Stat 154 HW8

Mokssh Surve

4/19/2020

General

```
set.seed(12345)
library('geosphere')
library('cluster')
library('FactoMineR')
library('factoextra')
```

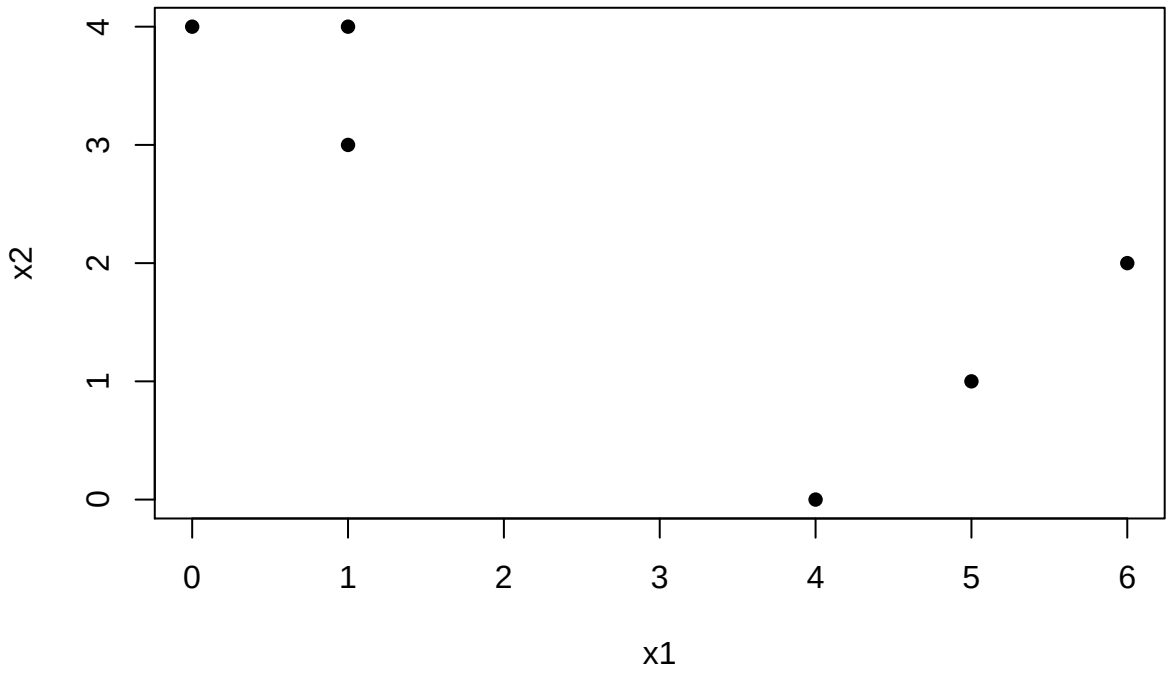
```
## Loading required package: ggplot2
```

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

```
library('cluster')
library('matrixStats')

#Note: each p1,p2 are vectors of length 2 (x,y) point coordinates
euclidean <- function(p1, p2){

  return(sqrt( sum((p1-p2)^2) ))

}
```

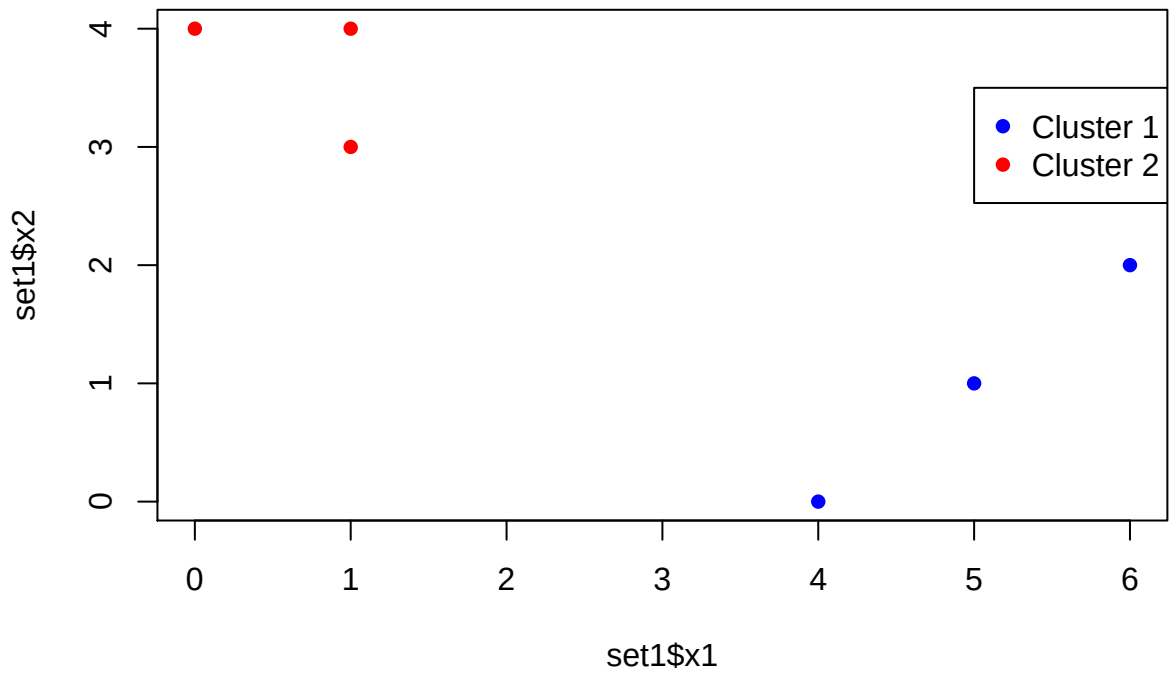
Q1) K-Means

```
## a)
K <- 2
x1 <- c(1,1,0,5,6,4)
x2 <- c(4,3,4,1,2,0)
set1 <- as.data.frame(cbind(x1,x2))
plot(set1, pch = 16)
```




```r
## b)
n <- nrow(set1)
rand_cluster <- sample(rep(1:K, 3), size=n)
set1$cluster_old <- rand_cluster
cluster_1 <- set1[which(set1$cluster_old == 1), ]
cluster_2 <- set1[which(set1$cluster_old == 2), ]

## c)
centroid_1 <- as.vector(centroid(cluster_1[, 1:2]))
centroid_2 <- as.vector(centroid(cluster_2[, 1:2]))

## d) #asssigning each observation to closest cluster
set1$cluster_new <- c(0,0,0,0,0,0)
for (i in 1:n){

  dist_1 <- euclidean(as.numeric(set1[i, 1:2]), centroid_1)
  dist_2 <- euclidean(as.numeric(set1[i, 1:2]), centroid_2)

  if(dist_1 < dist_2){
    set1$cluster_new[i] <- 1
  }
  else{
    set1$cluster_new[i] <- 2
  }
}

## e) repeat till the clusters don't change
while (any(set1$cluster_old != set1$cluster_new)){

  set1$cluster_old <- set1$cluster_new
  cluster_1 <- set1[which(set1$cluster_old == 1), ]
  cluster_2 <- set1[which(set1$cluster_old == 2), ]
```

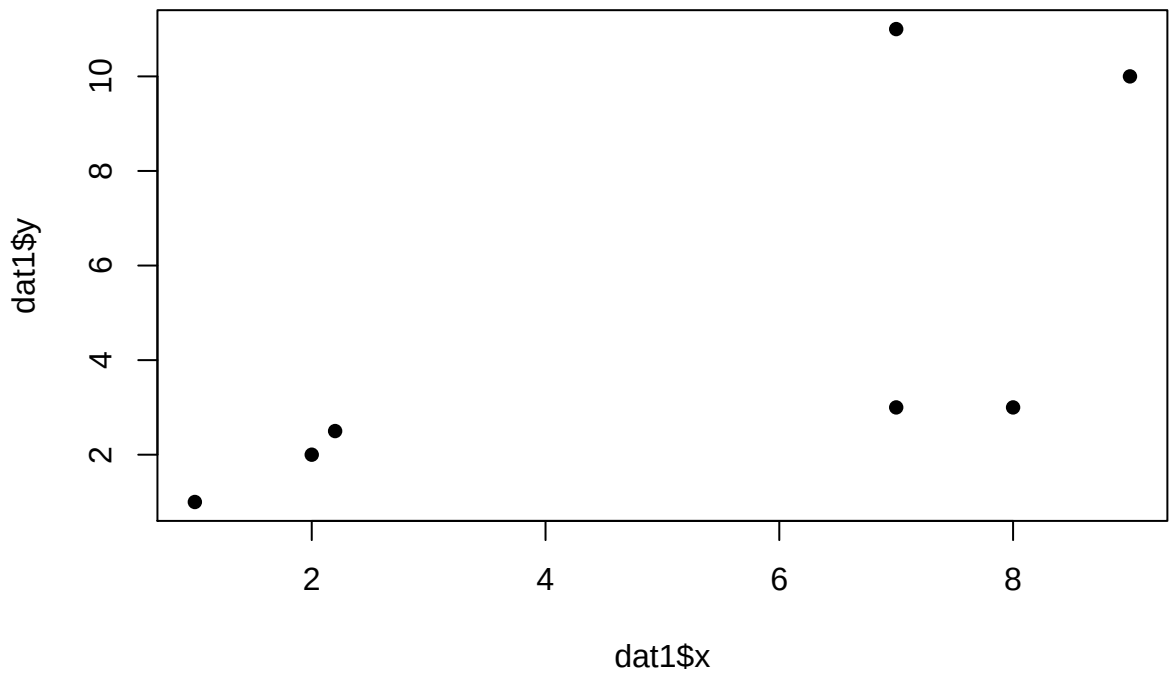
```
  centroid_1 <- as.vector(centroid(cluster_1[, 1:2]))
  centroid_2 <- as.vector(centroid(cluster_2[, 1:2]))

  for (i in 1:n){
    dist_1 <- euclidean(as.numeric(set1[i, 1:2]), centroid_1)
    dist_2 <- euclidean(as.numeric(set1[i, 1:2]), centroid_2)

    if(dist_1 < dist_2){
      set1$cluster_new[i] <- 1
    }
    else{
      set1$cluster_new[i] <- 2
    }
  }

}

# f)
set1$color <- rep('white', 6)
set1$color[which(set1$cluster_new == 1)] <- 'blue'
set1$color[which(set1$cluster_new == 2)] <- 'red'

plot(set1$x1, set1$x2, col = set1$color, pch = 16)
legend(5, 3.5, col=c('blue', 'red'), legend = c('Cluster 1', 'Cluster 2'), pch = 16)
```



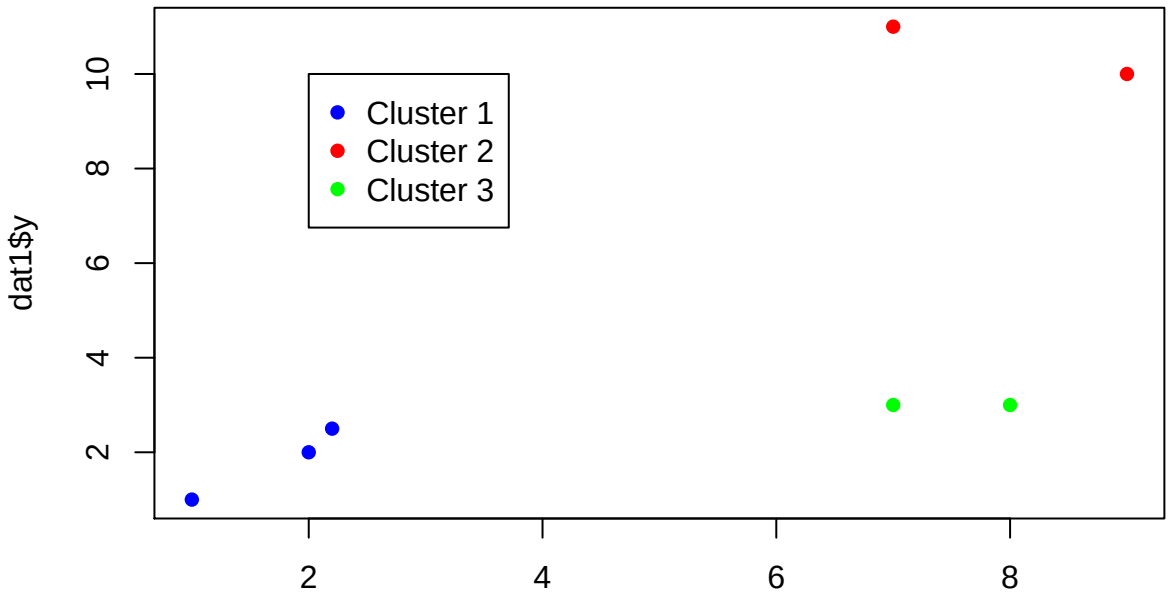
Q2) More K-Means

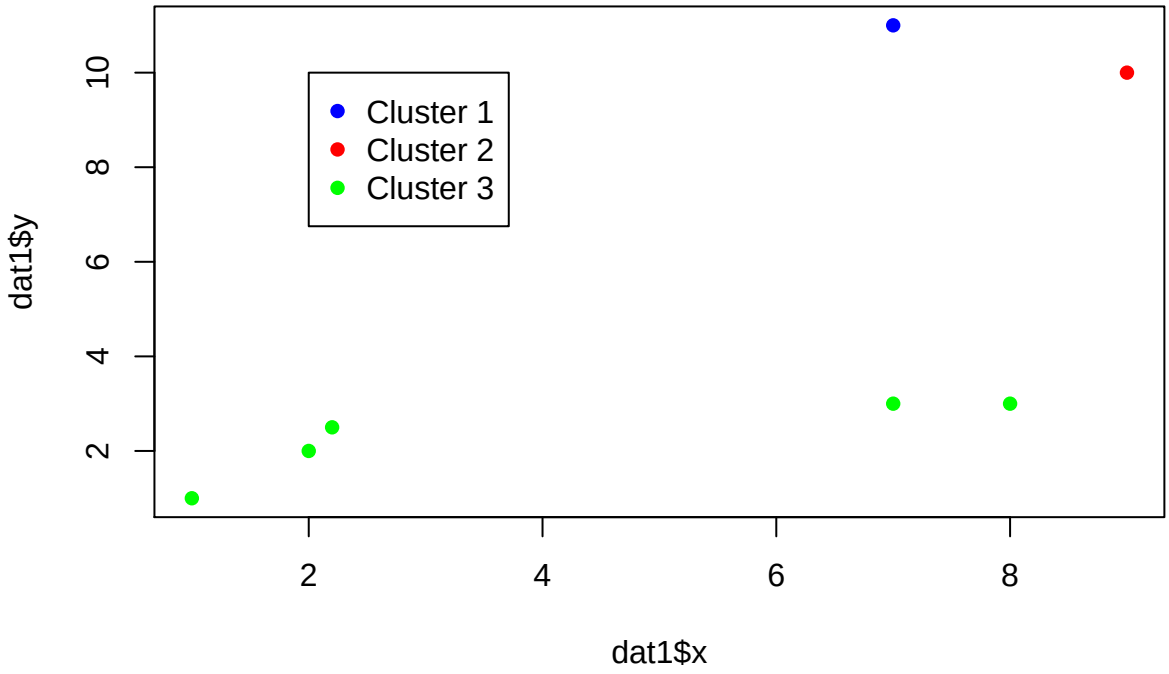
```
dat1 <- read.csv('/Users/moksshsurve/Desktop/Semesters/SPRING/Spring\ 20/Stat\ 154/Homeworks/HW\ 8/data
n <- nrow(dat1)
```

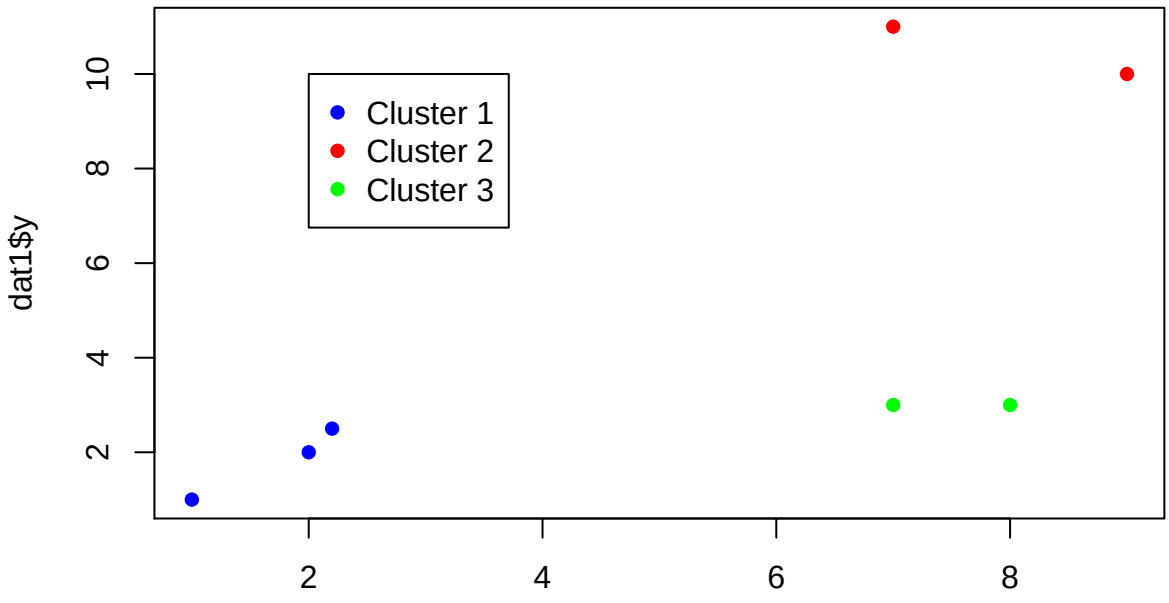
```r
dat1$color <- rep('white', n)

#a
plot(dat1$x, dat1$y, pch = 16)
```

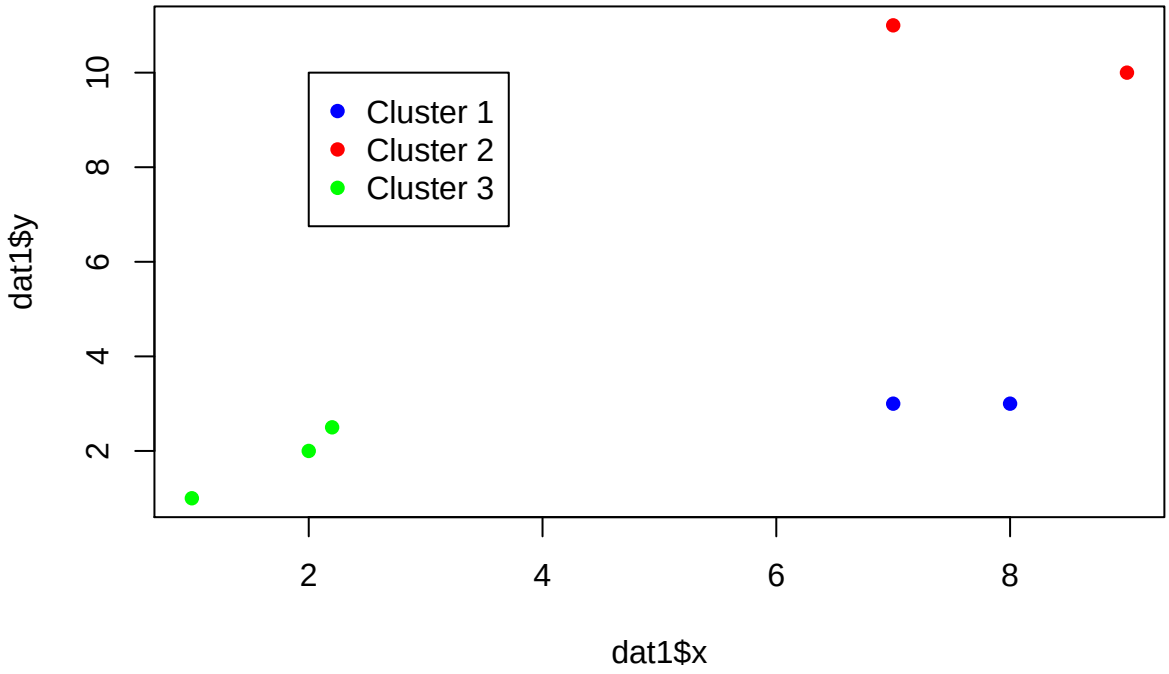


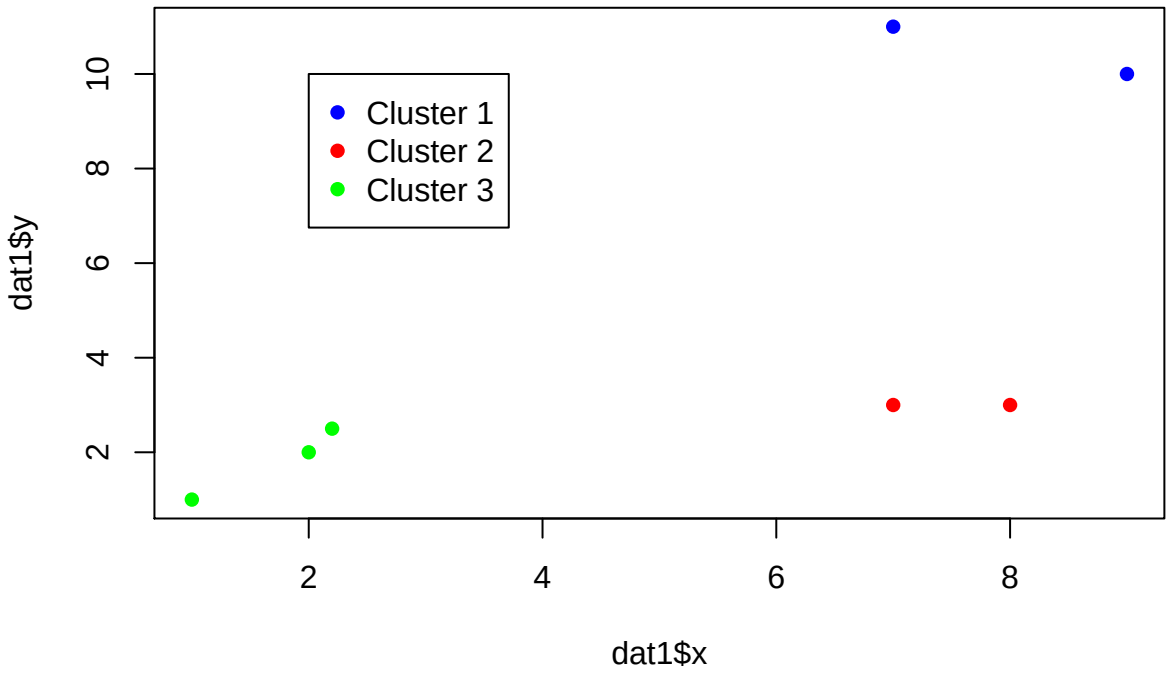

```r
#b, #c
for (i in 1:6){

  k1 <- kmeans(dat1[, 2:3], 3)
  dat1$cluster <- as.vector(k1$cluster)
  dat1$color[which(dat1$cluster == 1)] <- 'blue'
  dat1$color[which(dat1$cluster == 2)] <- 'red'
  dat1$color[which(dat1$cluster == 3)] <- 'green'
  plot(dat1$x, dat1$y, col = dat1$color, pch = 16)
  legend(2, 10, col=c('blue', 'red', 'green'), legend = c('Cluster 1', 'Cluster 2', 'Cluster 3'), pch =
}
```


Cluster 1
Cluster 2
Cluster 3
dat1$y
dat1$x
2
4
6
8
10


Cluster 1
Cluster 2
Cluster 3
dat1$y
dat1$x
2
4
6
8
10


Cluster 1
Cluster 2
Cluster 3
dat1$y
dat1$x
2
4
6
8
2
4
6
8
10


Cluster 1
Cluster 2
Cluster 3
dat1$y
dat1$x
2
4
6
8
2
4
6
8
10



###

Q2c) It is notable to mention that 5 of the 6 iteration of the kmeans() algorithm and plot yielded the same *"global optimum result"*. However, there is one clutering iteration that split the observations as 5 in 1 cluster, and 1 in the remaining 2 clusters each. This is clearly not an optimal result. The possible reasons for this are alluded to below:
The **kmeans()** function in R allocates a **random set of 'k' centroids** each time it is called (unless the centroids are specified). Thus, each time, when the k-means algorithm starts from scratch from a random set of 3 centres, the clustering decisions can vary. Although the algorithm WILL generate a minimum, it **will be the local minimum, and need not always be the global minimum** of the distance between all the points and their respective clustering centres.

Q3) Comparing Agglomerative Hierarchical Clustering Methods

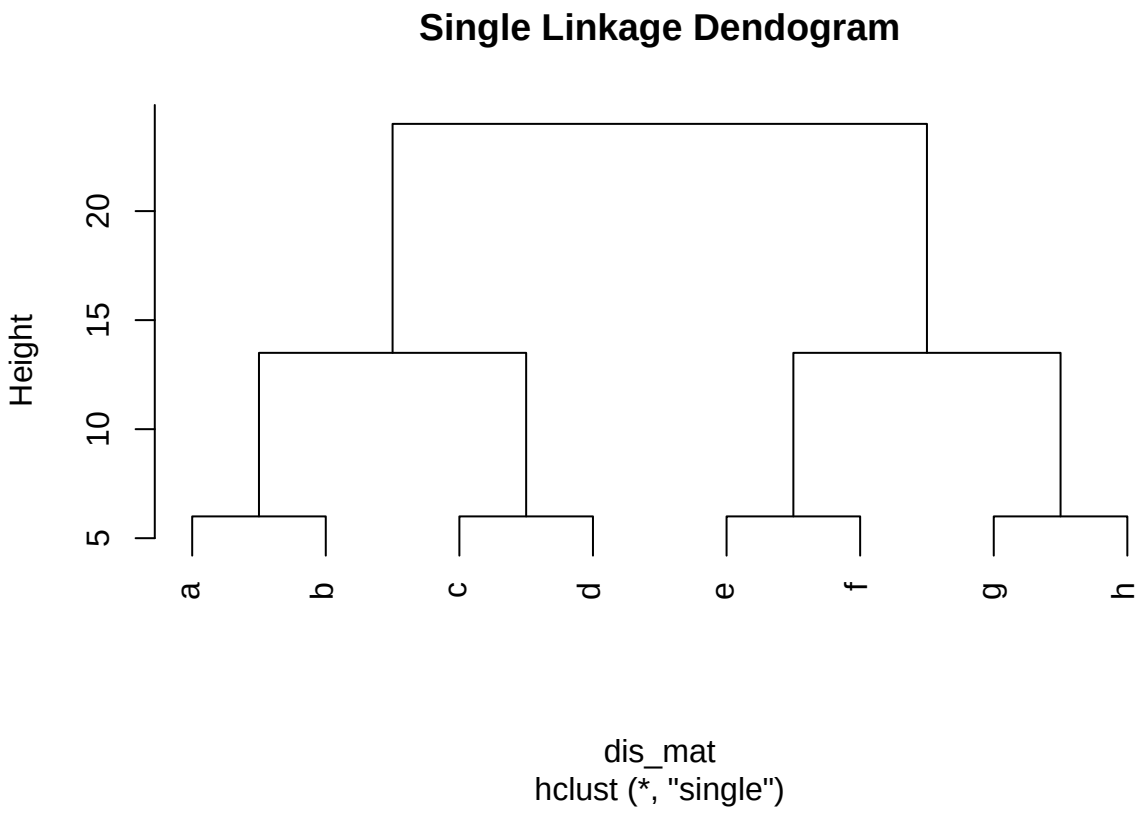
```
# a)
dat2 <- read.csv('/Users/moksshsurve/Desktop/Semesters/SPRING/Spring\ 20/Stat\ 154/Homeworks/HW\ 8/data1
row.names(dat2) <- dat2[, 1]
dis_mat <- dist(dat2)^2
```

```
## Warning in dist(dat2): NAs introduced by coercion
```

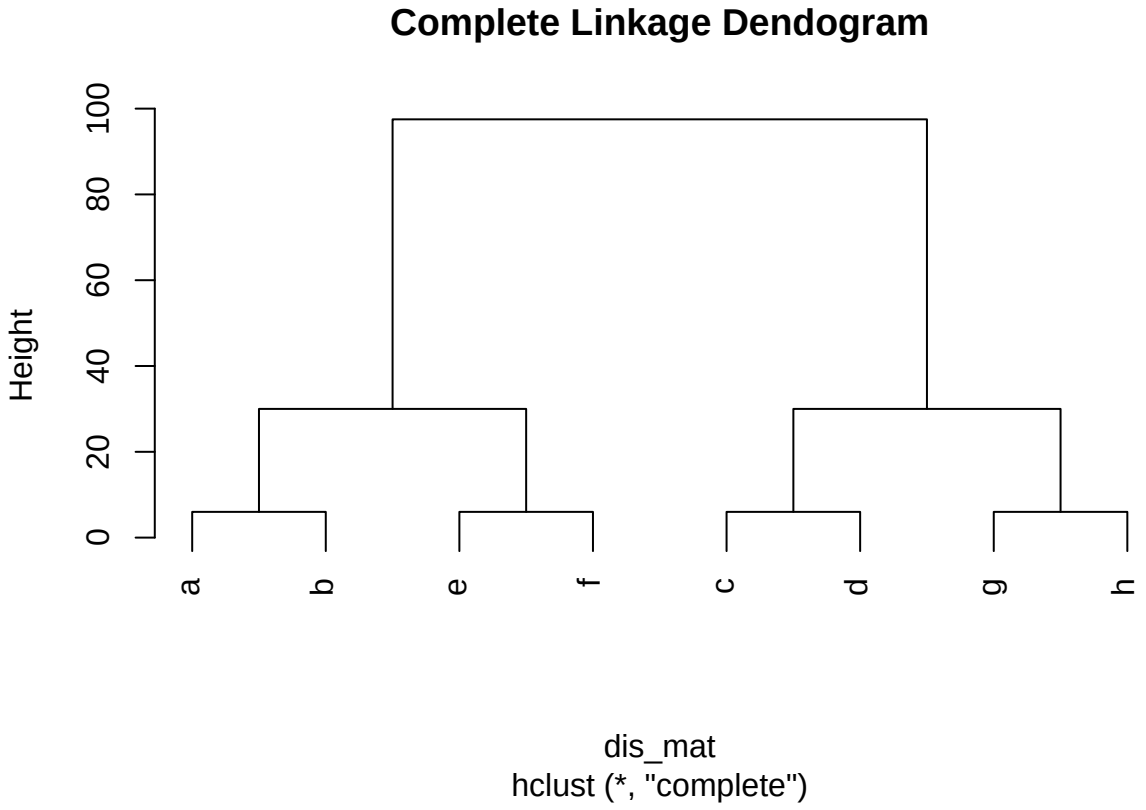
```
dis_mat
```

```
##      a    b    c    d    e    f    g
## b  6.0
## c 37.5 13.5
## d 73.5 37.5  6.0
## e 24.0 30.0 61.5 97.5
## f 30.0 24.0 37.5 61.5  6.0
## g 61.5 37.5 24.0 30.0 37.5 13.5
## h 97.5 61.5 30.0 24.0 73.5 37.5  6.0
```

```r
# b)
hc.single <- hclust(dis_mat, method = 'single')
plot(hc.single, main='Single Linkage Dendogram')
```




```r
# c)
hc.complete <- hclust(dis_mat, method = 'complete')
plot(hc.complete, main='Complete Linkage Dendogram')
```



3d)

If we cut the Single Linkage Cluster Dendogram such that 2 clusters exist, Observations {a,b,c,d} form the 1st cluster, and observations {e,f,g,h} form the 2nd cluster:

- **Cluster 1: {a,b,c,d}**
- **Cluster 2: {e,f,g,h}**

3e)

If we cut the Complete Linkage Cluster Dendogram such that 2 clusters exist, Observations {a,b,e,f} form the 1st cluster, and observations {c,d,g,h} form the 2nd cluster:

- **Cluster 1: {a,b,e,f}**
- **Cluster 2: {c,d,g,h}**

3f)

No, both the results obtained in part c) and part d) are **not similar**. This is intrinsically the case owing to the **verying aggregating methods** adopted by the single and complete linkage algorithms for hierarchical clustering as listed below:

- Single Linkage treats the distance between an already formed cluster to a point as:

$\mathbf{d^2(ab, c) = min[d^2(a, c), d^2(b, c)]}$

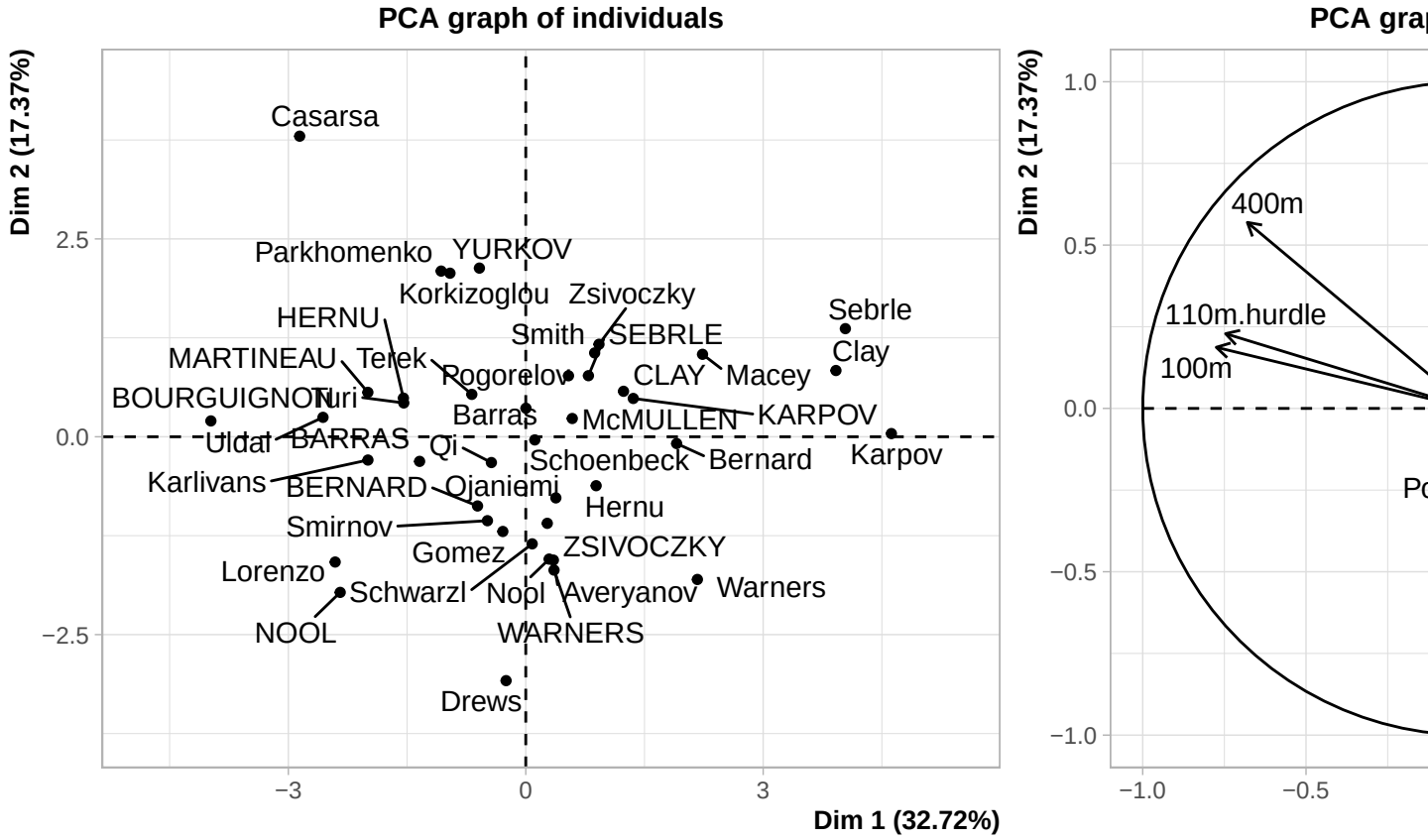
- Complete Linkage treats the distance between an already formed cluster to a point as:

$\mathbf{d^2(ab, c) = max[d^2(a, c), d^2(b, c)]}$

Note: ab refers to an already formed cluster of points a and b, and c refers to a point not in the ac cluster

Q4) Clustering Variables

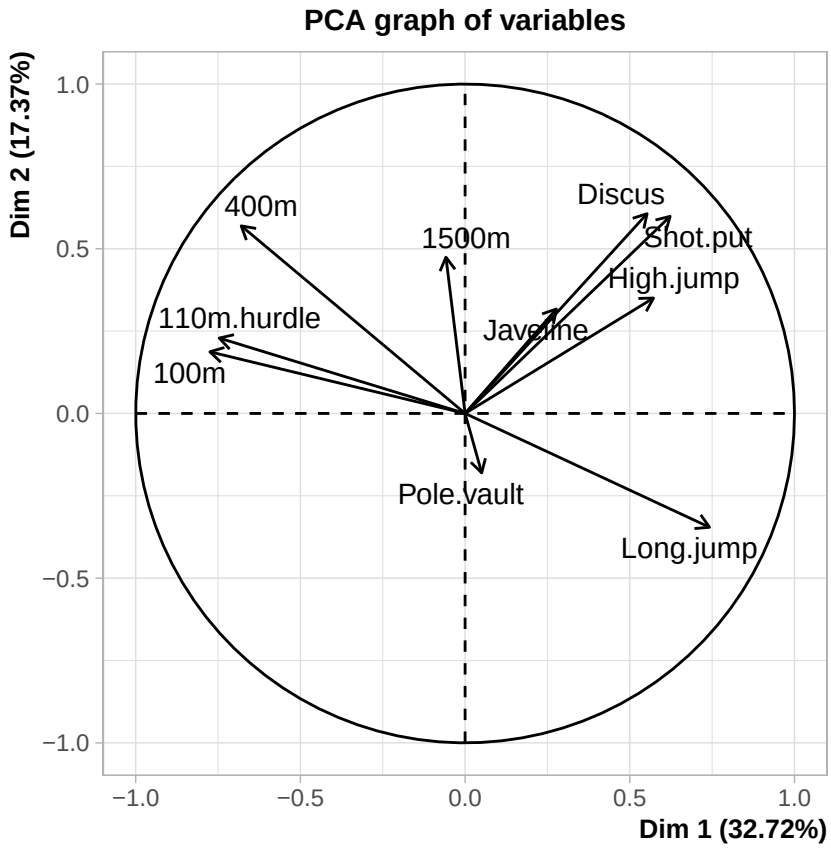
```
data("decathlon")
dec <- decathlon[, 1:10]
dec.pca <- PCA(dec, scale.unit = TRUE, ncp=2)
```




```
### PS: Ignore these plots below please - some glitch wasn't letting me get them off.
```

Q4.1) PCA and Circle of Correlations

```
plot(dec.pca, choix='varcor')
```



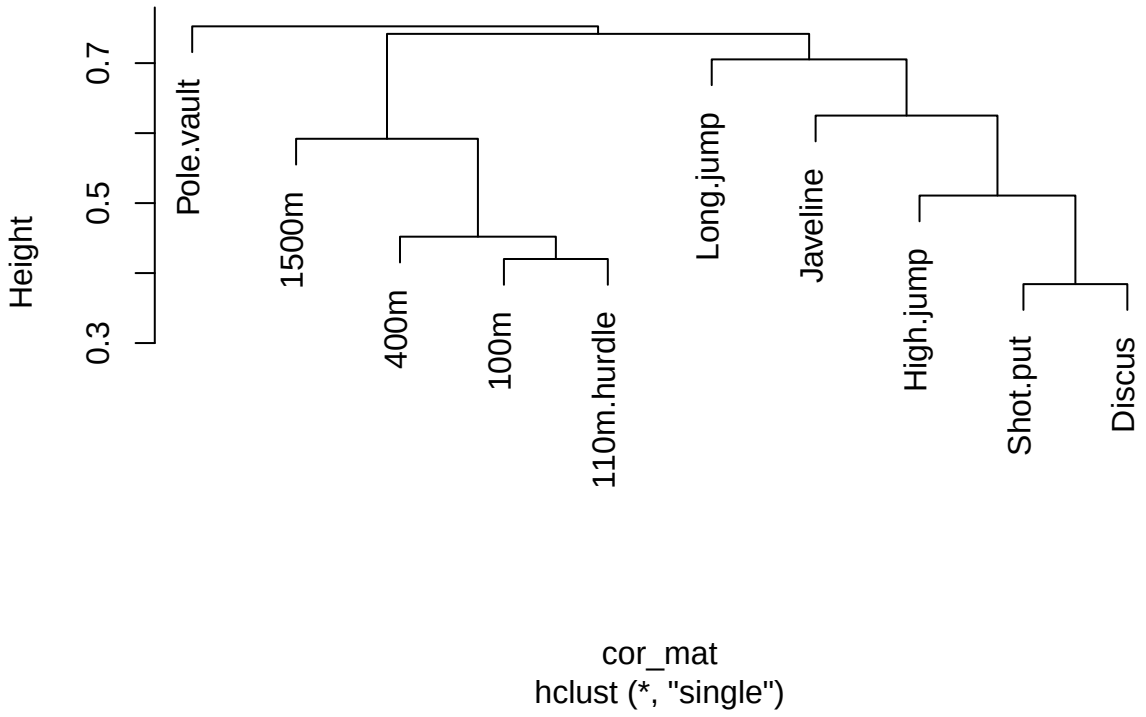
Visaully examining the Circle of correlations on the 2 most prominent dimensions intuitively gives us **3 clusters**:

- **Cluster 1:** 400m; 110m Hurdle; 100m; 1500m
- **Cluster 2:** Discus; Shot Put; High Jump; Javeline
- **Cluster 3:** Pole Vault; Long Jump

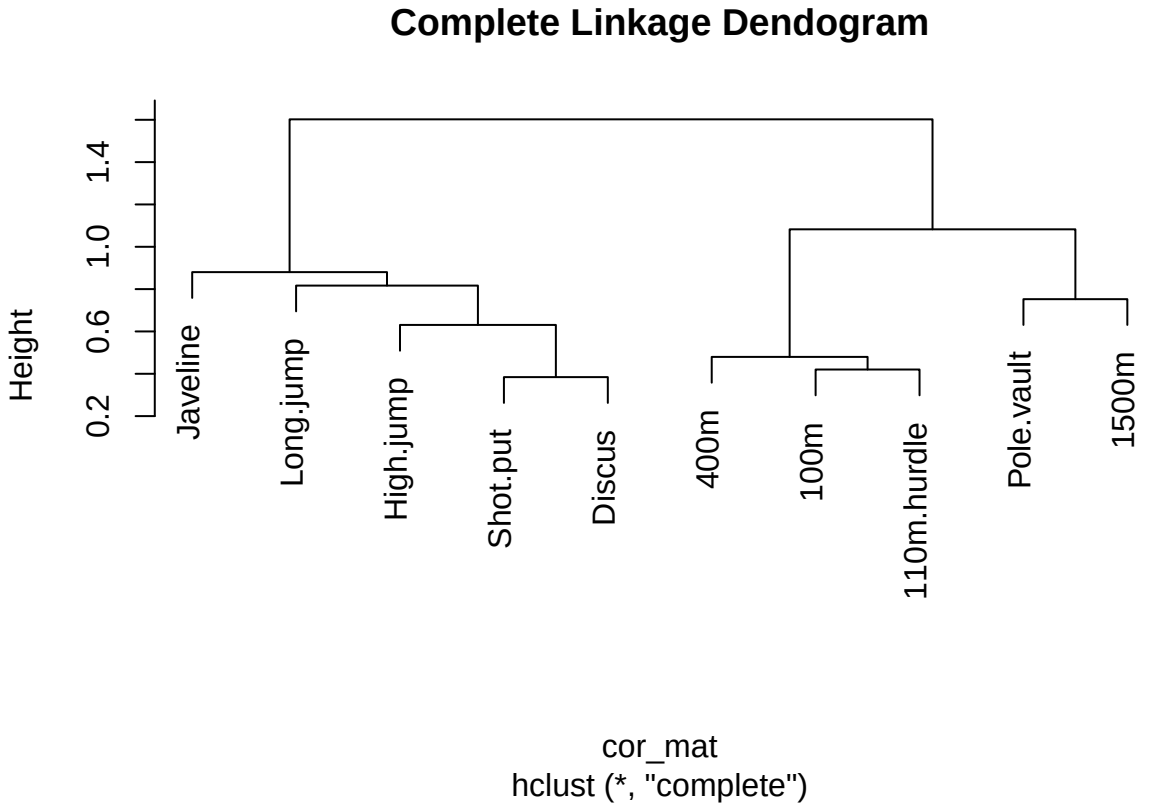
Cluster 1 can be qualitatively be clustered as **track events**, **Cluster 2** can be qualitatively be clustered as **field events**, & **Cluster 3** can be qualitatively be clustered as **field events to do something with the participants' heights.** It should be noted that the variables in Cluster 3 is a little more vaguely similar to each other.

4.2) Distances based on correlations

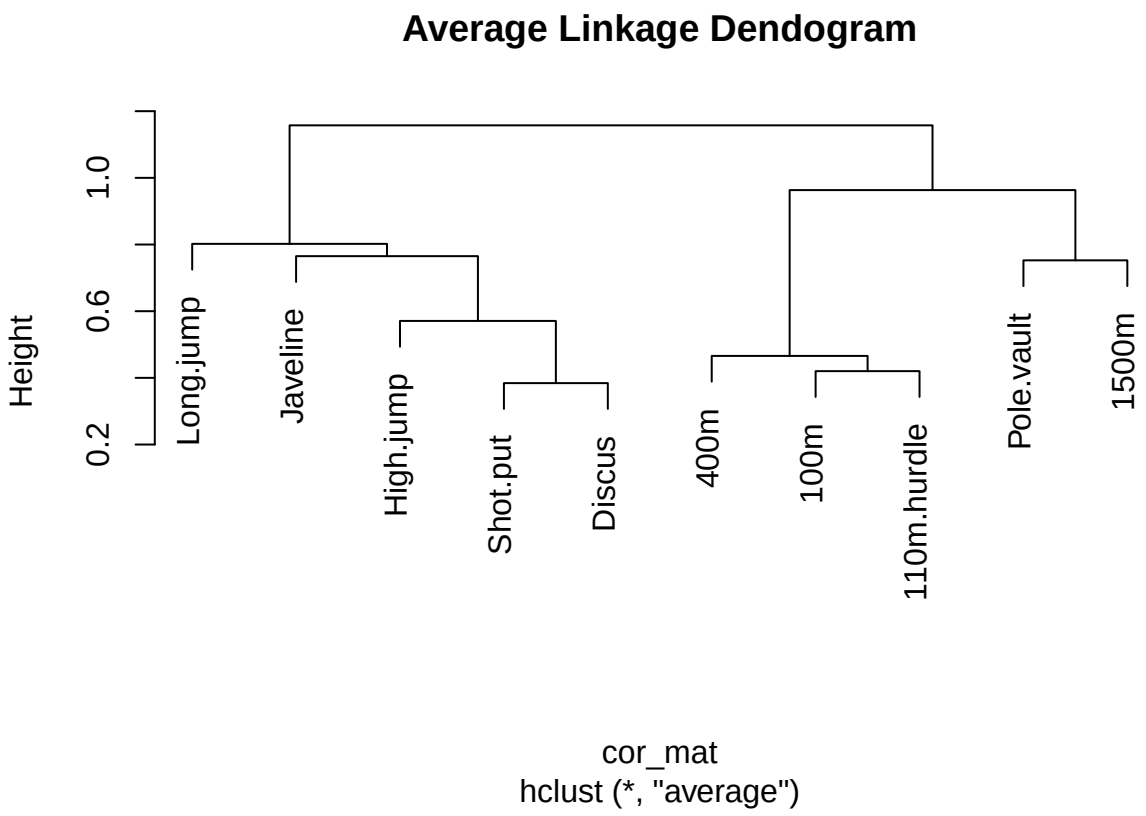
```
#a,b
cor_mat <- as.dist(1 - cor(dec, method = 'pearson'))

#c
hc.single <- hclust(cor_mat, method = 'single')
plot(hc.single, main = 'Single Linkage Dendogram')
```




```
#d
hc.complete <- hclust(cor_mat, method = 'complete')
plot(hc.complete, main = 'Complete Linkage Dendogram')
```

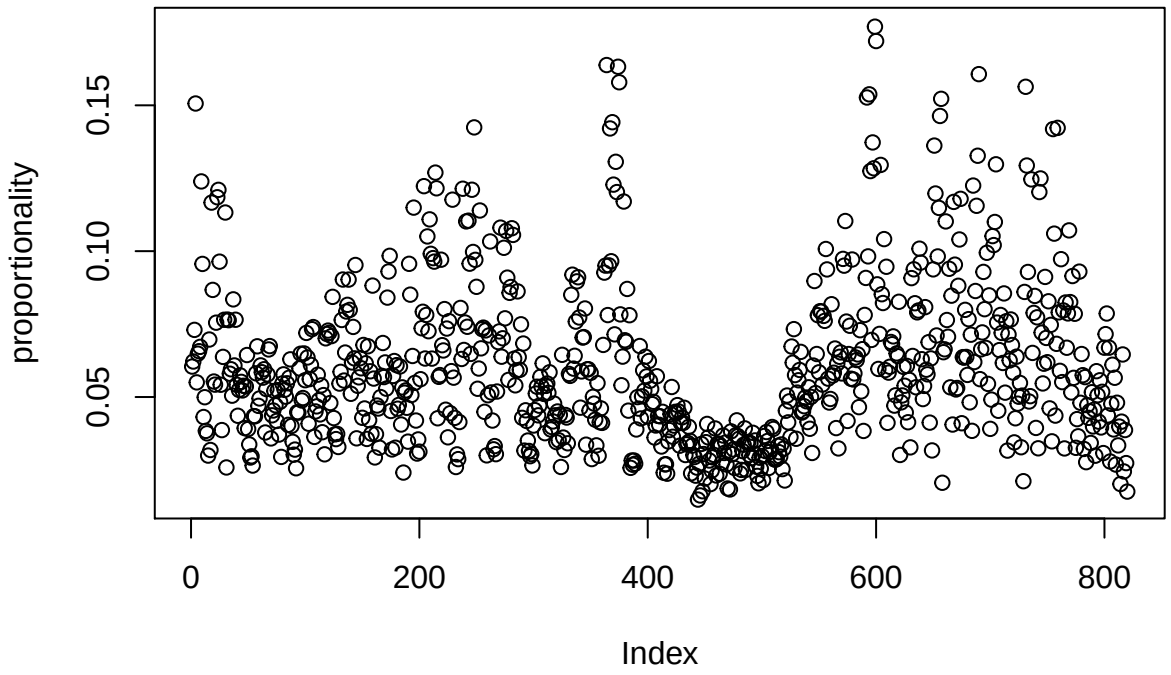



```
#e
hc.average <- hclust(cor_mat, method = 'average')
plot(hc.average, main = 'Average Linkage Dendogram')
```



Q5) Squared Euclidean distance, and Correlation-based distance

```
#a
scaled_dec <- scale(dec)
scaled_cor_mat <- as.dist( 1 - cor(t(scaled_dec), method = 'pearson'))
scaled_euclidean_dist <- (dist(scaled_dec))^2
proportionality <- scaled_cor_mat / scaled_euclidean_dist
plot(proportionality)
```

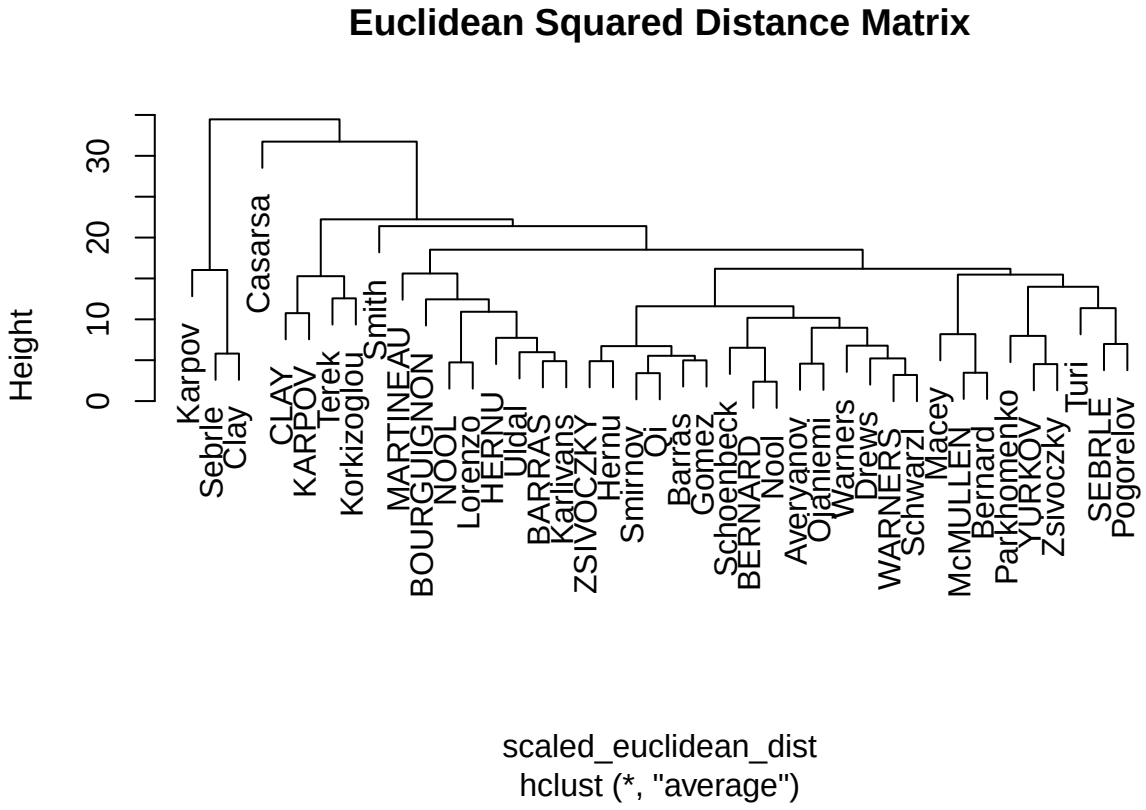



```
summary(proportionality)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.01485 0.03972 0.05493 0.06011 0.07293 0.17695
```

As is evident from the scatter plot and summary, there seems to be some proportionality around the value of 0.05 with some variation involved. It should be noted that variations are bound to be present since this is not supposed to be a perfect statistical relationship.
Note: This could be why there are mild discrepancies in the dendograms generated in part b) below.

```
#b
hc.avg.euclidean <- hclust(scaled_euclidean_dist, method = 'average')
plot(hc.avg.euclidean, main = 'Euclidean Squared Distance Matrix')
```




```r
hc.avg.cor <- hclust(scaled_cor_mat, method = 'average')
plot(hc.avg.cor, main = 'Correlation Based Distance Matrix')
```