Sridhar_Sriram_HW10

Lab 10.4, Lab 1 (PCA)

Exploring the `USArrests` Dataset

```
states = row.names(USArrests)

names(USArrests)

## [1] "Murder"   "Assault"  "UrbanPop" "Rape"
```

Using the `apply` function:

- Allows us to apply any given function onto a specified dataset
- Syntax is `apply(<dataset>, <margin:1,2>,<function>)`
  - **margin**:
    - **1**: applying to rows
    - **2**: applying to columns

```
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
```

Standardizing dataset to avoid overwhelming influence by the Assault data points (due to very high variance and very high mean)

`prcomp` –> PCA function in R

- Centers the variables to have a mean of 0

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

names(pr.out)

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

pr.out$center

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

```
pr.out$sdev
```

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

```
pr.out$scale
```

```
##    Murder   Assault  UrbanPop      Rape
##  4.355510 83.337661 14.474763  9.366385
```

The `rotation` matrix provides PCA loadings; each column has the respective PC loading vector

```
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
```

We expect to see 4 Principal components because the rule of thumb is

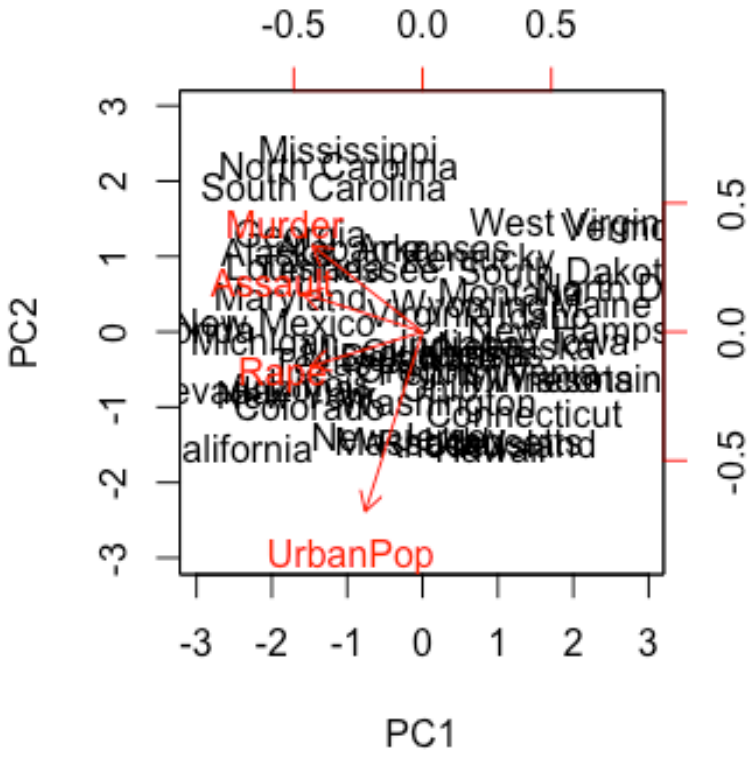
- number of principal components = min(observations -1, variables)

```
dim(pr.out$x)
```

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

** Plotting the first two principal components** :

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

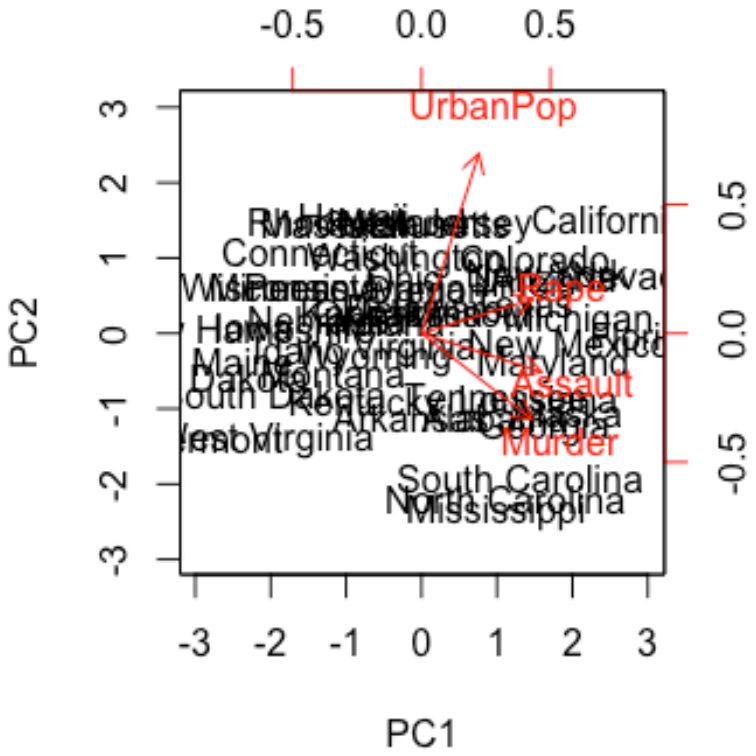


Fixing the problem of principal components being unique up to a sign change

```
pr.out$rotation = -pr.out$rotation
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
```

```
pr.out$x = -pr.out$x

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



** Obtaining the variance related to each principal component**

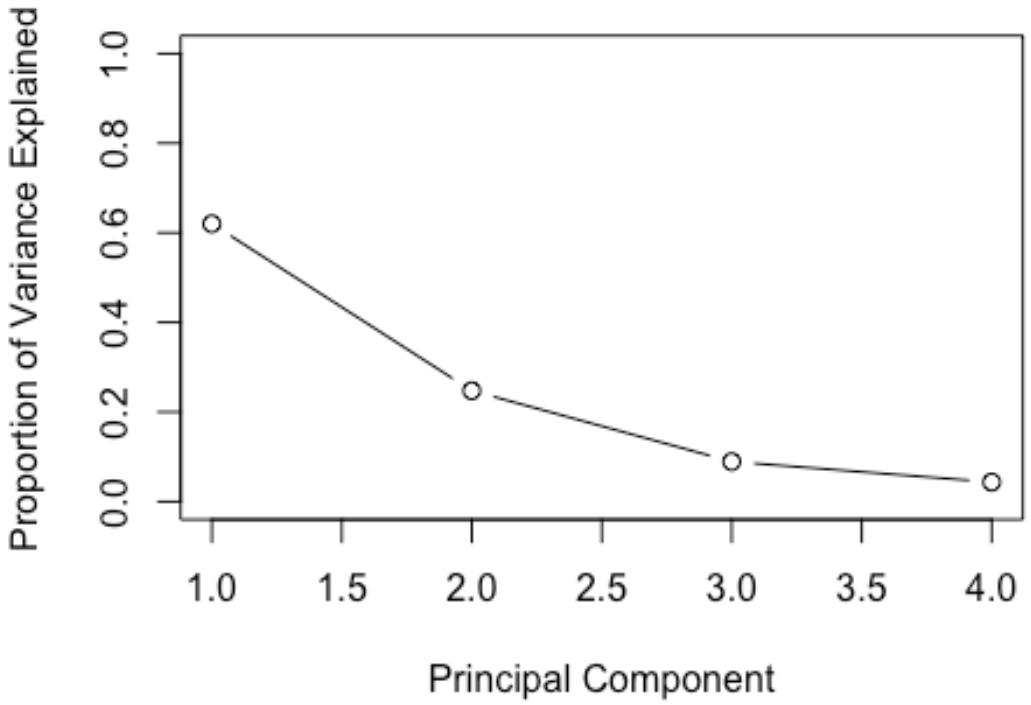
```
pr.var = pr.out$sdev^2

pr.var

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

Variance explained by each principal component

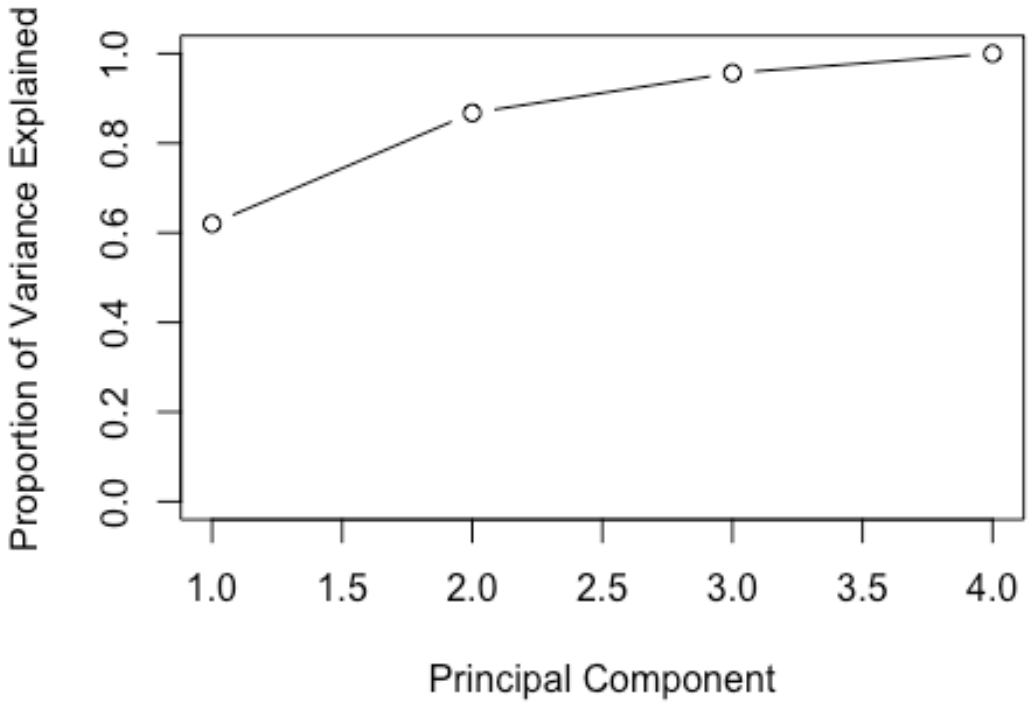
```
variance.by.component = pr.var/sum(pr.var)
variance.by.component

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

Visualizing these effects

```
plot(variance.by.component,
     xlab = "Principal Component",
     ylab = "Proportion of Variance Explained",
     ylim = c(0,1),
     type = 'b')
```




```r
plot(cumsum(variance.by.component),
     xlab = "Principal Component",
     ylab = "Proportion of Variance Explained",
     ylim = c(0,1),
     type = 'b')
```



`cumsum()` calculates cumulative sums of elements in a numeric vector

```
a = c(1,2,8,-3)

cumsum(a)

## [1]  1  3 11  8
```

Lab 10.5.1, Lab 2 (K-means Clustering)

Simulated 2-Cluster K-Means

```
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
```

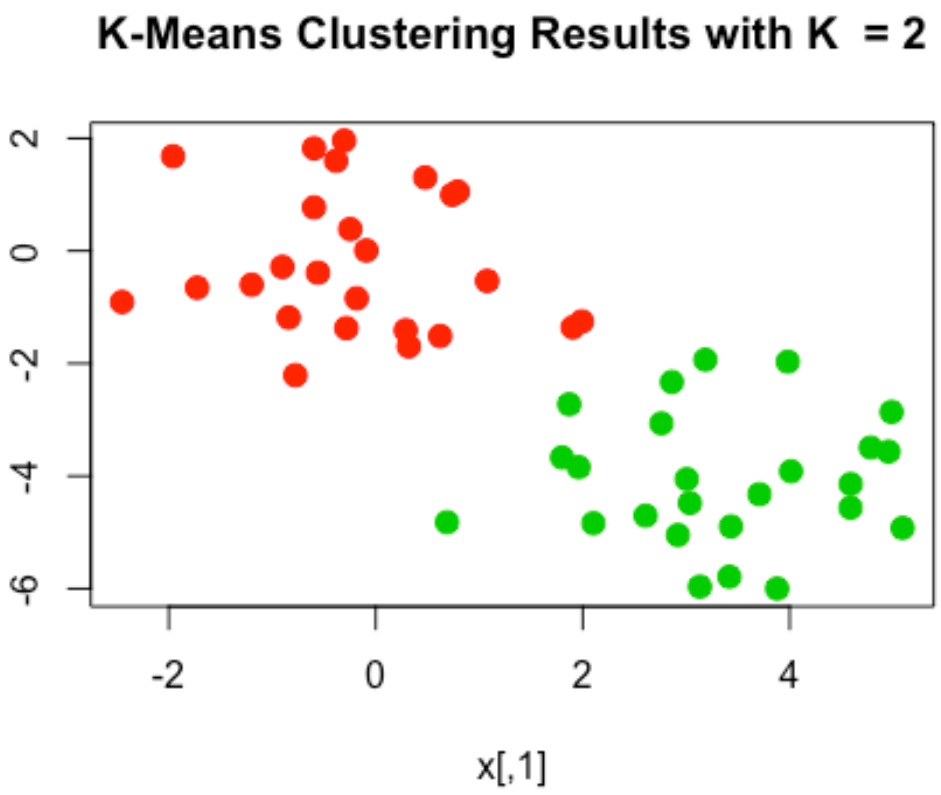
Performing the clustering with K = 2

```
km.out = kmeans(x,2,nstart = 20)
km.out

## K-means clustering with 2 clusters of sizes 25, 25
##
```

```
## Cluster means:
##         [,1]       [,2]
## 1 -0.1956978 -0.1848774
## 2  3.3339737 -4.0761910
##
## Clustering vector:
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1
## [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## Within cluster sum of squares by cluster:
## [1] 65.40068 63.20595
##  (between_SS / total_SS =  72.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

Plotting the kmeans clustering with colored distinction

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



FOR FUTURE APPLICATIONS:

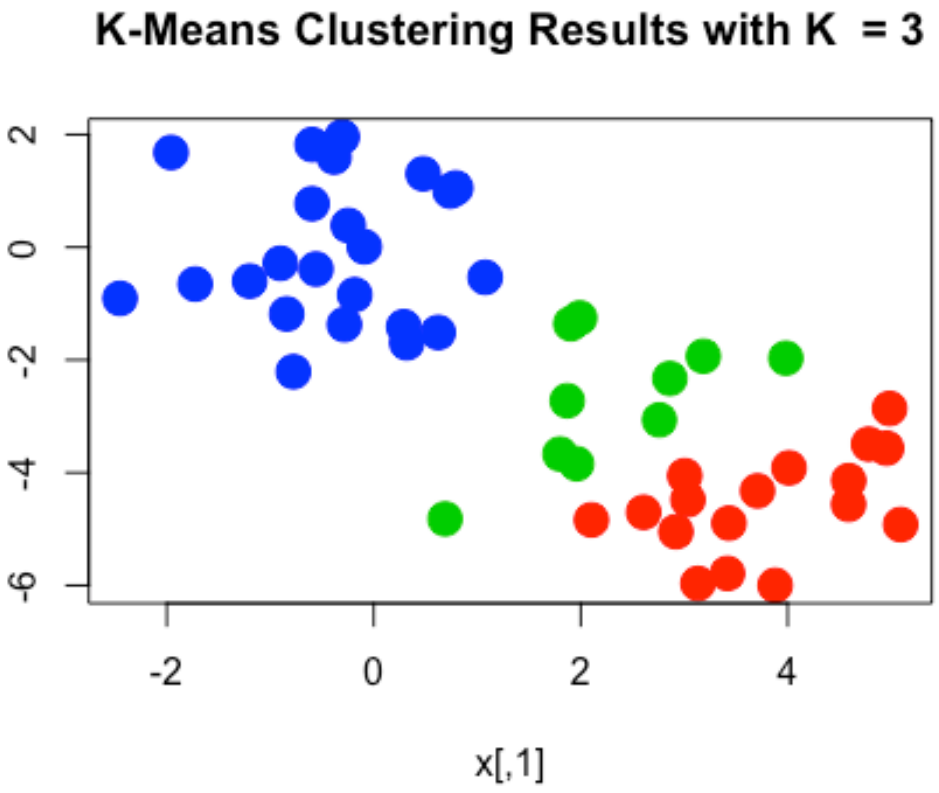
If we have clusters/components that are more than just 2 in number, then we can perform PCA and plot/visualize clusters relating to the first two most significant cluster

Same data, now with 3 clusters

```
set.seed(1234)

km.out.3 = kmeans(x,3,nstart = 20)
km.out.3

## K-means clustering with 3 clusters of sizes 17, 10, 23
##
## Cluster means:
##         [,1]        [,2]
## 1  3.7789567 -4.56200798
## 2  2.3001545 -2.69622023
## 3 -0.3820397 -0.08740753
##
## Clustering vector:
##  [1] 1 2 1 2 1 1 1 2 1 2 1 2 1 2 1 2 1 1 1 1 1 2 1 1 1 3 3 3 3 3 3 3 3 3
## [36] 3 3 3 3 3 3 3 3 2 3 2 3 3 3 3
##
```

```
## Within cluster sum of squares by cluster:
## [1] 25.74089 19.56137 52.67700
##  (between_SS / total_SS =  79.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

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



Within kmeans: `nstart` :

- parameter for multiple initial cluster assignments

** kmeans components: **

`tot.withinss` :

- Total within-cluster sum of squares –> minimize this!!!

`withinss`:

- individual within-cluster sum of squares

Lab 10.5.2, Lab (hierarchial clustering)

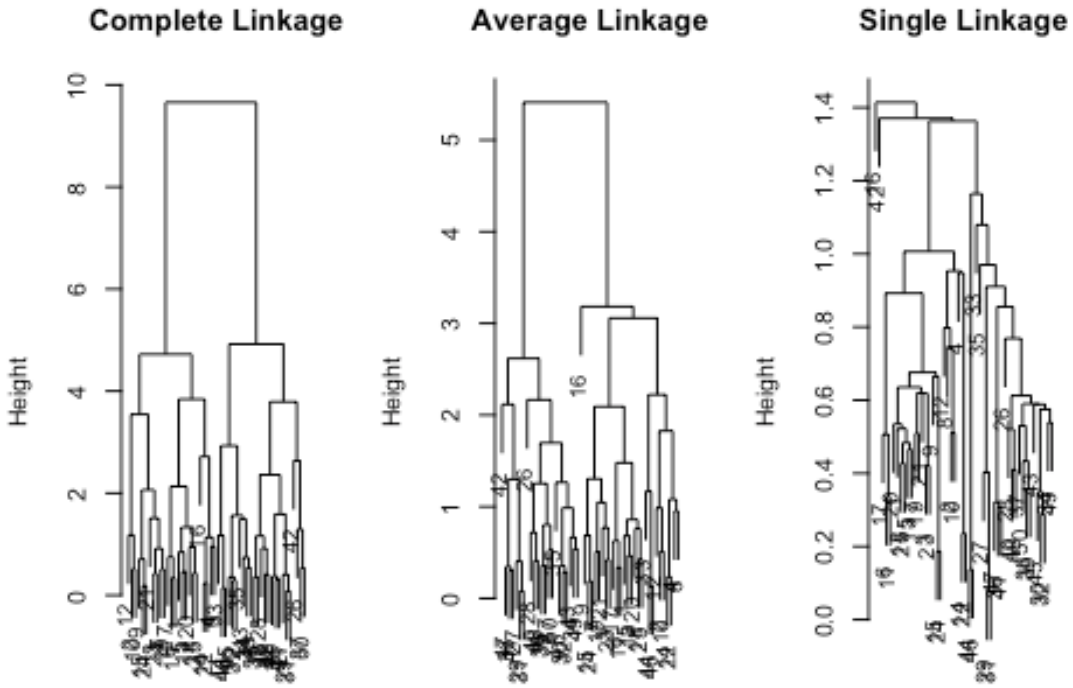
Using same data from previous lab (x)

Complete, average, single linkage

```
hc.complete = hclust(dist(x),method = "complete")

hc.single = hclust(dist(x),method = "single")

hc.average = hclust(dist(x),method = "average")
```

Plotting the associated dendrograms

```
par(mfrow = c(1,3))

plot(hc.complete,
     main = "Complete Linkage",
     sub = "",
     xlab = "",
     cex = 0.9)

plot(hc.average,
     main = "Average Linkage",
     sub = "",
     xlab = "",
     cex = 0.9)

plot(hc.single,
     main = "Single Linkage",
     sub = "",
     xlab = "",
     cex = 0.9)
```



Determining cluster labels for each observation

```
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
## [36] 2 2 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 2 1 2 2
## [36] 2 2 2 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 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

The `cutree` using `hc.single` returns a list of `1`s because single linkage identifies one point as belonging to its own cluster

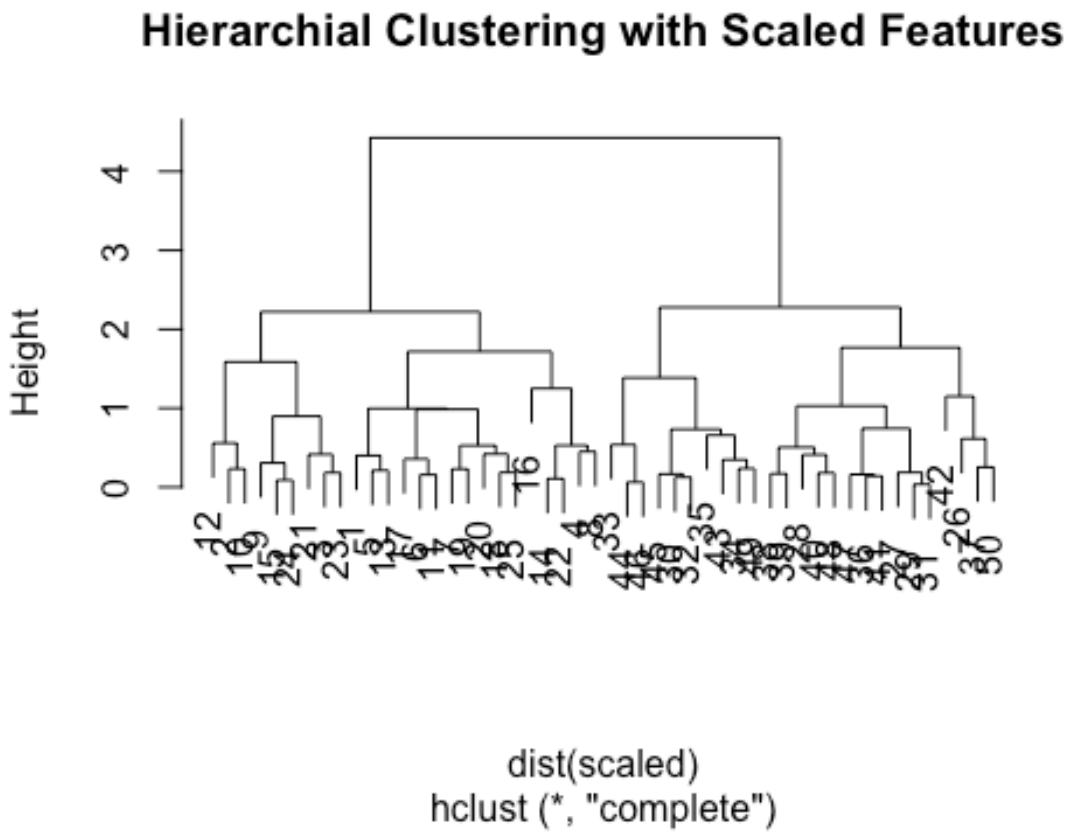
** Fixing the above (using 4 clusters) **

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

```
##  [1] 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
## [36] 3 3 3 3 3 3 4 3 3 3 3 3 3 3 3
```

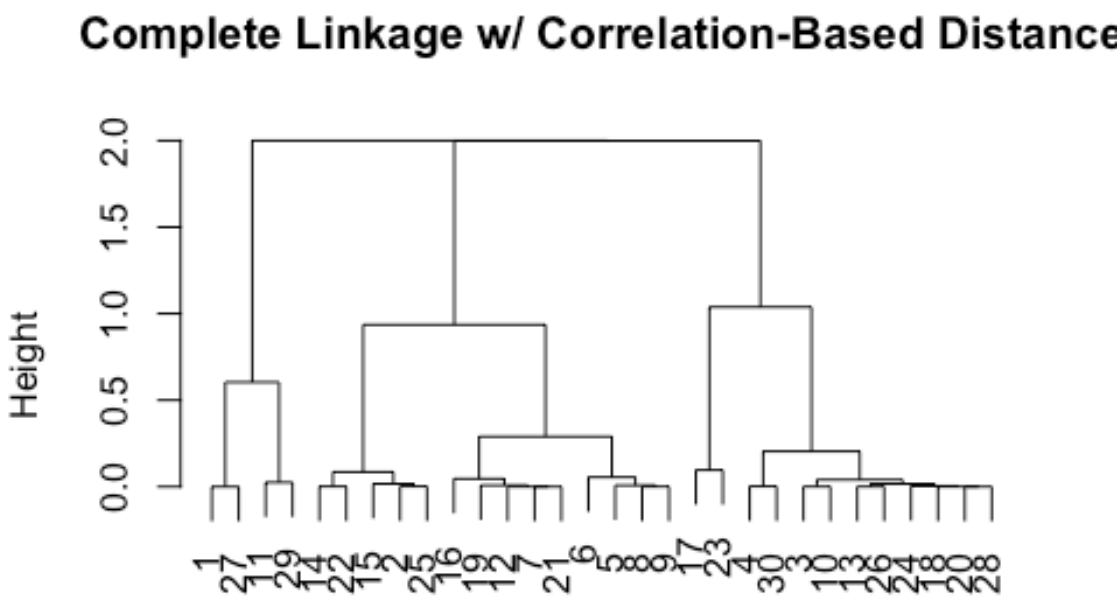
Scaling before performing hierarchial clustering

```
scaled = scale(x)

plot(hclust(dist(scaled), method = "complete"),
     main = "Hierarchial Clustering with Scaled Features")
```



Correlation-based distance

```
x = matrix(rnorm(30*3),ncol = 3)

dd = as.dist(1-cor(t(x)))

plot(hclust(dd,method = "complete"),
     main = "Complete Linkage w/ Correlation-Based Distance",
     xlab = "",
     sub = "")
```


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