Example - USArrests

.

```
library(cluster)    # silohuette()
library(factoextra) # get_dist()
#
df0 <- USArrests
str(df0)
```

```
## '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
```

```
head(df0)
```

```
##            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
```

Example - 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
```

```
#
# scale dataframe
#
df <- scale(df0)
class(df)
```

```
## [1] "matrix"
```

```
head(df)
```

```
##                Murder   Assault   UrbanPop         Rape
## Alabama    1.24256408 0.7828393 -0.5209066 -0.003416473
## Alaska     0.50786248 1.1068225 -1.2117642  2.484202941
## Arizona    0.07163341 1.4788032  0.9989801  1.042878388
## Arkansas   0.23234938 0.2308680 -1.0735927 -0.184916602
## California 0.27826823 1.2628144  1.7589234  2.067820292
## Colorado   0.02571456 0.3988593  0.8608085  1.864967207
```

Example - USArrests

```r
distance = dist(df)
head(distance)
```

```
## [1] 2.703754 2.293520 1.289810 3.263110 2.651067 3.215297
```

```r
length(distance)
```

```
## [1] 1225
```

```r
# distance in a matrix display
#
distmat = as.matrix(distance)
dim(distmat)
```

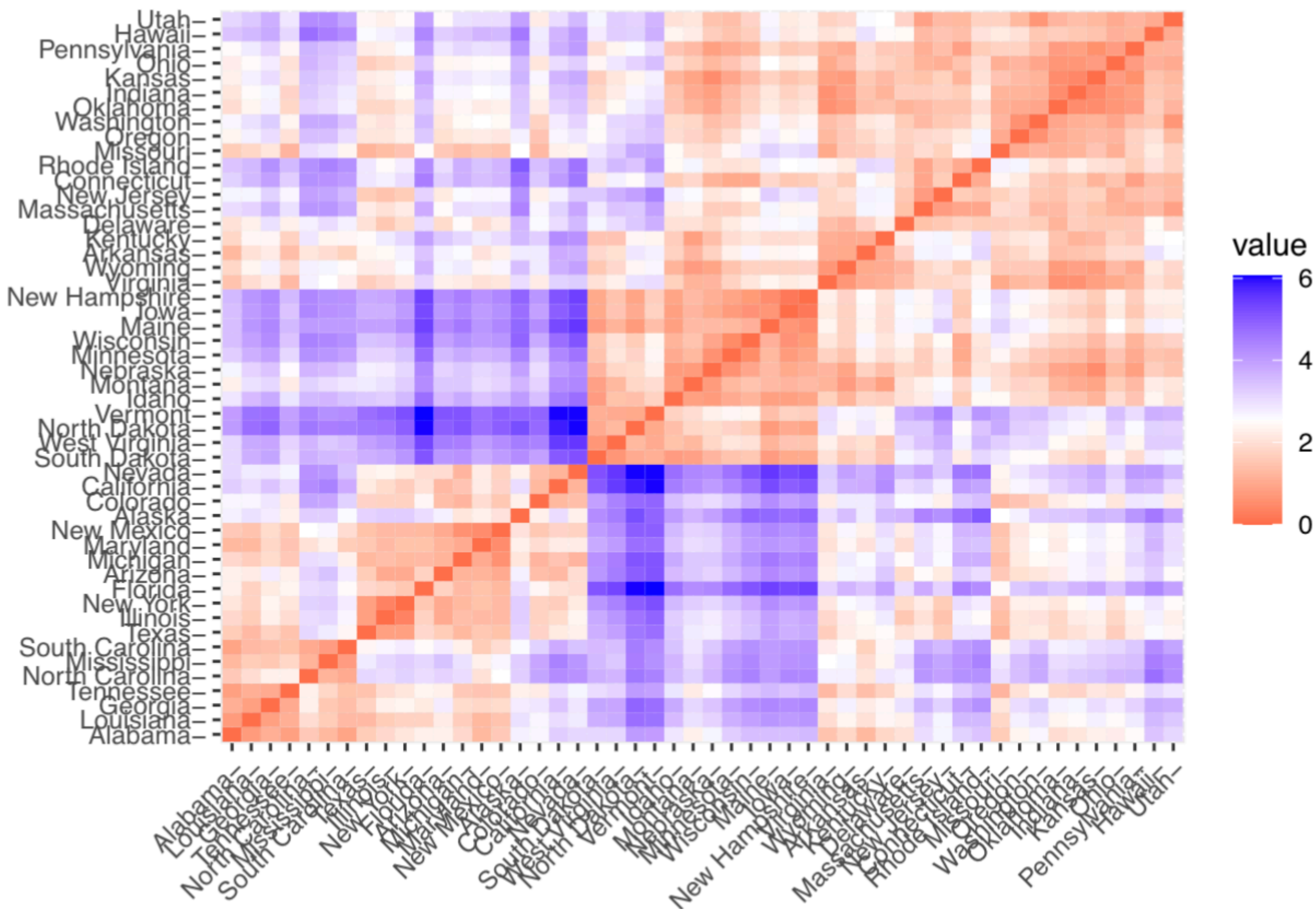
```
## [1] 50 50
```

```r
distmat[1:7,1:7]
```

```
##              Alabama   Alaska  Arizona Arkansas California Colorado Connecticut
## Alabama     0.000000 2.703754 2.293520 1.289810   3.263110 2.651067    3.215297
## Alaska      2.703754 0.000000 2.700643 2.826039   3.012541 2.326519    4.739912
## Arizona     2.293520 2.700643 0.000000 2.717758   1.310484 1.365031    3.262858
## Arkansas    1.289810 2.826039 2.717758 0.000000   3.763641 2.831051    2.607639
## California  3.263110 3.012541 1.310484 3.763641   0.000000 1.287619    4.066390
## Colorado    2.651067 2.326519 1.365031 2.831051   1.287619 0.000000    3.327992
## Connecticut 3.215297 4.739912 3.262858 2.607639   4.066390 3.327992    0.000000
```

Example - USArrests

```
fviz_dist(distance)
```



Example - USArrests

.

```
# K-means with 2 clusters
#
k2 = kmeans(df, centers = 2, nstart = 25)
str(k2)
```

```
## List of 9
##  $ cluster     : Named int [1:50] 2 2 2 1 2 2 1 1 2 2 ...
##   ..- attr(*, "names")= chr [1:50] "Alabama" "Alaska" "Arizona" "Arkansas"
##  $ centers     : num [1:2, 1:4] -0.67 1.005 -0.676 1.014 -0.132 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : chr [1:4] "Murder" "Assault" "UrbanPop" "Rape"
##  $ totss       : num 196
##  $ withinss    : num [1:2] 56.1 46.7
##  $ tot.withinss: num 103
##  $ betweenss   : num 93.1
##  $ size        : int [1:2] 30 20
##  $ iter        : int 1
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

Example - USArrests

.

```
# K-means with 2 clusters
#
k2 = kmeans(df, centers = 2, nstart = 25)
str(k2)
```

```
## List of 9
##  $ cluster     : Named int [1:50] 2 2 2 1 2 2 1 1 2 2 ...
##   ..- attr(*, "names")= chr [1:50] "Alabama" "Alaska" "Arizona" "Arkansas"
##  $ centers     : num [1:2, 1:4] -0.67 1.005 -0.676 1.014 -0.132 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : chr [1:4] "Murder" "Assault" "UrbanPop" "Rape"
##  $ totss       : num 196
##  $ withinss    : num [1:2] 56.1 46.7
##  $ tot.withinss: num 103
##  $ betweenss   : num 93.1
##  $ size        : int [1:2] 30 20
##  $ iter        : int 1
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

Example - USArrests

Use multiple random assignments (step 1) find clusters from each one, and report the best performance

```
# K-means with 2 clusters
#
k2 = kmeans(df, centers = 2, nstart = 25)
str(k2)
```

```
## List of 9
##  $ cluster     : Named int [1:50] 2 2 2 1 2 2 1 1 2 2 ...
##   ..- attr(*, "names")= chr [1:50] "Alabama" "Alaska" "Arizona" "Arkansas"
##  $ centers     : num [1:2, 1:4] -0.67 1.005 -0.676 1.014 -0.132 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : chr [1:4] "Murder" "Assault" "UrbanPop" "Rape"
##  $ totss       : num 196
##  $ withinss    : num [1:2] 56.1 46.7
##  $ tot.withinss: num 103
##  $ betweenss   : num 93.1
##  $ size        : int [1:2] 30 20
##  $ iter        : int 1
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

WCV for each cluster
TWCV = sum(withinss)

cluster sizes

Example - USArrests

.

```
k2
```

centroids

row assignments

```
## K-means clustering with 2 clusters of sizes 30, 20
##
## Cluster means:
##      Murder   Assault   UrbanPop       Rape
## 1 -0.669956 -0.6758849 -0.1317235 -0.5646433
## 2  1.004934  1.0138274  0.1975853  0.8469650

## Clustering vector:
##        Alabama         Alaska        Arizona       Arkansas     California
##              2              2              2              1              2
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              2              1              1              2              2
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              1              1              2              1              1
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              1              1              2              1              2
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##              1              2              1              2              2
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##              1              1              2              1              1
```

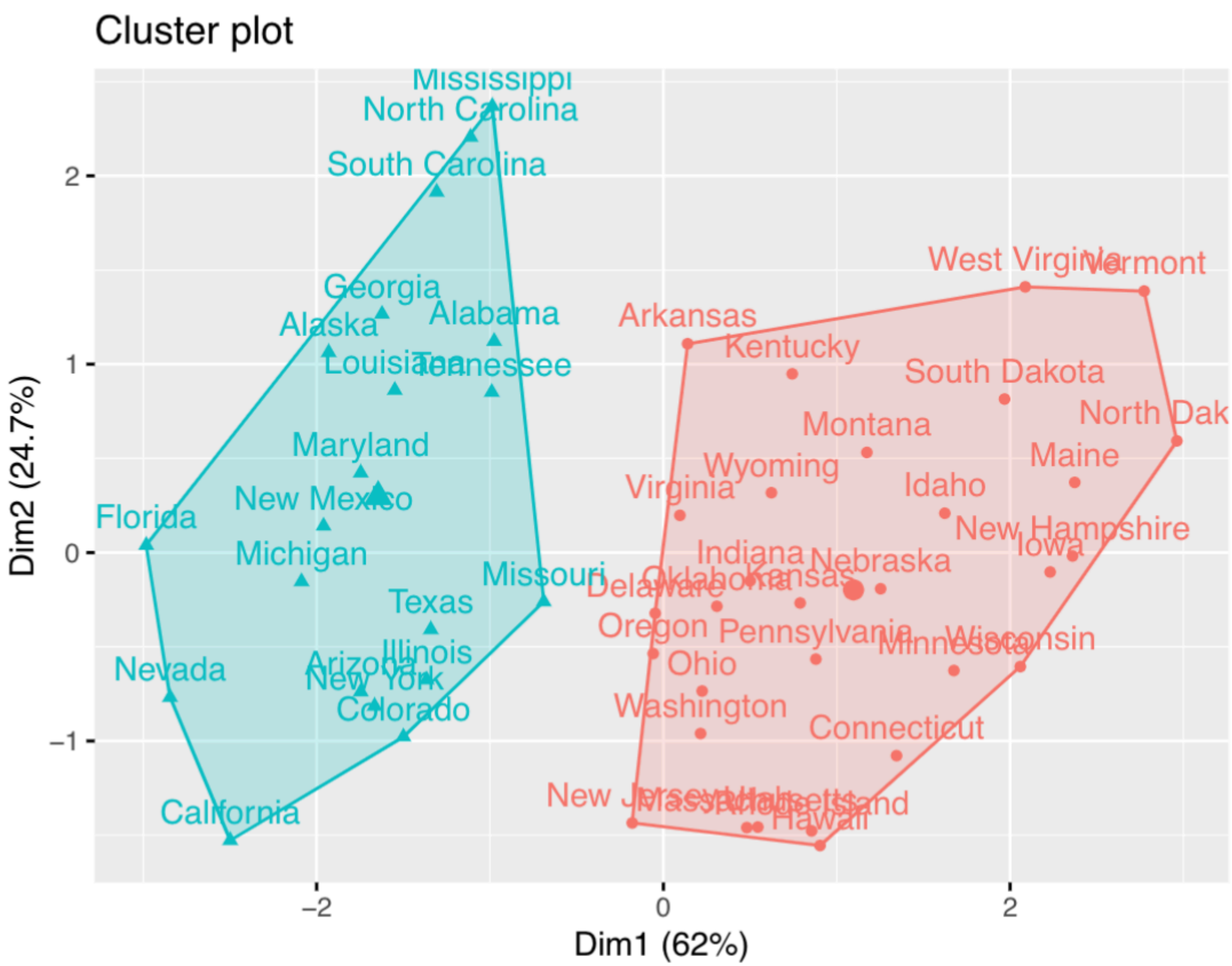
Example - USArrests

```
k2
```

```
## K-means clustering with 2 clusters of sizes 30, 20
##
## Cluster means:
##       Murder    Assault   UrbanPop        Rape
## 1 -0.669956 -0.6758849 -0.1317235 -0.5646433
## 2  1.004934  1.0138274  0.1975853  0.8469650
## Clustering vector:
##        Alabama         Alaska        Arizona       Arkansas     California
##              2              2              2              1              2
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              2              1              1              2              2
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              1              1              2              1              1
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              1              1              2              1              2
## Within cluster sum of squares by cluster:
## [1] 56.11445 46.74796
##  (between_SS / total_SS =  47.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Example - USArrests

```
fviz_cluster(k2, data = df)
```

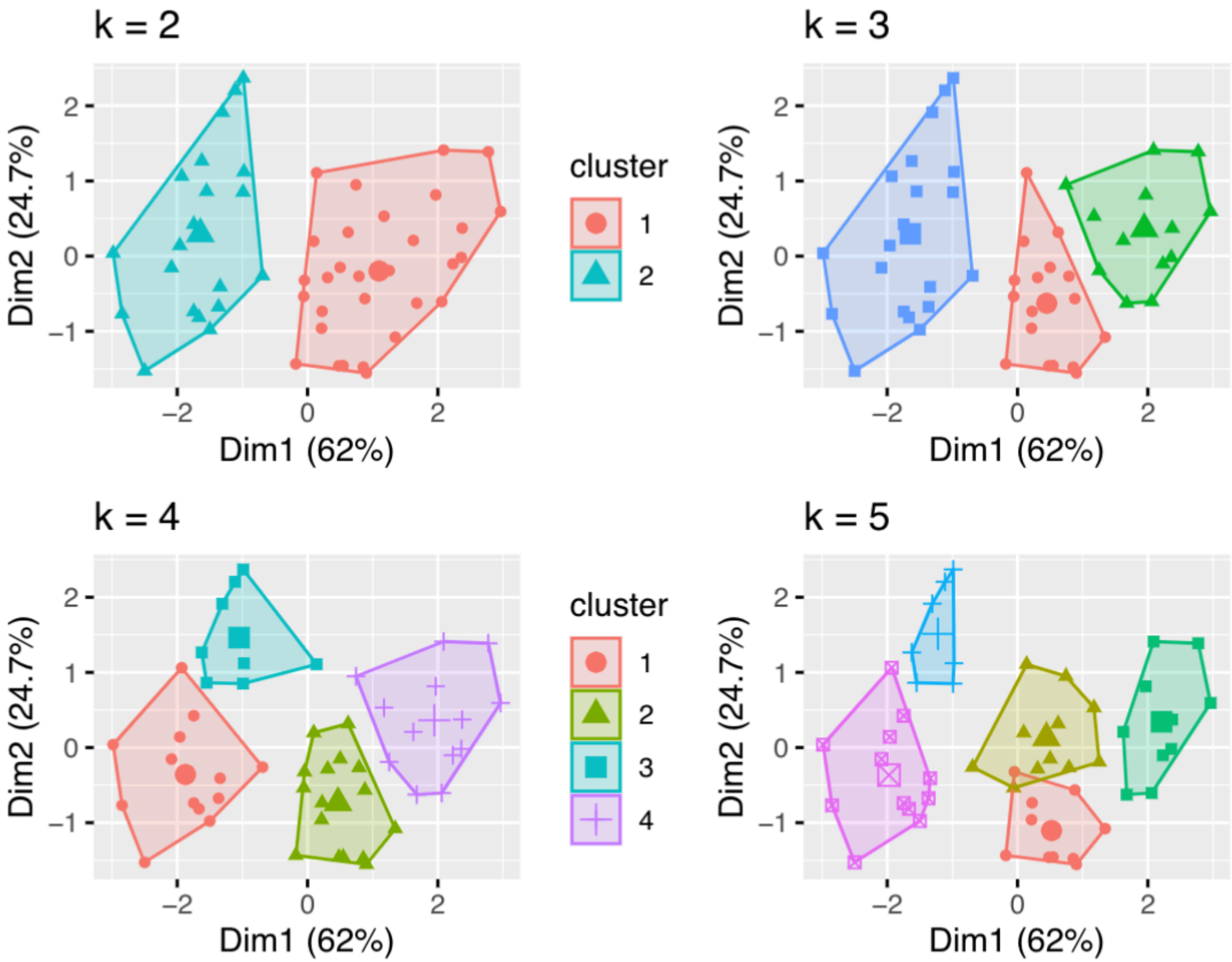


Example - USArrests

.

```
# K-means with k=3,4,5
#
k3 <- kmeans(df, centers = 3, nstart = 25)
k4 <- kmeans(df, centers = 4, nstart = 25)
k5 <- kmeans(df, centers = 5, nstart = 25)
#
p1 <- fviz_cluster(k2, geom = "point", data = df) + ggtitle("k = 2")
p2 <- fviz_cluster(k3, geom = "point",  data = df) + ggtitle("k = 3")
p3 <- fviz_cluster(k4, geom = "point",  data = df) + ggtitle("k = 4")
p4 <- fviz_cluster(k5, geom = "point",  data = df) + ggtitle("k = 5")
#
library(gridExtra)
grid.arrange(p1, p2, p3, p4, nrow = 2)
```

Example - USArrests



Example - USArrests

.

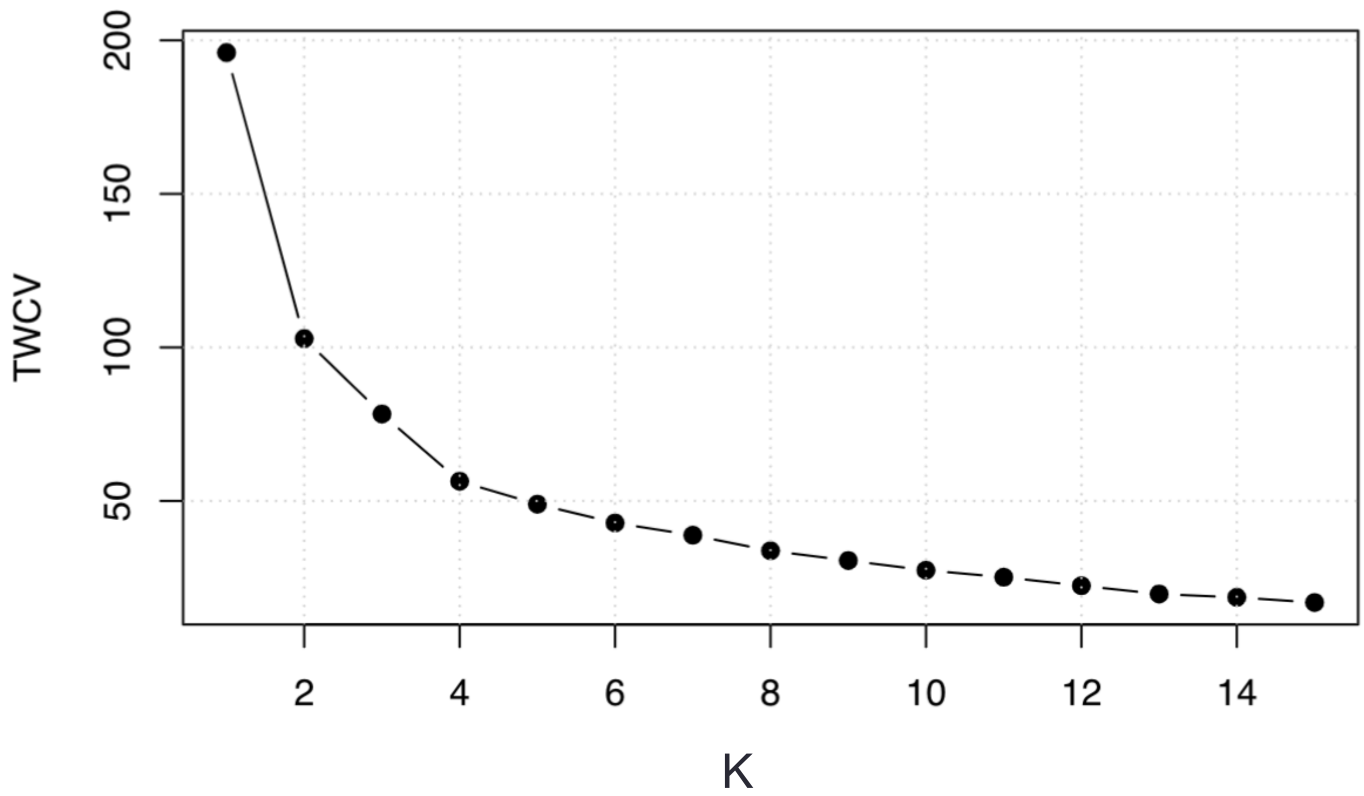
```r
# TWCV for k=1 to 15
#
set.seed(123)
twcv = function(k) kmeans(df, k, nstart = 10 )$tot.withinss
#
# Plot twcv for k = 1 to k = 15
#
k <- 1:15
twcv_values <- sapply(k,twcv)
head(twcv_values)
```

```
## [1] 196.00000 102.86240  78.32327  56.40317  48.94420  42.83303
```

Example - USArrests

.

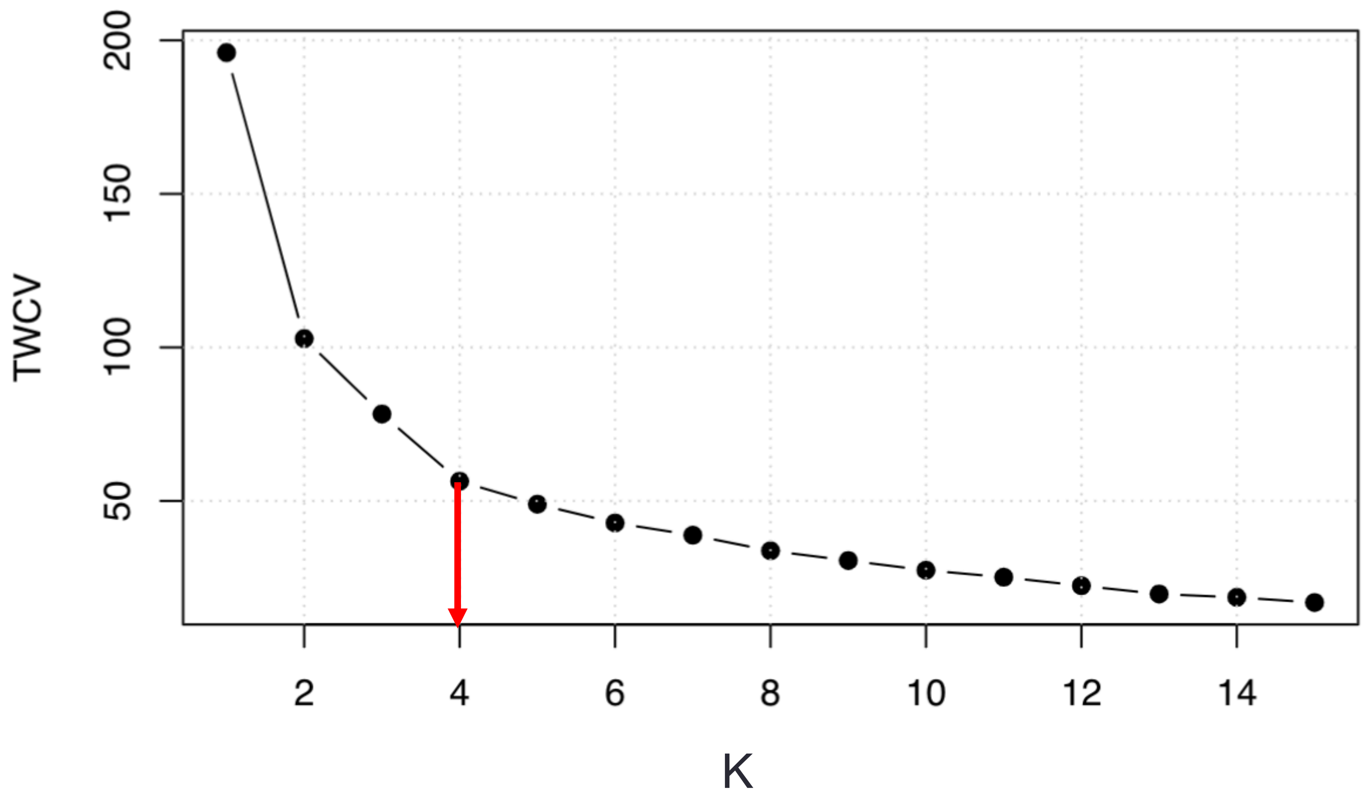
```
plot(k, twcv_values,type="b",pch = 19,
     xlab="Number of clusters K",ylab="TWCV")
grid()
```



Example – Elbow chart

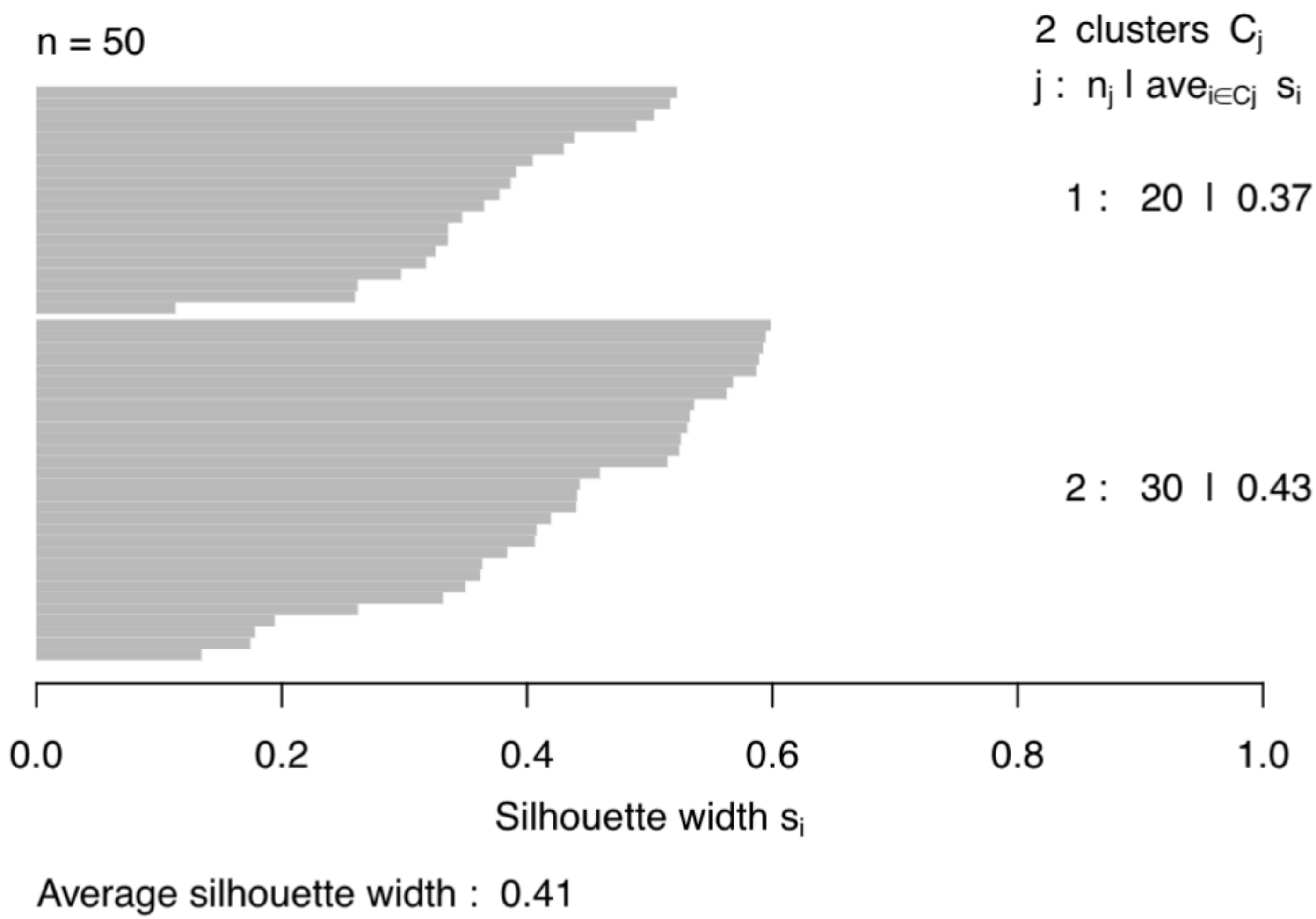
.

```
plot(k, twcv_values,type="b",pch = 19,
     xlab="Number of clusters K",ylab="TWCV")
grid()
```



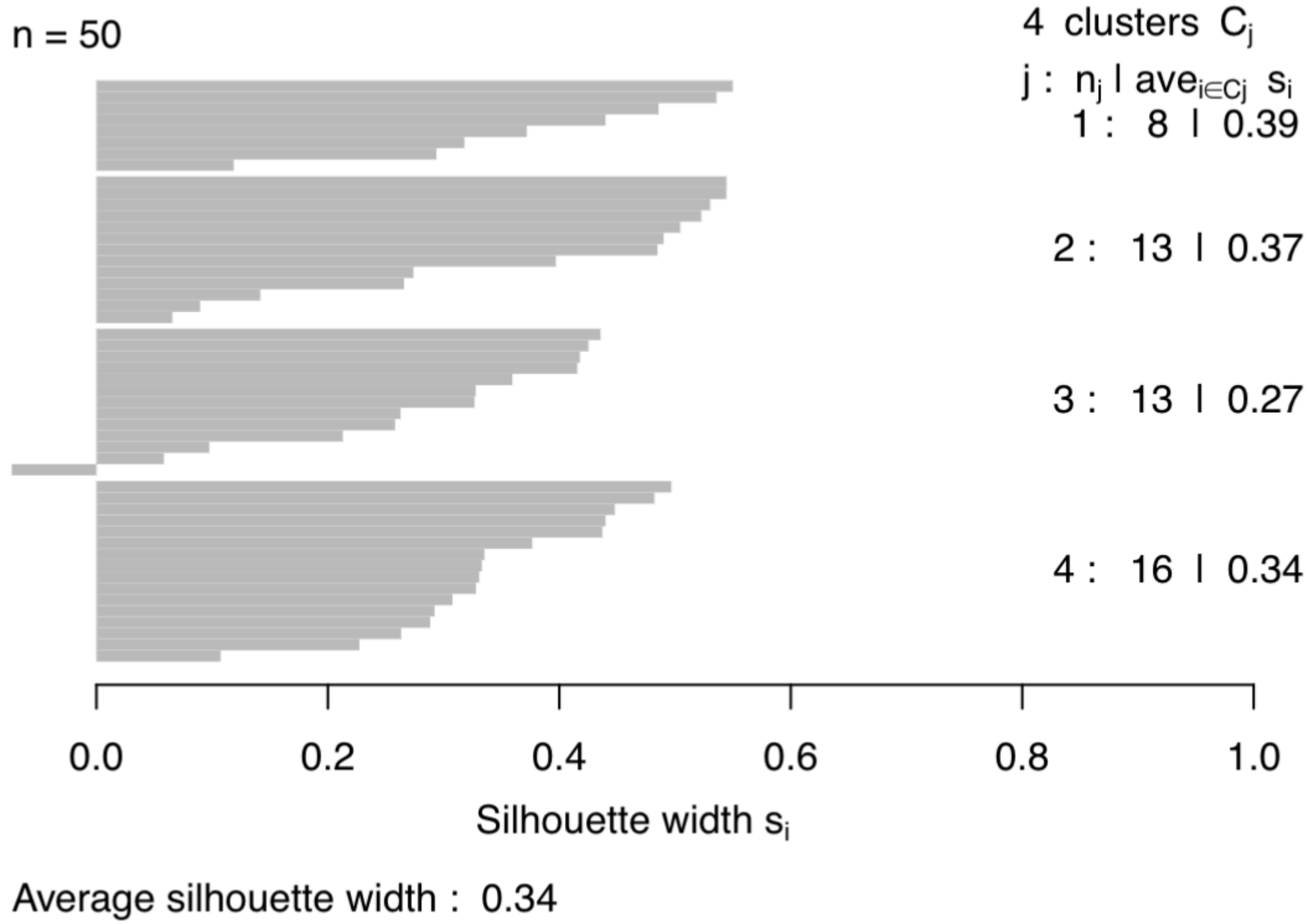
Example – USArrests silhouette diagram k = 2

```
ss = silhouette(k2$cluster, distance)
plot(ss,main="")
```



Example – USArrests silhouette diagram k = 4

```
ss = silhouette(k4$cluster, distance)
plot(ss,main="")
```



Example – USArrests average silhouette

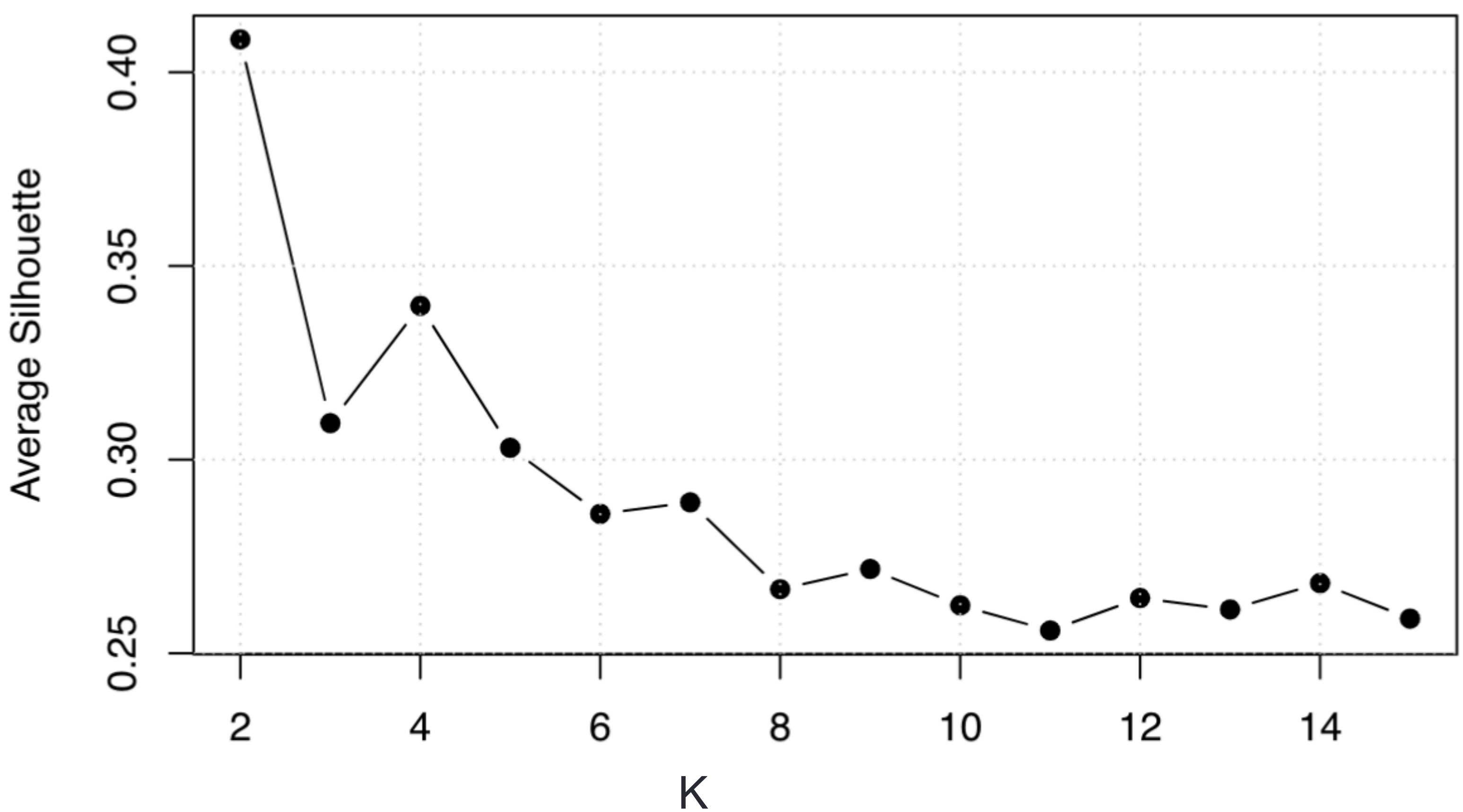
.

```r
# function to find the average silhouette for k clusters
#
avg_sil <- function(k)
  {
  km.res <- kmeans(df, centers = k, nstart = 25)
  ss <- silhouette(km.res$cluster, dist(df))
  mean(ss[, 3])
}
#
# avg silhouette for 2-15 clusters
#
j <- 2:15
avg_sil_values <- sapply(j, avg_sil)
avg_sil_values
```

```
##  [1] 0.4084890 0.3094312 0.3396889 0.3030781 0.2859821 0.2889692
##  [8] 0.2717609 0.2623532 0.2558806 0.2642560 0.2613045 0.2681210
```

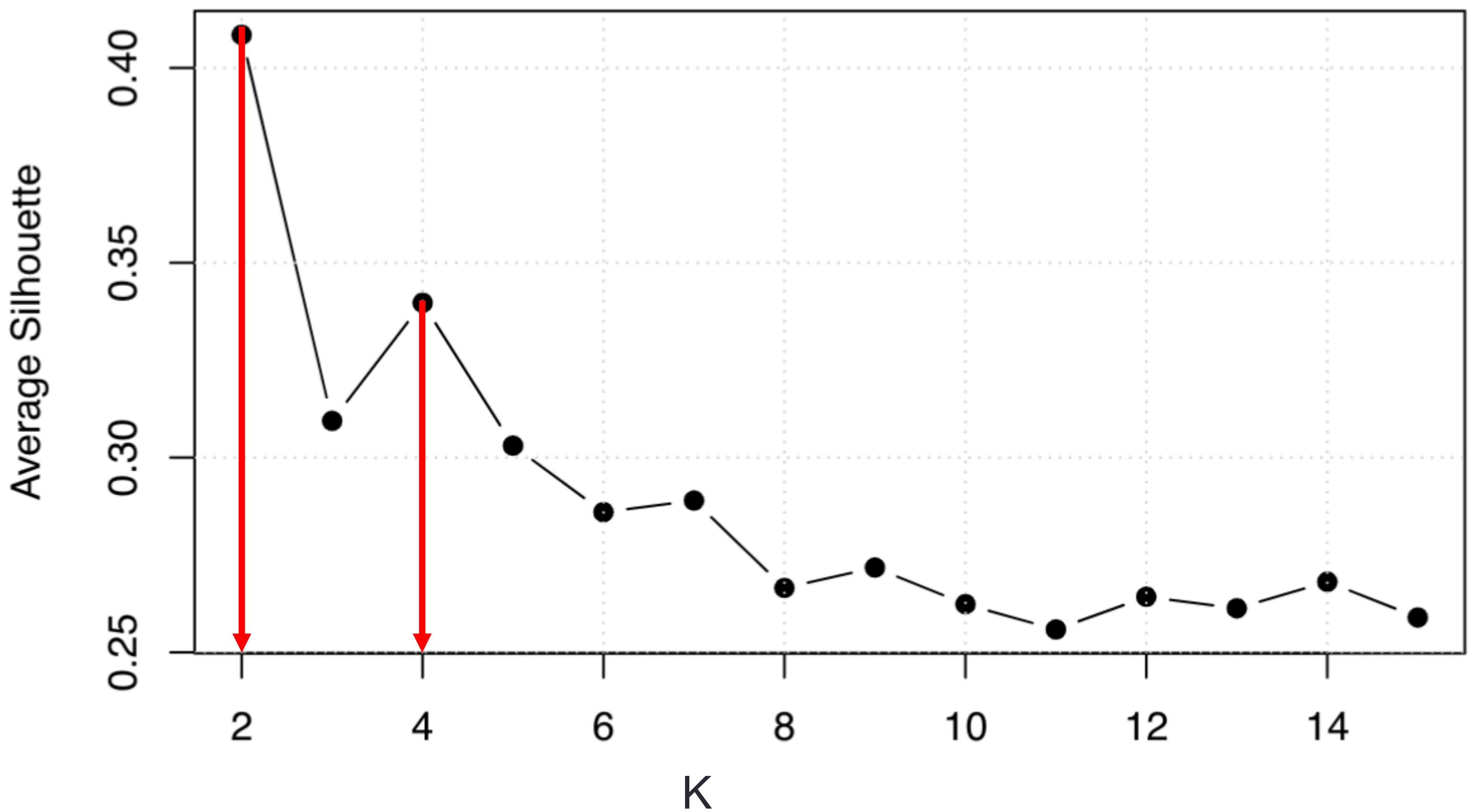
Example – USArrests average silhouette

```
## [1] 0.4084890 0.3094312 0.3396889 0.3030781 0.2859821 0.2889692
## [8] 0.2717609 0.2623532 0.2558806 0.2642560 0.2613045 0.2681210
```



Example – USArrests average silhouette

```
## [1] 0.4084890 0.3094312 0.3396889 0.3030781 0.2859821 0.2889692
## [8] 0.2717609 0.2623532 0.2558806 0.2642560 0.2613045 0.2681210
```



Example – USArrests final choice k = 4

.

```
set.seed(123)
final <- kmeans(df, 4, nstart = 25)
final
```

```
## K-means clustering with 4 clusters of sizes 8, 13, 16, 13
##
## Cluster means:
##       Murder    Assault   UrbanPop        Rape
## 1  1.4118898  0.8743346 -0.8145211  0.01927104
## 2 -0.9615407 -1.1066010 -0.9301069 -0.96676331
## 3 -0.4894375 -0.3826001  0.5758298 -0.26165379
## 4  0.6950701  1.0394414  0.7226370  1.27693964
```

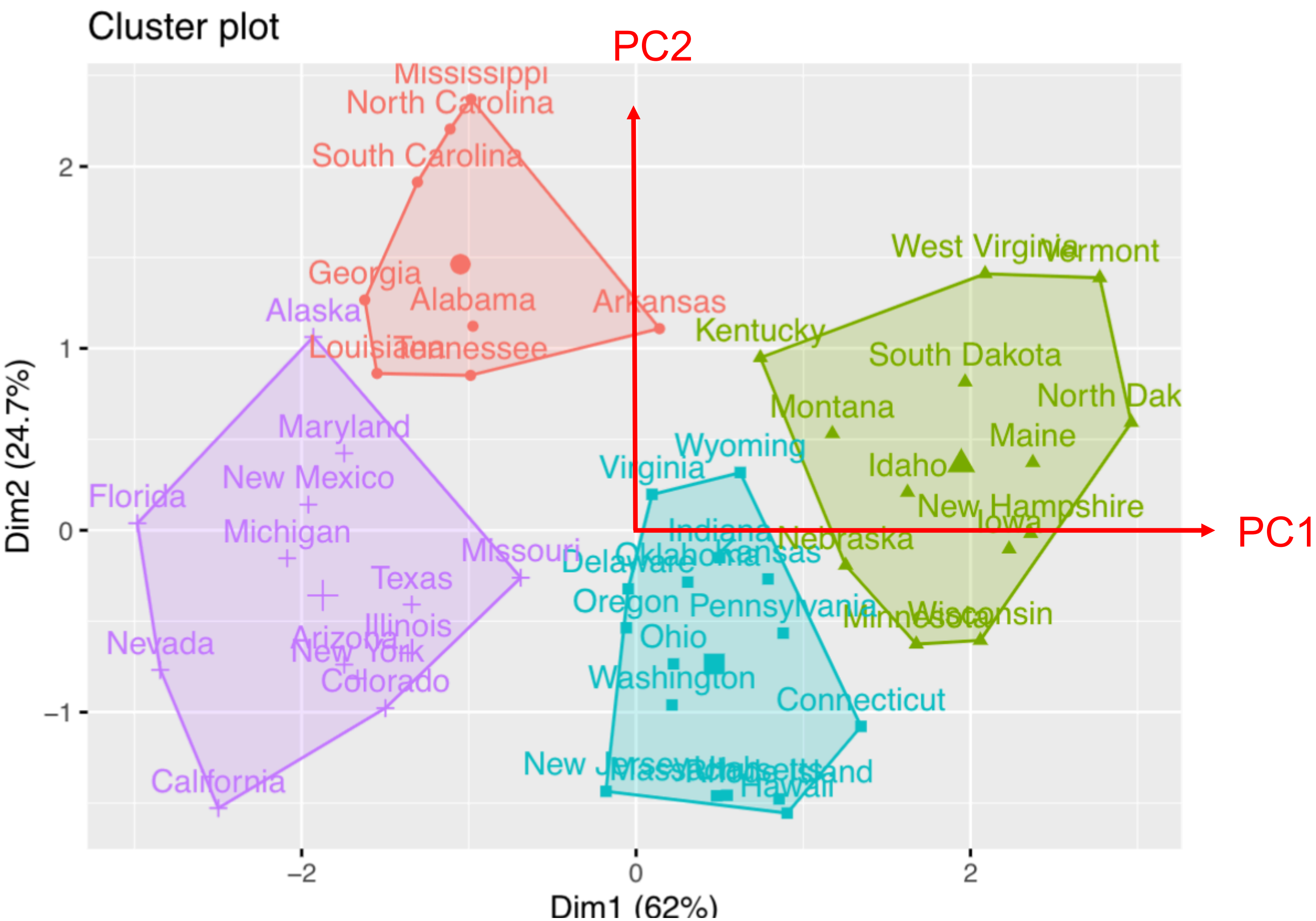
Example – USArrests final choice k = 4

.

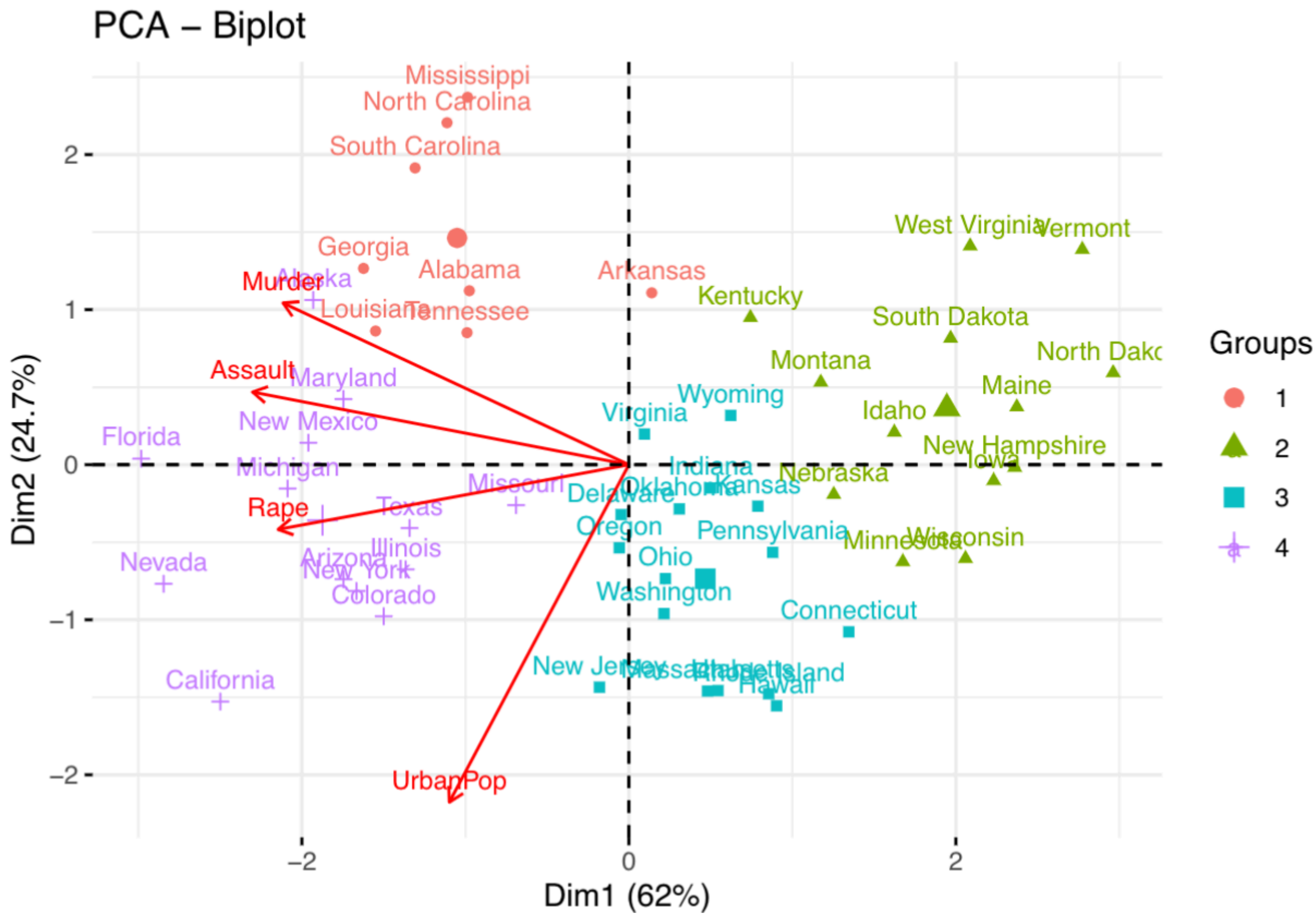
```
# add cluster to dataframe
#
cluster_number = as.factor(final$cluster)
df0$cluster = cluster_number
head(df0)
```

```
##            Murder Assault UrbanPop Rape cluster
## Alabama      13.2     236       58 21.2       1
## Alaska       10.0     263       48 44.5       4
## Arizona       8.1     294       80 31.0       4
## Arkansas      8.8     190       50 19.5       1
## California    9.0     276       91 40.6       4
## Colorado      7.9     204       78 38.7       4
```

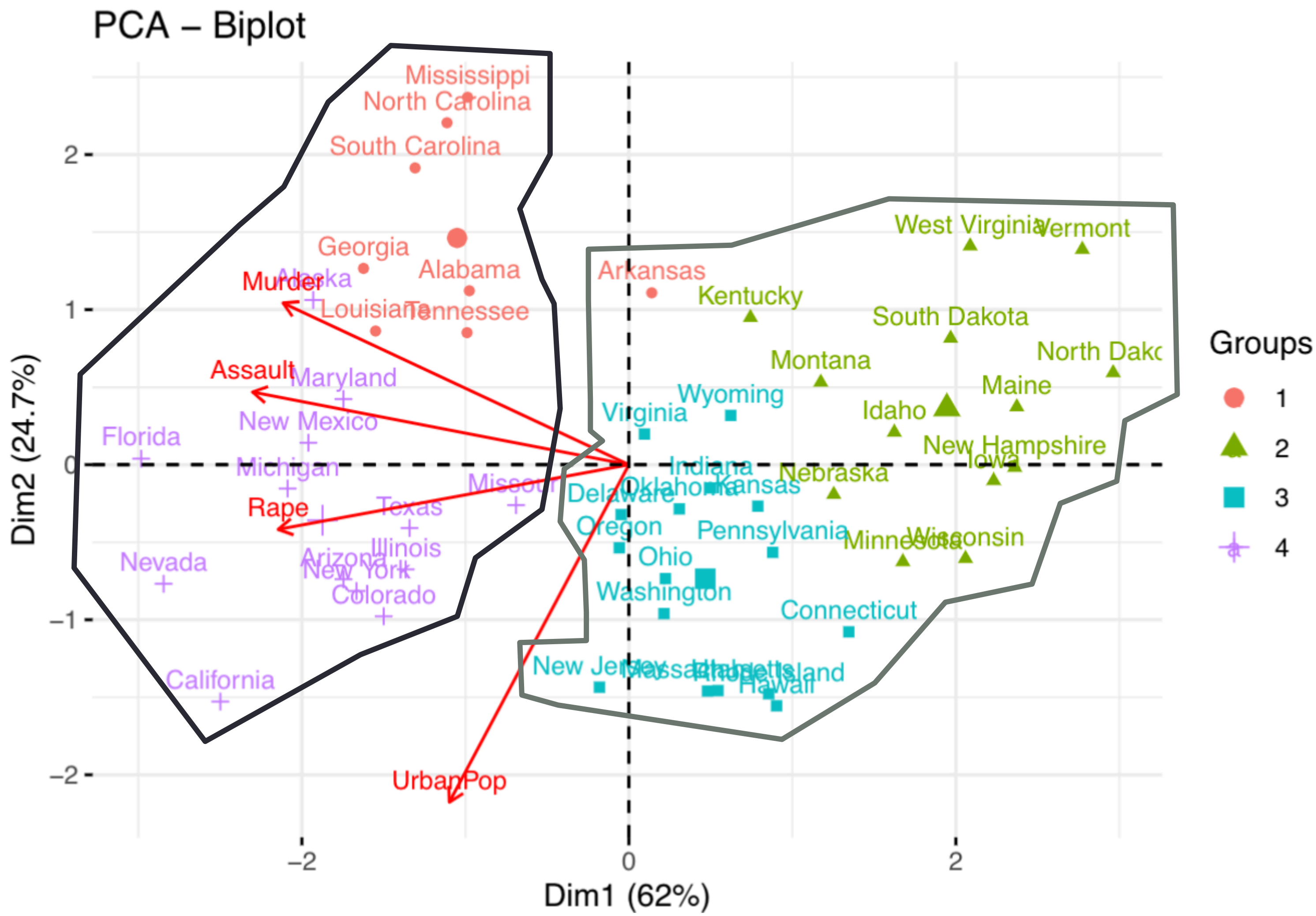
Example – USArrests final choice k = 4



Example – USArrests final choice k = 4

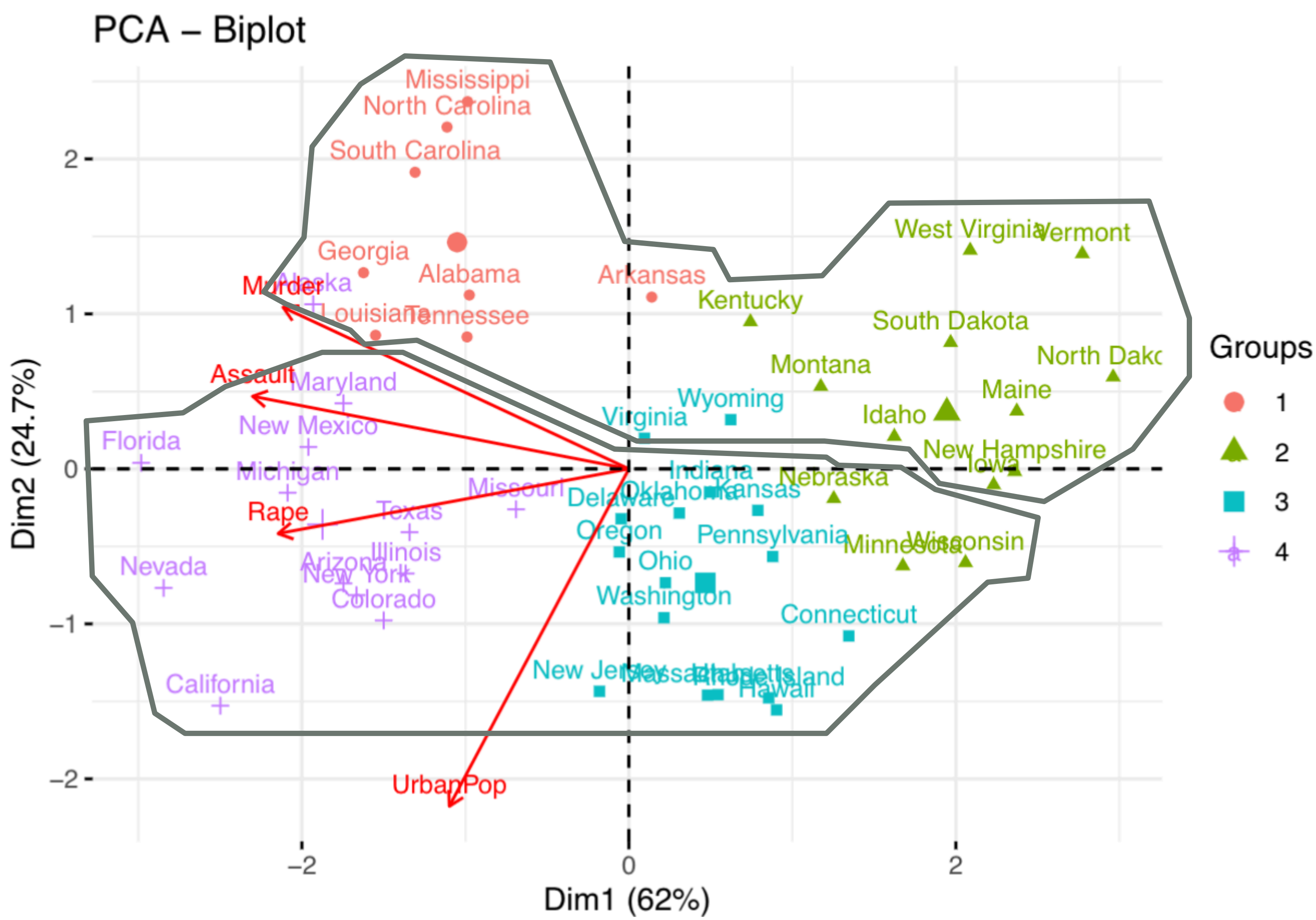


Example – USArrests final choice k = 4



Example – USArrests final choice k = 4

.



Choice of Dissimilarity Measure

- So far, we have considered using *Euclidean* distance as the dissimilarity measure
- An alternative measure that could make sense in some cases is the *correlation-based* distance

Correlation-based distance

Pearson correlation distance:

$$d_{cor}(x,y) = 1 - \frac{\sum_{i=1}^{p}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{p}(x_i - \bar{x})^2 \sum_{i=1}^{p}(y_i - \bar{y})^2}}$$

Spearman correlation distance:

The spearman correlation method computes the correlation between the rank of *x* and the rank of *y* variables.

$$d_{spear}(x,y) = 1 - \frac{\sum_{i=1}^{p}(x'_i - \bar{x}')(y'_i - \bar{y}')}{\sqrt{\sum_{i=1}^{p}(x'_i - \bar{x}')^2 \sum_{i=1}^{p}(y'_i - \bar{y}')^2}}$$

Where $x'_i = rank(x_i)$ and $y'_i = rank(y_i)$.

K-Means Clustering

Distance between two data points (rows)

Euclidean distance:

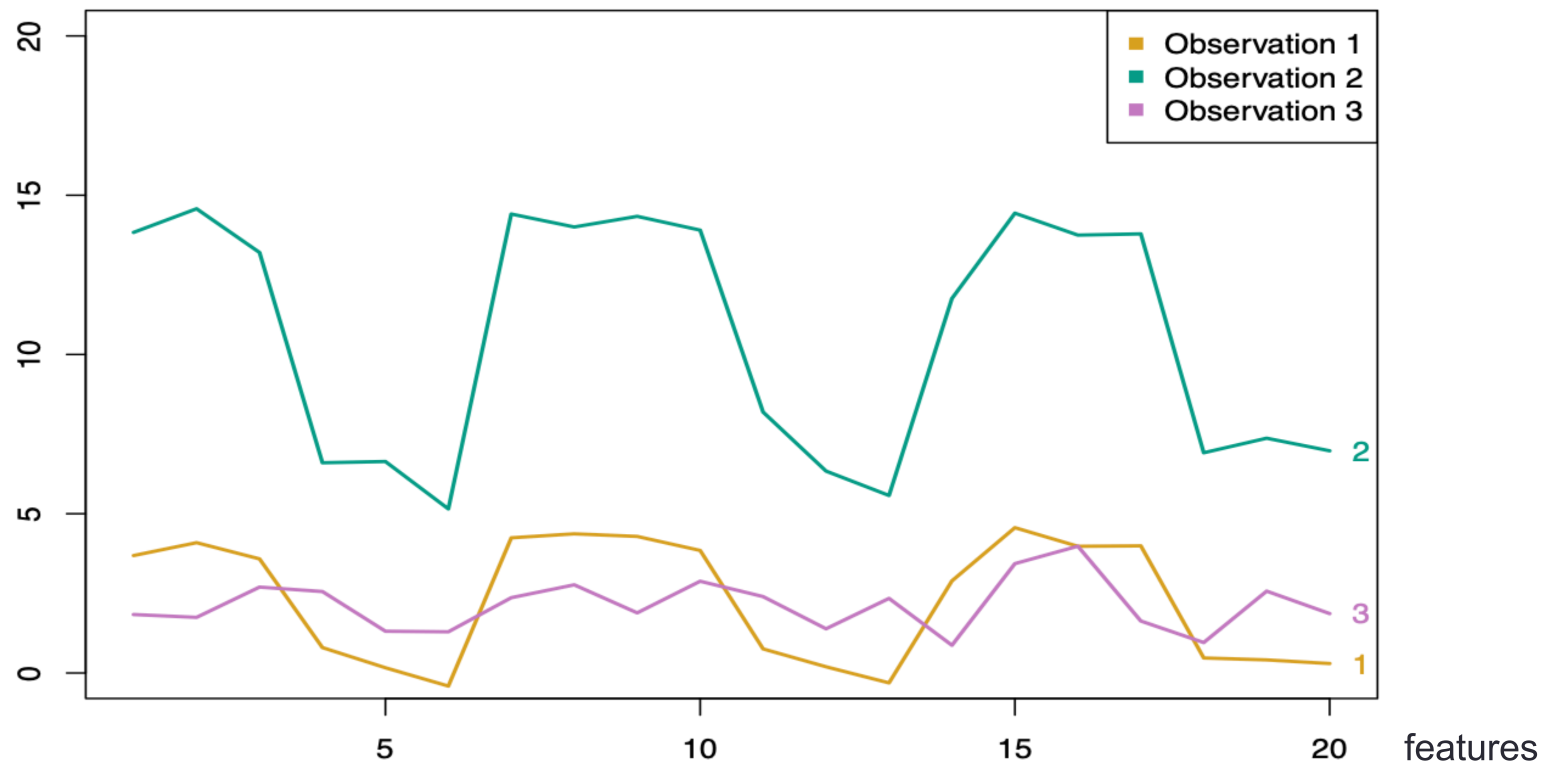
$$d_{euc}(x, y) = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}$$

Manhattan distance:

$$d_{man}(x, y) = \sum_{i=1}^{p} |(x_i - y_i)|$$

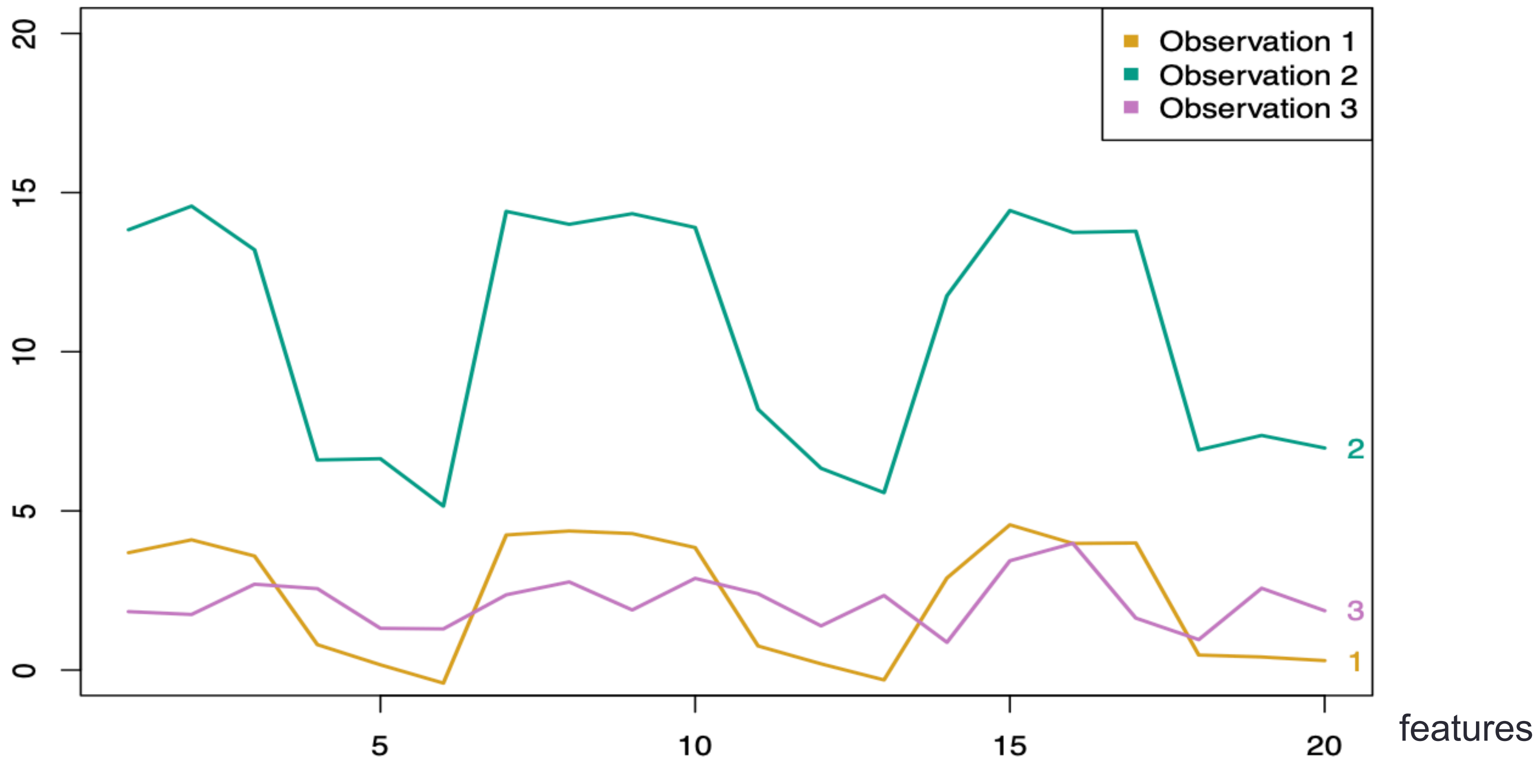
Euclidian vs Correlation-based distance

- Dataset with n = 3 observations (rows) and p=20 features
- Observations 1 and 3 have similar values for each feature, therefore there is a small distance between them



Euclidian vs Correlation-based distance

- Observations 1 and 3 are weakly correlated, therefore they should have a large correlation-based distance
- Observations 1 and 2 are highly correlated, and would be considered similar in terms of correlation measure



Euclidian vs Correlation-based distance

- Consider the number of purchases of each item (columns) for many customers (rows)
- Using Euclidean distance, customers who have purchases of similar dollar amount would be clustered together
- Using a correlation distance, customers who tend to purchase the same types of products will be clustered together (even if the dollar amount of the purchase is different)

Correlation-based distance

```
d0 = data.frame(x)
head(d0)
```

```
##            X1         X2         X3
## 1 -0.89691455  0.7389386 -1.7882422
## 2  0.18484918  0.3189604  2.0312425
## 3  1.58784533  1.0761644 -0.7031443
## 4 -1.13037567 -0.2841577  0.1581648
## 5 -0.08025176 -0.7766753  0.5062348
## 6  0.13242028 -0.5956605 -0.8199951
```

```
#
# correlations between cols (3x3 matrix)
#
dim(x)
```

```
## [1] 30  3
```

```
cor(x)
```

```
##              [,1]       [,2]         [,3]
## [1,]  1.000000000 -0.1478786  0.004264259
## [2,] -0.147878604  1.0000000 -0.128277680
## [3,]  0.004264259 -0.1282777  1.000000000
```

Correlation-based distance

```
# correlations between rows (30x30 matrix)
#
aux = cor(t(x))
dim(aux)
```

```
## [1] 30 30
```

```
aux[1:5,1:5]
```

```
##            [,1]       [,2]       [,3]       [,4]       [,5]
## [1,]  1.0000000 -0.7267216  0.6165294 -0.1751773 -0.9929226
## [2,] -0.7267216  1.0000000 -0.9888888  0.8036151  0.8031607
## [3,]  0.6165294 -0.9888888  1.0000000 -0.8831592 -0.7056721
## [4,] -0.1751773  0.8036151 -0.8831592  1.0000000  0.2908643
## [5,] -0.9929226  0.8031607 -0.7056721  0.2908643  1.0000000
```

Correlation-based distance

```
# adjust correlations by substracting from 1
#
dd = 1-cor(t(x))
dim(dd)
```

```
## [1] 30 30
```

```
dd[1:5,1:5]
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 0.0000000 1.7267216 0.3834706 1.1751773 1.9929226
## [2,] 1.7267216 0.0000000 1.9888888 0.1963849 0.1968393
## [3,] 0.3834706 1.9888888 0.0000000 1.8831592 1.7056721
## [4,] 1.1751773 0.1963849 1.8831592 0.0000000 0.7091357
## [5,] 1.9929226 0.1968393 1.7056721 0.7091357 0.0000000
```

```
dd=as.dist(dd)
head(dd)
```

```
## [1] 1.7267216 0.3834706 1.1751773 1.9929226 0.9412891 1.0767251
```