Homework 4

Hong Xu

September 24, 2015

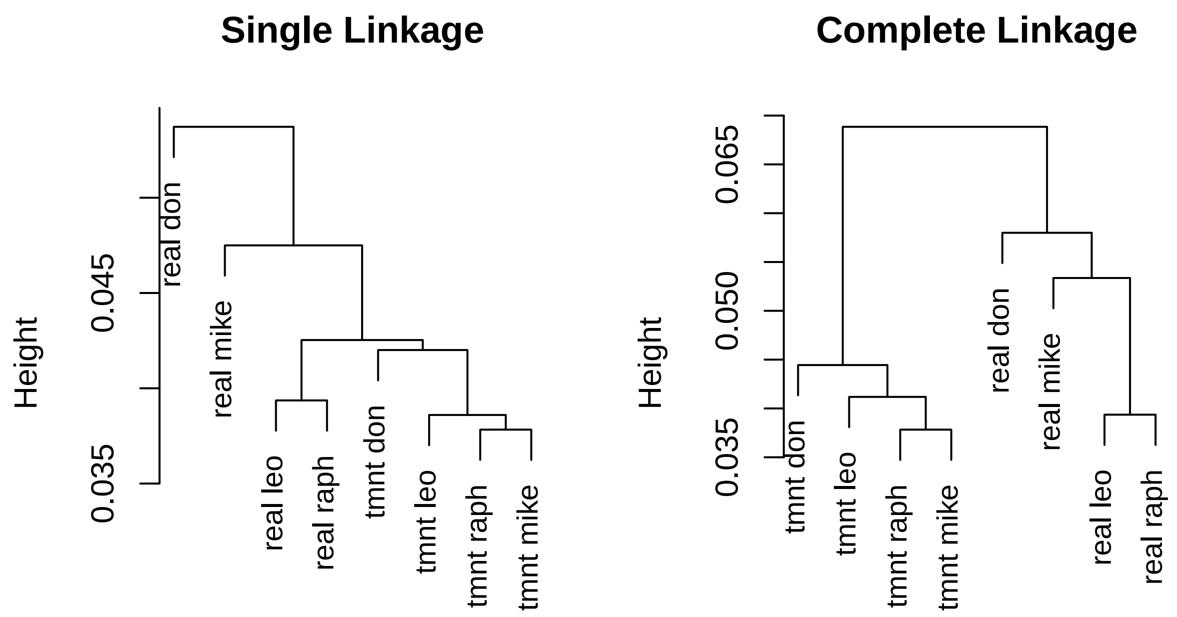
Problem 1

```r
load("hw1prob1.Rdata")
D <- nrow(dtm)
## a)
## dtm1
# first normalize
dtm1 <- t(apply(dtm, 1, function(x) x/sum(x)))
# then IDF transformation
dtm1 <- scale(dtm1, center = FALSE, scale = apply(dtm1, 2, function(x) 1/log(D/sum(x > 0))))

## dtm2
# first IDF transformation
dtm2 <- scale(dtm, center = FALSE, scale = apply(dtm, 2, function(x) 1/log(D/sum(x > 0))))
# then normalize
dtm2 <- t(apply(dtm2, 1, function(x) x/sum(x)))


# compare
all(dtm1 == dtm2)
```

```
## [1] FALSE
```

```r
## They are not supposed to be equal. IDF transformation will impose different weights on
## different terms so the normalized results will not be proportional to the term frequencies.
## We should always normalize first then perform IDF weighting.


## b)
# just normalize
dtm3 <- t(apply(dtm, 1, function(x) x/sum(x)))
# compute the distance
computeDistance <- function(x, q) {
  # input:
  # x: a vector of normalized terms for a document
  # q: a vector of normalized terms for the query
  sqrt(sum((x - q)^2))
}
dist3 <- apply(dtm3, 1, computeDistance, q = dtm3[3,])
# find the closet
dist3[order(dist3)]
```

```
##  tmnt mike  tmnt raph   tmnt leo   tmnt don  real raph  real mike
## 0.00000000 0.03782949 0.04118468 0.04200612 0.04487837 0.04750018
##   real leo   real don
## 0.04887327 0.06376091
## tmnt raph is the closest document


## c)
dist3_dist <- dist(dtm3)
hc_single <- hclust(dist3_dist, method = "single")
hc_complete <- hclust(dist3_dist, method = "complete")
# plot them together
oldpar <- par(mfrow =c(1,2))
plot(hc_single , main=" Single Linkage ", xlab="", sub ="", cex =.9)
plot(hc_complete ,main =" Complete Linkage ", xlab="", sub ="", cex =.9)
```



```
par(oldpar)
## Obviously the Complete Linkage method gives a more reasonable clustering.

## d)
# count the total occurance
word_count <- apply(dtm, 2, sum)
# print the top 20 and their counts
word_count[order(word_count, decreasing = TRUE)[1:20]]
```

```
##          the          and          his          was     leonardo
##         2664         1127          636          453          342
##         that          for         with michelangelo      raphael
##          297          282          279          277          212
##         from         this        which      turtles   donatello
##          209          189          129          127         117
##          him       series         were          who         one
##          116          115          111          110         103
```

```
# how many percentage do the top 20 account for?
sum(word_count[order(word_count, decreasing = TRUE)[1:20]])/sum(word_count)
```

```
## [1] 0.2434249
```

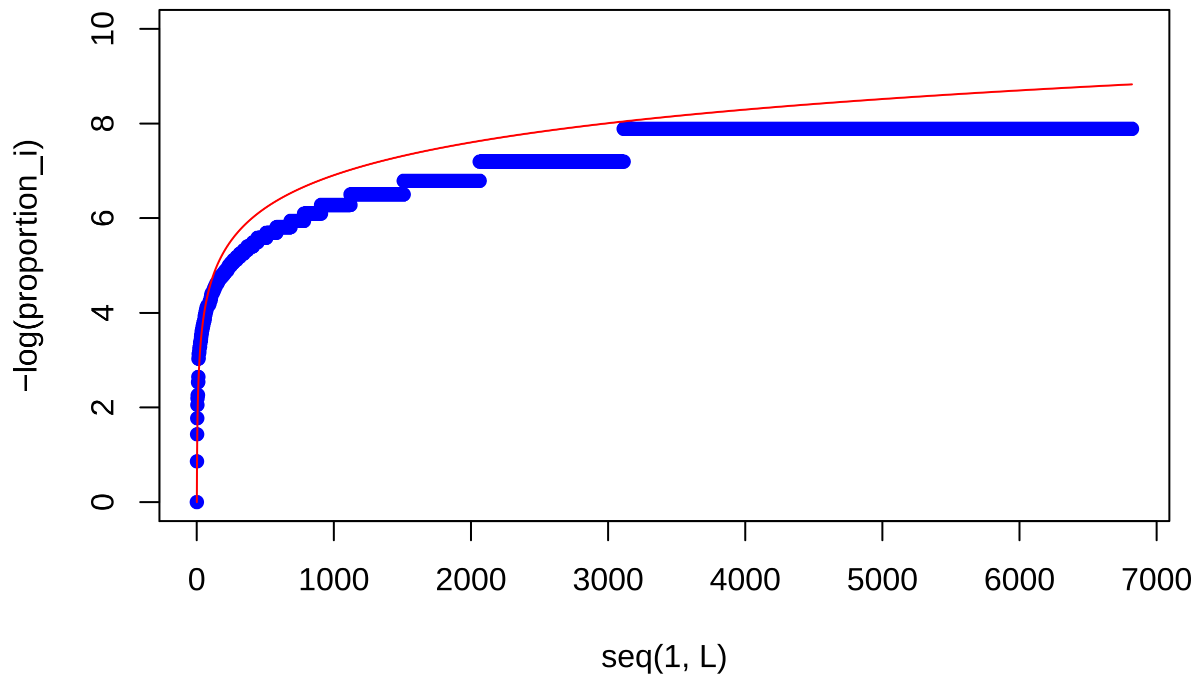
```
# how many top words count for 50% of the occurance?
L <- length(word_count)
percentage_i <- rep(0, L)
# find the count
for (i in 1:L) {
  percentage_i[i] <- sum(word_count[order(word_count, decreasing = TRUE)[1:i]])/sum(word_count)

}
which(signif(percentage_i, 3) == 0.5)
```

```
## [1] 253 254
```

```
which(signif(percentage_i, 3) == 0.5) / L
```

```
## [1] 0.03709677 0.03724340
```

```
## 253 (or 254) top words (top 3.7% of the total words) account for 50% of the total occurance

## e)
## Test Zipf's law
# compute the proportion of occurance to the most common
ordered_count <- word_count[order(word_count, decreasing = TRUE)]
proportion_i <- ordered_count / ordered_count[1]
# plot the word count proportion and y = 1/x function
# use log transformation -log(y) = log(x)
plot(x = seq(1, L), y = -log(proportion_i), type = "p", pch = 16, col = "blue", ylim = c(0, 10))
lines(x = seq(1, L), y = log(seq(1, L)), col = "red")
```



```
## It appears that our collection of documents loosely follow the Zipf's law in the first ~200 ranks.
```

Problem 2

```
# Compose A the link matrix
L <- matrix(rep(0, 100), 10, 10)
idx <- rbind(c(10, 1),
             c(3, 2), c(6, 2),
             c(4, 3), c(10, 3),
             c(9, 4),
             c(4, 5), c(10, 5),
             c(4, 6),
             c(5, 7), c(8, 7), c(10, 7),
             c(4, 8),
             c(8, 9),
             c(4, 10))
L[idx] <- 1

## Compute PageRank
my_pagerank <- function(L, d=0.85) {
  ##
  ## The function that calculate the PageRank vector
  ## for a given link matrix.
  ##
  ## Input:
  ## L, the link matrix
  ## d, numerical, the dampening parameter
  ##
  ## Output:
  ## p, the PageRank vector
  ##

  # compose the matrix to find the PageRank vector
  M <- diag(colSums(L))
  MInv <- solve(M)
  A <- L %*% MInv
  # build a strong conneted Markov Chain
  n <- nrow(A)
  E <- matrix(rep(1, n*n), n, n)
  A <- ((1 - d) / n) * E + d * A
  # find the eigenvector
  A_eig <- eigen(A)
  PR_vector <- A_eig$vectors[, which(signif(A_eig$values, 6) == 1)]
  # normalize
  p <- scale(PR_vector, center = FALSE,
                   scale = as.numeric(apply(as.matrix(PR_vector), 2, sum)))
  return(as.numeric(p))
}

my_pagerank(L, 0.85)
```

```
##  [1] 0.01500000 0.01500000 0.02137500 0.30971210 0.01925000 0.02137500
##  [7] 0.01500000 0.25576699 0.27825528 0.04926563
```

```
## Compute BrokenRank
my_pagerank(L, 1)
```

```
##  [1] 0.0000000 0.0000000 0.0000000 0.3333333 0.0000000 0.0000000 0.0000000
##  [8] 0.3333333 0.3333333 0.0000000
```

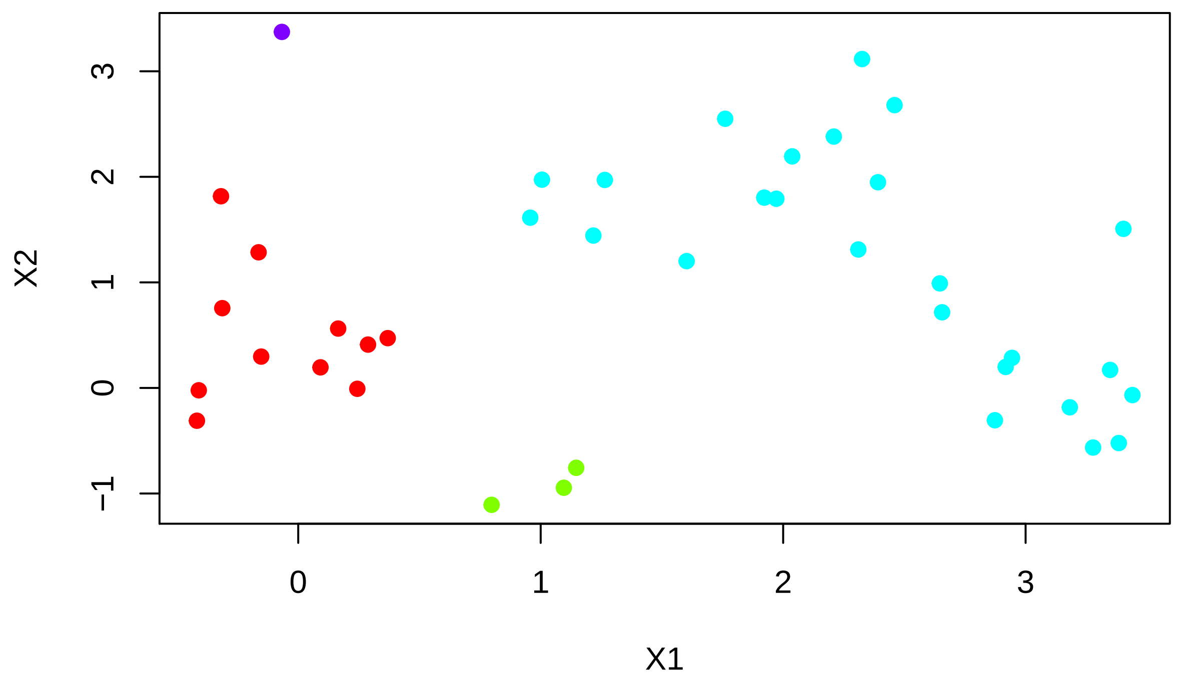
```
## BrokenRank gives us unreasonale results:
## we are only getting the ranks for page 4, 8, and 9.
```

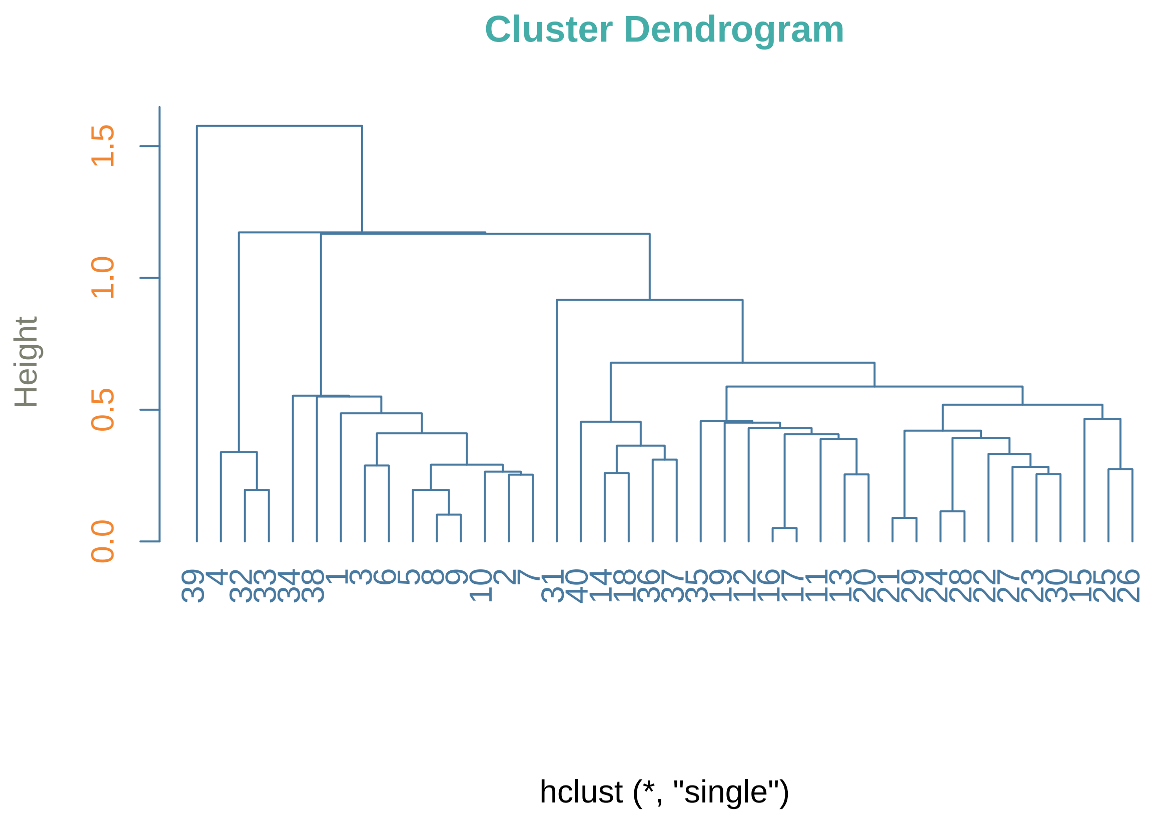
Problem 3

Problem 4

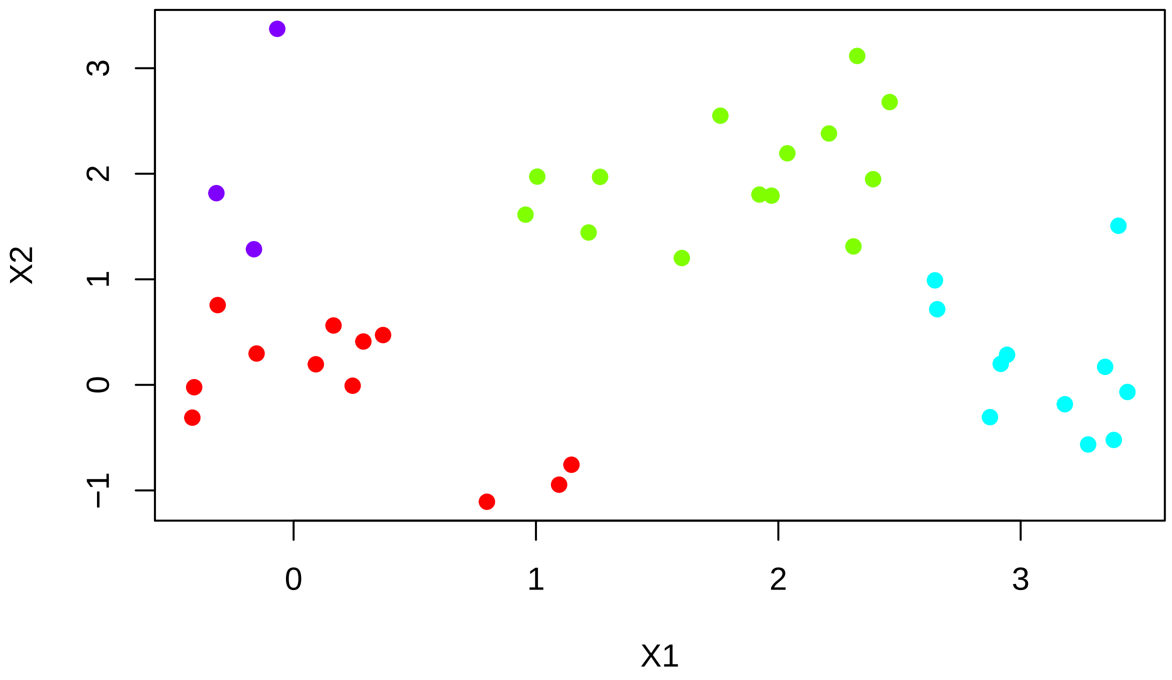
```
## a)
load("hw1prob3.Rdata")
# reference: ISLR 10.6.1
Cols <- function (vec) {
  cols=rainbow (length (unique (vec )))
  return (cols[as.numeric (as.factor (vec))])
}
```

```
hclust_and_plot <- function(x, d, method) {
  ##
  ## Input:
  ## x, the data matrix
  ## d, the pairwise Euclidean distance
  ## method, the hierarchical clustering linkage
  ##
  ## Returns:
  ## No returned object
  ##

  hc_4 <- hclust(d, method = method)
  # cut into 4 clusters
  c_labels <- cutree(hc_4, k = 4)
  # plot points in x
  plot(x[,1:2], col =Cols(c_labels), pch =19, xlab ="X1",ylab="X2")
  # plot dendorgram
  plot(hc_4, col = "#487AA1", col.main = "#45ADA8", col.lab = "#7C8071",
                    col.axis = "#F38630", hang = -1, xlab = "")

}
# run hierarchical agglomerative clustering with single linkage
hclust_and_plot(x, d, "single")
```

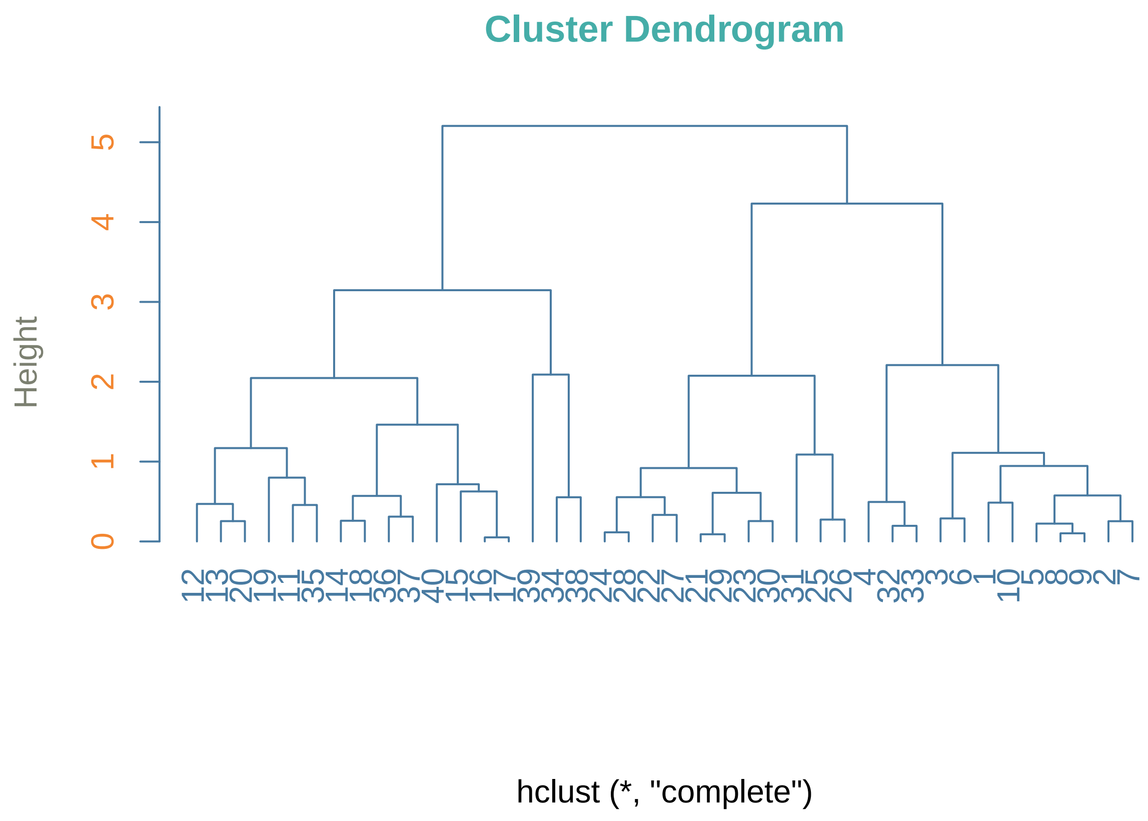




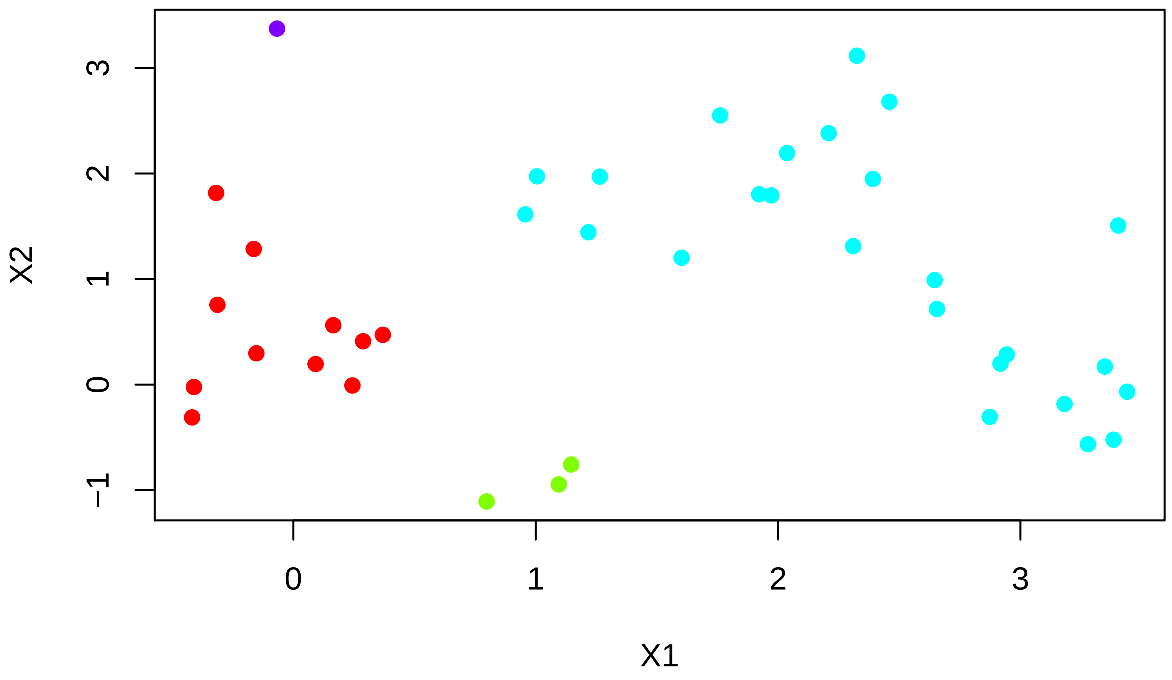
```
## b)
# repeat a for complete linkage
hclust_and_plot(x, d, "complete")
```

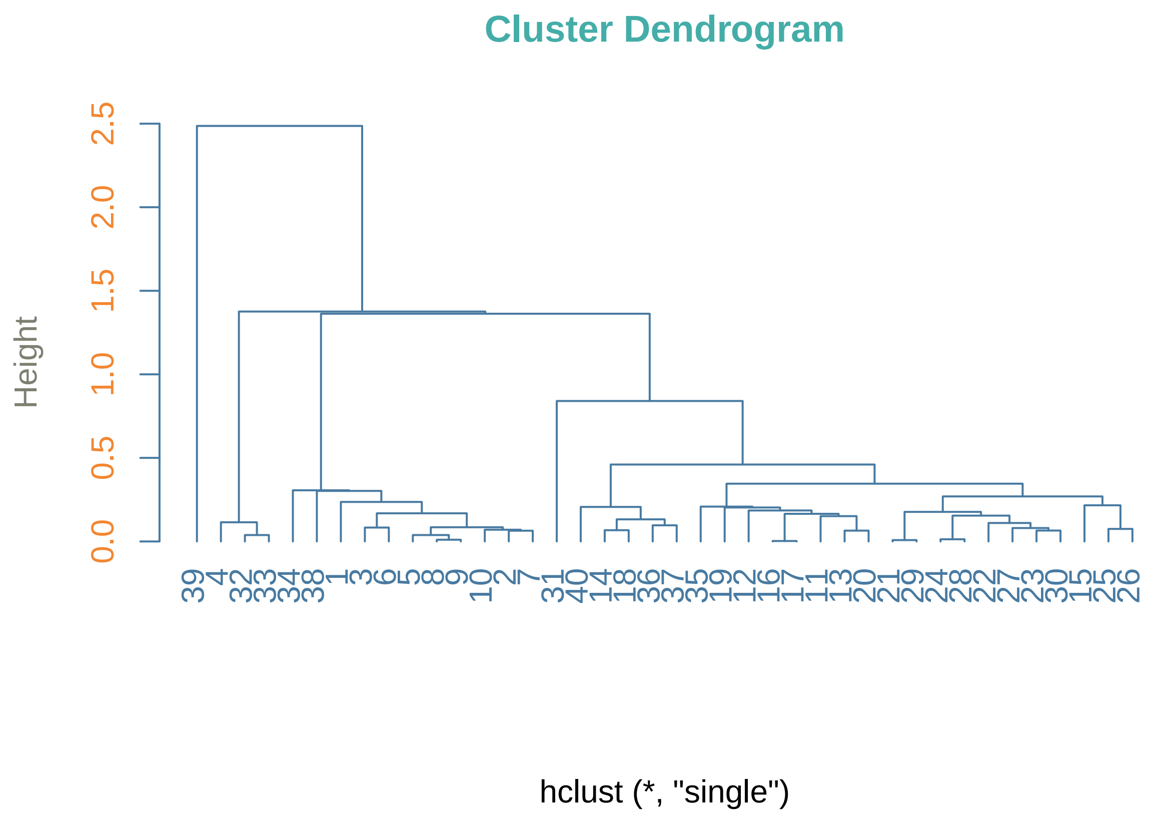


X2
X1
3
2
1
0
−1
0
1
2
3

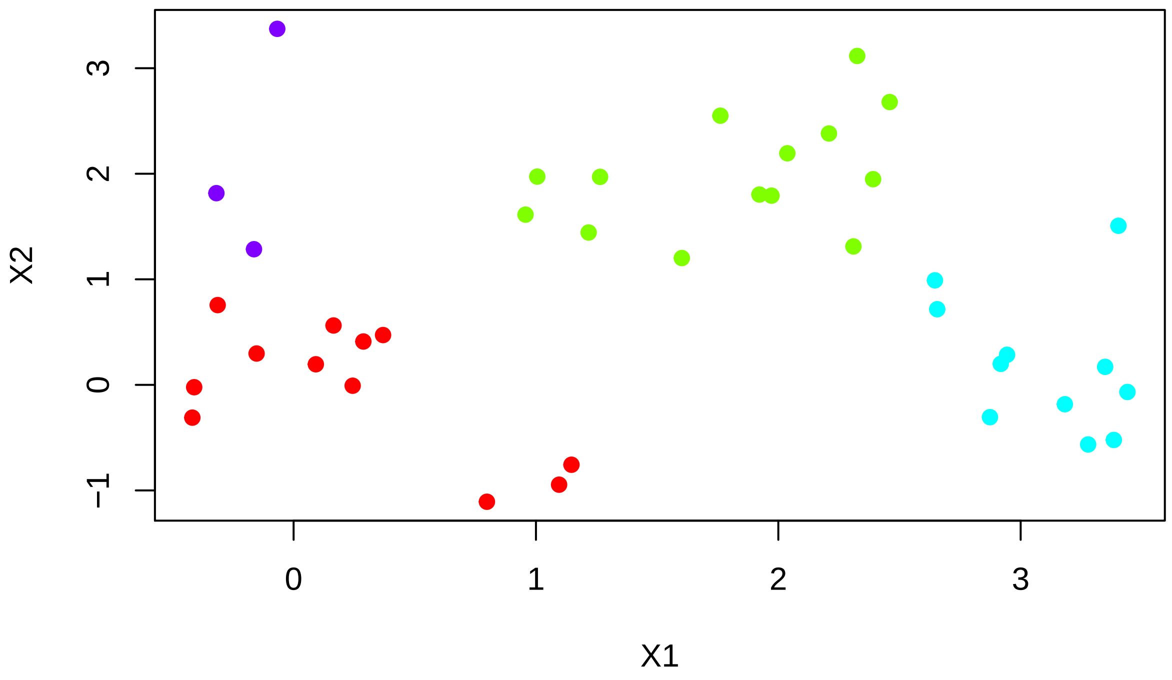


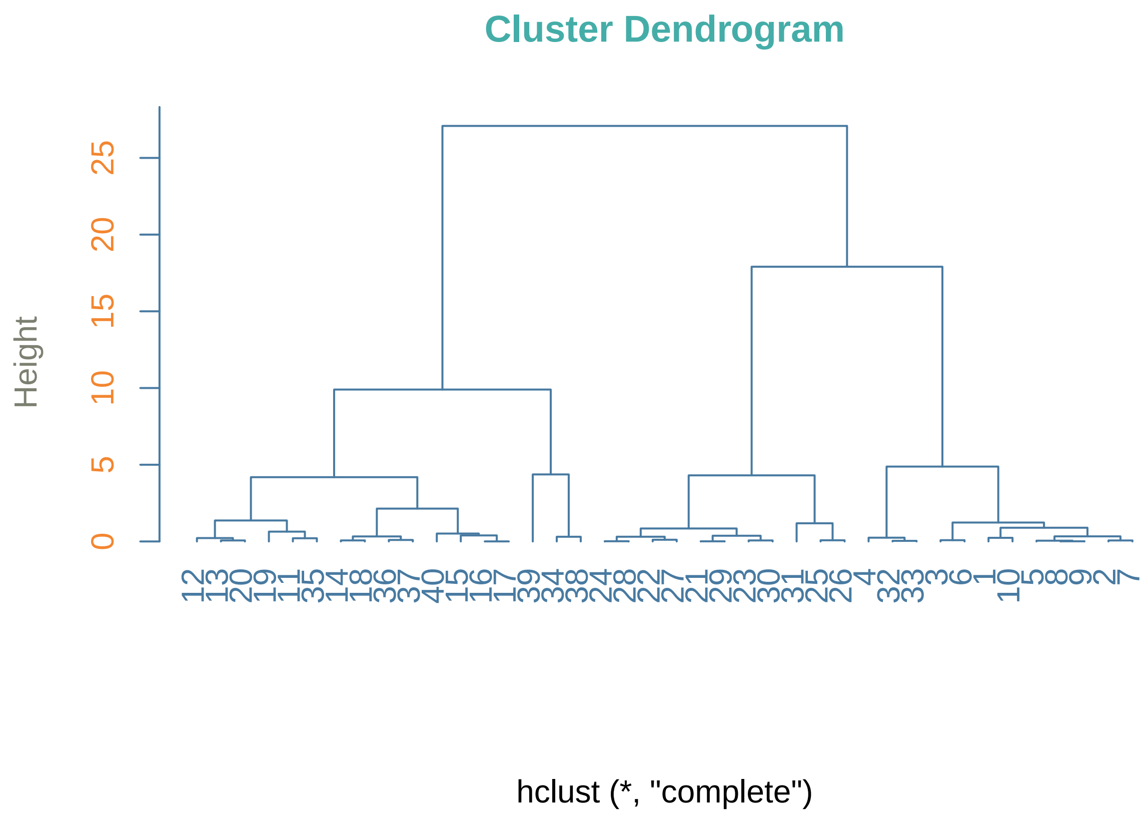
```
## c)
## Repeat a and b but passing d^2
hclust_and_plot(x, d^2, "single")
```


X2
X1
3
2
1
0
−1
0
1
2
3



```
hclust_and_plot(x, d^2, "complete")
```


X2
X1
3
2
1
0
−1
0
1
2
3



```
## A monotone increasing transformation does not change the clustering assignment
## or the dendrogram for single or complete linkage.
```

d)

At step $m \in 1, \cdots, N-1$, N = number of points:
Suppose we have clusters $1, \cdots, K_m$ at step m.
First, for $G, H \in 1, \cdots, K_m$, the cluster dissimilarities:

$$d_{single}(G,H) = \min_{g \in G, h \in H} h(d_{gh})$$

The pair of points g_m, h_m chosen to represent the cluster dissimilarites between G, H would be the same as the one before transformation:

$$g_m, h_m = \arg\min_{g \in G, h \in H} h(d_{gh}) = \arg\min_{g \in G, h \in H} d_{gh}$$

Then, the pair of clusters I_m, J_m that has smallest simliarity:

$$I, J = \arg\min_{I, J \in 1, \cdots, K_m} d_{single}(I,J) = \arg\min_{I, J \in 1, \cdots, K_m} h(d_{i_m, j_m}) = \arg\min_{I, J \in 1, \cdots, K_m} d_{i_m, j_m}$$

So at step m, the same pair of clusters I_m, J_m would be chosen to merge as prior to the transformation. Since every step we make the same merging choice, the result will be the same.

For complete linkage, the idea is the same: At step m, the pair of points g_m, h_m chosen to represent the cluster dissimilarites between G, H would be the same as the one before transformation:

$$g_m, h_m = \underset{g \in G, h \in H}{\arg\max} h(d_{gh}) = \underset{g \in G, h \in H}{\arg\max} d_{gh}$$

The rest of the proof is the same as the pair of clusters I_m, J_m is also chosen based on the smallest dissimilairity.