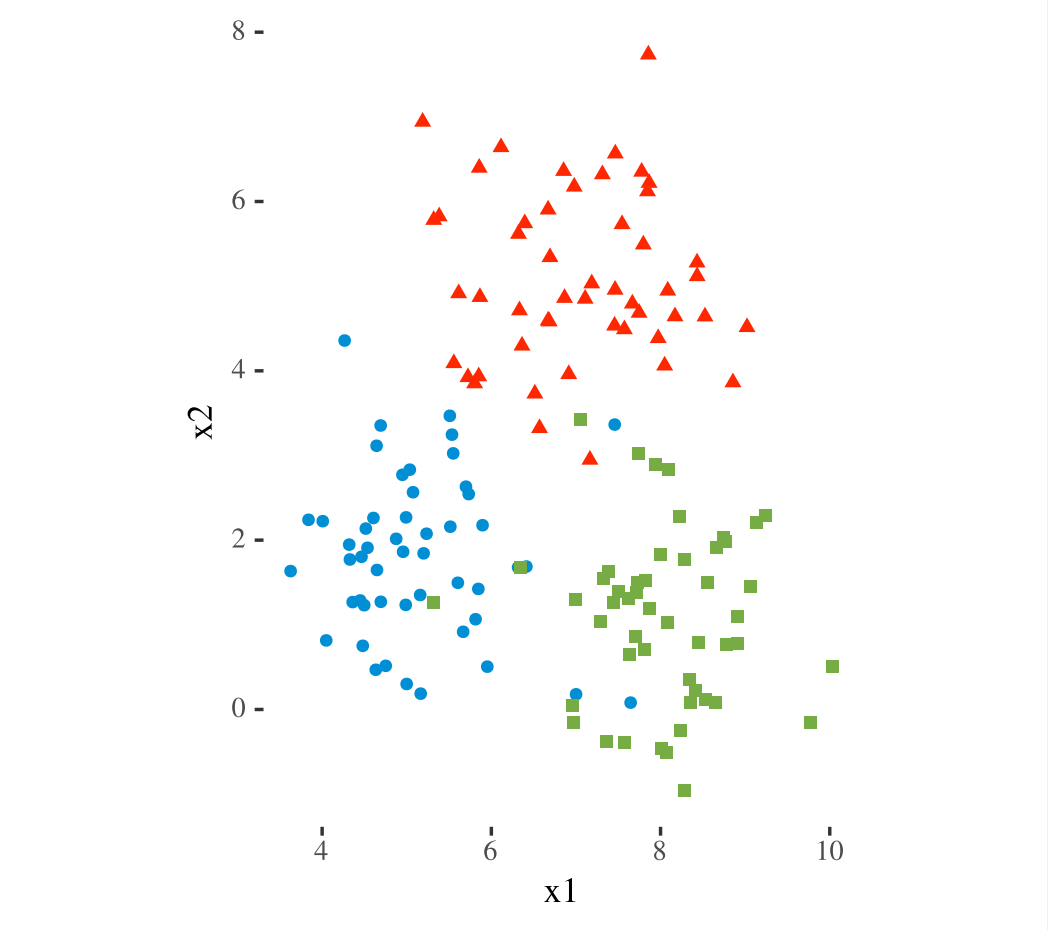
Intro to Clustering

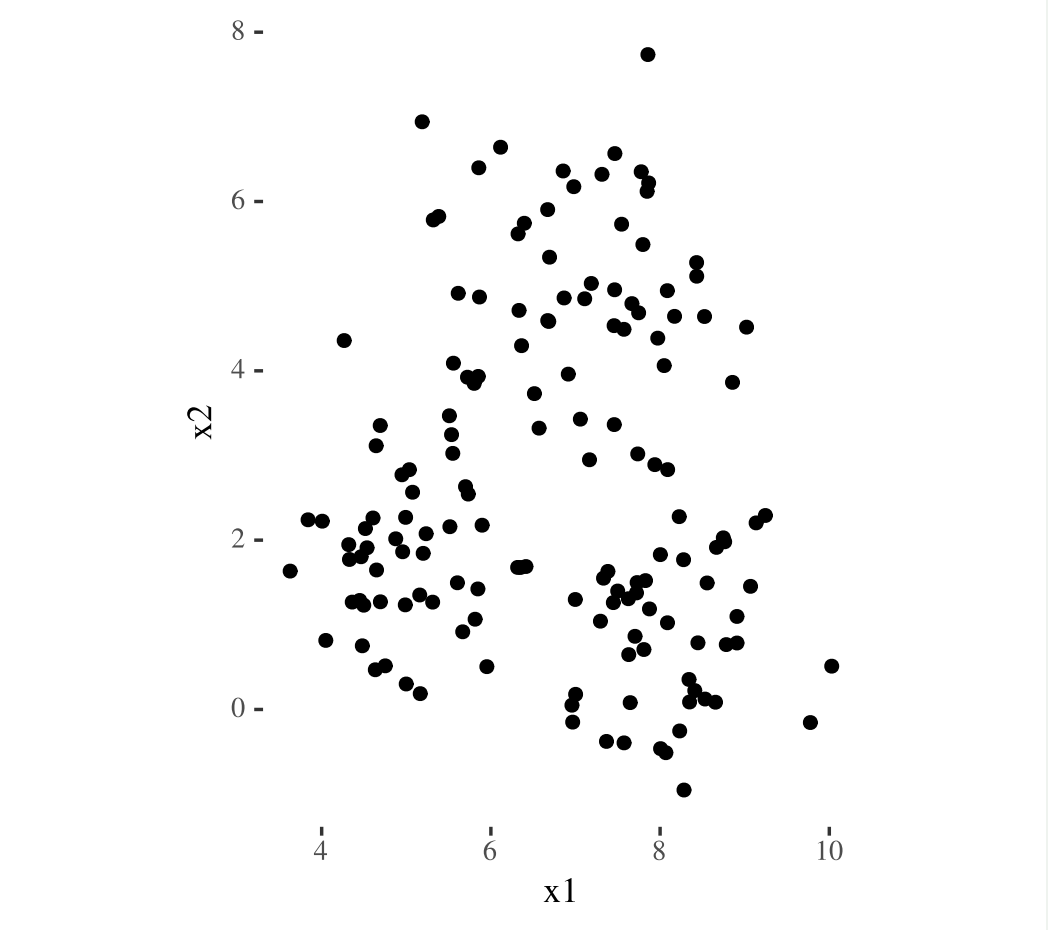
Stat 220

Bastola

March 04 2022

Supervised to unsupervised





K-means Basics

- Algorithm to group data into K clusters
- Starts with an initial clustering of data
- Iteratively improves the cluster assignments
- Stops until the assignments cannot be improved further

Algorithm

1. Randomly assign a number, from 1 to K, to each of the observations
2. Compute the centroid of each of the K clusters
3. Assign each point to the nearest centroid and redefine the cluster
4. Repeat steps 2 and 3 until no point change clusters

Main Idea

To minimize the total within cluster variation

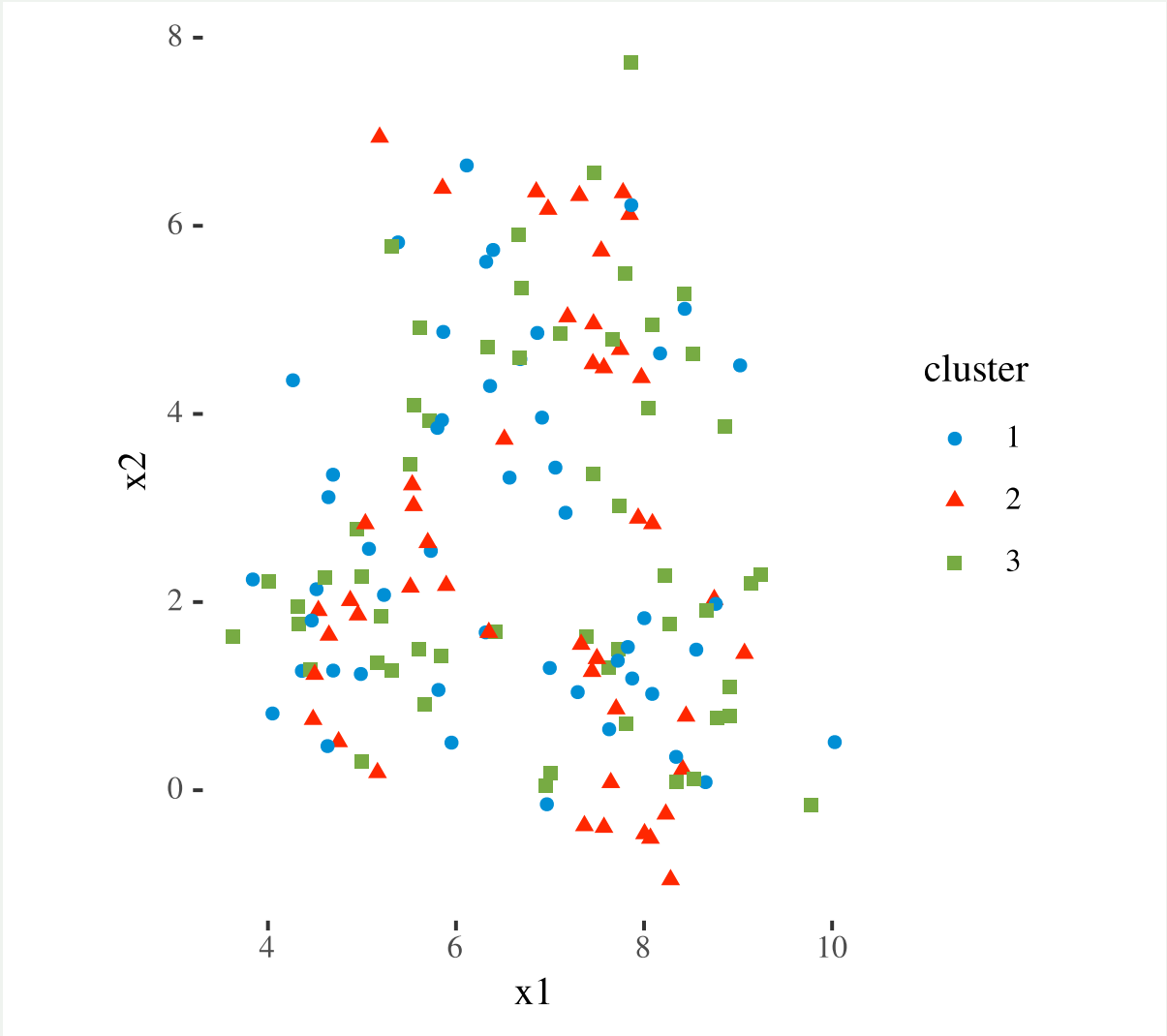
The total within-cluster variation is the sum of squared Euclidean distances between items and the corresponding centroid:

$$W = \sum_{k=1}^{K} W(C_k) = \sum_{k=1}^{K} \sum_{x_i \in C_k} (x_i - \mu_k)^2$$

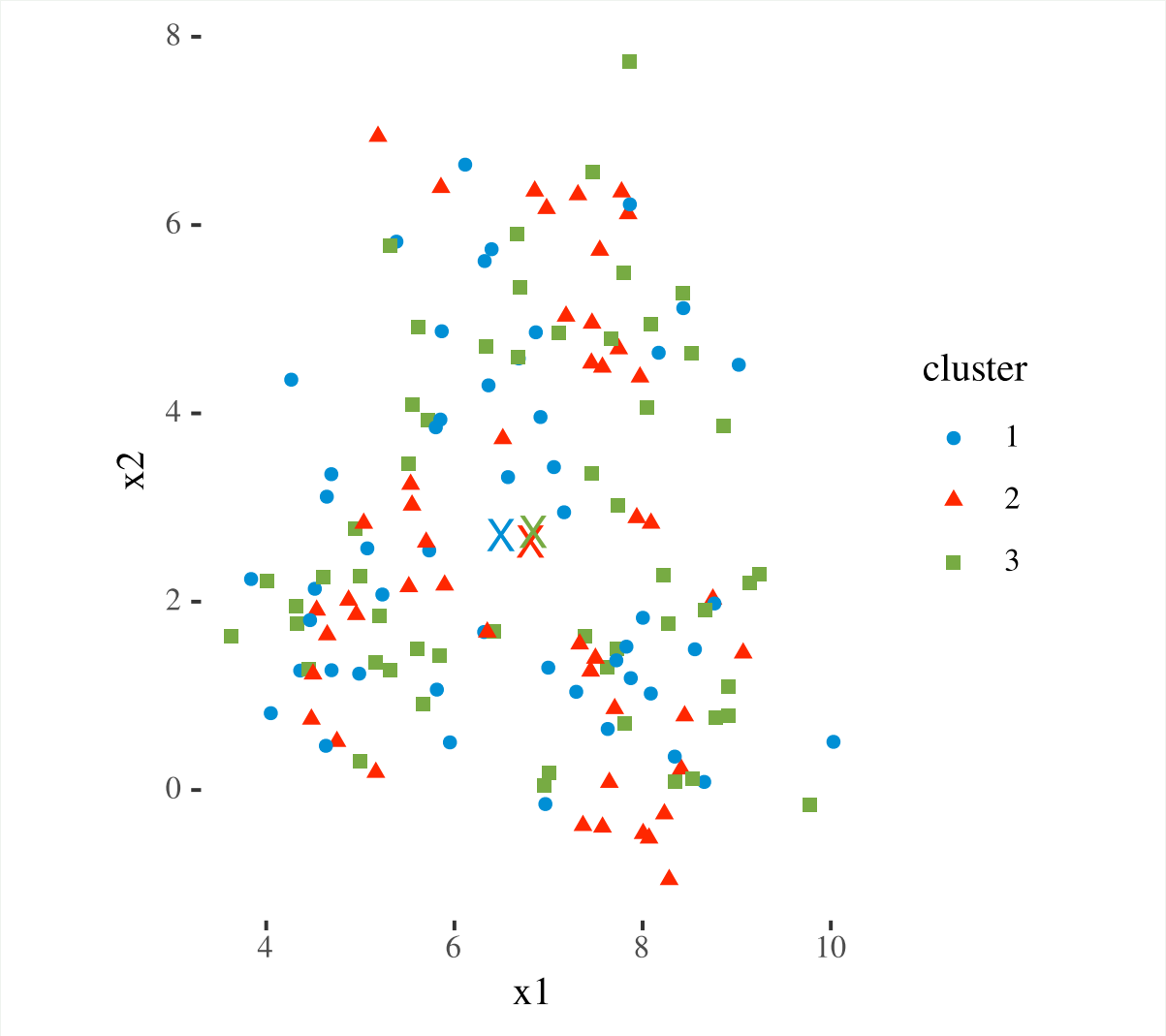
where:

- x_i is a data point in the cluster C_k
- μ_k is the mean value of the points assigned to the cluster C_k

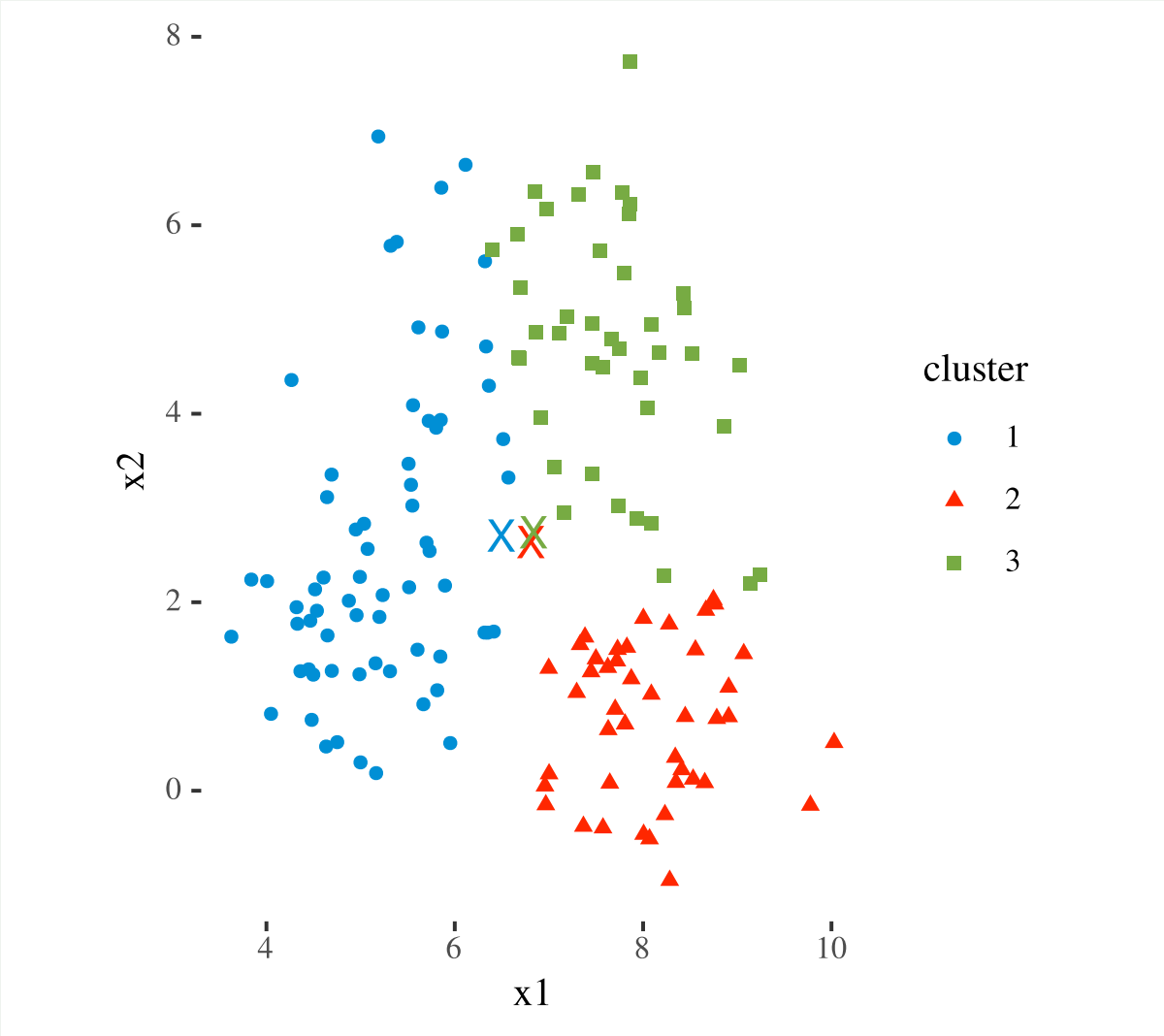
Randomly assign a number, from 1 to *K*, to each of the observations



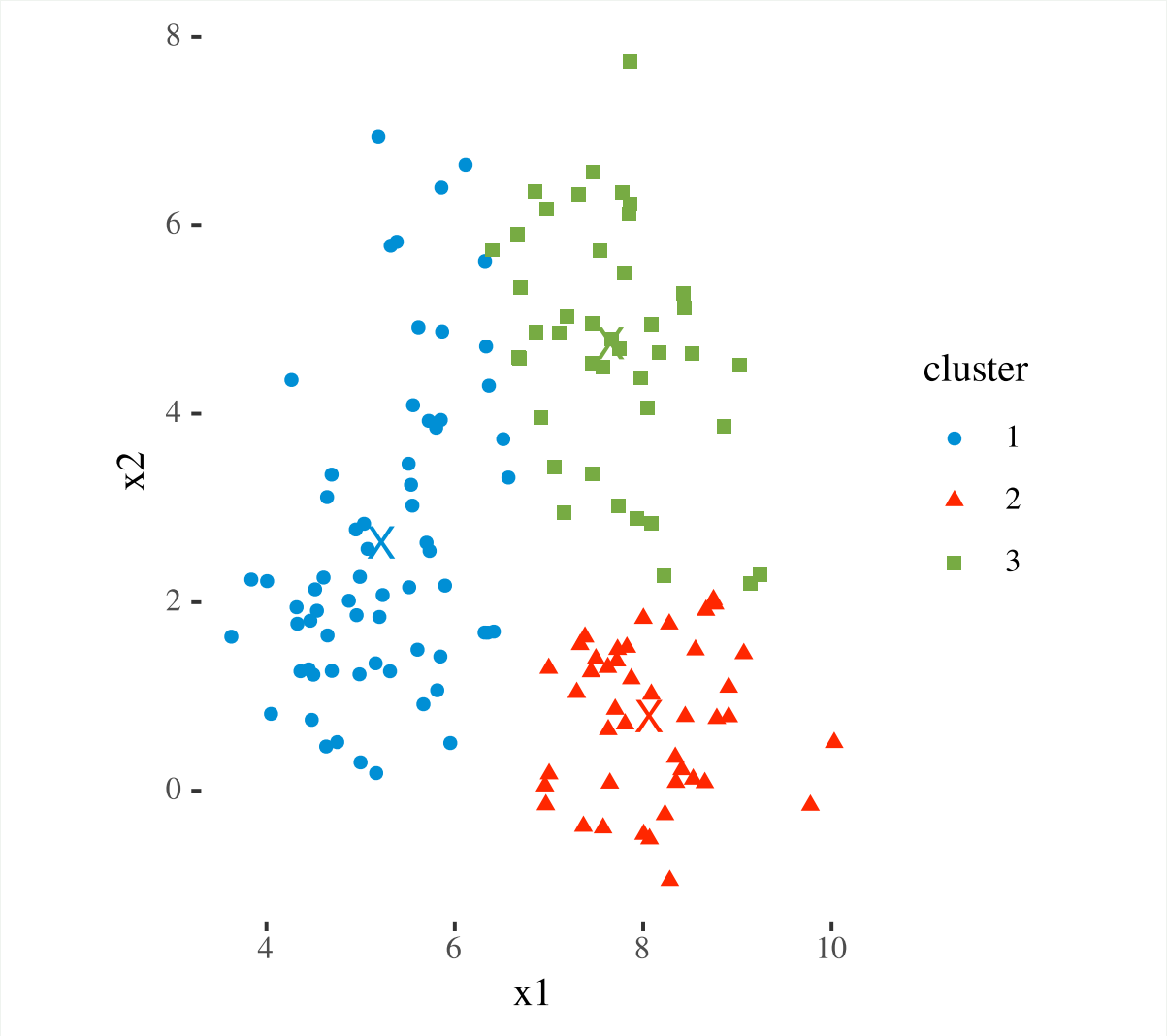
Compute the centroid of each cluster



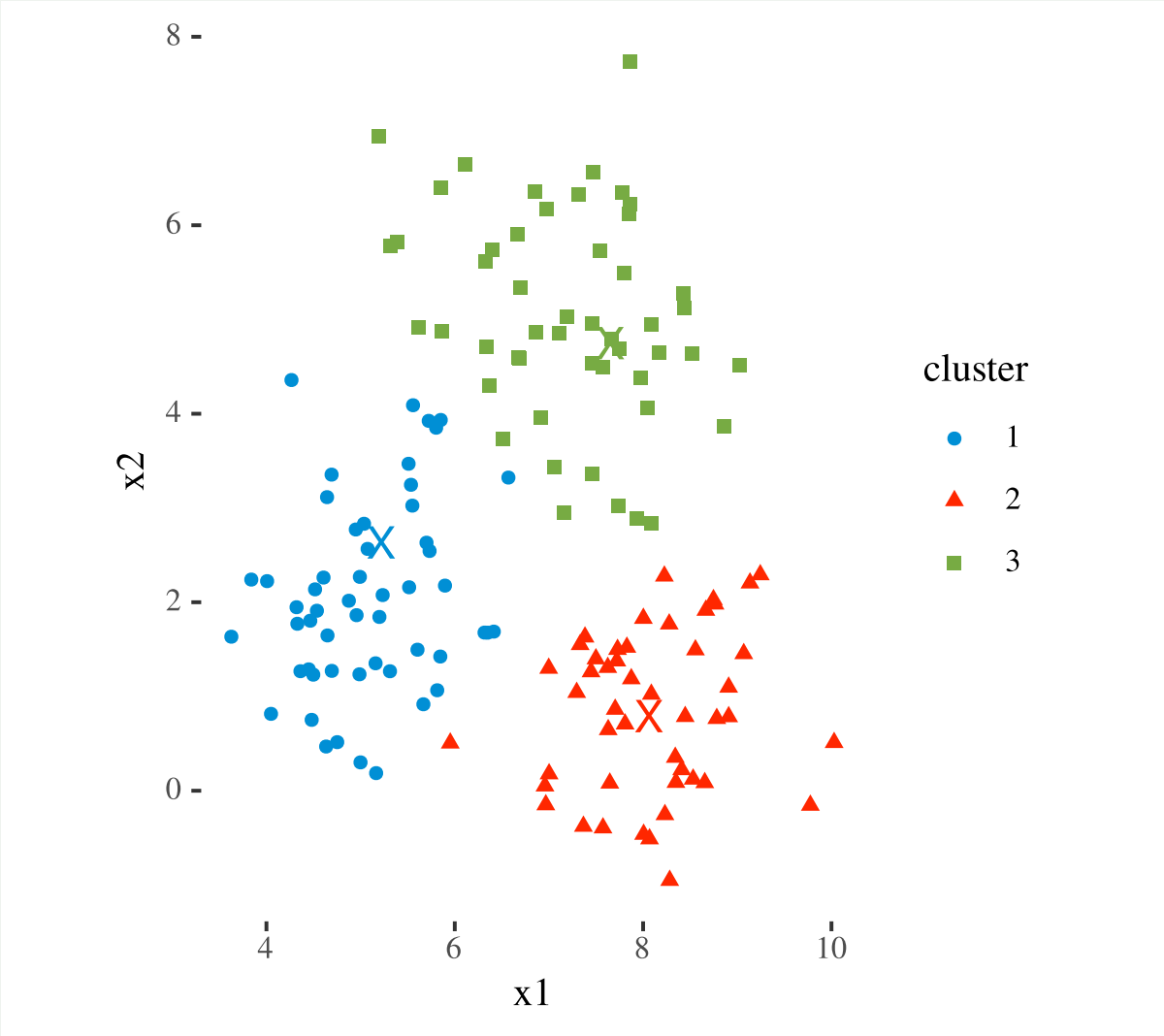
Re-assign each observation to the cluster whose centroid is closest



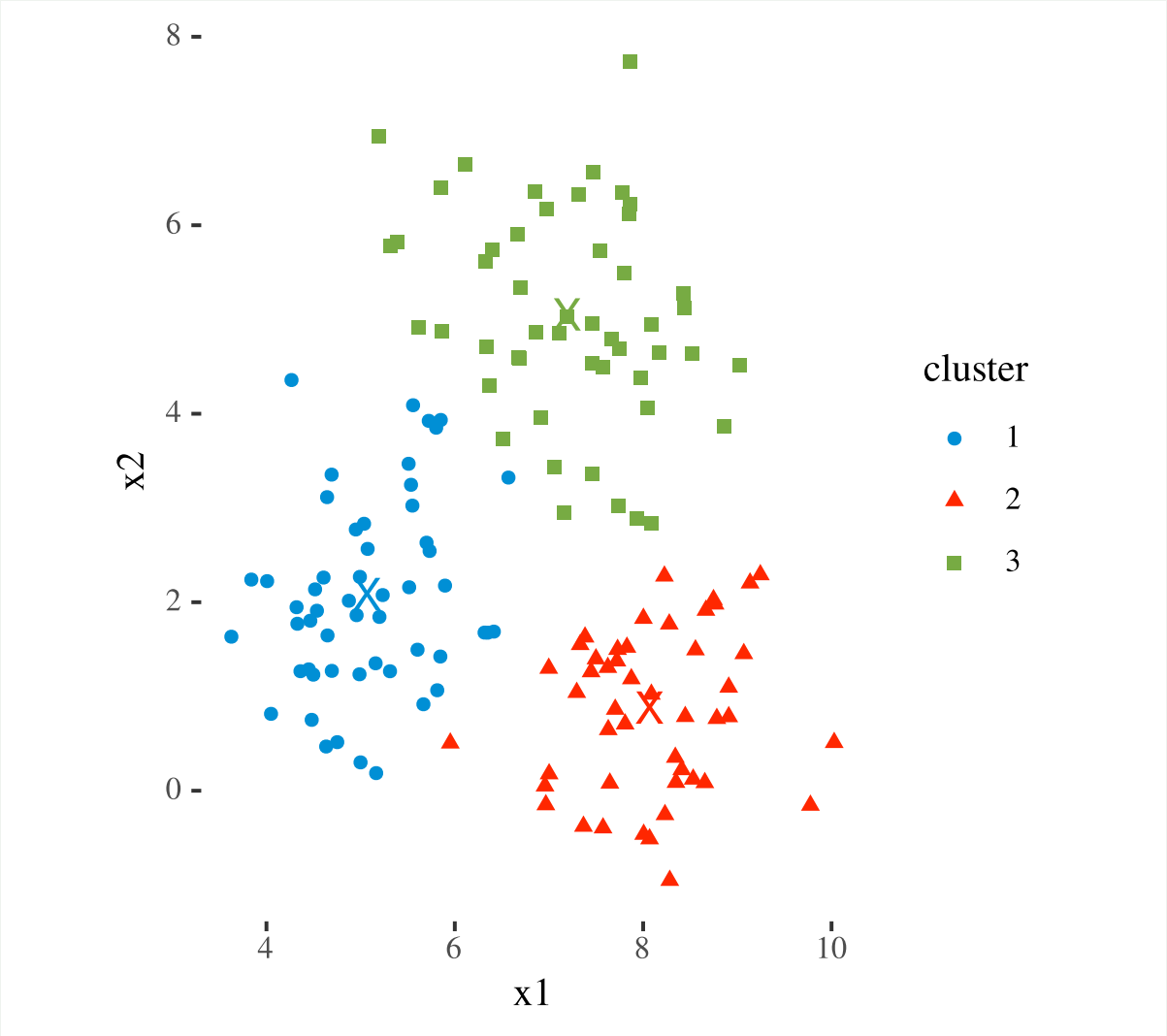
Re-compute the centroid of each cluster



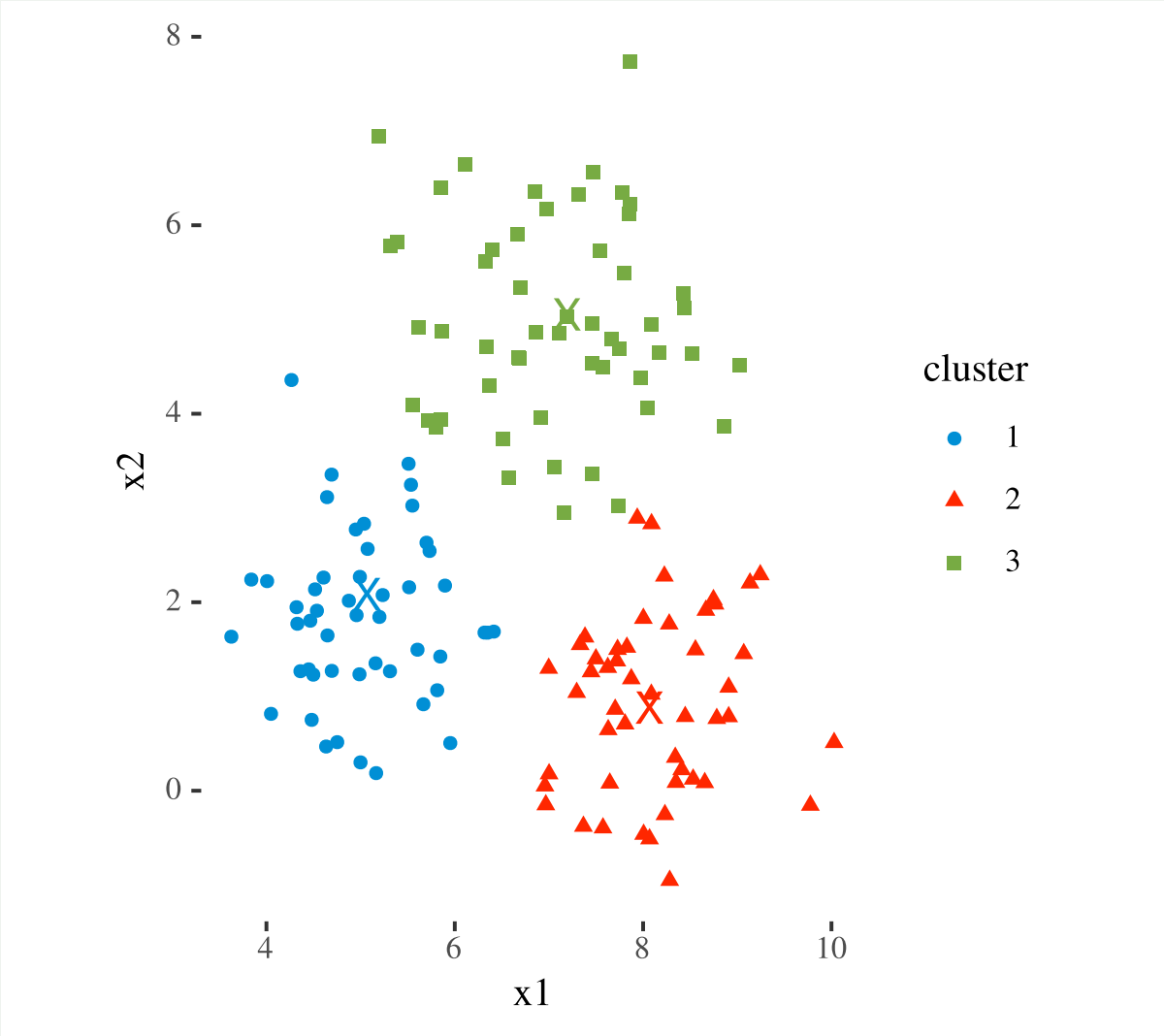
Re-assign each observation to the cluster whose centroid is closest



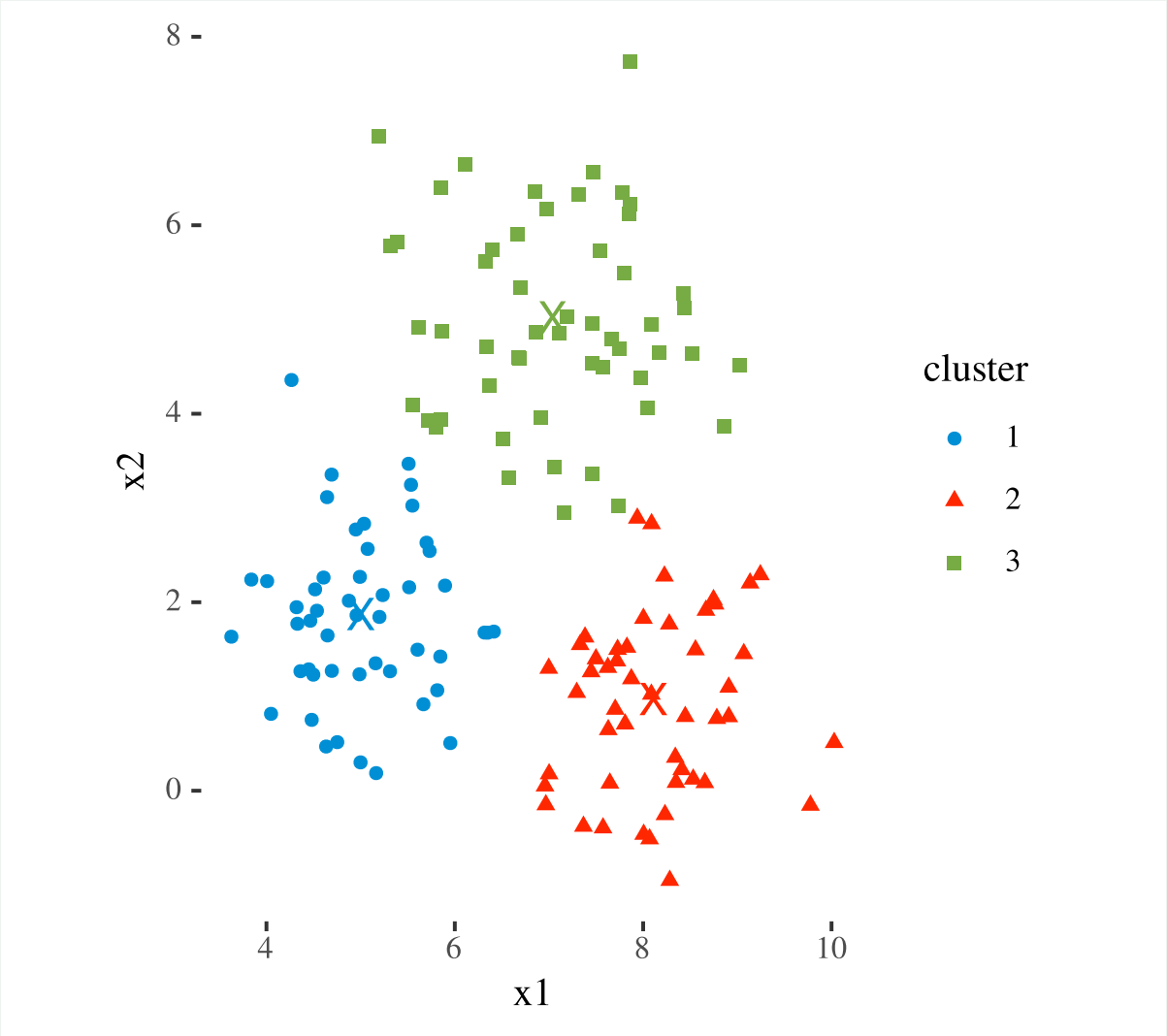
Re-compute the centroid of each cluster



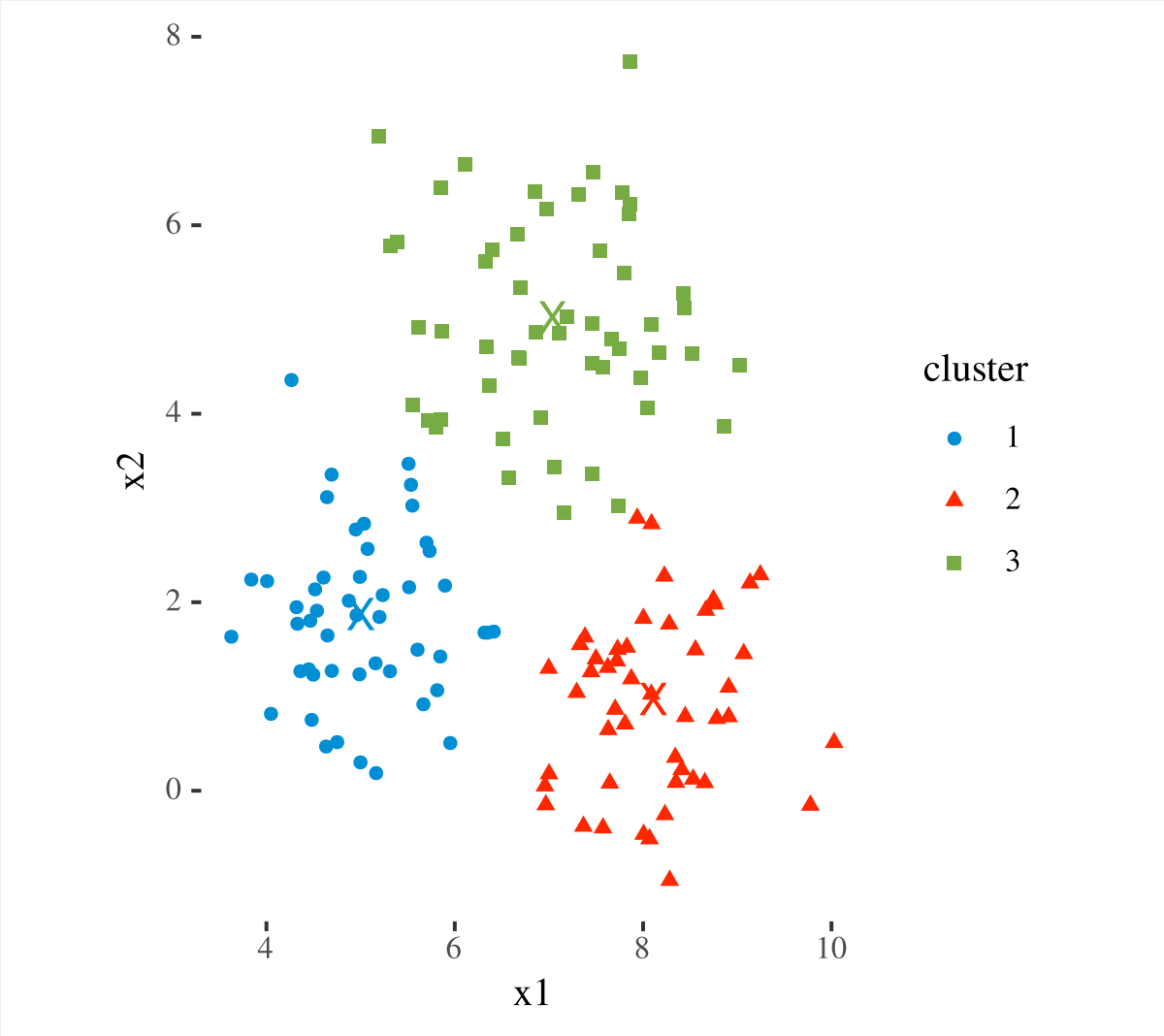
Re-assign each observation to the cluster whose centroid is closest



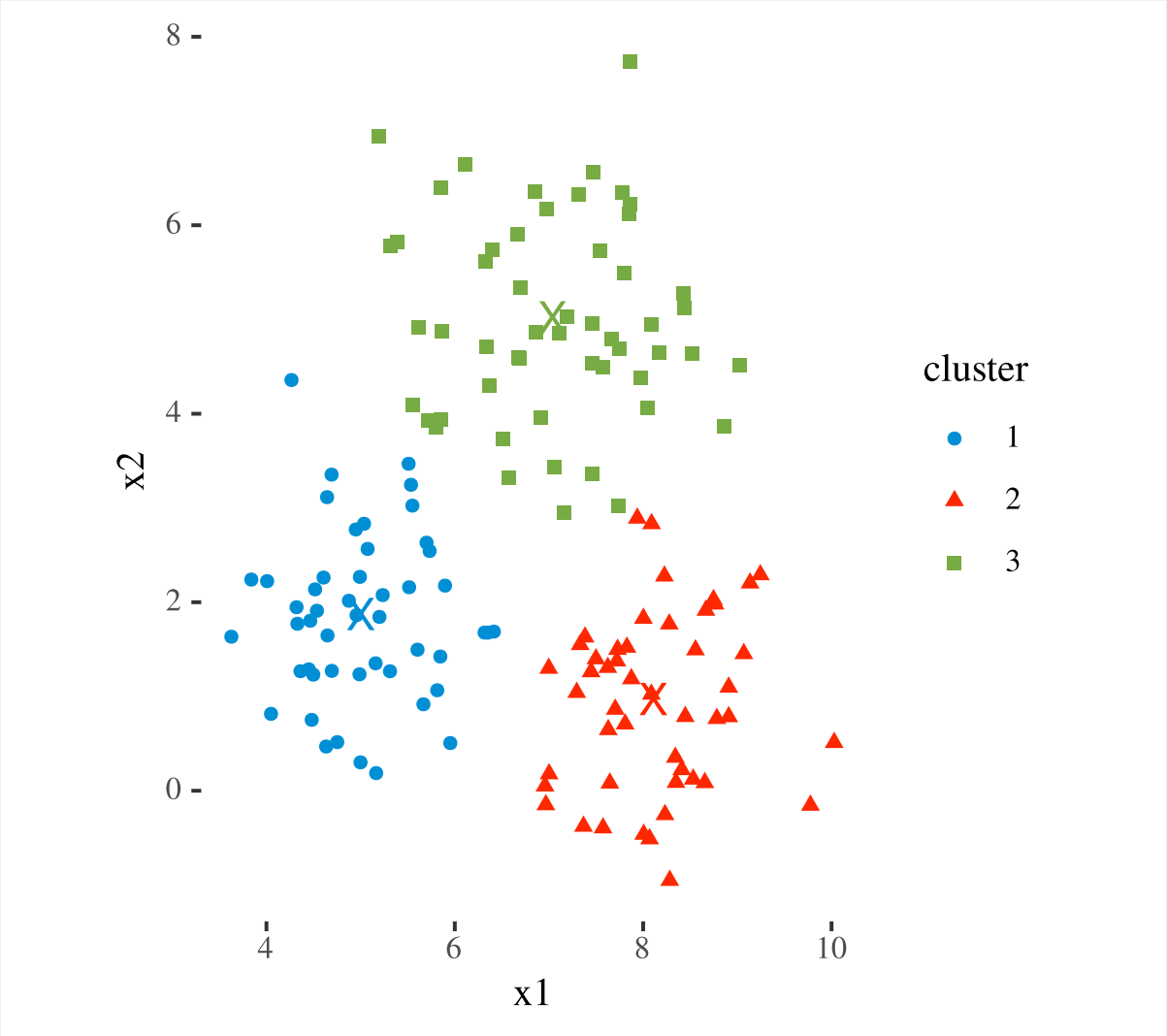
Re-compute the centroid of each cluster



Re-assign each observation to the cluster whose centroid is closest



Re-compute the centroid of each cluster



USArrests

```
USAData <- as_tibble(USArrests, rownames = "state") %>% na.omit() %>%
  column_to_rownames("state") %>%
  select(Murder, UrbanPop)
```

```
head(USAData, 10)
            Murder UrbanPop
Alabama       13.2       58
Alaska        10.0       48
Arizona        8.1       80
Arkansas       8.8       50
California     9.0       91
Colorado       7.9       78
Connecticut    3.3       77
Delaware       5.9       72
Florida       15.4       80
Georgia       17.4       60
```

Means and standard deviations

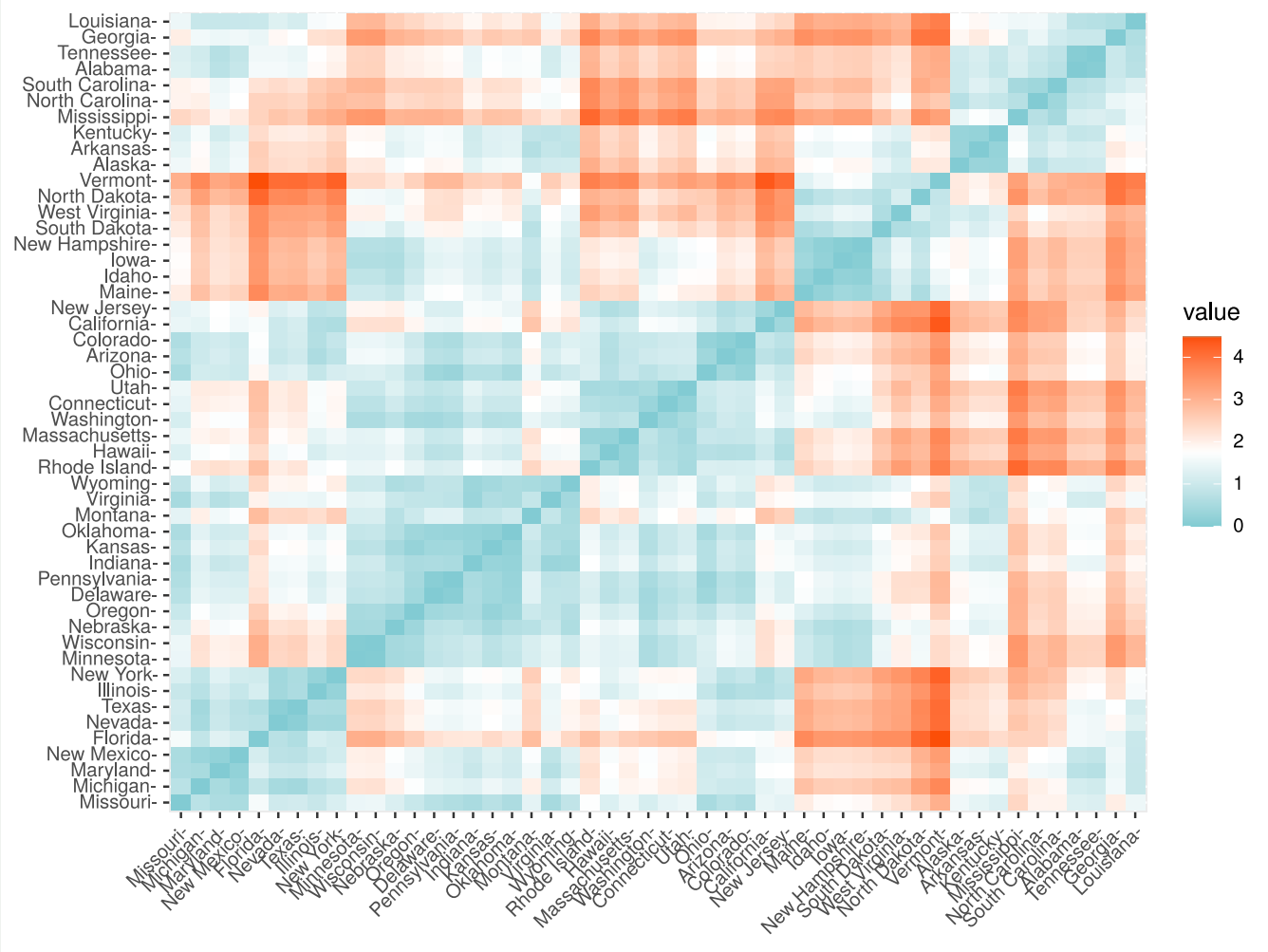
```
USAData %>%
  map_dfr(mean)
# A tibble: 1 × 2
  Murder UrbanPop
   <dbl>    <dbl>
1   7.79     65.5
```

```
USAData %>%
  map_dfr(sd)
# A tibble: 1 × 2
  Murder UrbanPop
   <dbl>    <dbl>
1   4.36     14.5
```

Standardize the data

```
USAData <- USAData %>% mutate(across(where(is.numeric), standardize))
```

```
head(USAData,10)
                Murder   UrbanPop
Alabama      1.24256408 -0.5209066
Alaska       0.50786248 -1.2117642
Arizona      0.07163341  0.9989801
Arkansas     0.23234938 -1.0735927
California   0.27826823  1.7589234
Colorado     0.02571456  0.8608085
Connecticut -1.03041900  0.7917228
Delaware    -0.43347395  0.4462940
Florida      1.74767144  0.9989801
Georgia      2.20685994 -0.3827351
```



Euclidean Distances

So, how do we fit all of this in R?

`kmeans()`

- `kmeans()` function takes a matrix or data-frame or tibble and the number of centers/clusters we want to find.
- We also set `nstart = 20-25` to have multiple initial starting positions in the hope of finding global optimal solution instead of local optimal solution
- Use `set.seed()` for reproducibility

Within Cluster Sum of Squared Errors (WSS)

- Calculate WSS for different values of K.
- Choose K for which WSS first starts to diminish.
- Visually deciphered with an **elbow graph**.
- The number of clusters is taken at the elbow joint point.

K-means

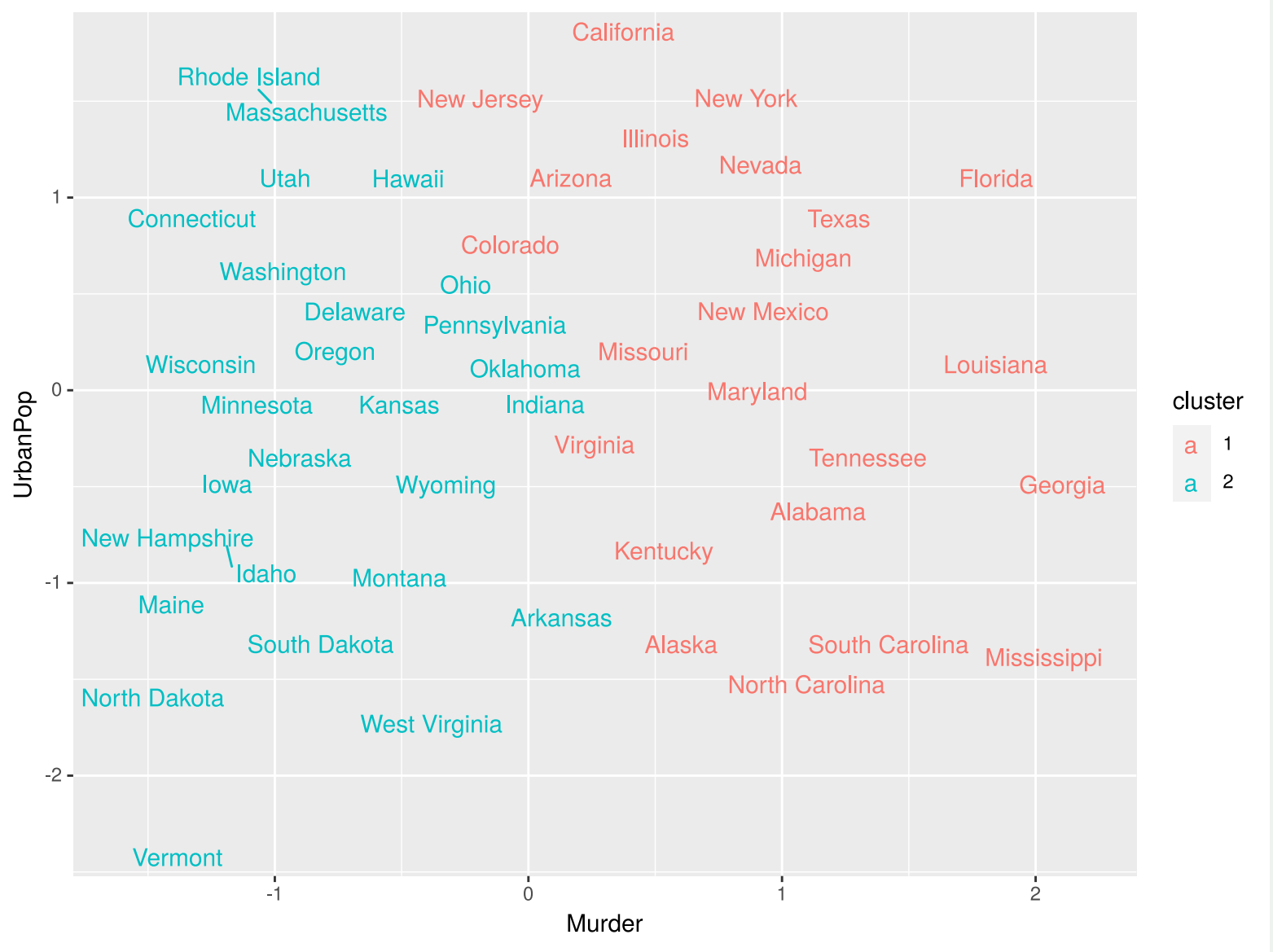
```
set.seed(1234)
k.means <- kmeans(USAData, centers = 2, nstart = 25)
```

```
k.means %>% tidy()
# A tibble: 2 × 5
  Murder UrbanPop  size withinss cluster
   <dbl>    <dbl> <int>    <dbl> <fct>
1  0.896    0.194    23     31.6 1
2 -0.763   -0.165    27     30.6 2
```

```
glance(k.means)
# A tibble: 1 × 4
  totss tot.withinss betweenss  iter
  <dbl>        <dbl>     <dbl> <int>
1    98         62.2      35.8     1
```

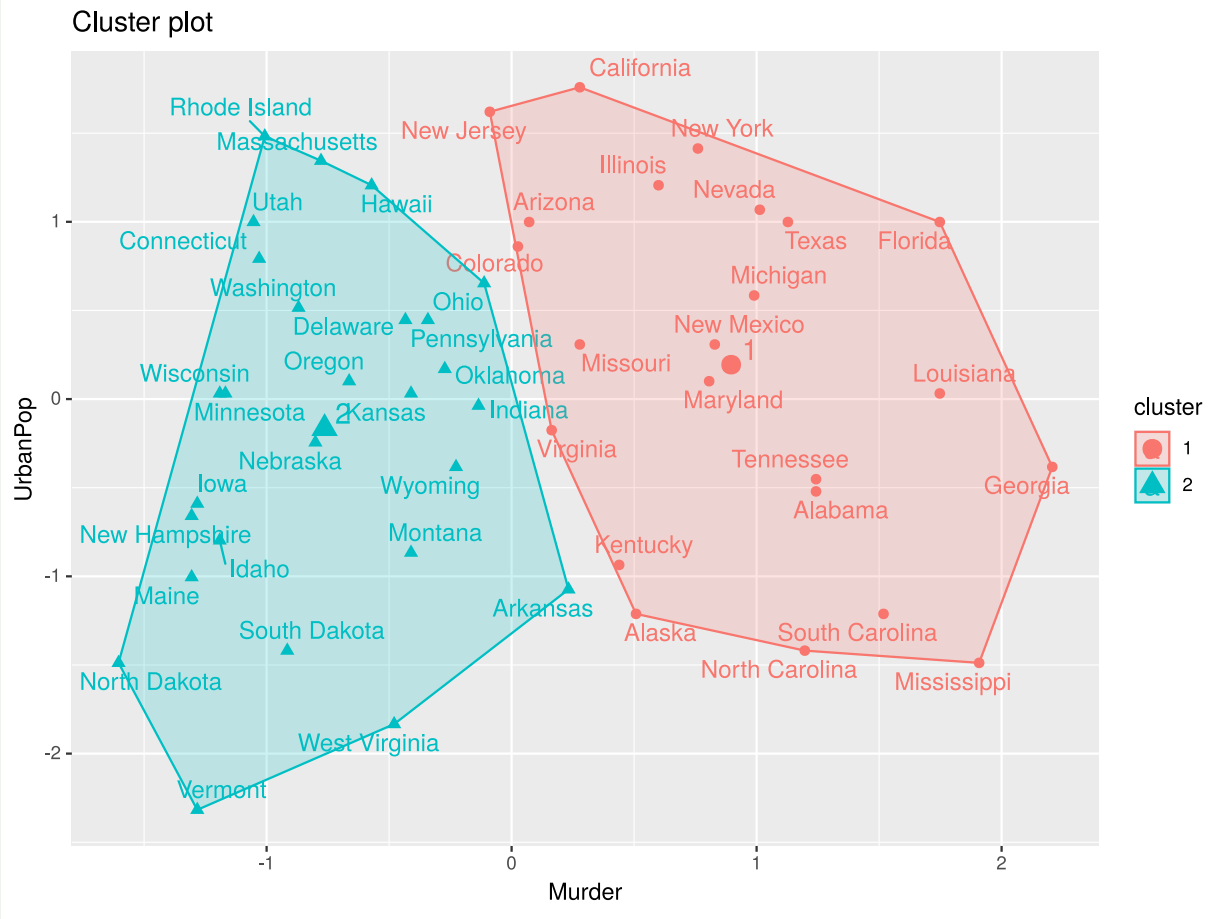
augment from broom package

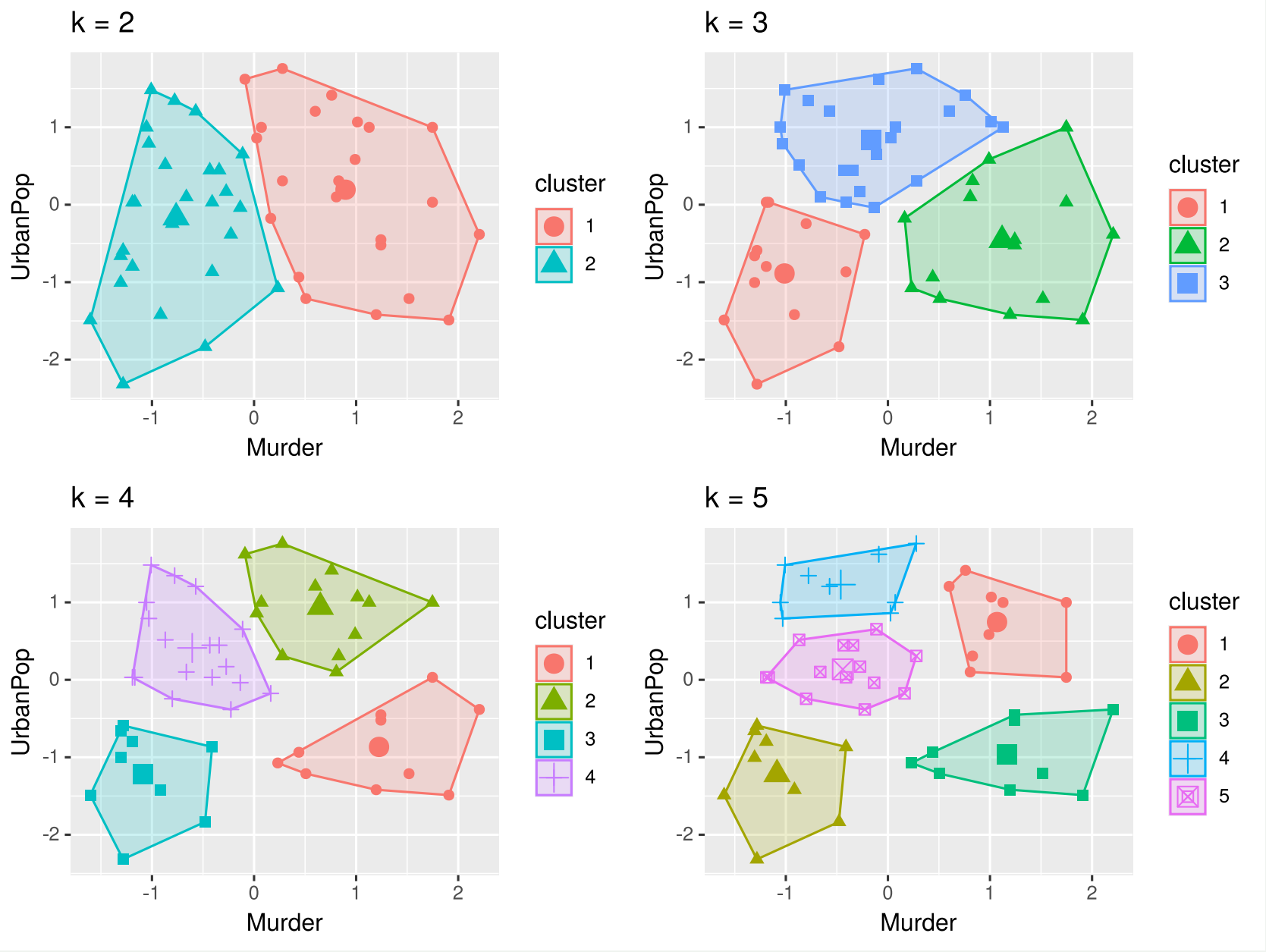
```
augment(k.means, data = USAData)
# A tibble: 50 × 4
   .rownames    Murder UrbanPop .cluster
   <chr>         <dbl>    <dbl> <fct>
 1 Alabama       1.24    -0.521 1
 2 Alaska        0.508   -1.21  1
 3 Arizona       0.0716   0.999 1
 4 Arkansas      0.232   -1.07  2
 5 California    0.278    1.76  1
 6 Colorado      0.0257   0.861 1
 7 Connecticut  -1.03     0.792 2
 8 Delaware     -0.433    0.446 2
 9 Florida       1.75     0.999 1
10 Georgia       2.21    -0.383 1
# … with 40 more rows
```


California
Rhode Island
Massachusetts
New Jersey
New York
Illinois
Nevada
Utah
Hawaii
Arizona
Florida
Connecticut
Texas
Colorado
Michigan
Washington
Ohio
Delaware
Pennsylvania
New Mexico
Oregon
Missouri
Wisconsin
Oklahoma
Louisiana
Maryland
Minnesota
Kansas
Indiana
Virginia
Nebraska
Tennessee
Iowa
Wyoming
Georgia
Alabama
New Hampshire
Kentucky
Idaho
Montana
Maine
Arkansas
South Dakota
Alaska
South Carolina
Mississippi
North Carolina
North Dakota
West Virginia
Vermont
UrbanPop
1
0
-1
-2
-1
0
1
2
Murder
cluster
a 1
a 2

In-built function for visuals

```
library(factoextra)
fviz_cluster(k.means, data = USAData, repel = TRUE)
```




k = 2
k = 3
k = 4
k = 5
UrbanPop
Murder
cluster
1
2
3
4
5

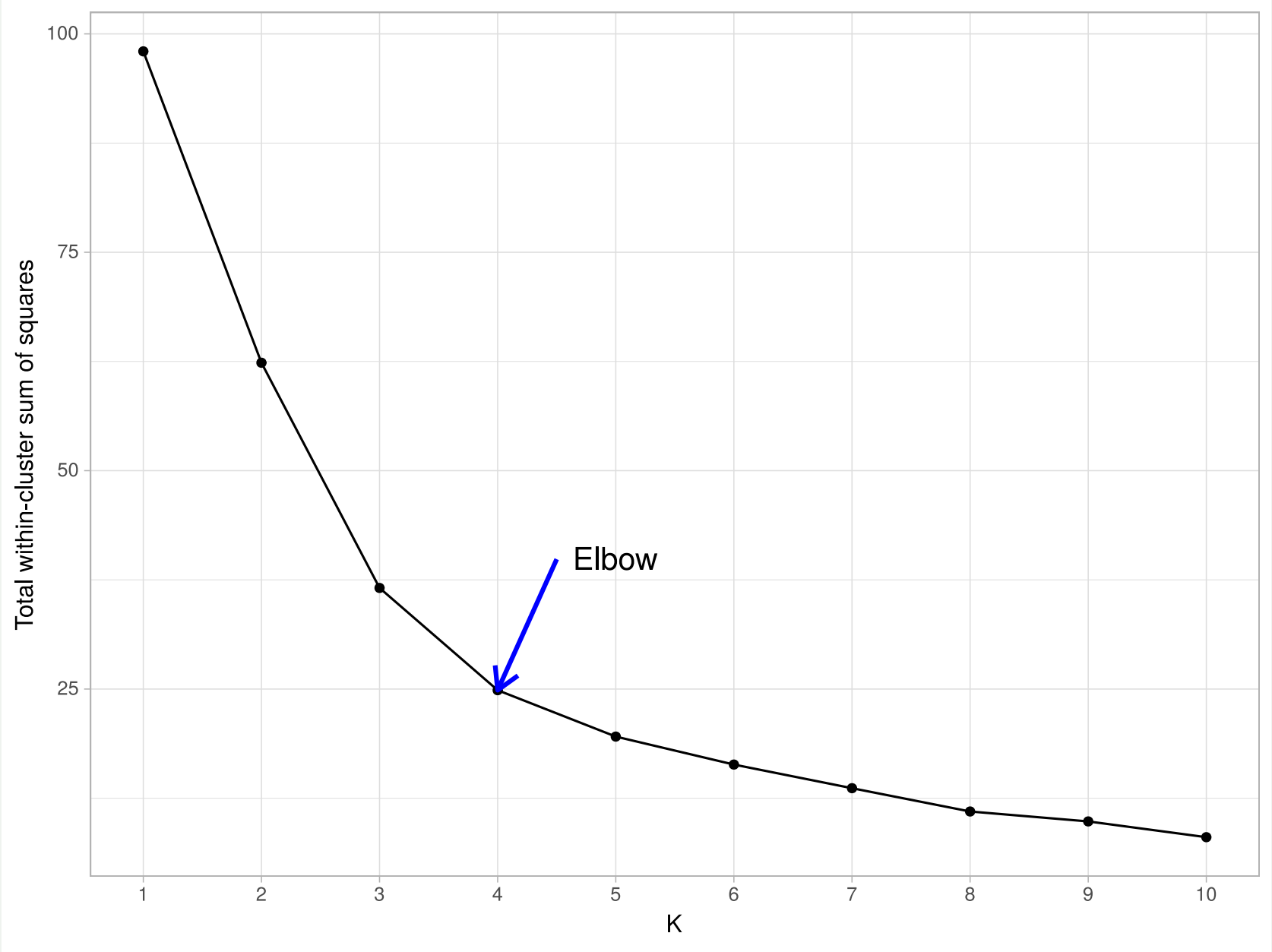
Visuals do not tell all the story

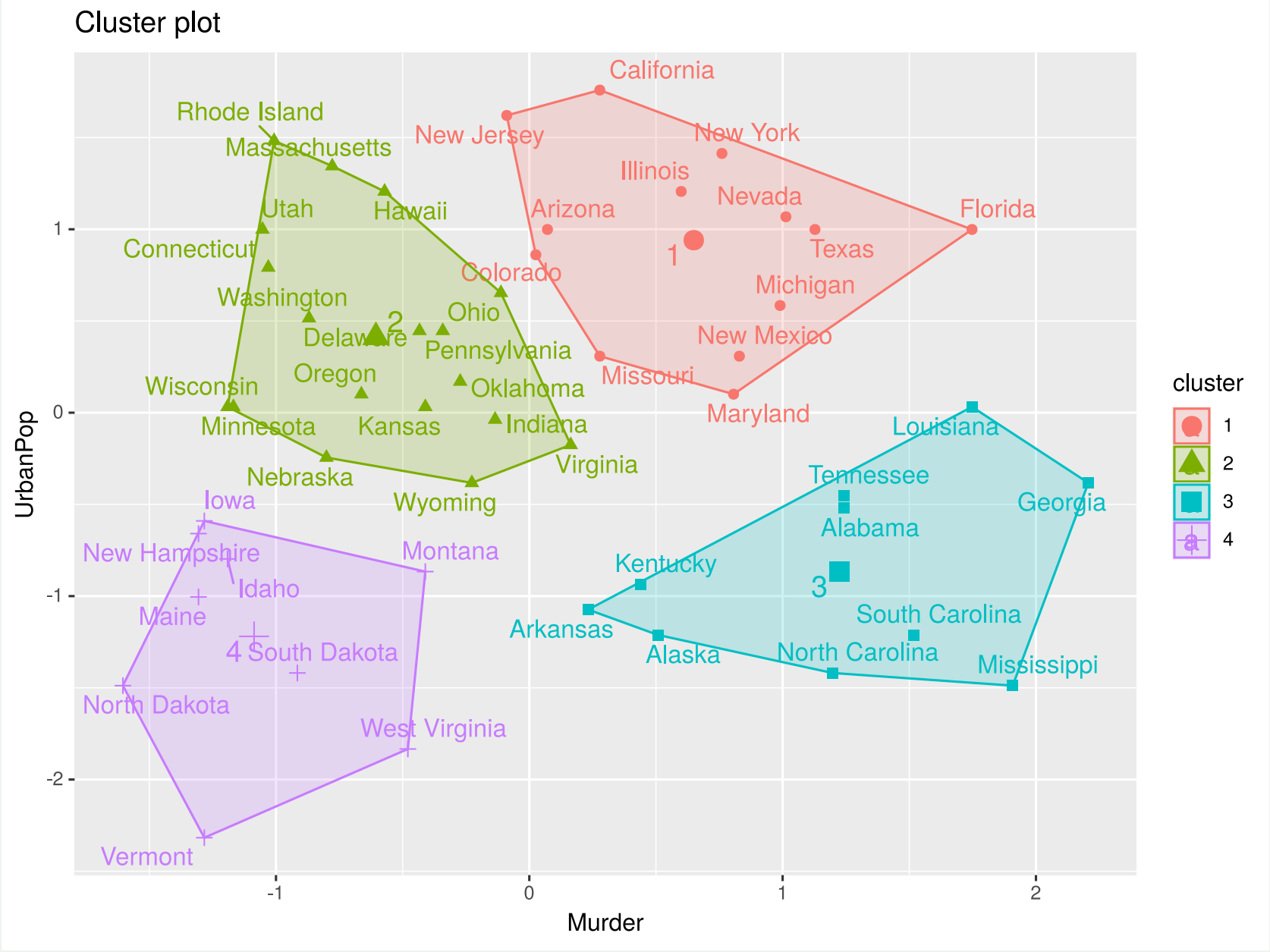
Visuals tell us where the true delineations occur, but do not tell us what the optimal number of clusters is.

Determine the optimal number of clusters

```
set.seed(1234)
multi_kmeans <- tibble(k = 1:10) %>%
  mutate(
    model = purrr::map(k, ~ kmeans(USAData, centers = .x, nstart = 25)),
    tot.withinss = purrr::map_dbl(model, ~ glance(.x)$tot.withinss)
  )

multi_kmeans
# A tibble: 10 × 3
       k model    tot.withinss
   <int> <list>           <dbl>
 1     1 <kmeans>         98
 2     2 <kmeans>         62.4
 3     3 <kmeans>         36.6
 4     4 <kmeans>         24.9
 5     5 <kmeans>         19.6
 6     6 <kmeans>         16.4
 7     7 <kmeans>         13.7
 8     8 <kmeans>         11.0
 9     9 <kmeans>          9.85
10    10 <kmeans>          8.04
```


Total within-cluster sum of squares
100
75
50
25
Elbow
1
2
3
4
5
6
7
8
9
10
K


Cluster plot
California
Rhode Island
New Jersey
New York
Massachusetts
Illinois
Nevada
Utah
Hawaii
Arizona
Florida
Connecticut
1
Texas
Colorado
Michigan
Washington
Ohio
2
New Mexico
Delaware
Pennsylvania
Missouri
Oregon
Wisconsin
Oklahoma
Maryland
Louisiana
Minnesota
Kansas
Indiana
Tennessee
Virginia
Georgia
Nebraska
Alabama
Iowa
Wyoming
Kentucky
New Hampshire
Montana
3
Idaho
South Carolina
Maine
Arkansas
Alaska
North Carolina
4 South Dakota
Mississippi
North Dakota
West Virginia
Vermont
UrbanPop
Murder
cluster
1
2
3
4
1
0
-1
-2
-1
0
1
2

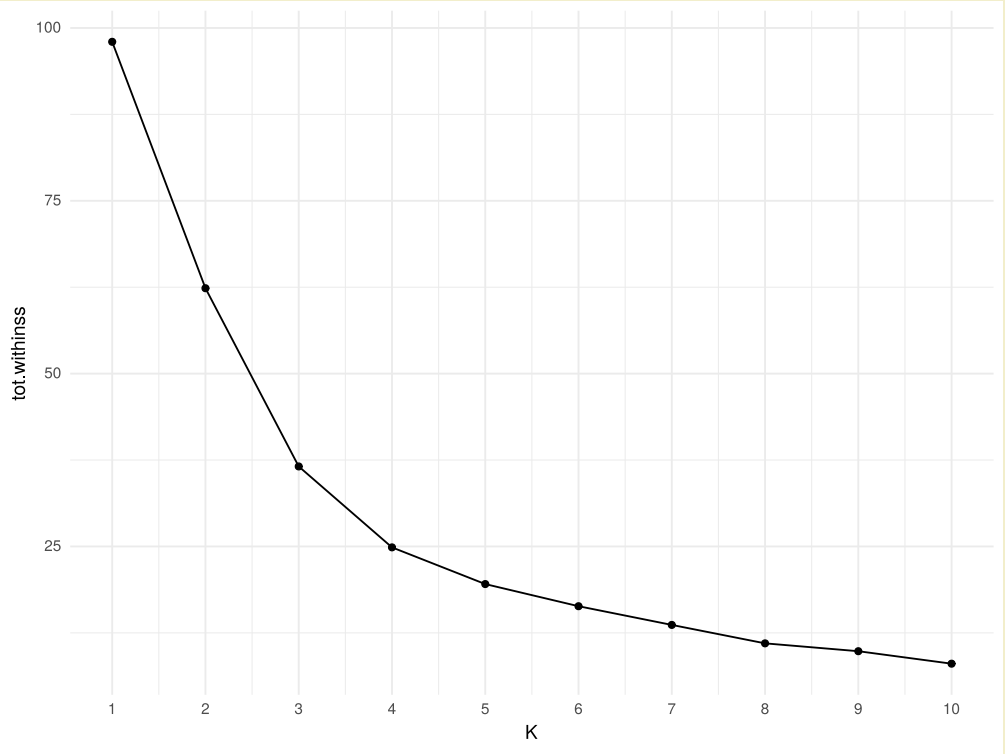
Extract

```
USAData %>%
  mutate(Cluster = kmeans.final$cluster) %>%
  group_by(Cluster) %>%
  summarise_all("mean")
# A tibble: 4 × 3
  Cluster Murder UrbanPop
    <int>  <dbl>    <dbl>
1       1  0.649    0.941
2       2 -0.606    0.412
3       3  1.22    -0.866
4       4 -1.09    -1.22
```

Your Turn 1

05:00

Please clone the repository on clustering to your local folder.



Complete the questionnaires to find the optimal number of clusters using the total within sum of squares criteria.

Hierarchical clustering

- A method used to group objects based on similarity
- Focus on agglomerative hierarchical clustering (bottom-up approach)
- To form an attractive tree-based representation of the observations called **dendogram**
- Flexible cut-off choice for the number of clusters

Algorithm

1. Starts by calculating the distance between every pair of observation points and store it in a distance matrix.
2. Puts every point in its own cluster.
3. Starts merging the closest pairs of points based on the distances from the distance matrix and as a result the amount of clusters goes down by 1.
4. Recomputes the distance between the new cluster and the old ones and stores them in a new distance matrix.
5. Repeats steps 2 and 3 until all the clusters are merged into one single cluster.

Dissimilarity values

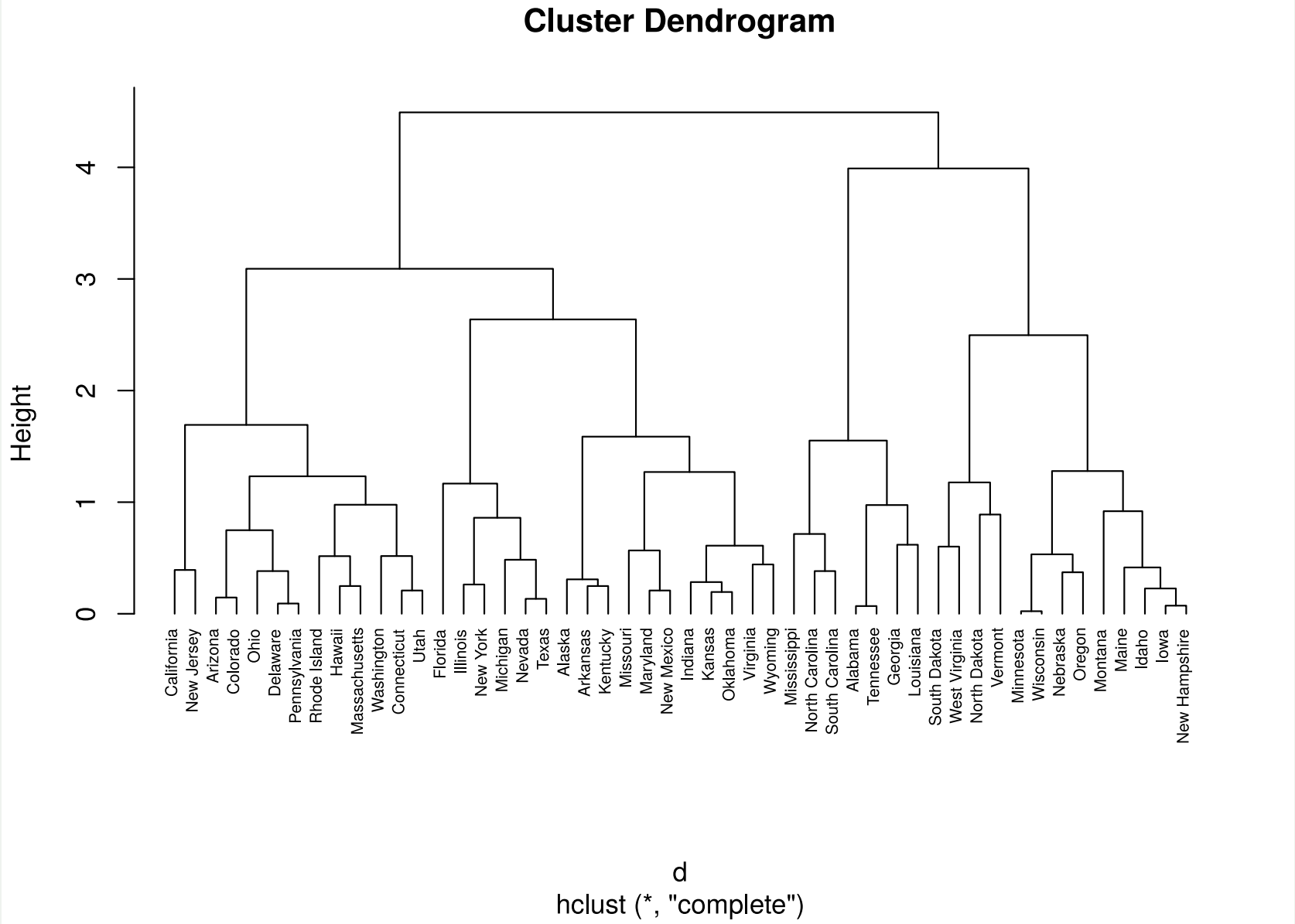
```
d <- dist(USAData, method = "euclidean")
```

Agglomeration method

```
# Hierarchical clustering
hc1 <- hclust(d)
```

Plot the dendogram

```
# Plot the obtained dendrogram
plot(hc1, cex = 0.6, hang = -1)
```

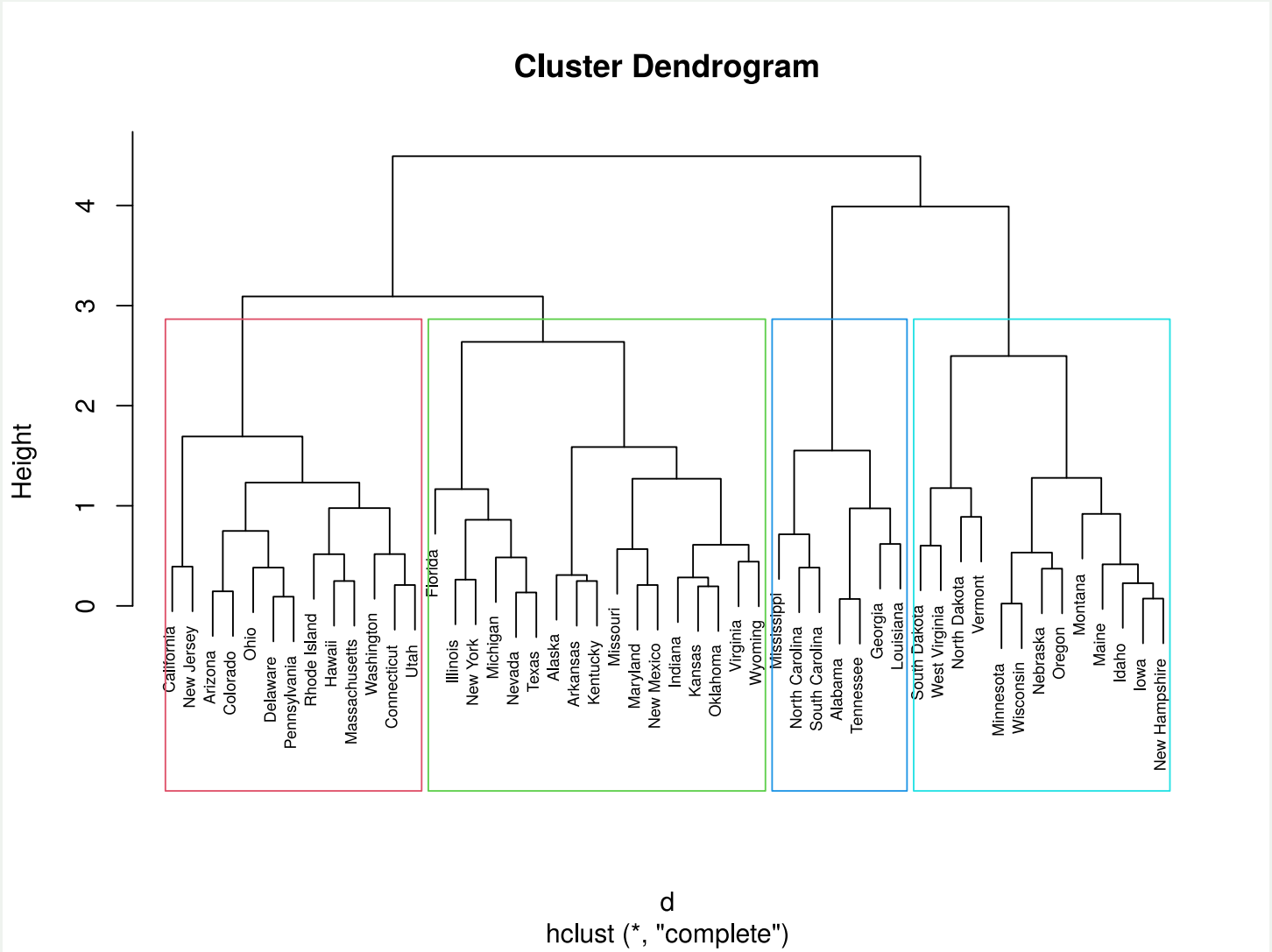

Cluster Dendrogram
Height
0
1
2
3
4
California
New Jersey
Arizona
Colorado
Ohio
Delaware
Pennsylvania
Rhode Island
Hawaii
Massachusetts
Washington
Connecticut
Utah
Florida
Illinois
New York
Michigan
Nevada
Texas
Alaska
Arkansas
Kentucky
Missouri
Maryland
New Mexico
Indiana
Kansas
Oklahoma
Virginia
Wyoming
Mississippi
North Carolina
South Carolina
Alabama
Tennessee
Georgia
Louisiana
South Dakota
West Virginia
North Dakota
Vermont
Minnesota
Wisconsin
Nebraska
Oregon
Montana
Maine
Idaho
Iowa
New Hampshire
d
hclust (*, "complete")

```r
# Cut tree into 4 groups
sub_grp <- cutree(hc1, k = 4)
```

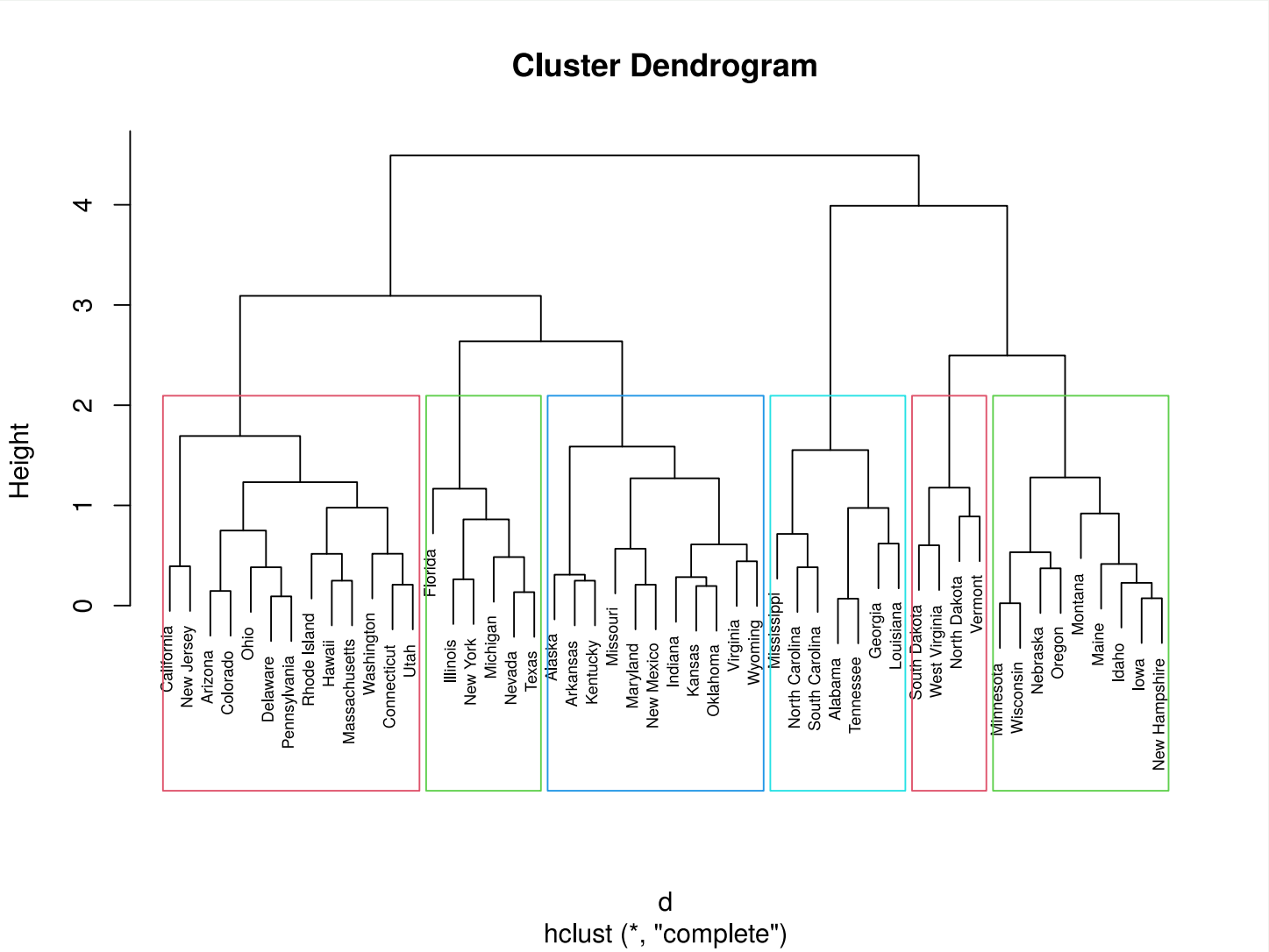
```r
# Number of members in each cluster
table(sub_grp)
sub_grp
 1  2  3  4
 7 17 13 13
```

```
USAData %>%
  mutate(cluster = sub_grp) %>%
  head()
               Murder   UrbanPop cluster
Alabama    1.24256408 -0.5209066       1
Alaska     0.50786248 -1.2117642       2
Arizona    0.07163341  0.9989801       3
Arkansas   0.23234938 -1.0735927       2
California 0.27826823  1.7589234       3
Colorado   0.02571456  0.8608085       3
```

```r
plot(hc1, cex = 0.6)
rect.hclust(hc1, k = 4, border = 2:5)    # Number of clusters
```

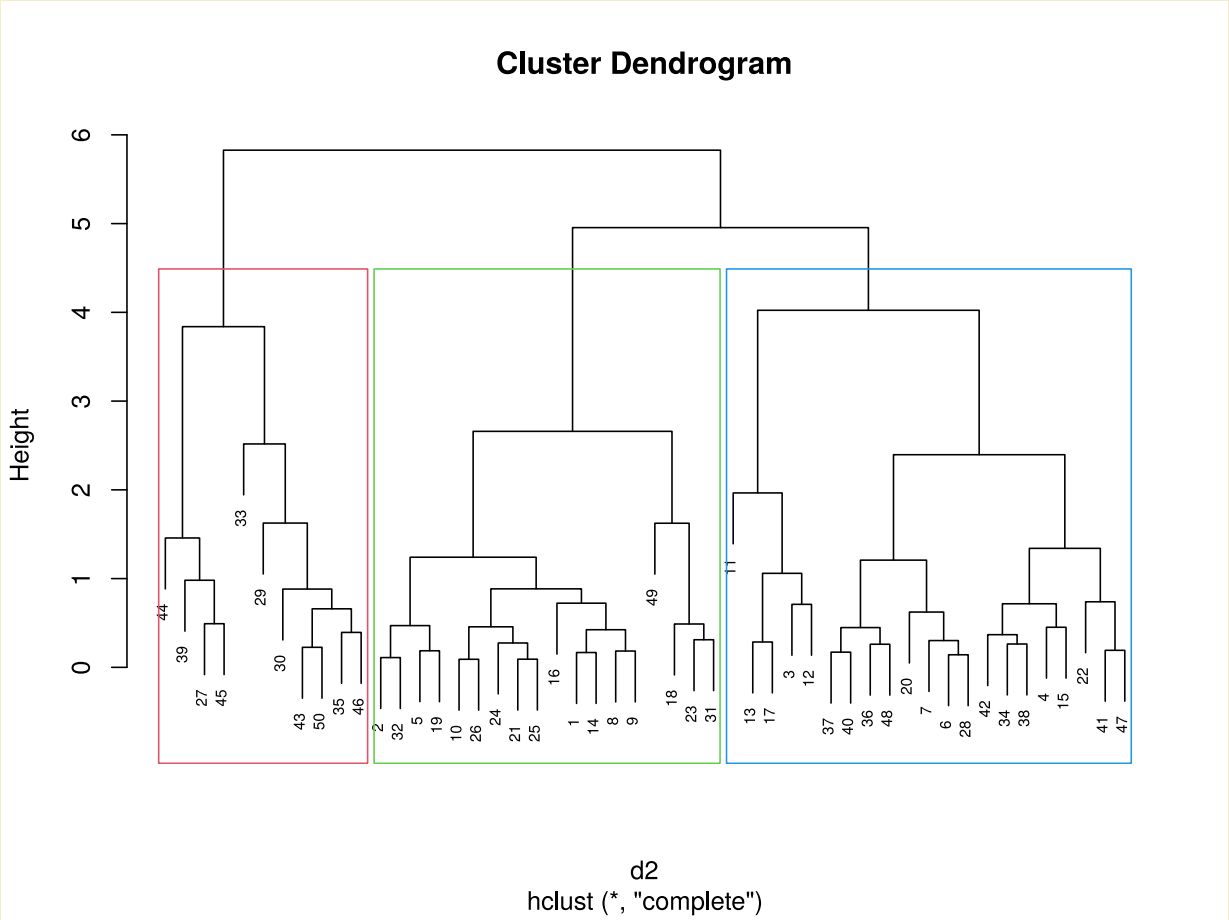


```r
plot(hc1, cex = 0.6)
rect.hclust(hc1, h = 2, border = 2:5)    # Height of branch
```



Your Turn 2

03:00



Explore further about hierarchical clustering using the instructions provided to produce the dendogram above.