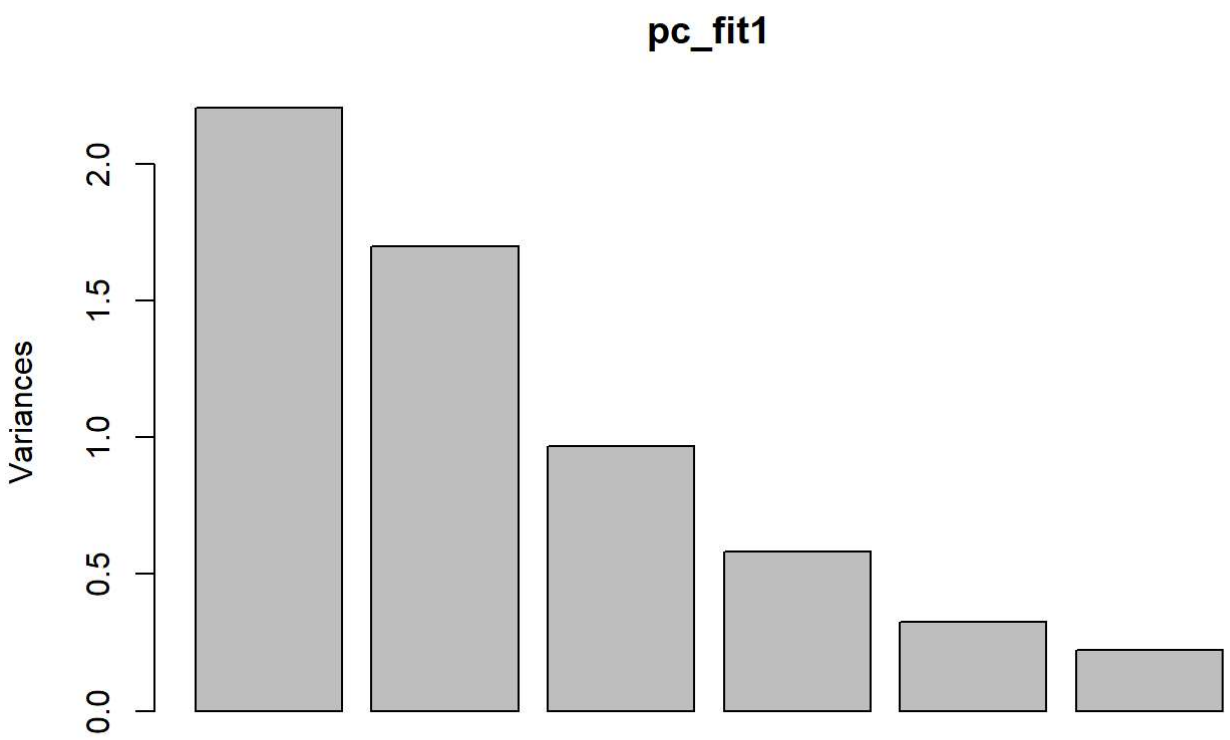
```
summary(sbn)
```

```
##      length       height.left     height.right    inner.lower
##  Min.   :213.8   Min.   :129.0   Min.   :129.0   Min.   : 7.200
##  1st Qu.:214.6   1st Qu.:129.9   1st Qu.:129.7   1st Qu.: 8.200
##  Median :214.9   Median :130.2   Median :130.0   Median : 9.100
##  Mean   :214.9   Mean   :130.1   Mean   :130.0   Mean   : 9.418
##  3rd Qu.:215.1   3rd Qu.:130.4   3rd Qu.:130.2   3rd Qu.:10.600
##  Max.   :216.3   Max.   :131.0   Max.   :131.1   Max.   :12.700
##   inner.upper       diagonal
##  Min.   : 7.70   Min.   :137.8
##  1st Qu.:10.10   1st Qu.:139.5
##  Median :10.60   Median :140.4
##  Mean   :10.65   Mean   :140.5
##  3rd Qu.:11.20   3rd Qu.:141.5
##  Max.   :12.30   Max.   :142.4
```

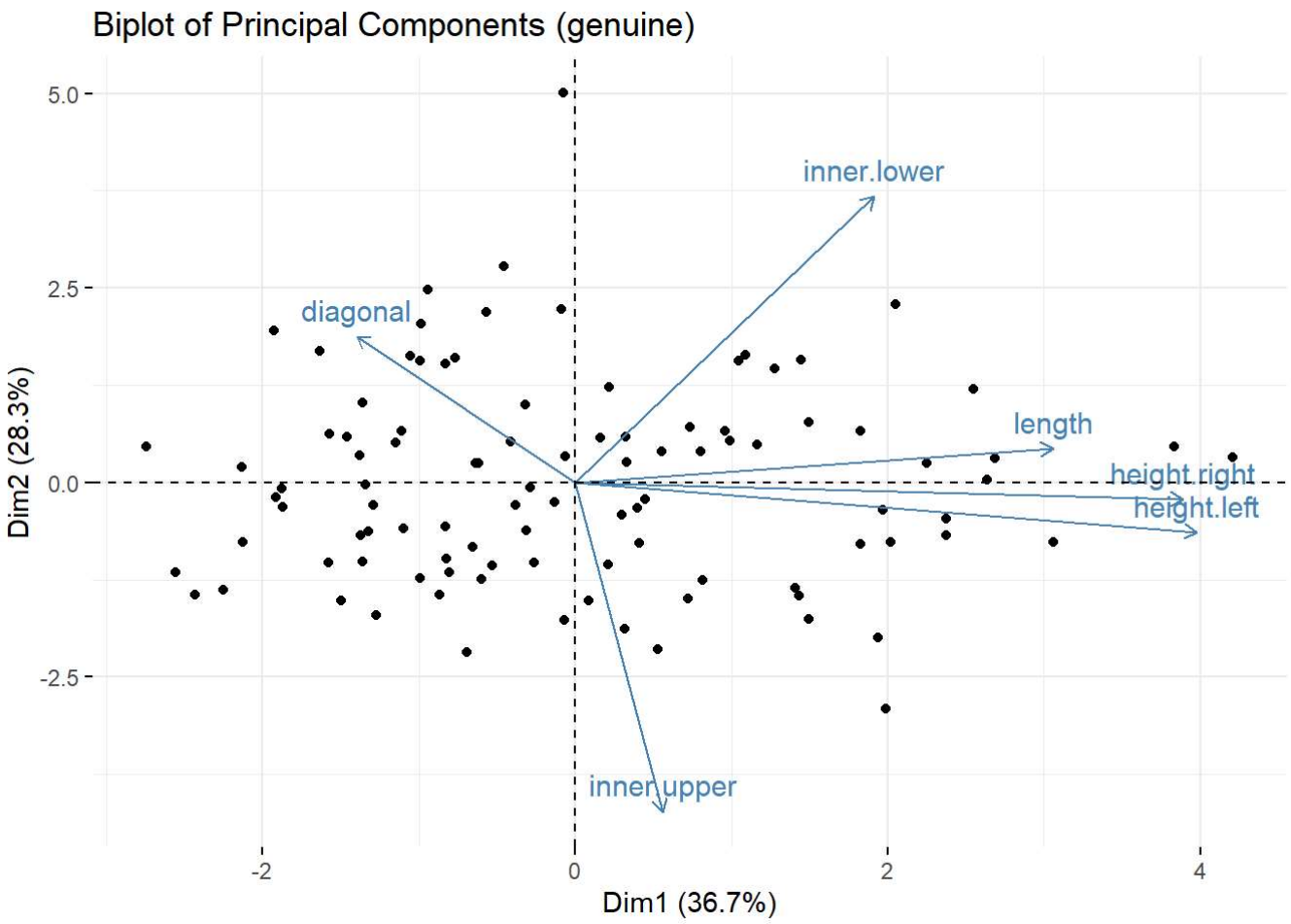
```
sbn1 <- sbn[1:100, ]
sbn2 <- sbn[101:200, ]


# PCA for First 100 genuine
pc_fit1 <- prcomp(sbn1, center = TRUE, scale = TRUE)
pc_fit1
```

```
## Standard deviations (1, .., p=6):
## [1] 1.4845355 1.3025778 0.9827302 0.7634784 0.5715609 0.4733979
##
## Rotation (n x k) = (6 x 6):
##                      PC1         PC2         PC3          PC4         PC5
## length        0.44955352  0.07399926 -0.47874839 -0.741567882  0.06494094
## height.left   0.58504531 -0.10733153  0.04828325  0.286154105 -0.69956514
## height.right  0.57218242 -0.03575517 -0.07727321  0.455925195  0.67581429
## inner.lower   0.28120306  0.61627568  0.27955689 -0.044657130 -0.11600754
## inner.upper   0.08238938 -0.70885999 -0.17047020 -0.005561638 -0.09546841
## diagonal     -0.20583422  0.31535285 -0.80949702  0.397869237 -0.16460367
##                      PC6
## length        0.09531775
## height.left   0.26943479
## height.right  0.02765774
## inner.lower  -0.66897314
## inner.upper  -0.67269924
## diagonal     -0.13230698
```

```
plot(pc_fit1)
```



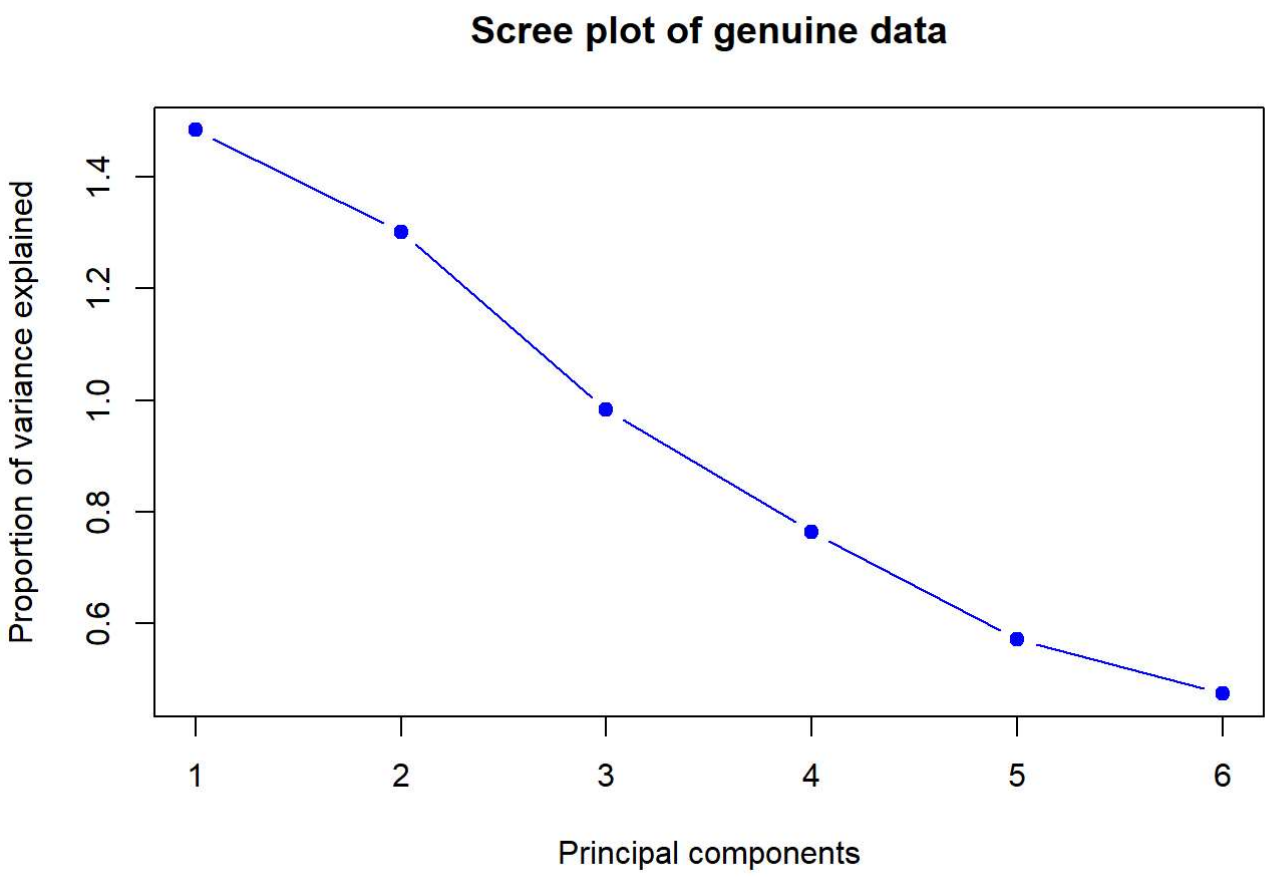

```
fviz_pca_biplot(pc_fit1,label ="var",title = 'Biplot of Principal Components (genuine)')
```




```
pc_fit1$sdev
```

```
## [1] 1.4845355 1.3025778 0.9827302 0.7634784 0.5715609 0.4733979
```

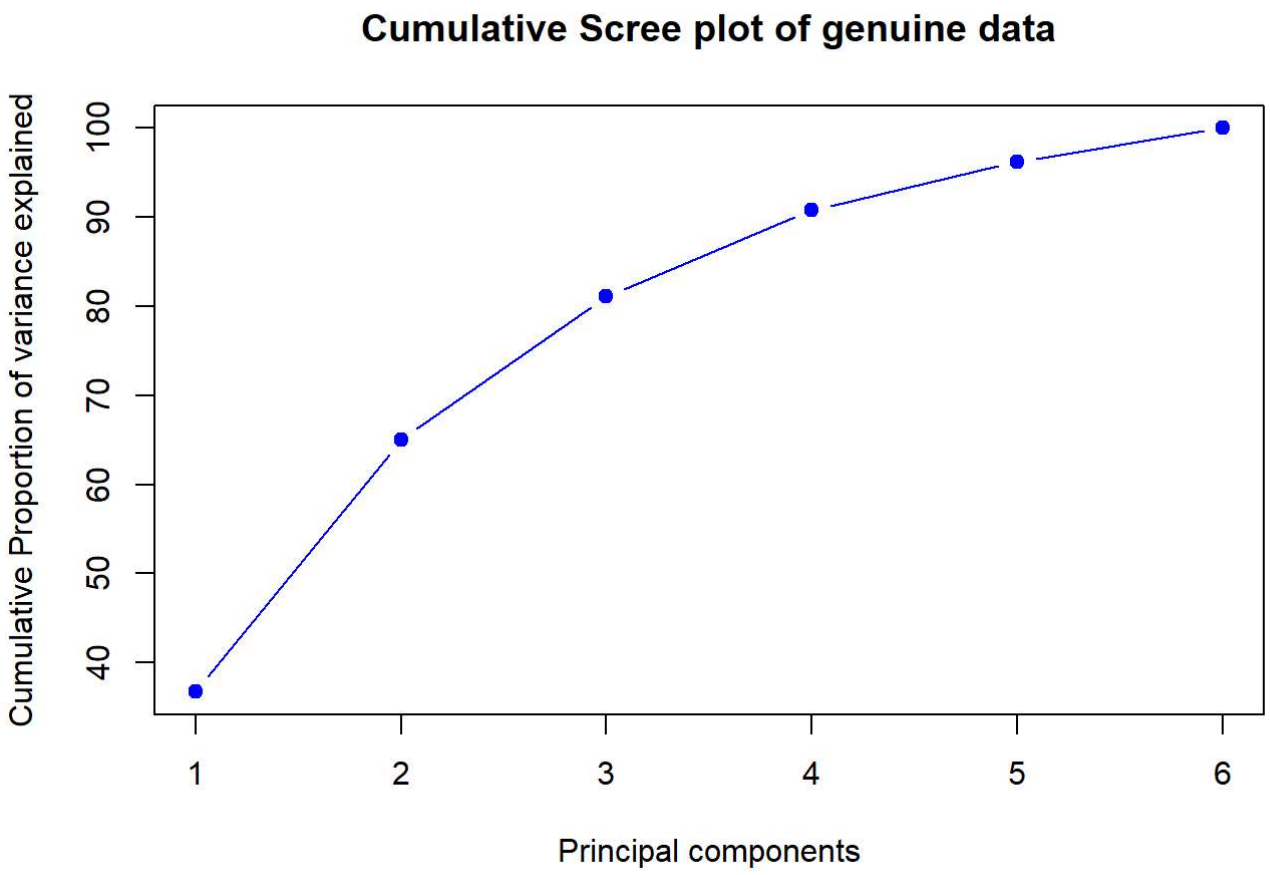
```
# elbow method or scree plot
x_val <- numeric()
y_val <- numeric()

# Iterate over the sequence 1:6 and collect values
for (i in 1:6) {
  x_val <- c(x_val, i)
  y_val <- c(y_val, pc_fit1$sdev[i])
}

# Plot the collected values
plot(x_val, y_val,
     type = "b",
     pch = 19,
     col = "blue",
     xlab = "Principal components",
     ylab = "Proportion of variance explained",
     main = "Scree plot of genuine data")
```




```
per_var_expl1 <- 100 * ((pc_fit1$sdev)^2) / (sum(((pc_fit1$sdev)^2)))
per_var_expl1
```

```
## [1] 36.730761 28.278484 16.095978  9.714988  5.444697  3.735093
```

```
y_v <- numeric()

# Iterate over the sequence 1:6 and collect values
for (i in 1:6) {
  y_v[i] <- sum(per_var_expl1[1:i])
}

# Plot the collected values
plot(1:6, y_v,
     type = "b",
     pch = 19,
     col = "blue",
     xlab = "Principal components",
     ylab = "Cumulative Proportion of variance explained",
     main = "Cumulative Scree plot of genuine data")
```



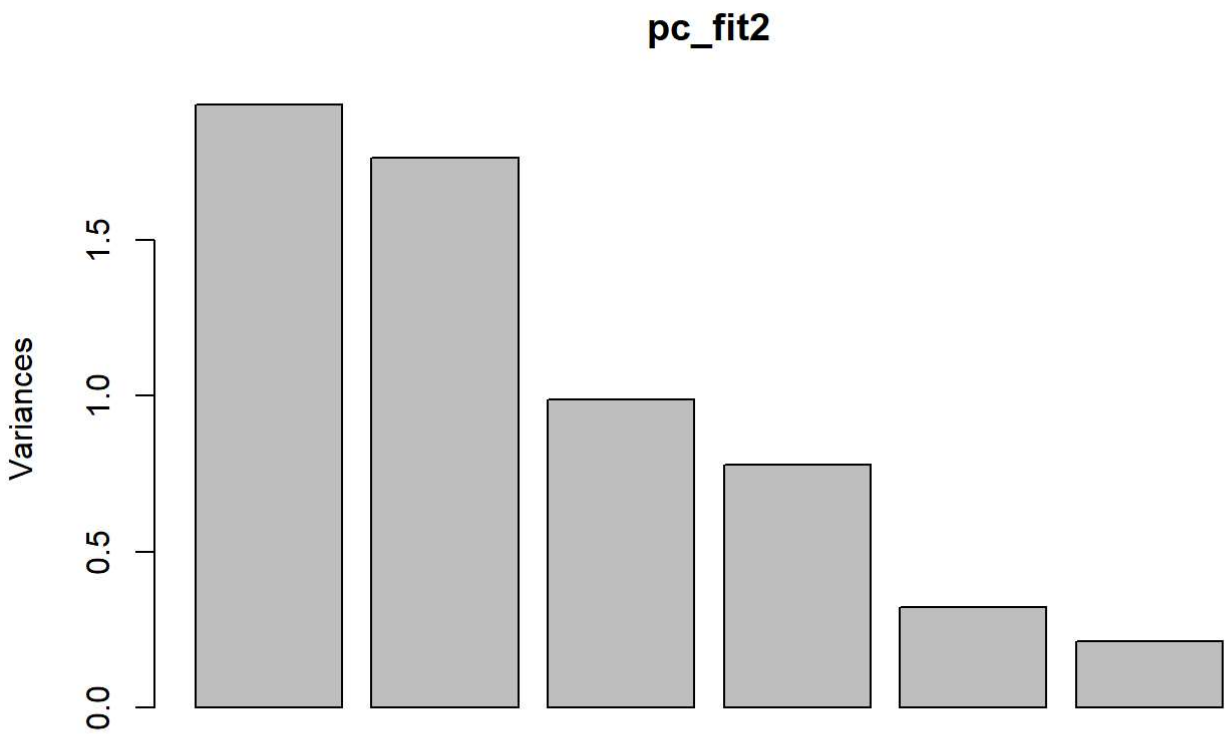

```
cat("We can select first three PCs based on the total variance for the first (100 genuine) to ge
t more than 80%:",per_var_expl1[1] + per_var_expl1[2] + per_var_expl1[3],"%")
```

```
## We can select first three PCs based on the total variance for the first (100 genuine) to get
more than 80%: 81.10522 %
```

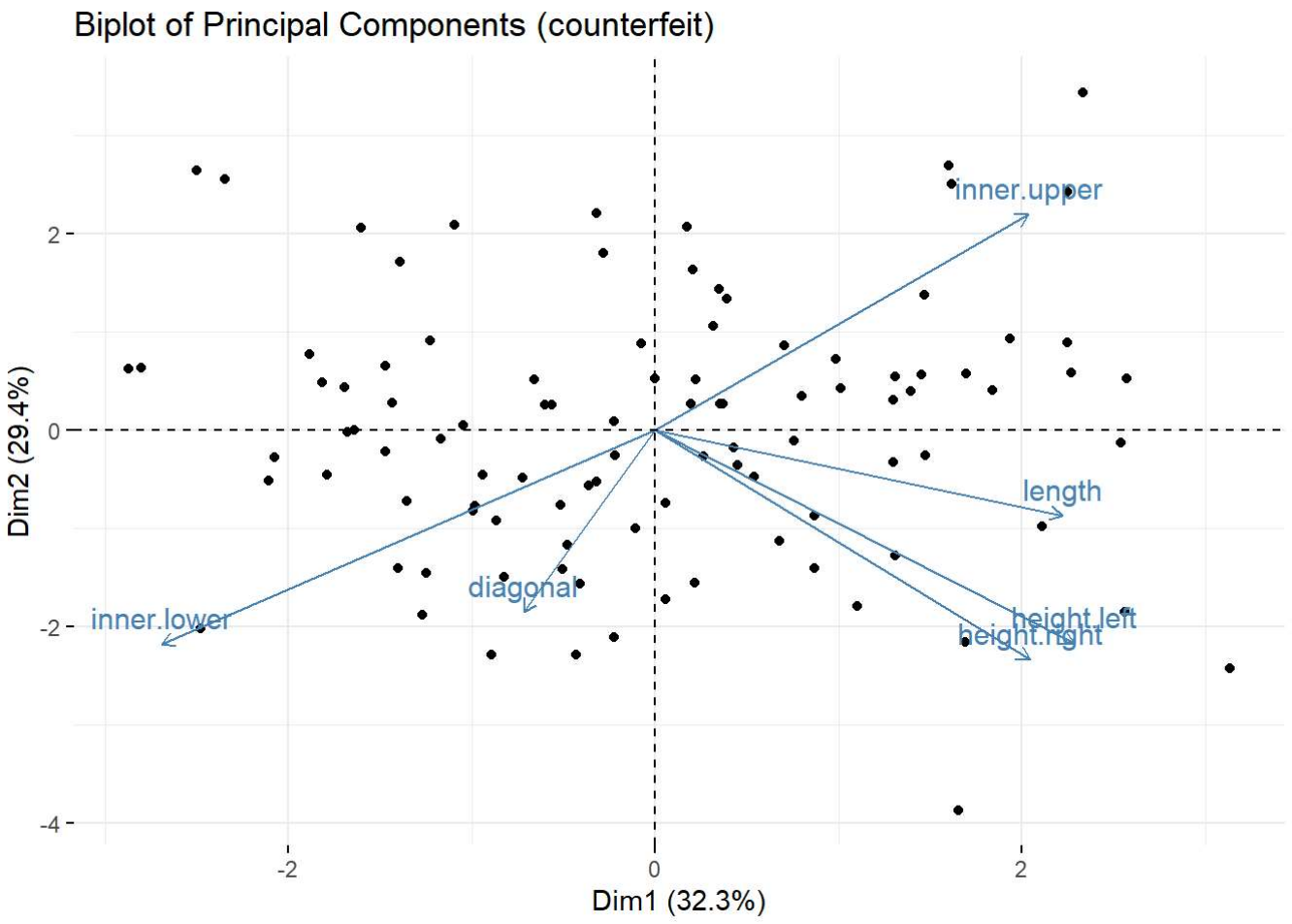
```
# PCA for second 100 counterfeit
pc_fit2 <- prcomp(sbn2, center = TRUE, scale = TRUE)
pc_fit2
```

```
## Standard deviations (1, .., p=6):
## [1] 1.3914793 1.3284814 0.9941399 0.8822512 0.5675470 0.4584009
##
## Rotation (n x k) = (6 x 6):
##                     PC1        PC2         PC3         PC4        PC5
## length        0.4340833 -0.1775842 -0.08198922  0.84763264 -0.1368643
## height.left   0.4456165 -0.4433000  0.31011754 -0.15585668  0.6875964
## height.right  0.3989756 -0.4785602 -0.05857034 -0.44783468 -0.6154210
## inner.lower  -0.5256536 -0.4452589  0.02353654  0.04687108  0.1641716
## inner.upper   0.3972111  0.4491523 -0.46020595 -0.23253889  0.2730199
## diagonal     -0.1404554 -0.3780684 -0.82542771  0.02001165  0.1680597
##                     PC6
## length        0.1900009
## height.left  -0.1079723
## height.right  0.1704968
## inner.lower   0.7040807
## inner.upper   0.5477972
## diagonal     -0.3568769
```

```
plot(pc_fit2)
```



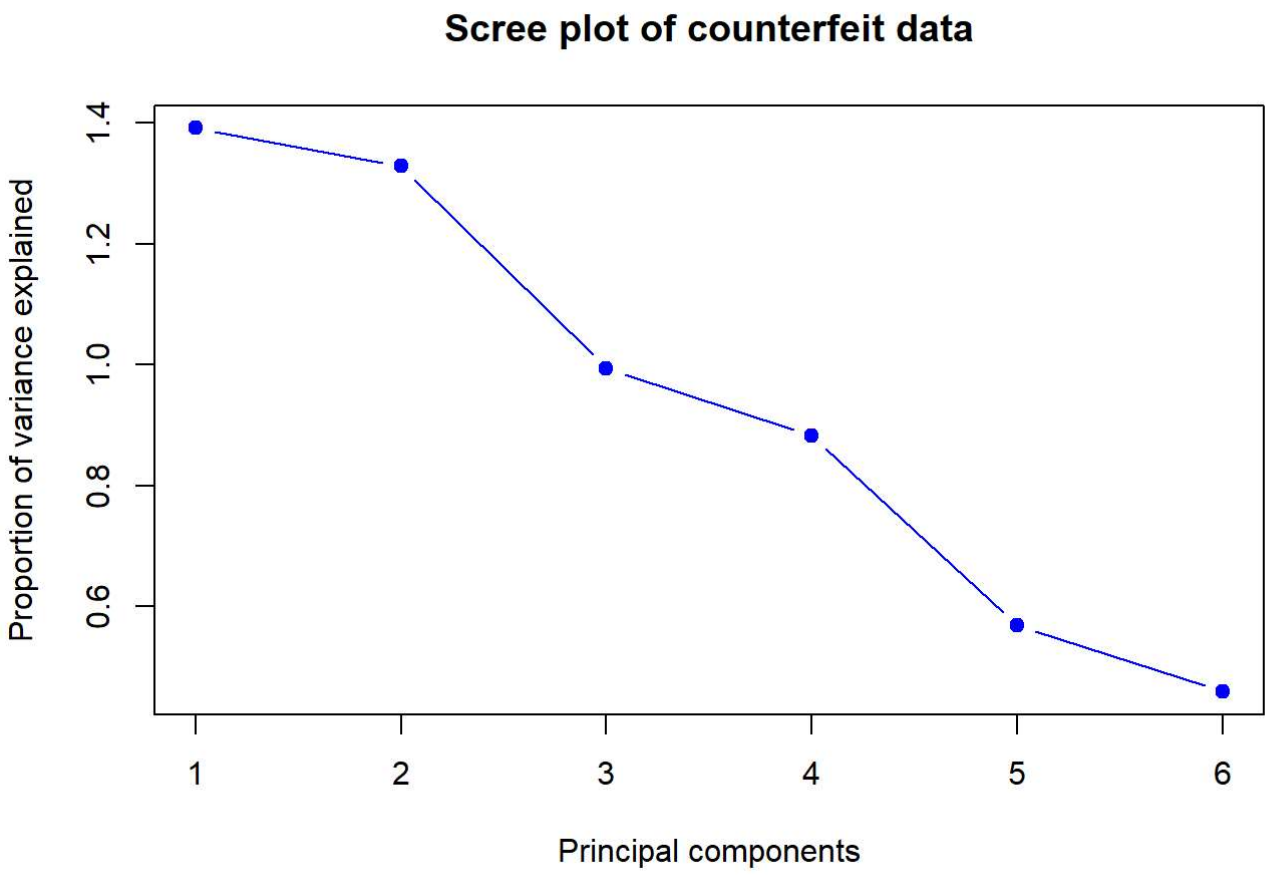

```
fviz_pca_biplot(pc_fit2,label ="var",title = 'Biplot of Principal Components (counterfeit)')
```




```
pc_fit2$sdev
```

```
## [1] 1.3914793 1.3284814 0.9941399 0.8822512 0.5675470 0.4584009
```

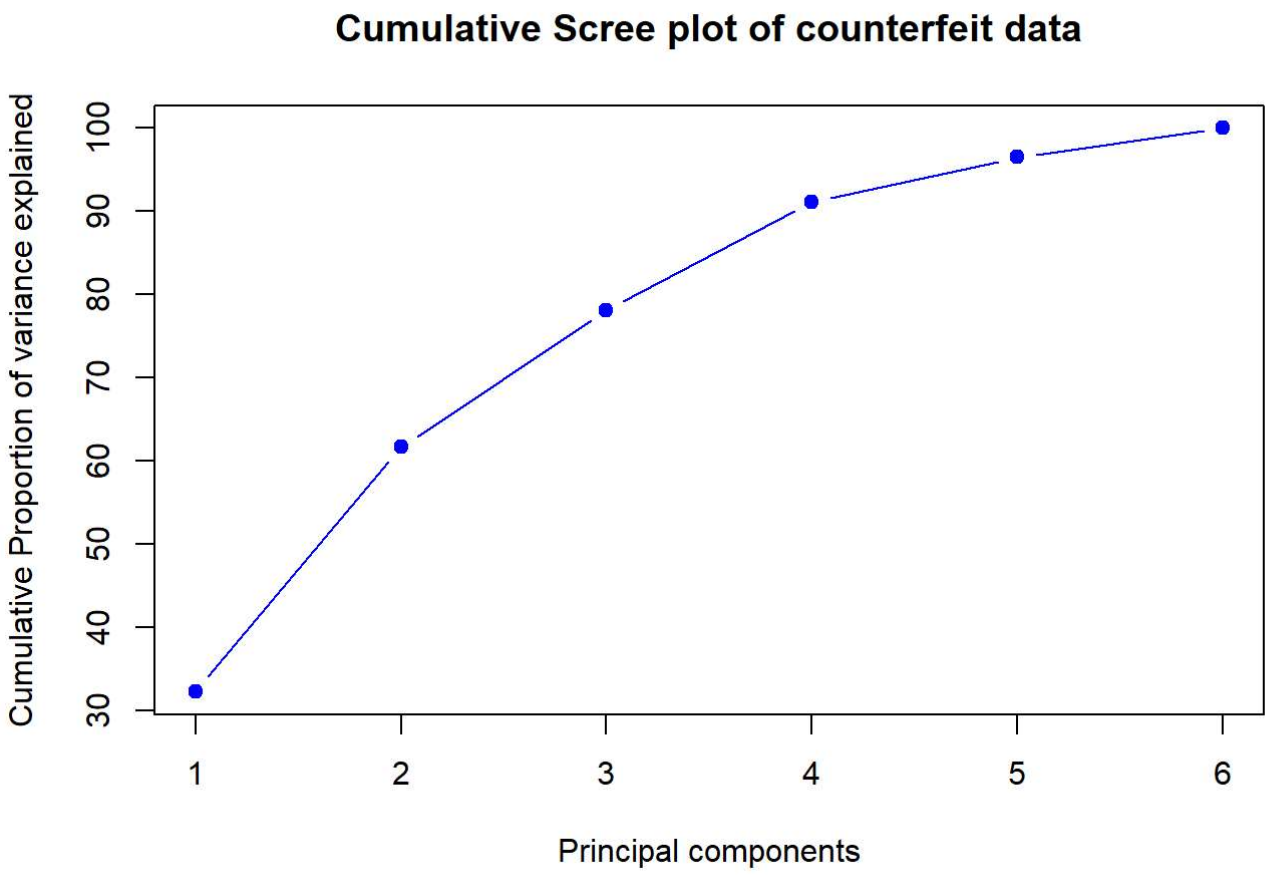
```
# elbow method or scree plot
x_val1 <- numeric()
y_val1 <- numeric()

# Iterate over the sequence 1:6 and collect values
for (i in 1:6) {
  x_val1 <- c(x_val1, i)
  y_val1 <- c(y_val1, pc_fit2$sdev[i])
}

# Plot the collected values
plot(x_val1, y_val1,
     type = "b",
     pch = 19,
     col = "blue",
     xlab = "Principal components",
     ylab = "Proportion of variance explained",
     main = "Scree plot of counterfeit data")
```




```
per_var_expl2 <- 100 * ((pc_fit2$sdev)^2) / (sum(((pc_fit2$sdev)^2)))
per_var_expl2
```

```
## [1] 32.270246 29.414380 16.471903 12.972787  5.368494  3.502190
```

```
y_v1 <- numeric()

# Iterate over the sequence 1:6 and collect values
for (i in 1:6) {
  y_v1[i] <- sum(per_var_expl2[1:i])
}

# Plot the collected values
plot(1:6, y_v1,
     type = "b",
     pch = 19,
     col = "blue",
     xlab = "Principal components",
     ylab = "Cumulative Proportion of variance explained",
     main = "Cumulative Scree plot of counterfeit data")
```




```
cat("We can select first four PCs based on the total variance for the second (100 counterfeit) t
o get more than 80%:",per_var_expl2[1] + per_var_expl2[2] + per_var_expl2[3] + per_var_expl2
[4],"%")
```

```
## We can select first four PCs based on the total variance for the second (100 counterfeit) to
get more than 80%: 91.12932 %
```

```
# PCA for combined data
sbn1$y <- 0
dim(sbn1)
```

```
## [1] 100   7
```

```
sbn2$y <- 1
dim(sbn2)
```

```
## [1] 100   7
```

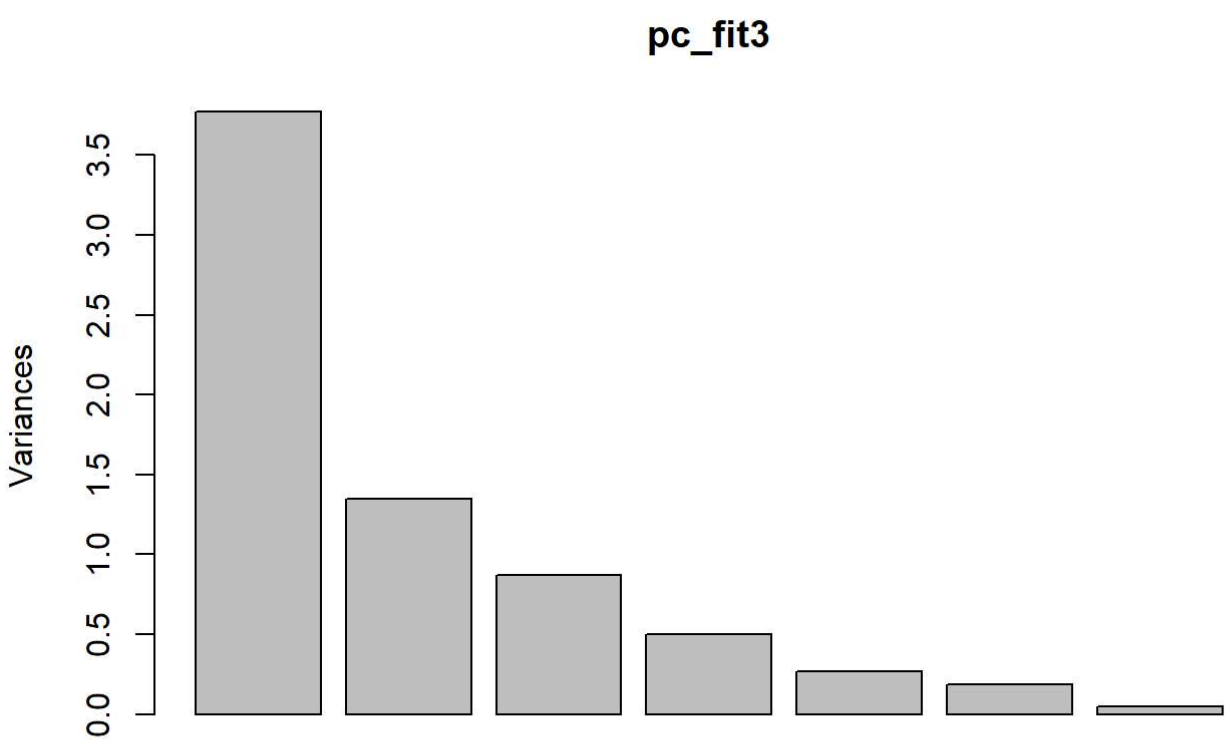
```
sbn_combined <- rbind(sbn1,sbn2)
dim(sbn_combined)
```

```
## [1] 200   7
```

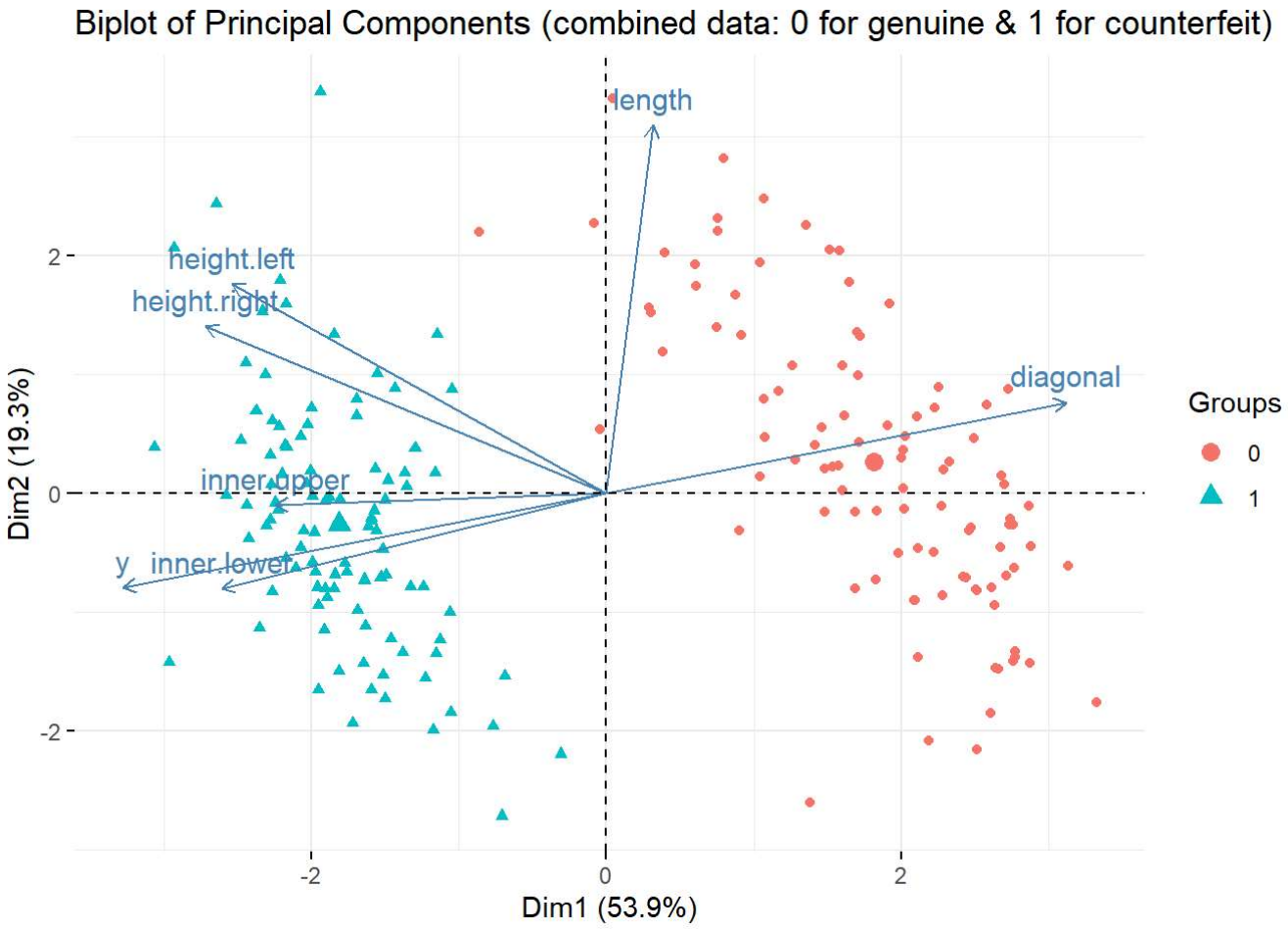
```
pc_fit3 <- prcomp(sbn_combined, center = TRUE, scale = TRUE)
pc_fit3
```

```
## Standard deviations (1, .., p=7):
## [1] 1.9422379 1.1612061 0.9322194 0.7075962 0.5198661 0.4350941 0.2236500
##
## Rotation (n x k) = (7 x 7):
##                      PC1         PC2           PC3         PC4         PC5
## length        0.04645028  0.76250217 -0.0170425080  0.64301302 -0.04223112
## height.left  -0.37315888  0.43265514  0.1032202471 -0.42883870  0.62412344
## height.right -0.39998131  0.34450893  0.1232829374 -0.43304731 -0.62806903
## inner.lower  -0.38328629 -0.19739802  0.5832967625  0.27684012 -0.16620412
## inner.upper  -0.33023142 -0.02502119 -0.7877726220  0.03204421 -0.19390002
## diagonal      0.45870648  0.18767848  0.1141716609 -0.24363606 -0.38075321
## y            -0.48190972 -0.19467491  0.0006823915  0.27942053 -0.06341706
##                      PC6         PC7
## length        0.02544574 -0.01567383
## height.left  -0.28385286  0.09435260
## height.right  0.34451182 -0.07376169
## inner.lower  -0.48876551 -0.36157234
## inner.upper  -0.43397445 -0.20682740
## diagonal     -0.60913842  0.40736818
## y             0.04262056  0.80371025
```

```
plot(pc_fit3)
```



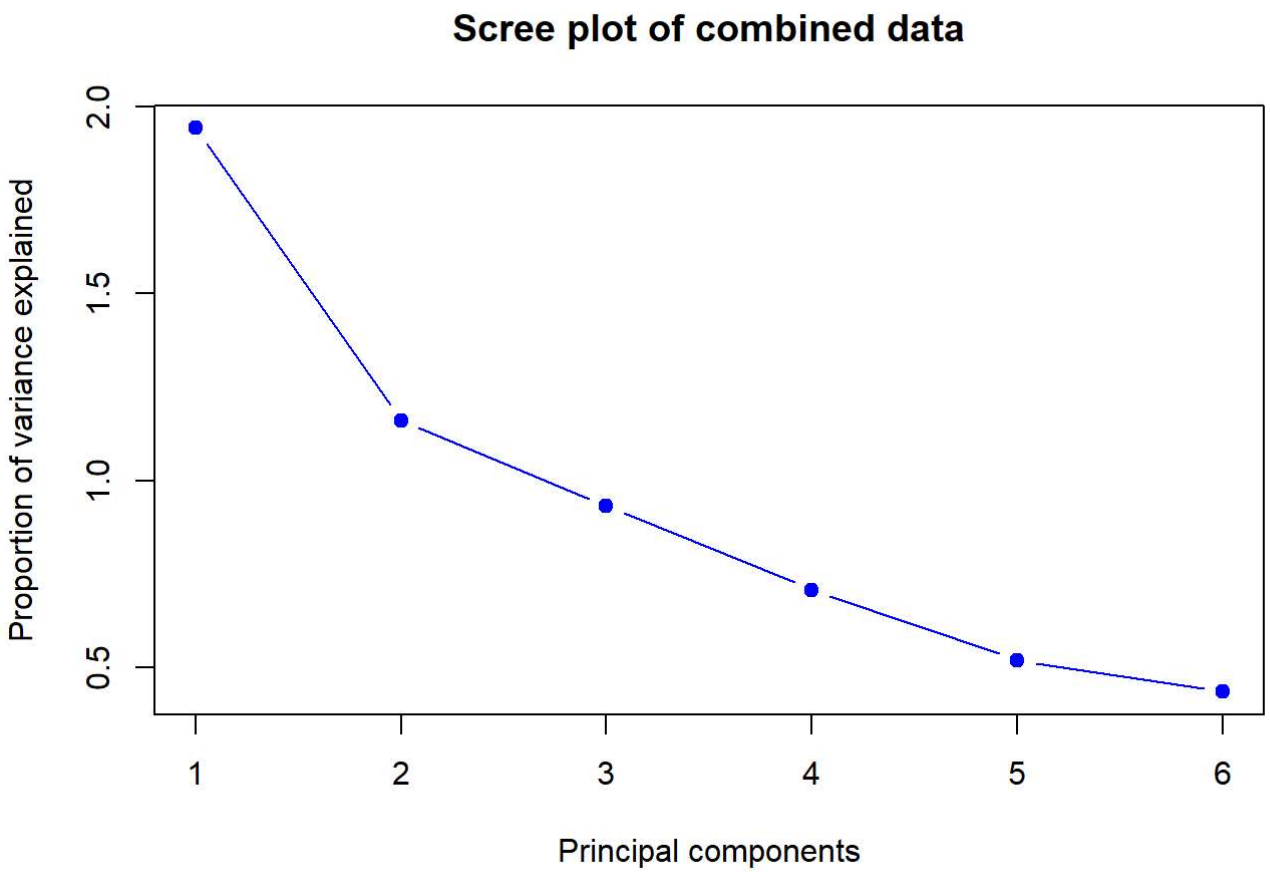

```
fviz_pca_biplot(pc_fit3,label ="var",habillage = sbn_combined$y,title = 'Biplot of Principal Com
ponents (combined data: 0 for genuine & 1 for counterfeit)')
```




```
pc_fit3$sdev
```

```
## [1] 1.9422379 1.1612061 0.9322194 0.7075962 0.5198661 0.4350941 0.2236500
```

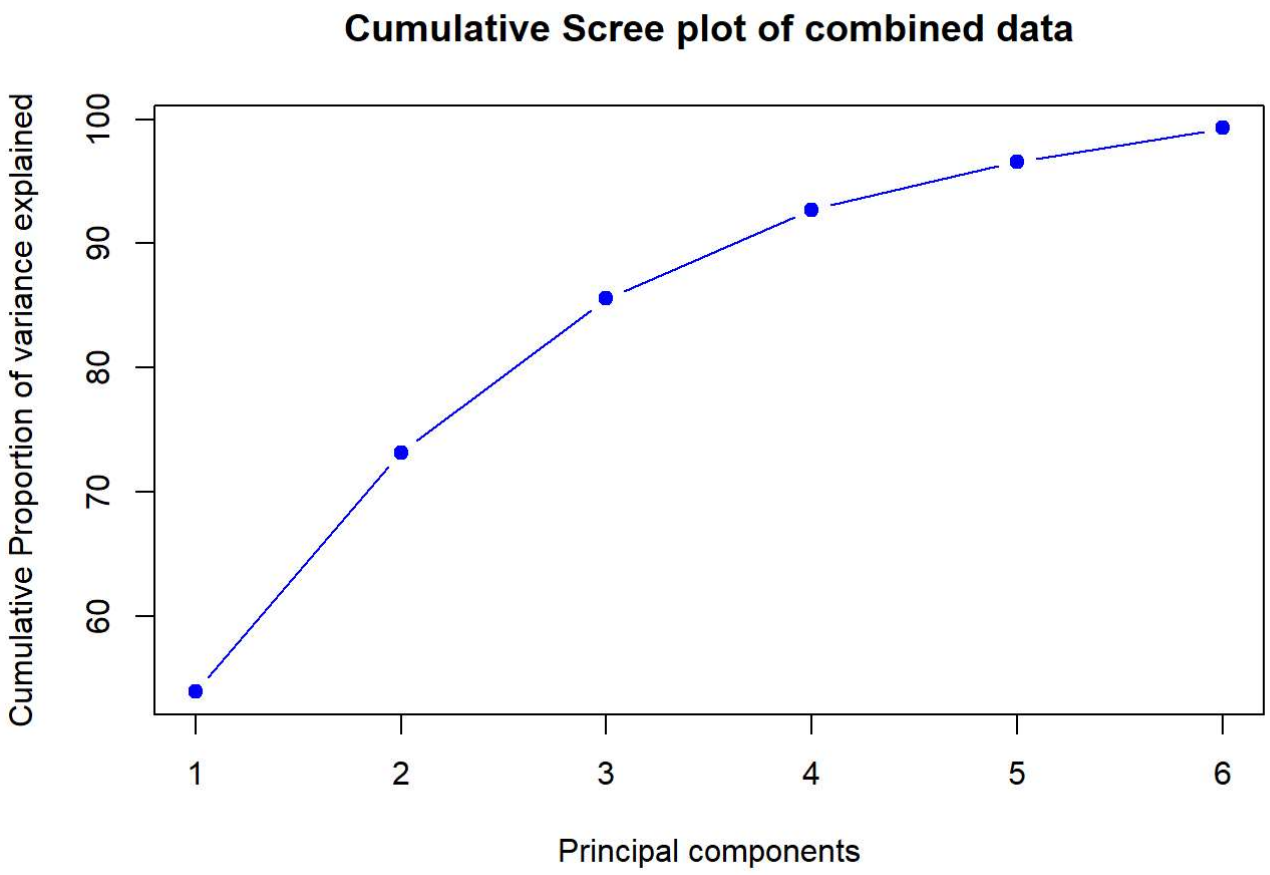
```
# elbow method or scree plot
x_values <- numeric()
y_values <- numeric()

# Iterate over the sequence 1:6 and collect values
for (i in 1:6) {
  x_values <- c(x_values, i)  # Append current iteration value to x_values
  y_values <- c(y_values, pc_fit3$sdev[i])  # Append corresponding pc_fit3$sdev value
}

# Plot the collected values
plot(x_values, y_values,
     type = "b",
     pch = 19,
     col = "blue",
     xlab = "Principal components",
     ylab = "Proportion of variance explained",
     main = "Scree plot of combined data")
```




```
per_var_expl3 <- 100 * ((pc_fit3$sdev)^2) / (sum(((pc_fit3$sdev)^2)))
per_var_expl3
```

```
## [1] 53.8898288 19.2628513 12.4147586  7.1527476  3.8608684  2.7043835  0.7145618
```

```
y_v2 <- numeric()

# Iterate over the sequence 1:6 and collect values
for (i in 1:6) {
  y_v2[i] <- sum(per_var_expl3[1:i])
}

# Plot the collected values
plot(1:6, y_v2,
     type = "b",
     pch = 19,
     col = "blue",
     xlab = "Principal components",
     ylab = "Cumulative Proportion of variance explained",
     main = "Cumulative Scree plot of combined data")
```




```
cat("We can select first three PCs based on the total variance for the combined data to get more than 80%:",per_var_expl3[1] + per_var_expl3[2] + per_var_expl3[3],"%")
```

```
## We can select first three PCs based on the total variance for the combined data to get more than 80%: 85.56744 %
```

We can select first three Principal Components for genuine notes & combined notes to get more than 80% of the total variance but for counterfeit notes we need to select first 4 PC's to get more than 80% of total variance.

Question 3) Consider the USArrests data. library(ISLR); data(USArrests); head(USArrests)

```
library(ISLR2)
data(USArrests)

dim(USArrests)
```

```
## [1] 50  4
```

```
head(USArrests)
```

```
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
```

```
str(USArrests)
```

```
## 'data.frame':    50 obs. of  4 variables:
##  $ Murder  : num  13.2 10 8.1 8.8 9 7.9 3.3 5.9 15.4 17.4 ...
##  $ Assault : int  236 263 294 190 276 204 110 238 335 211 ...
##  $ UrbanPop: int  58 48 80 50 91 78 77 72 80 60 ...
##  $ Rape    : num  21.2 44.5 31 19.5 40.6 38.7 11.1 15.8 31.9 25.8 ...
```

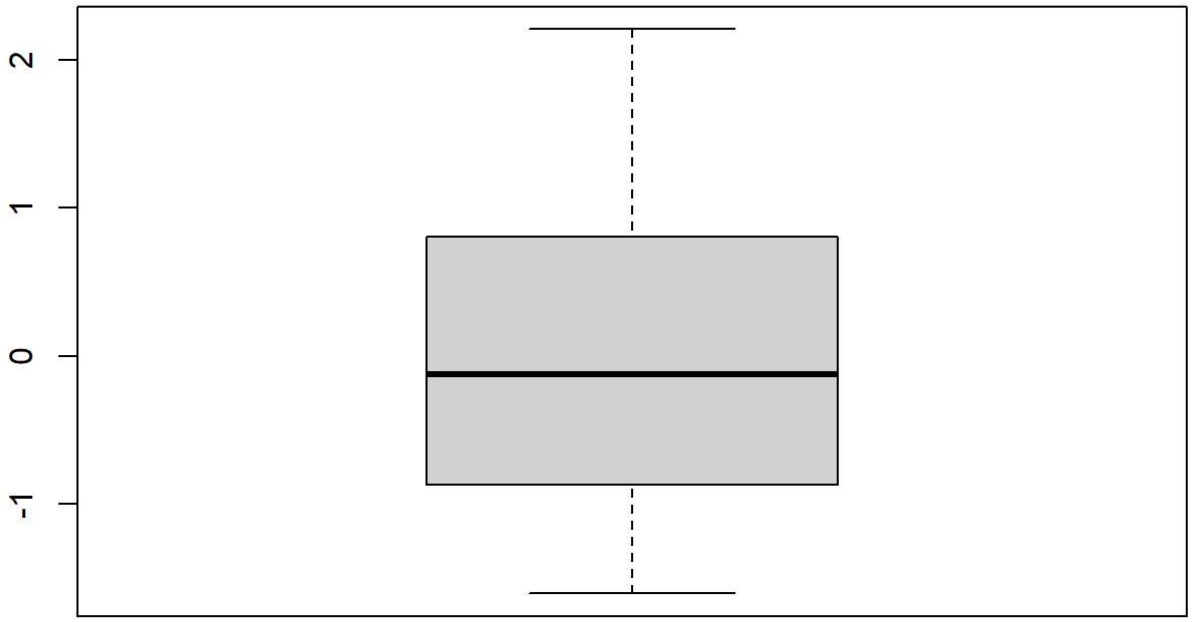
```
summary(USArrests)
```

```
##      Murder          Assault         UrbanPop          Rape
##  Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   : 7.30
##  1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st Qu.:15.07
##  Median : 7.250   Median :159.0   Median :66.00   Median :20.10
##  Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean   :21.23
##  3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd Qu.:26.18
##  Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.   :46.00
```

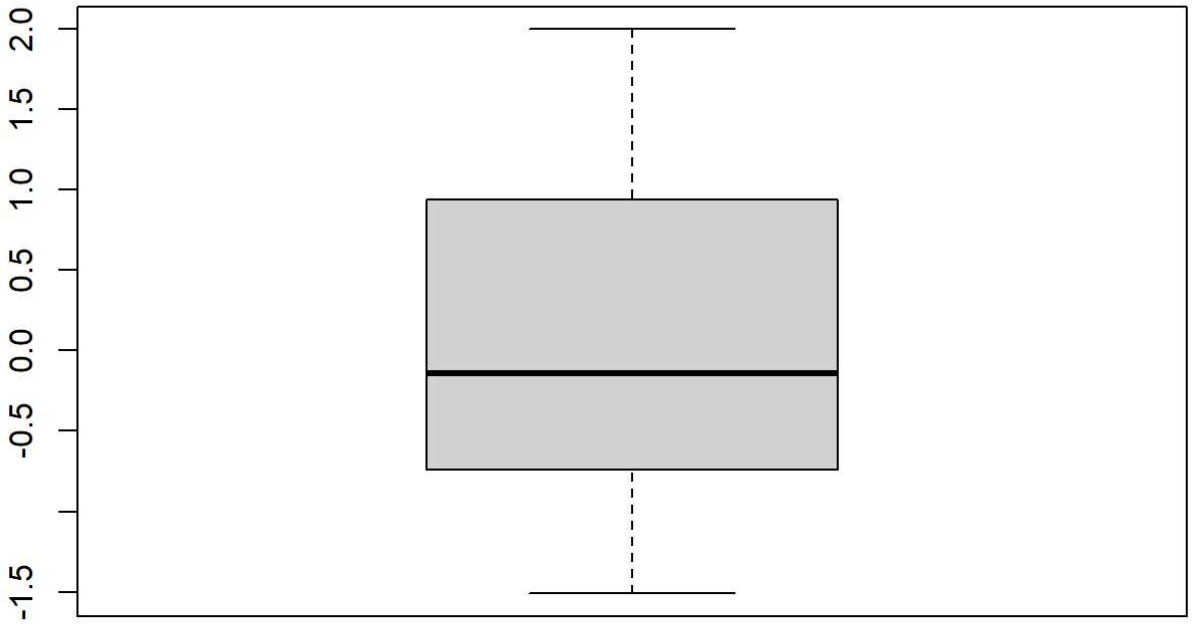
```
colSums(is.na(USArrests))
```

```
##   Murder  Assault UrbanPop     Rape
##        0        0        0        0
```

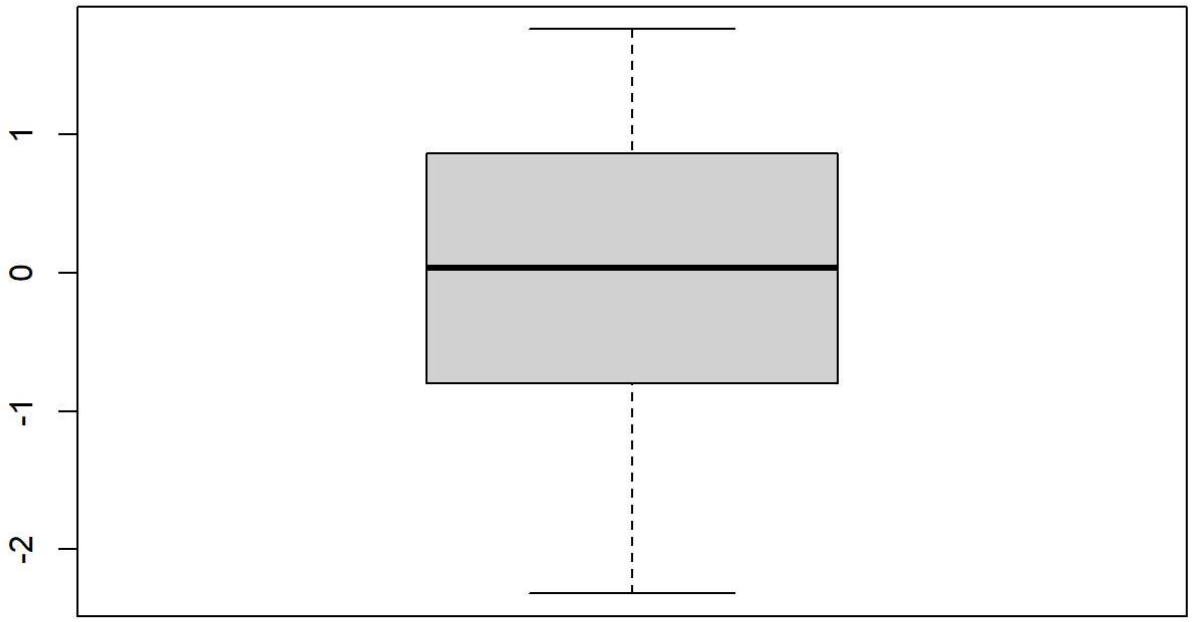
```
# Scaling the data
USA_scale <- as.data.frame(scale(USArrests))

boxplot(USA_scale$Murder)
```



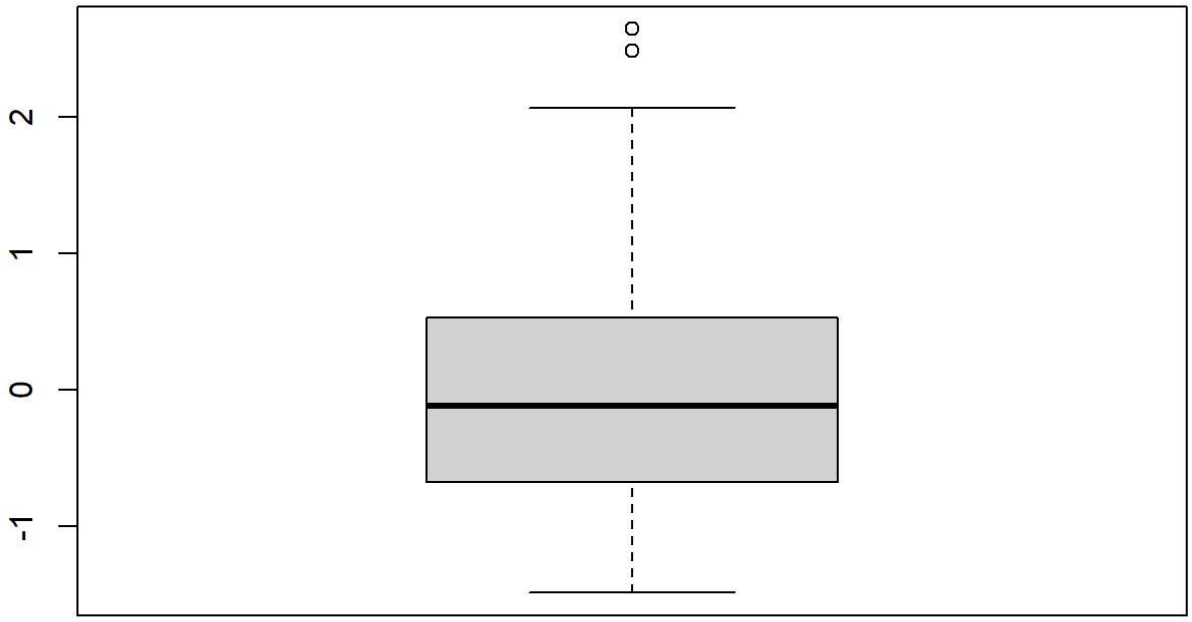

```
boxplot(USA_scale$Assault)
```




```
boxplot(USA_scale$UrbanPop)
```

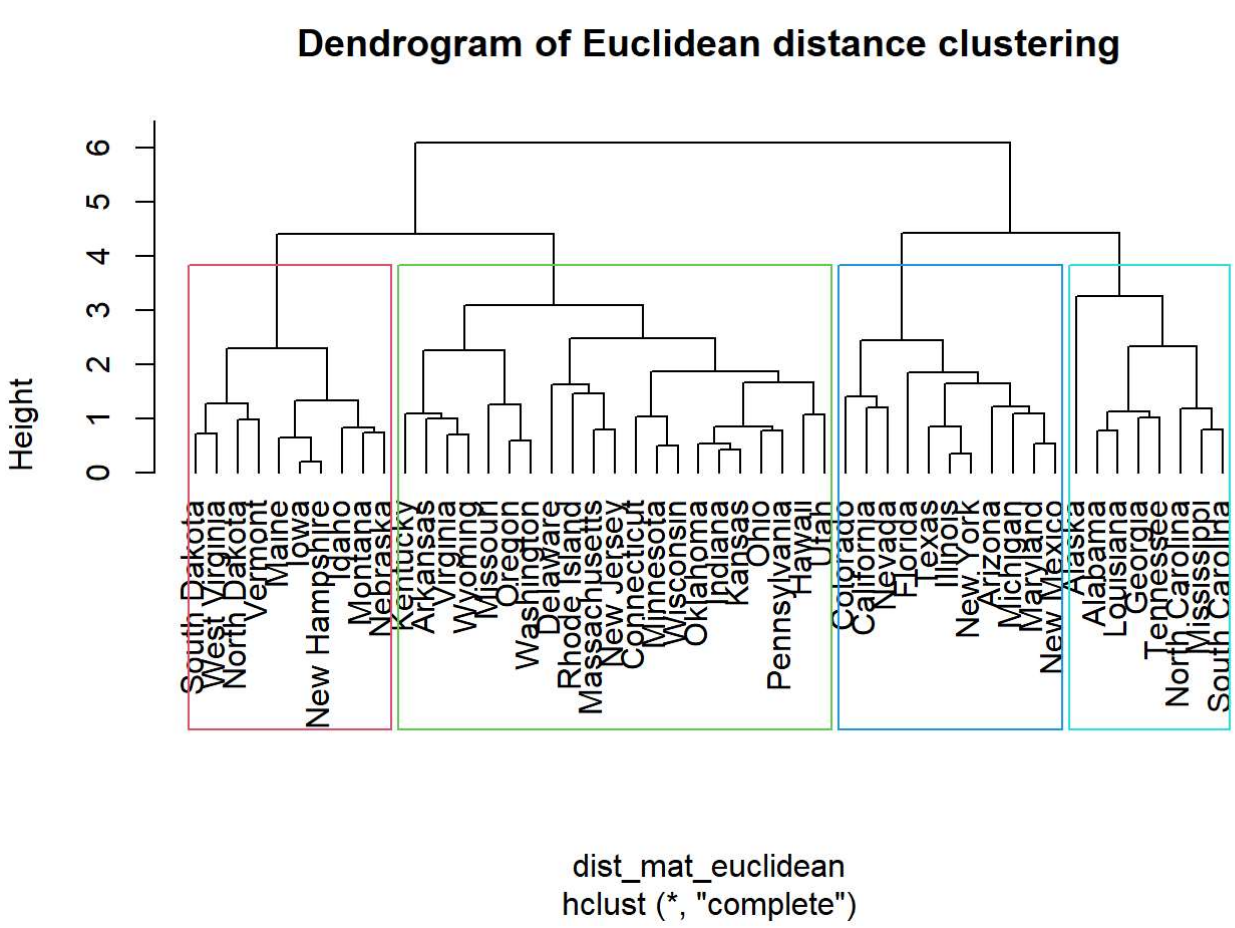



```
boxplot(USA_scale$Rape)
```

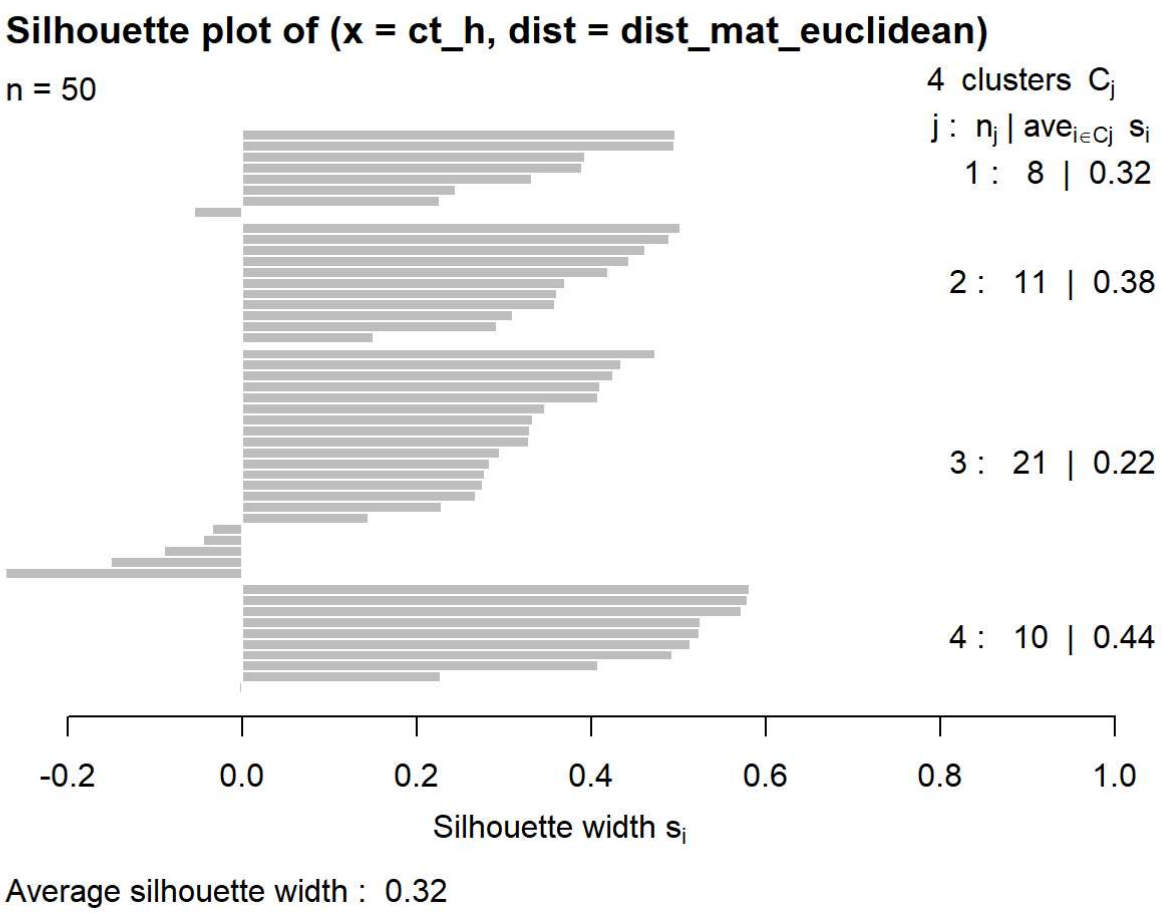


a) Perform hierarchical clustering with complete linkage and Euclidean distance to cluster the states'. Include an illustration of the dendrogram. How many clusters do you detect. Is this what you expected?

```
library("cluster")

set.seed(123)
# using method = 'euclidean'
dist_mat_euclidean <- dist(USA_scale, method = 'euclidean')

h_clust_com <- hclust(dist_mat_euclidean, method = 'complete')
plot(h_clust_com, hang = -1, main = "Dendrogram of Euclidean distance clustering")
rect.hclust(h_clust_com, k = 4, border = 2:5)
```



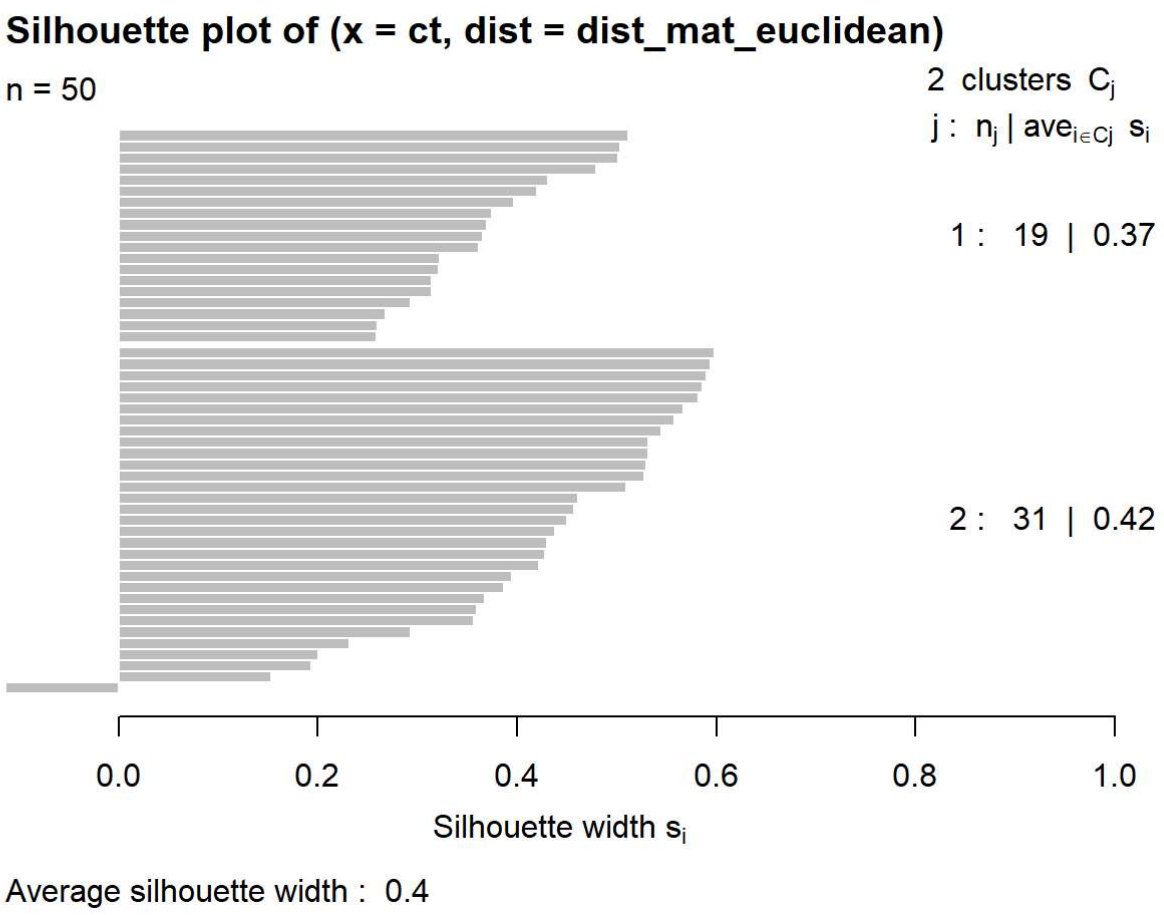

```
# Based on the above dendrogram i am cutting at h = 4 because of maximum vertical distance.
ct_h <- cutree(h_clust_com, h = 4)
si_h <- silhouette(ct_h, dist = dist_mat_euclidean)

plot(si_h)
```



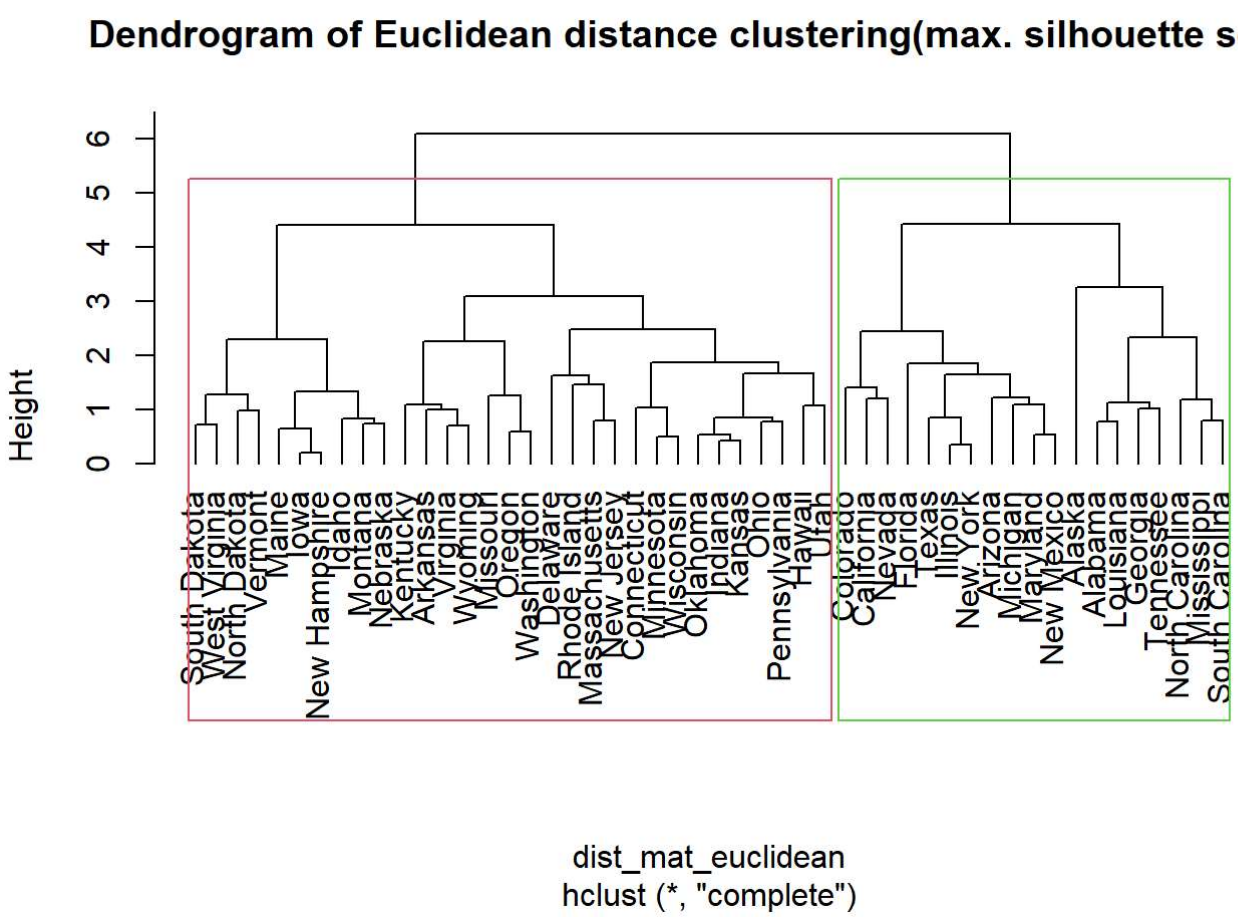

```
store <- c()
for (i in 2:6){
    ct <- cutree(h_clust_com, k=i)
    si <- silhouette(ct, dist = dist_mat_euclidean)
    avg_width <- summary(si)$avg.width
    store <- c(store, avg_width)
}
store
```

```
## [1] 0.4047945 0.3692431 0.3159551 0.3174349 0.2621063
```

```
# From the above average silhouette scores we are getting maximum at k=2.
ct <- cutree(h_clust_com, k = 2)
si <- silhouette(ct, dist = dist_mat_euclidean)

plot(si)
```



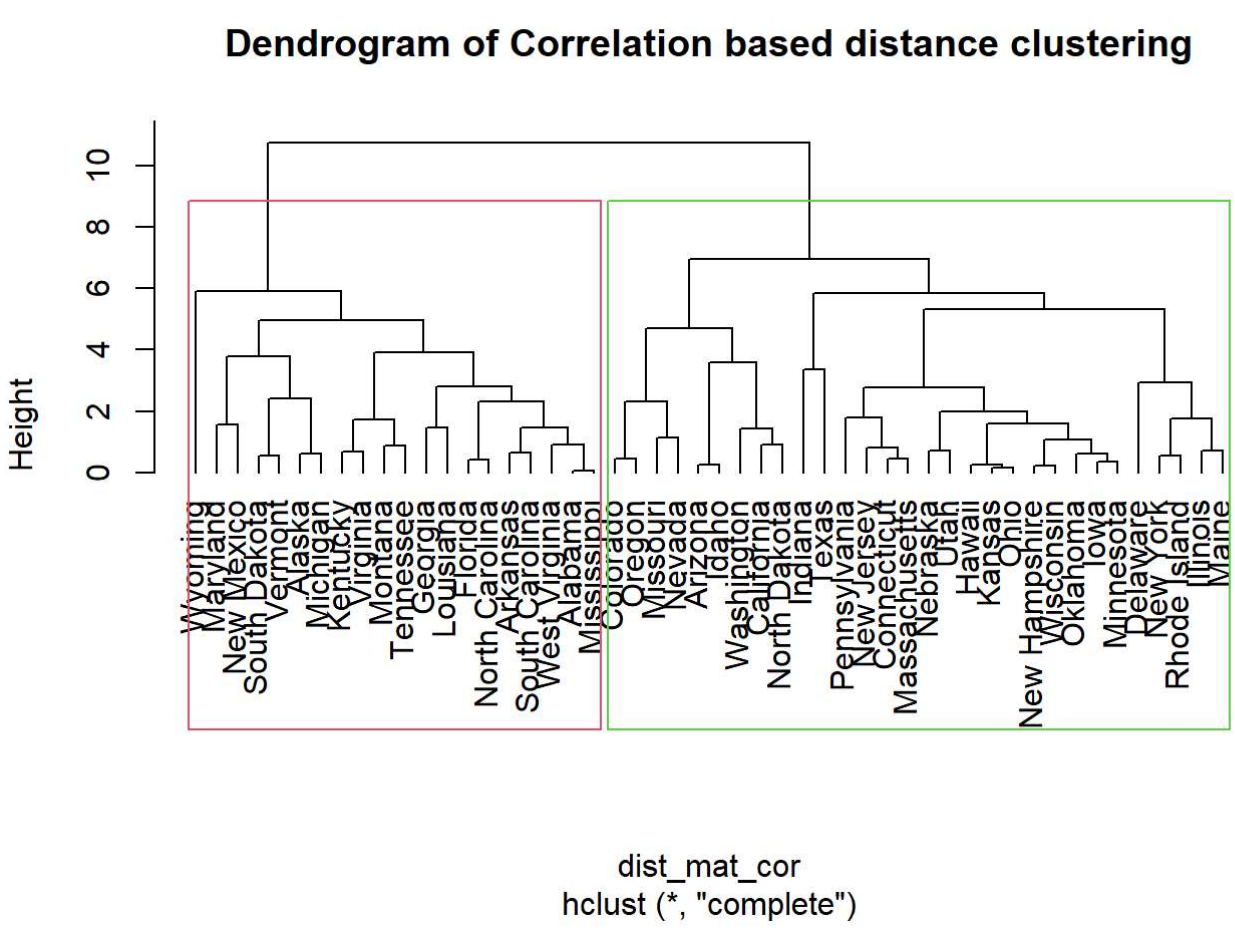

```
plot(h_clust_com, hang = -1, main = "Dendrogram of Euclidean distance clustering(max. silhouette
score)")
rect.hclust(h_clust_com, k = 2, border = 2:3, cluster = ct)
```



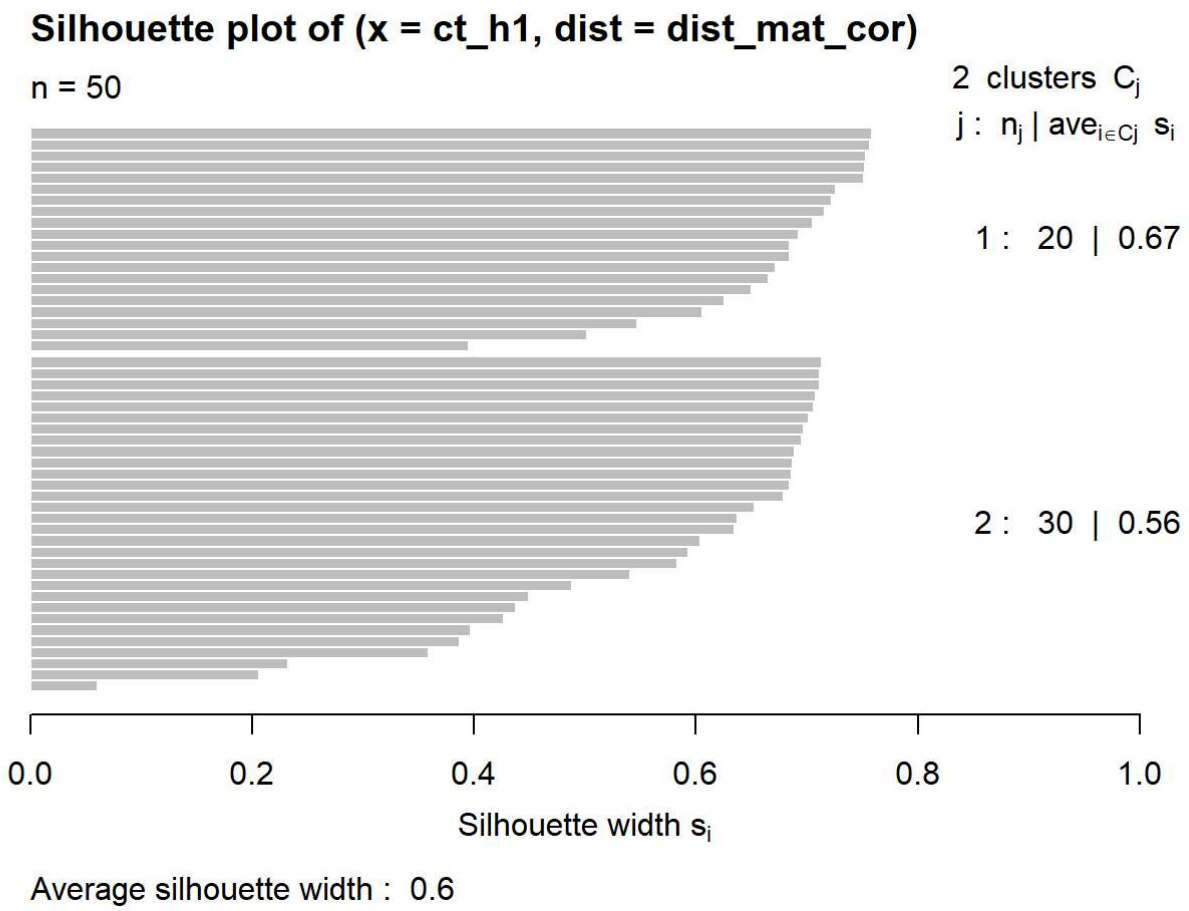
I expected 4 clusters by looking at dendrogram and thought to cut the tree at height 4 based on the maximum distance between clusters but according to silhouette scores got the maximum score at k = 2. Therefore, 2 clusters.

b) Perform hierarchical clustering using a correlation based distance and complete linkage. Include an illustration of the dendrogram. How many clusters do you detect. Is this what you expected?

```
set.seed(123)

dist_mat_cor <- dist(1 - cor(t(USA_scale)))

h_clust_com1 <- hclust(dist_mat_cor, method = 'complete')
plot(h_clust_com1, hang = -1, main = "Dendrogram of Correlation based distance clustering")
rect.hclust(h_clust_com1, k = 2, border = 2:3)
```



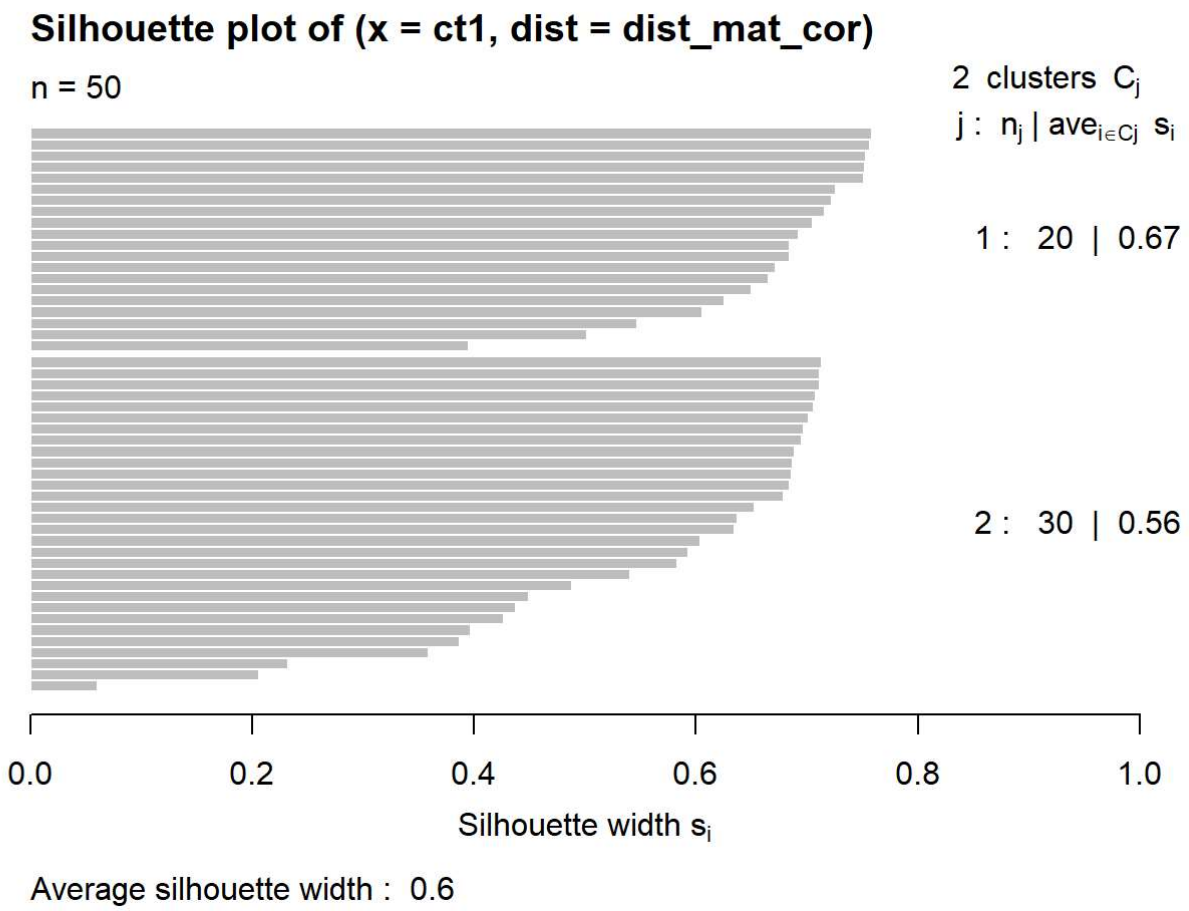

```
# Based on the above dendrogram i am cutting at h = 8 because of maximum vertical distance.
ct_h1 <- cutree(h_clust_com1, h = 8)
si_h1 <- silhouette(ct_h1, dist = dist_mat_cor)

plot(si_h1)
```



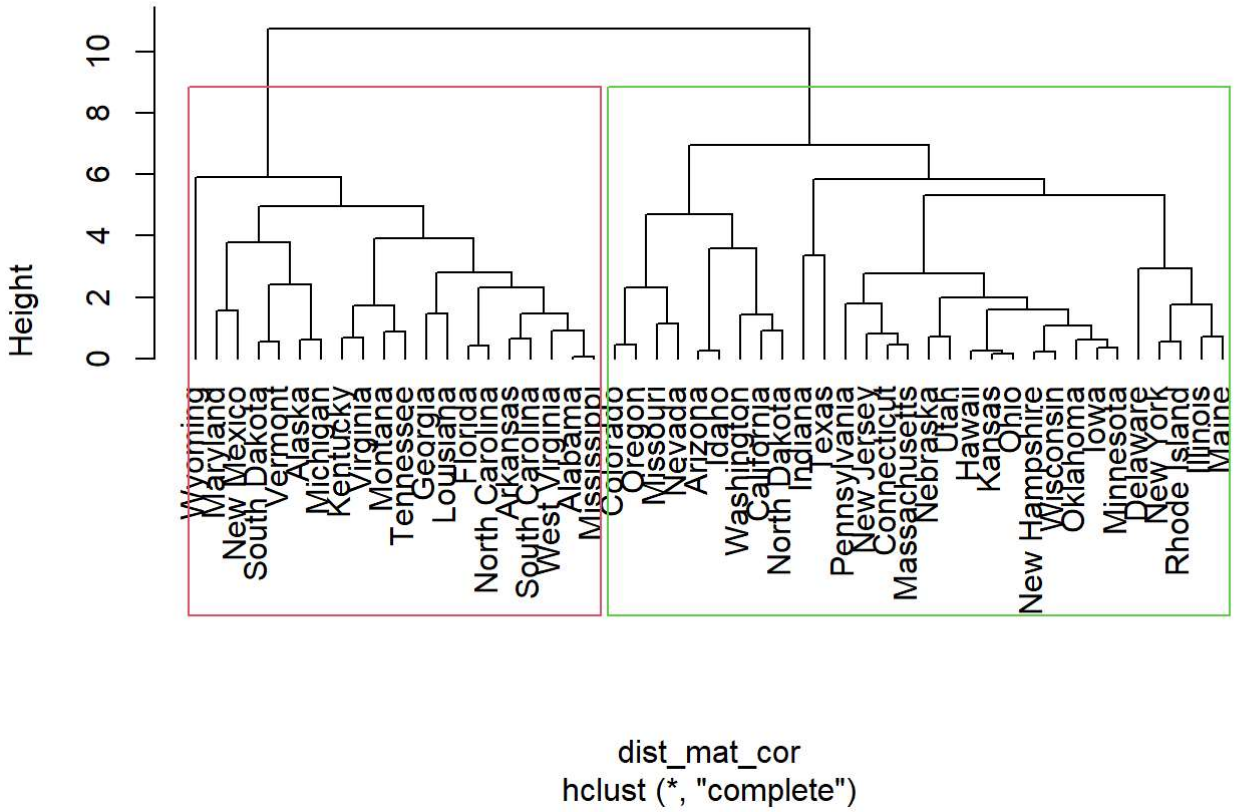

```
store1 <- c()
for (i in 2:6){
    ct <- cutree(h_clust_com1, k=i)
    si <- silhouette(ct, dist = dist_mat_cor)
    avg_width <- summary(si)$avg.width
    store1 <- c(store1, avg_width)
}
store1
```

```
## [1] 0.6026494 0.5023592 0.3886554 0.3843002 0.4087648
```

```
# From the above average silhouette scores we are getting maximum at k=2.
ct1 <- cutree(h_clust_com1, k = 2)
si1 <- silhouette(ct1, dist = dist_mat_cor)

plot(si1)
```




```
plot(h_clust_com1, hang = -1, main = "Dendrogram of Correlation distance clustering(max. silhoue
tte score)")
rect.hclust(h_clust_com1, k = 2, border = 2:3, cluster = ct1)
```



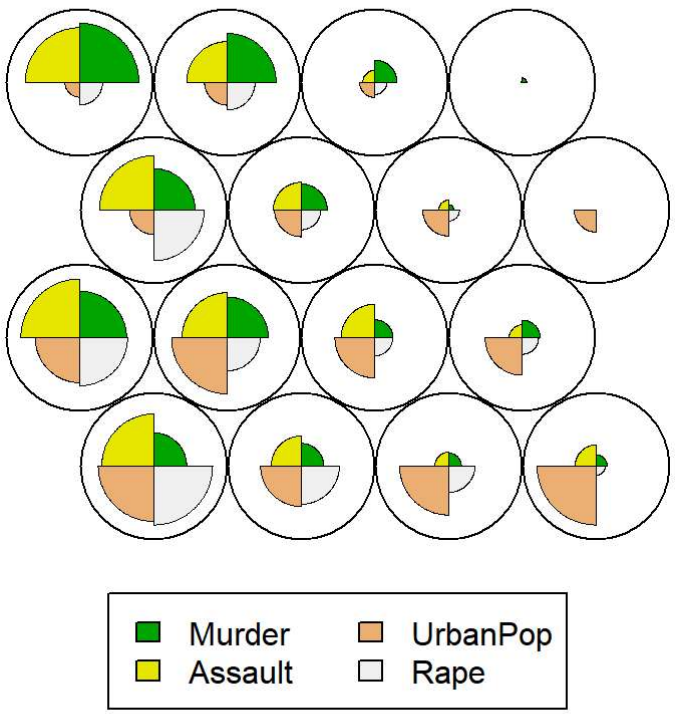
I expected 2 clusters by looking at dendrogram and thought to cut the tree at height 8 based on the maximum distance between clusters and according to silhouette scores also we got the maximum score at k = 2. Therefore, 2 clusters.

c) Fit a SOM to the data and present the results (e.g., classic visualizations). Perform hclust on the codebook vectors of the SOM. Is this what you expected? Does this result generally support your results in Part A.

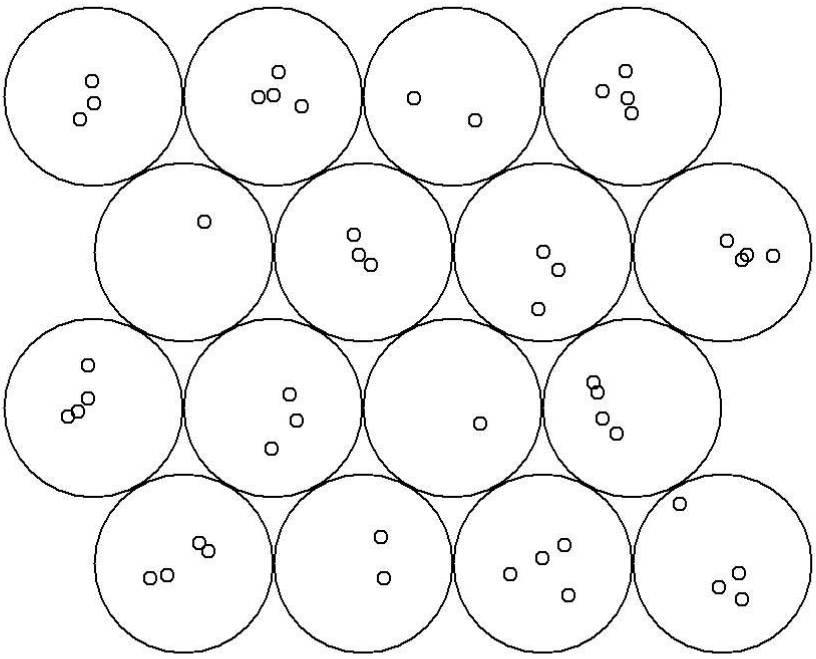
```
library(kohonen)
```

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

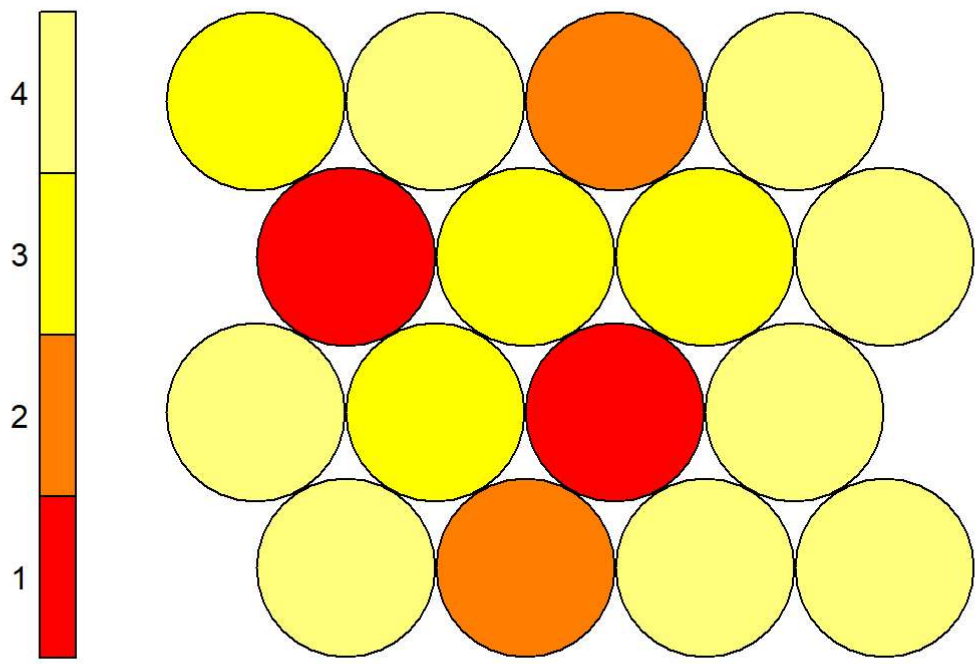
```
set.seed(123)
# setting the SOM grid dimensions
grid_size <- somgrid(xdim = 4, ydim = 4, topo = "hexagonal")
# Train the SOM
som_model <- som(as.matrix(USA_scale), grid = grid_size, rlen = 100)

# Plot the SOM
plot(som_model, type = "codes", main = "USArrests data SOM Prototypes")
```



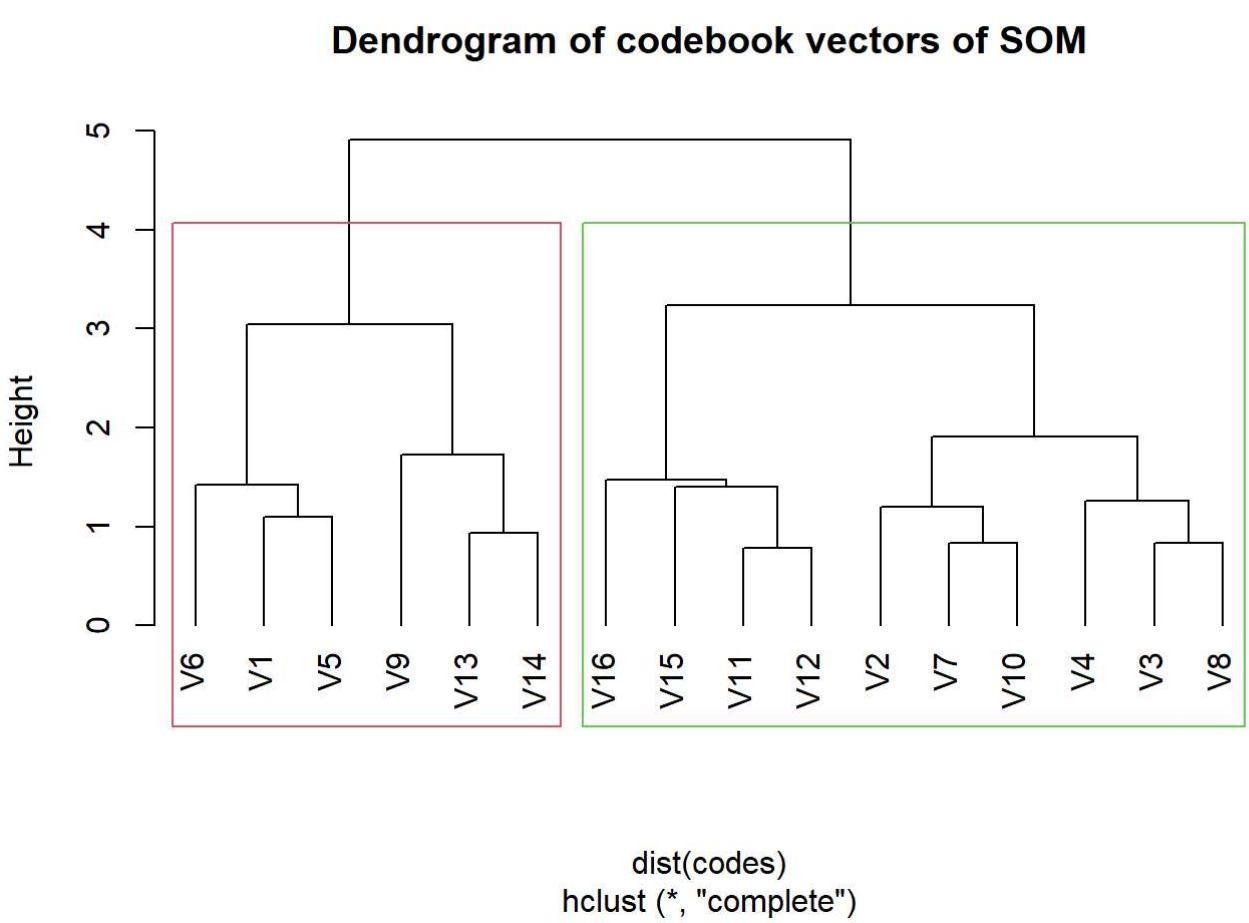

```
plot(som_model, type = "mapping", main = "USArrests data mapping")
```



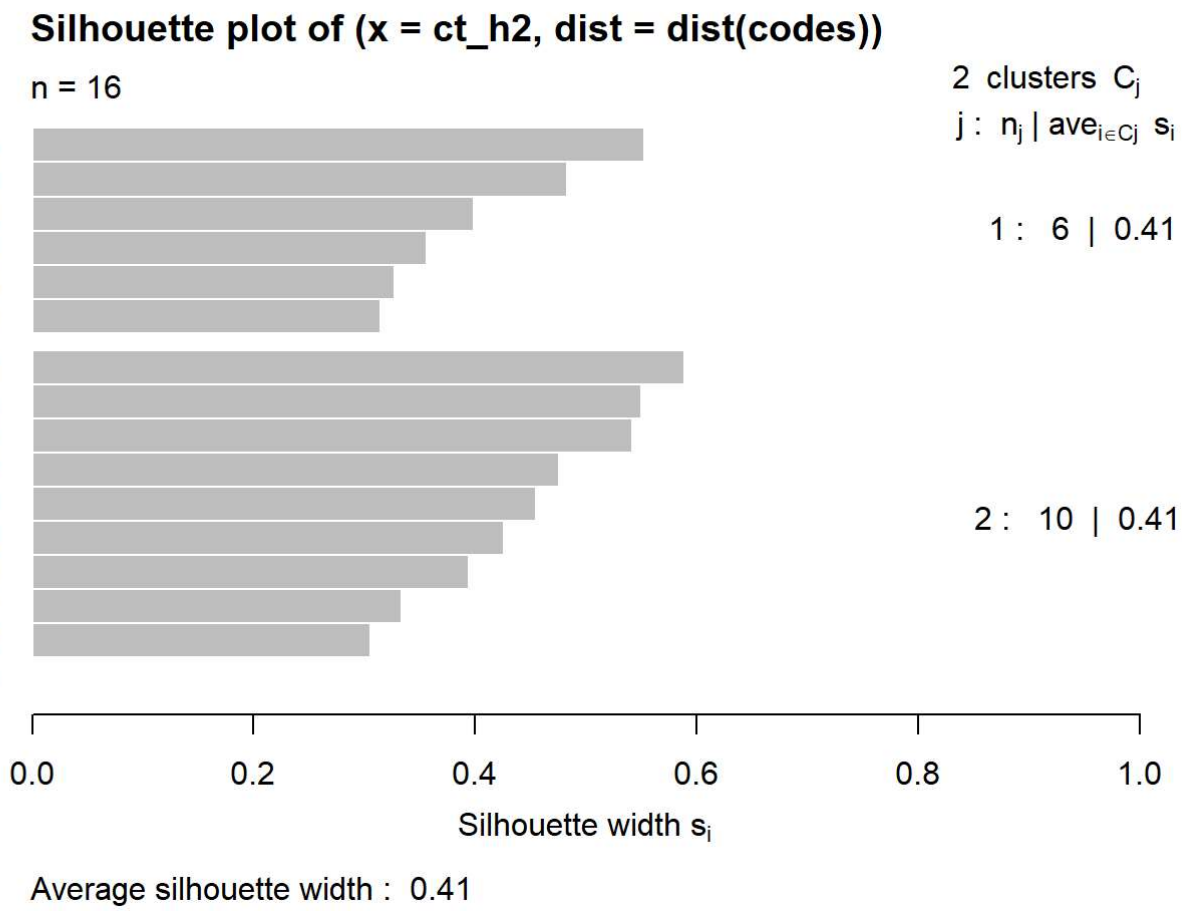

```
plot(som_model, type = "count", main = "USArrests data count")
```




```
# hclust on codebook vectors
codes <- som_model$codes[[1]]
hc <- hclust(dist(codes))
plot(hc, hang = -1, main = "Dendrogram of codebook vectors of SOM")
rect.hclust(hc, k = 2, border = 2:3)
```



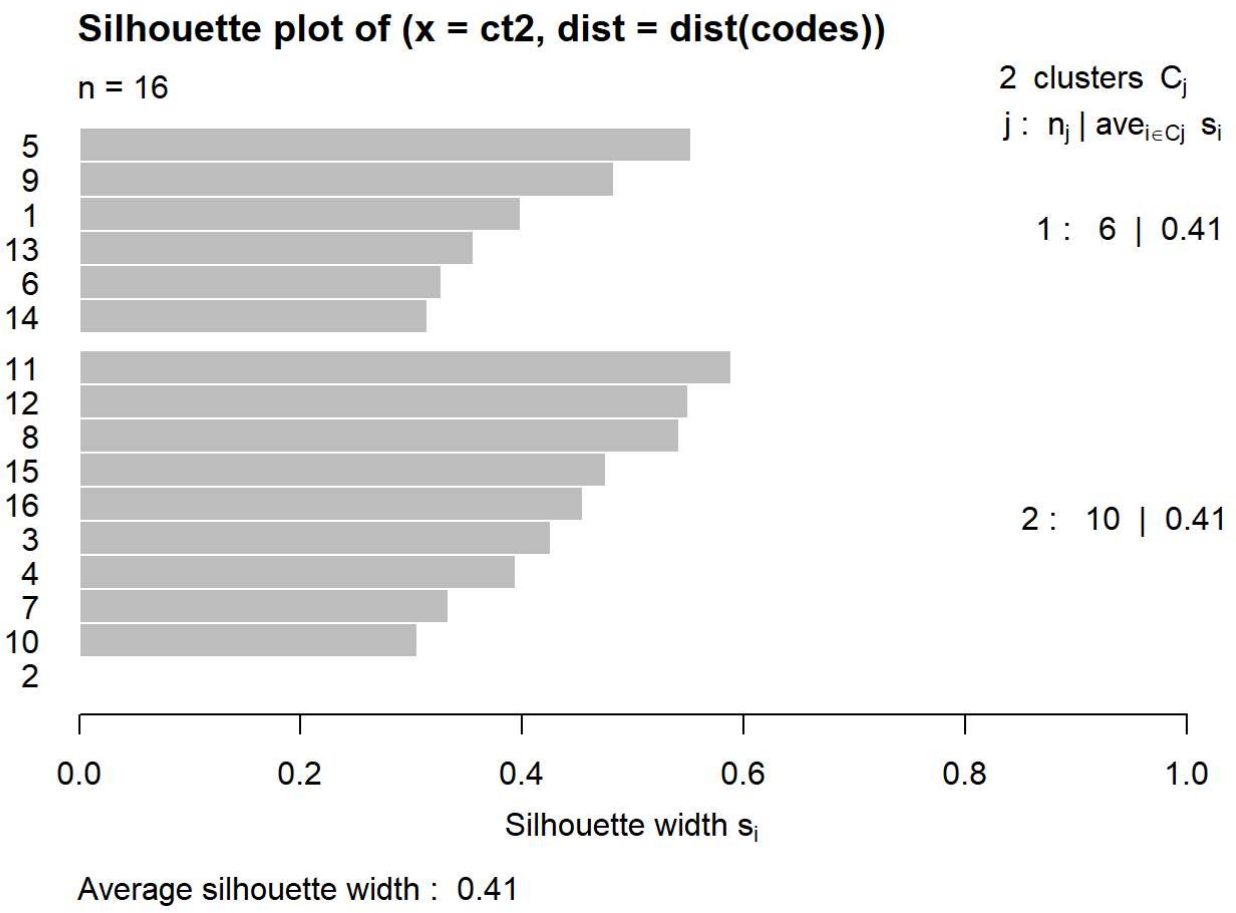

```
ct_h2 <- cutree(hc, h = 4)
si_h2 <- silhouette(ct_h2, dist = dist(codes))

plot(si_h2)
```



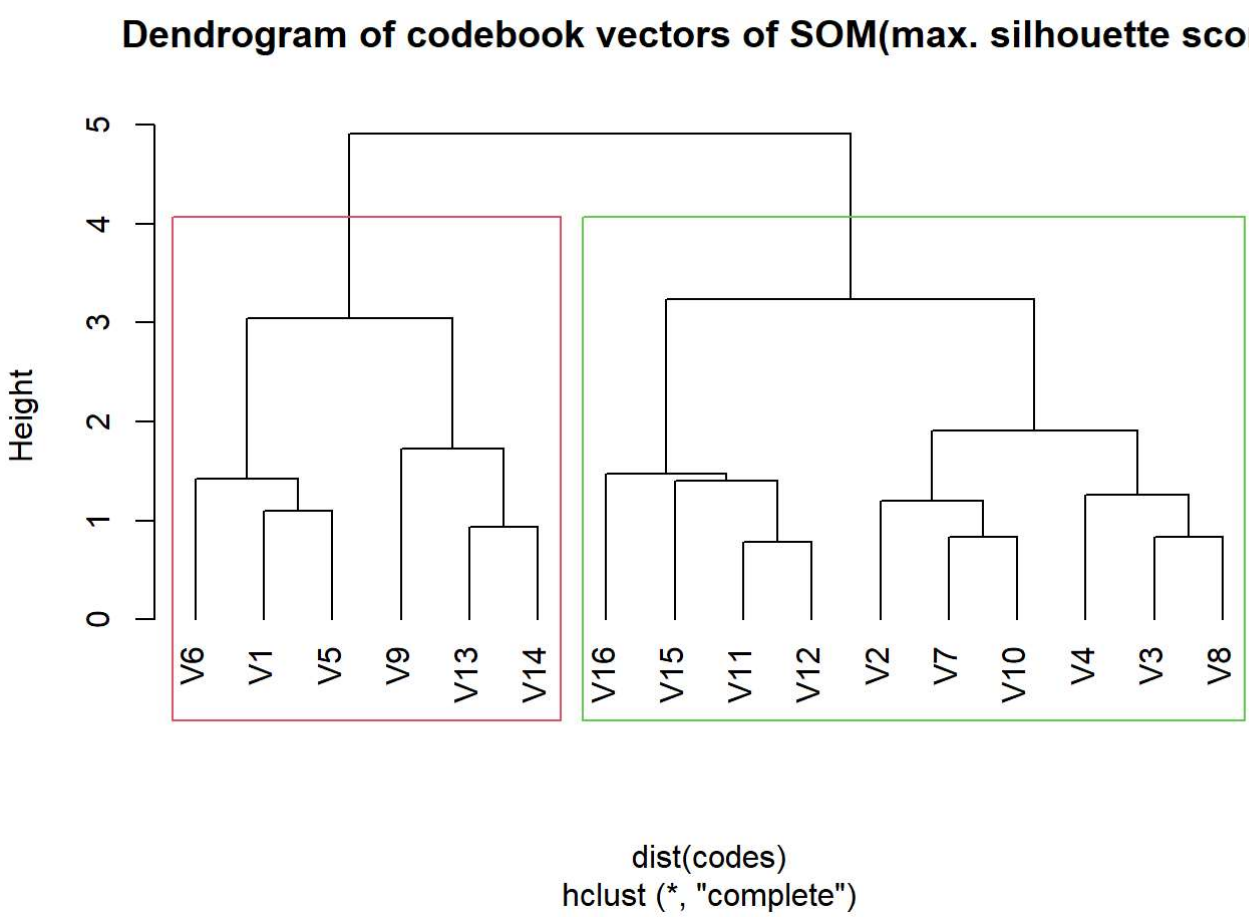

```
store2 <- c()
for (i in 2:6){
    ct <- cutree(hc, k=i)
    si <- silhouette(ct, dist = dist(codes))
    avg_width <- summary(si)$avg.width
    store2 <- c(store2, avg_width)
}
store2
```

```
## [1] 0.4064783 0.3359761 0.3478668 0.2741793 0.2415991
```

```
# From the above average silhouette scores we are getting maximum at k=2.
ct2 <- cutree(hc, k = 2)
si2 <- silhouette(ct2, dist = dist(codes))

plot(si2)
```




```
plot(hc, hang = -1, main = "Dendrogram of codebook vectors of SOM(max. silhouette score)")
rect.hclust(hc, k = 2, border = 2:3, cluster = ct2)
```



I expected 2 clusters by looking at dendrogram and if we cut the tree at height 4 based on the maximum vertical distance between clusters and according to silhouette scores also we got the maximum score at k = 2. Therefore, 2 clusters. In part A also we got the optimal clusters as k = 2.

d) Comment on the advantages and limitations of hierarchical clustering to SOM, and discuss when one would be preferred over the other. Comments should capture a general comparison, in addition to the comments specific to this exercise.

Advantages of Hierarchical Clustering:

Interpretability: Hierarchical clustering produces a dendrogram, which provides a visual representation of the clustering and the hierarchical relationships between clusters. This can help in understanding the structure of the data and determining the appropriate number of clusters.

Flexibility: Hierarchical clustering does not require specifying the number of clusters, which can be advantageous when the optimal number of clusters is unknown.

Limitations of Hierarchical Clustering:

Sensitivity to Outliers: Hierarchical clustering algorithms can be sensitive to outliers, which can distort the results of clustering.

Handling Large Datasets: Hierarchical clustering may not be suitable for very large datasets, because of its computational complexity. And interpretation of clusters is also difficult.

Advantages of Self-Organizing Maps (SOM):

Topology preservation: SOMs preserve the input data's topological structure, which means that similar data points are assigned to the nearest neurons in the grid. This maintains the relationships and inherent structure between data points, making clusters easier to interpret.

Dimensionality reduction: SOMs can be used to reduce dimensionality by preserving the essential characteristics in the data. By mapping high-dimensional input data to a lower dimensional grid. SOMs can capture the main features of high-dimensional data in more interpretable and compact form.

Visualization: It is a useful technique for representing high-dimensional data in a low dimensional space. The organized grid structure provides an intuitive representation of clusters and their relationships, which helps with pattern recognition and exploratory data analysis.

Limitations of Self-Organizing Maps (SOM):

Difficult to interpret: Interpreting the SOM output can be subjective, especially when determining the number of clusters.

Lack of Hierarchical Structure: SOMs do not provide a hierarchical representation of the clusters, which may be desired in some applications.

In general, hierarchical clustering is preferred when:

• Interpretability of the hierarchical structure of the data is important.

• The number of clusters is unknown or needs to be determined from the data.

On the other hand, SOM is preferred when:

• Dimensionality reduction and visualization of high-dimensional data are desired.

• Preserving the topological structure of the data is important.

Both techniques have their strengths and weaknesses, and the choice between the 2 methods depends on the problem or nature of the data, and the desired interpretability and visualization of the results.

• **In the context of the USArrests dataset, both hierarchical clustering and SOM identified two clusters as optimal, suggesting that the results are consistent across the two methods. However, the interpretation and visualization of the clusters might differ.**