harinris_Homework1_lab

Harin Rishabh

2024-02-09

12.5 Lab: Unsupervised Learning

12.5.1 Principal Components Analysis

We are using the USArrests Dataset. We are calculating mean and variance.

```
states <- row.names(USArrests)
apply(USArrests , 2, mean)
```

```
##   Murder  Assault UrbanPop     Rape
##    7.788  170.760   65.540   21.232
```

```
apply(USArrests , 2, var)
```

```
##     Murder    Assault   UrbanPop       Rape
##   18.97047 6945.16571  209.51878   87.72916
```

We now perform principal components analysis using the prcomp() function. pr.out$rotation contains the corresponding principal component loading vector. Then we plot the first two principal components.

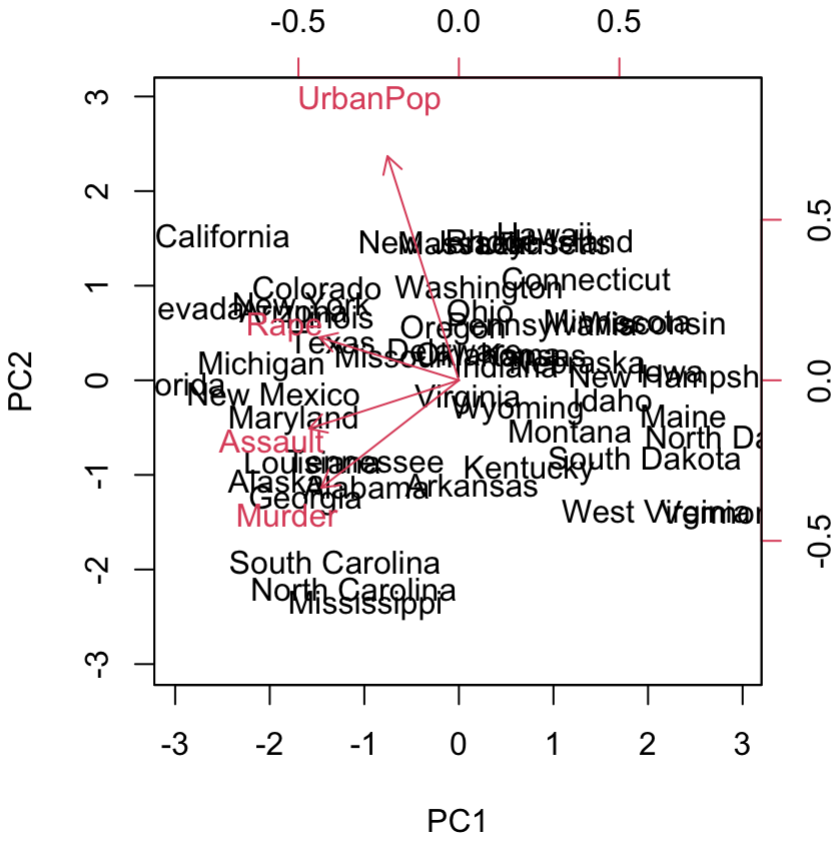
```
pr.out <- prcomp(USArrests , scale = TRUE)
names(pr.out)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

```
pr.out$rotation
```

```
##                 PC1        PC2        PC3         PC4
## Murder   -0.5358995 -0.4181809  0.3412327  0.64922780
## Assault  -0.5831836 -0.1879856  0.2681484 -0.74340748
## UrbanPop -0.2781909  0.8728062  0.3780158  0.13387773
## Rape     -0.5434321  0.1673186 -0.8177779  0.08902432
```

```
biplot(pr.out , scale = 0)
```



Calculating STD and var of each principal component.

```
pr.out$sdev
```

```
## [1] 1.5748783 0.9948694 0.5971291 0.4164494
```

```
pr.var <- pr.out$sdev^2
pr.var
```

```
## [1] 2.4802416 0.9897652 0.3565632 0.1734301
```

Computing the proportion of variance explained by each principal component. We see that the first principal component explains 62.0% of the variance in the data, the next principal component explains 24.7% of the variance, etc.

```
pve <- pr.var / sum(pr.var)
pve
```

```
## [1] 0.62006039 0.24744129 0.08914080 0.04335752
```

Plotting the PVE explained by each component, as well as the cumulative PVE.

```
plot(pve , xlab = "Principal Component", ylab = "Proportion of Variance Explained", y
lim = c(0, 1), type = "b")
plot(cumsum(pve), xlab = "Principal Component", ylab = "Cumulative Proportion of Vari
ance Explained", ylim = c(0, 1), type = "b")
```

12.5.2 Matrix Completion

Turning the data frame into a matrix, after centering and scaling each column to have mean zero and variance one.

```
X <- data.matrix(scale(USArrests))
pcob <- prcomp(X)
summary(pcob)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4
## Standard deviation     1.5749 0.9949 0.59713 0.41645
## Proportion of Variance 0.6201 0.2474 0.08914 0.04336
## Cumulative Proportion  0.6201 0.8675 0.95664 1.00000
```

Using SVD

```
sX <- svd(X)
names(sX)
```

```
## [1] "d" "u" "v"
```

```
round(sX$v, 3)
```

```
##        [,1]   [,2]   [,3]   [,4]
## [1,] -0.536 -0.418  0.341  0.649
## [2,] -0.583 -0.188  0.268 -0.743
## [3,] -0.278  0.873  0.378  0.134
## [4,] -0.543  0.167 -0.818  0.089
```

```
pcob$rotation
```

```
##                 PC1        PC2        PC3         PC4
## Murder   -0.5358995 -0.4181809  0.3412327  0.64922780
## Assault  -0.5831836 -0.1879856  0.2681484 -0.74340748
## UrbanPop -0.2781909  0.8728062  0.3780158  0.13387773
## Rape     -0.5434321  0.1673186 -0.8177779  0.08902432
```

We now omit 20 entries in the 50 × 2 data matrix at random. We do so by first selecting 20 rows (states) at random, and then selecting one of the four entries in each row at random. This ensures that every row has at least three observed values.

```
nomit <- 20
set.seed (15)
ina <- sample(seq (50) , nomit)
inb <- sample (1:4, nomit , replace = TRUE)
Xna <- X
index.na <- cbind(ina , inb)
Xna[index.na] <- NA
```

We first write a function that takes in a matrix, and returns an approximation to the matrix using the svd() function.

```
fit.svd <- function(X, M = 1) {
  svdob <-svd(X)
  with(svdob ,
  u[, 1:M, drop = FALSE] %*%
  (d[1:M] * t(v[, 1:M, drop = FALSE ])))
}
```

Step 1 of algorithm

```
Xhat <- Xna
xbar <- colMeans(Xna , na.rm = TRUE)
Xhat[index.na] <- xbar[inb]
```

Measuring progress of our iterations

```
thresh <- 1e-7
rel_err <- 1
iter <- 0
ismiss <- is.na(Xna)
mssold <- mean (( scale(Xna , xbar , FALSE)[!ismiss])^2)
mss0 <- mean(Xna[!ismiss ]^2)
```

Step 2

```
while(rel_err > thresh) {
  iter <- iter + 1
# Step 2(a)
  Xapp <- fit.svd(Xhat , M = 1)
# Step 2(b)
  Xhat [ ismiss ] <- Xapp [ ismiss ]
# Step 2(c)
  mss <- mean ((( Xna - Xapp)[! ismiss ])^2)
  rel_err <- ( mssold - mss ) / mss0
  mssold <- mss
  cat("Iter:", iter, "MSS:", mss,
  "Rel. Err:", rel_err, "\n")
}
```

```
## Iter: 1 MSS: 0.3821695 Rel. Err: 0.6194004
## Iter: 2 MSS: 0.3705046 Rel. Err: 0.01161265
## Iter: 3 MSS: 0.3692779 Rel. Err: 0.001221144
## Iter: 4 MSS: 0.3691229 Rel. Err: 0.0001543015
## Iter: 5 MSS: 0.3691008 Rel. Err: 2.199233e-05
## Iter: 6 MSS: 0.3690974 Rel. Err: 3.376005e-06
## Iter: 7 MSS: 0.3690969 Rel. Err: 5.465067e-07
## Iter: 8 MSS: 0.3690968 Rel. Err: 9.253082e-08
```

Finally, we compute the correlation between the 20 imputed values and the actual values.

```
cor(Xapp[ismiss], X[ismiss])
```

```
## [1] 0.6535043
```

12.5.3 Clustering

We begin with a simple simulated example in which there truly are two clusters in the data: the first 25 observations have a mean shift relative to the next 25 observations.

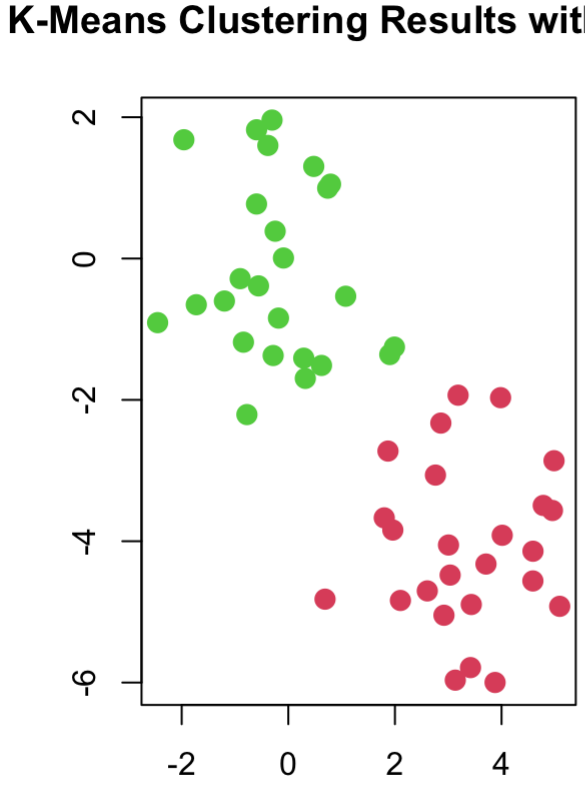
```
set.seed (2)
x <- matrix(rnorm (50 * 2), ncol = 2)
x[1:25, 1] <- x[1:25, 1] + 3
x[1:25, 2] <- x[1:25, 2] - 4
```

We now perform K-means clustering with K = 2.

```
km.out <- kmeans(x, 2, nstart = 20)
km.out$cluster
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
## [39] 2 2 2 2 2 2 2 2 2 2 2 2
```

```
par(mfrow = c(1, 2))
plot(x, col = (km.out$cluster + 1), main = "K-Means Clustering Results with K = 2", x
lab = "", ylab = "", pch = 20, cex = 2)
```

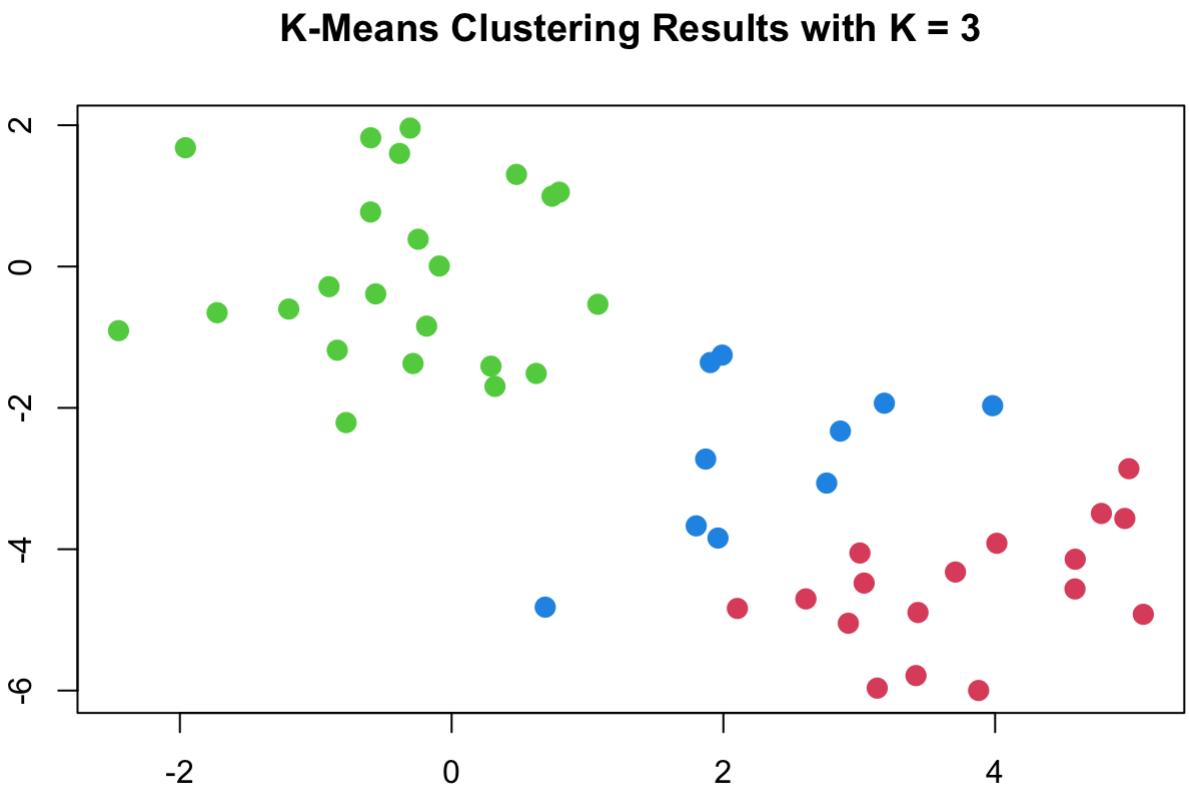


Clustering with k=3.

```
set.seed (4)
km.out <- kmeans(x, 3, nstart = 20)
km.out
```

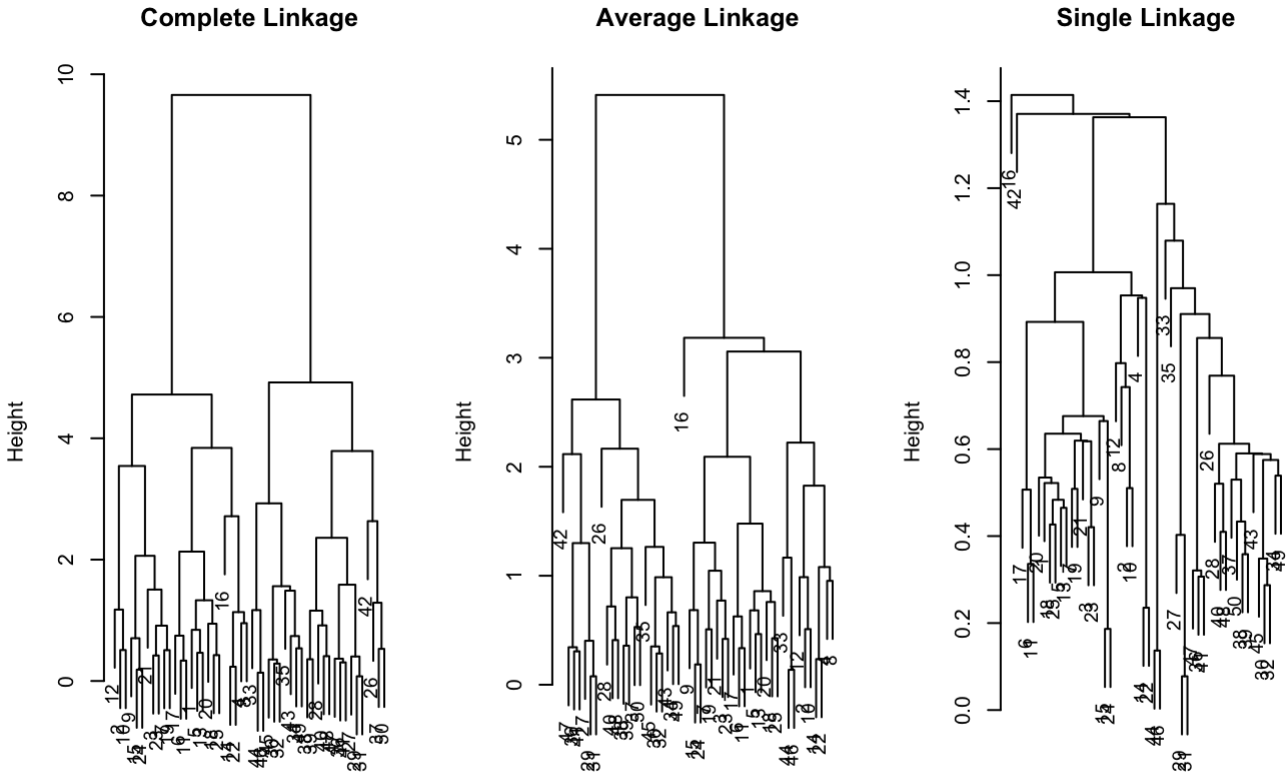
```
## K-means clustering with 3 clusters of sizes 17, 23, 10
##
## Cluster means:
##         [,1]        [,2]
## 1  3.7789567 -4.56200798
## 2 -0.3820397 -0.08740753
## 3  2.3001545 -2.69622023
##
## Clustering vector:
##  [1] 1 3 1 3 1 1 1 3 1 3 1 3 1 3 1 3 1 1 1 1 1 3 1 1 1 2 2 2 2 2 2 2 2 2 2 2
## [39] 2 2 2 2 2 3 2 3 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 25.74089 52.67700 19.56137
##  (between_SS / total_SS =  79.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
plot(x, col = (km.out$cluster + 1),
main = "K-Means Clustering Results with K = 3",
xlab = "", ylab = "", pch = 20, cex = 2)
```



Hierarchial Clusterig

```
hc.complete <- hclust(dist(x), method = "complete")
hc.average <- hclust(dist(x), method = "average")
hc.single <- hclust(dist(x), method = "single")
par(mfrow = c(1, 3))
plot(hc.complete , main = "Complete Linkage",
xlab = "", sub = "", cex = .9)
plot(hc.average , main = "Average Linkage",
xlab = "", sub = "", cex = .9)
plot(hc.single, main = "Single Linkage",
xlab = "", sub = "", cex = .9)
```



To determine the cluster labels for each observation associated with a given cut of the dendrogram, we can use the cutree() function.

```
cutree(hc.complete , 2)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
## [39] 2 2 2 2 2 2 2 2 2 2 2 2
```

```
cutree(hc.average , 2)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 1 2 2 2 2 2
## [39] 2 2 2 2 2 1 2 1 2 2 2 2
```

```
cutree(hc.single, 2)
```

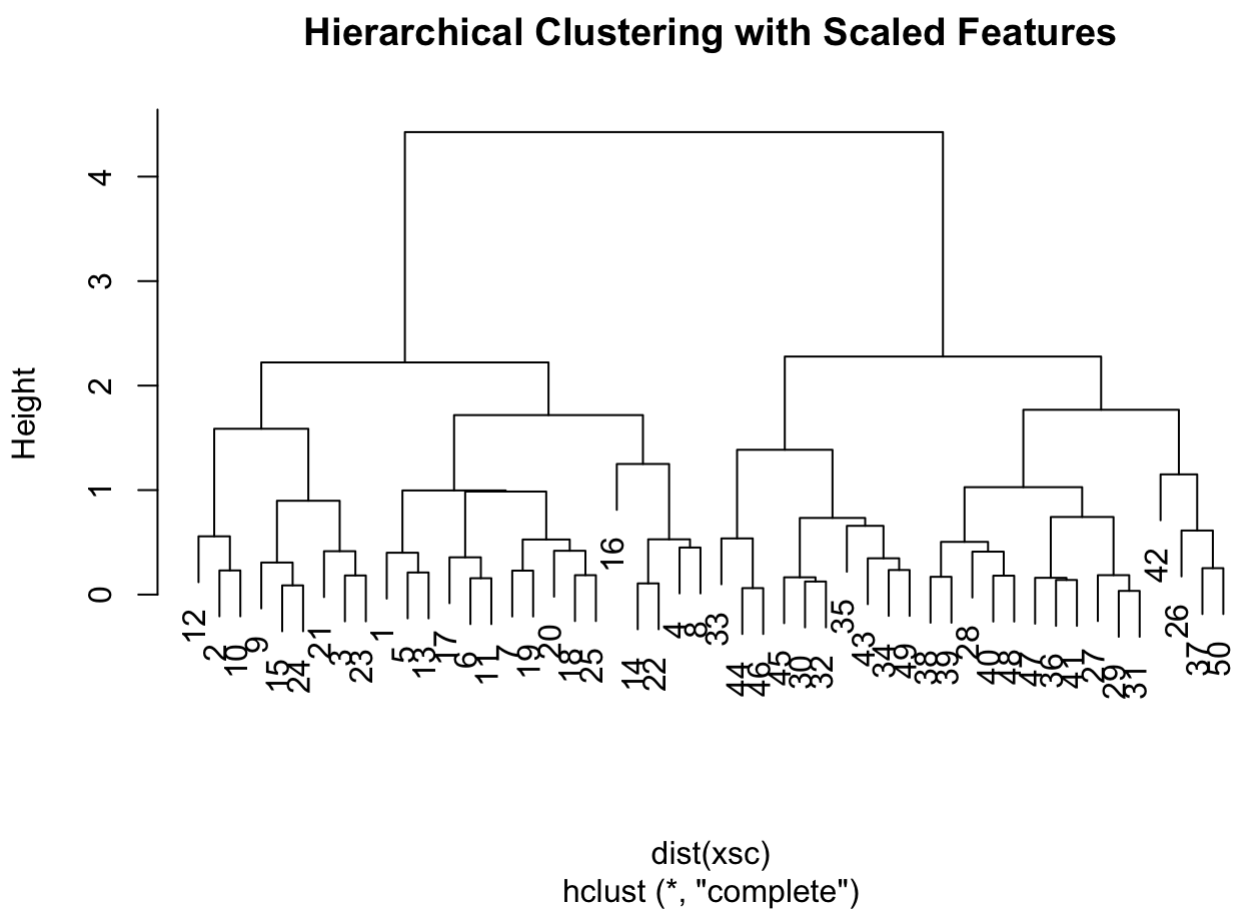
```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [39] 1 1 1 1 1 1 1 1 1 1 1 1
```

```
cutree(hc.single, 4)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3
## [39] 3 3 3 4 3 3 3 3 3 3 3 3
```

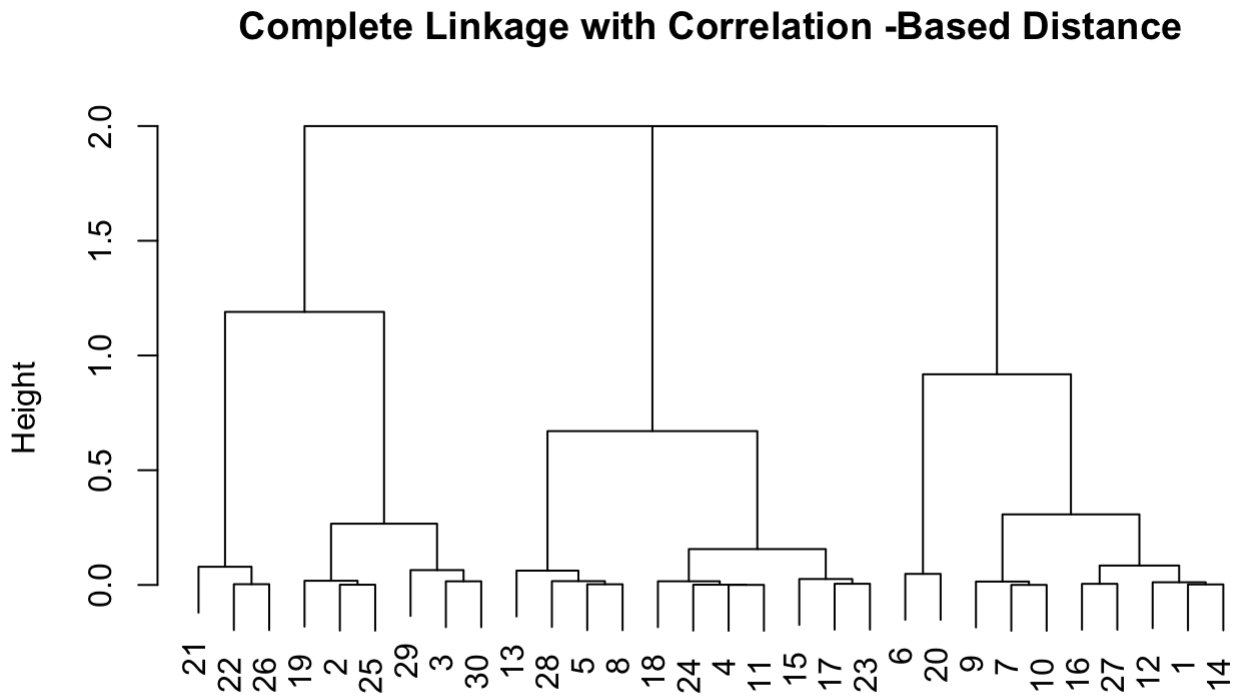
Scaling

```
xsc <- scale(x)
plot(hclust(dist(xsc), method = "complete"),
main = "Hierarchical Clustering with Scaled Features")
```



Correlation-based distance can be computed using the as.dist() function, which converts an arbitrary square symmetric matrix into a form that the hclust() function recognizes as a distance matrix.

```
x <- matrix(rnorm (30 * 3), ncol = 3)
dd <- as.dist (1 - cor(t(x)))
plot(hclust(dd, method = "complete"),
main = "Complete Linkage with Correlation -Based Distance",
xlab = "", sub = "")
```


Complete Linkage with Correlation -Based Distance
Height
0.0
0.5
1.0
1.5
2.0
21
22
26
19
2
25
29
3
30
13
28
5
8
18
24
4
11
15
17
23
6
20
9
7
10
16
27
12
1
14