Cluster Analysis: Discovery
Winter Institute in Data Science

Ryan T. Moore

2026-01-07

Matrices and Lists

Matrices and Lists

- `matrix`: rectangular array with a few more restrictions than `data.frame`

Matrices and Lists

- `matrix`: rectangular array with a few more restrictions than `data.frame`
- `list`: more general type of array that can be rectangular, but need not be. A `data.frame` is an example of a `list`

Matrices and Lists

- `matrix`: rectangular array with a few more restrictions than `data.frame`
- `list`: more general type of array that can be rectangular, but need not be. A `data.frame` is an example of a `list`

Matrices and Lists

- `matrix`: rectangular array with a few more restrictions than `data.frame`
- `list`: more general type of array that can be rectangular, but need not be. A `data.frame` is an example of a `list`

In base R, can extract variable `v1` from data frame `df` with

- `df$v1`

Matrices and Lists

- `matrix`: rectangular array with a few more restrictions than `data.frame`
- `list`: more general type of array that can be rectangular, but need not be. A `data.frame` is an example of a `list`

In base R, can extract variable `v1` from data frame `df` with

- `df$v1`
- `df[["v1"]]`

Matrices and Lists

- `matrix`: rectangular array with a few more restrictions than `data.frame`
- `list`: more general type of array that can be rectangular, but need not be. A `data.frame` is an example of a `list`

In base R, can extract variable `v1` from data frame `df` with

- `df$v1`
- `df[["v1"]]`
- `df[[6]]`

Matrices and Lists

- `matrix`: rectangular array with a few more restrictions than `data.frame`
- `list`: more general type of array that can be rectangular, but need not be. A `data.frame` is an example of a `list`

In base R, can extract variable `v1` from data frame `df` with

- `df$v1`
- `df[["v1"]]`
- `df[[6]]`

Matrices and Lists

- `matrix`: rectangular array with a few more restrictions than `data.frame`
- `list`: more general type of array that can be rectangular, but need not be. A `data.frame` is an example of a `list`

In base R, can extract variable `v1` from data frame `df` with

- `df$v1`
- `df[["v1"]]`
- `df[[6]]` (if `v1` is 6th variable)

Matrices and Lists

- `matrix`: rectangular array with a few more restrictions than `data.frame`
- `list`: more general type of array that can be rectangular, but need not be. A `data.frame` is an example of a `list`

In base R, can extract variable `v1` from data frame `df` with

- `df$v1`
- `df[["v1"]]`
- `df[[6]]` (if `v1` is 6th variable)
- `df[, "v1"]`

Matrices and Lists

- `matrix`: rectangular array with a few more restrictions than `data.frame`
- `list`: more general type of array that can be rectangular, but need not be. A `data.frame` is an example of a `list`

In base R, can extract variable `v1` from data frame `df` with

- `df$v1`
- `df[["v1"]]`
- `df[[6]]` (if `v1` is 6th variable)
- `df[, "v1"]`
- `df[, 6]`

Matrices and Lists

- `matrix`: rectangular array with a few more restrictions than `data.frame`
- `list`: more general type of array that can be rectangular, but need not be. A `data.frame` is an example of a `list`

In base R, can extract variable `v1` from data frame `df` with

- `df$v1`
- `df[["v1"]]`
- `df[[6]]` (if `v1` is 6th variable)
- `df[, "v1"]`
- `df[, 6]`
- `df["v1"]`

Matrices and Lists

- `matrix`: rectangular array with a few more restrictions than `data.frame`
- `list`: more general type of array that can be rectangular, but need not be. A `data.frame` is an example of a `list`

In base R, can extract variable `v1` from data frame `df` with

- `df$v1`
- `df[["v1"]]`
- `df[[6]]` (if `v1` is 6th variable)
- `df[, "v1"]`
- `df[, 6]`
- `df["v1"]`

Matrices and Lists

- `matrix`: rectangular array with a few more restrictions than `data.frame`
- `list`: more general type of array that can be rectangular, but need not be. A `data.frame` is an example of a `list`

In base R, can extract variable `v1` from data frame `df` with

- `df$v1`
- `df[["v1"]]`
- `df[[6]]` (if `v1` is 6th variable)
- `df[, "v1"]`
- `df[, 6]`
- `df["v1"]` (keeps col name)

1. Describe list `ll` created below:

```
v1 <- 1:8
v2 <- letters[1:5]
m <- matrix(1:9, 3, 3)
ll <- list(x = v1, y = v2, z = m)
```

2. What is `ll$y`?
3. What is `ll[[1]]`?
4. What is `ll[[3]][2, 2]`?

Distance Matrix for Clustering

Each cell is a *distance* between unit `row` and unit `column`:

	a	b	c	d
a	0	1	2	3
b	1	0	6	7
c	2	6	0	8
d	3	7	8	0

Distance Matrix for Clustering

Each cell is a *distance* between unit `row` and unit `column`:

	a	b	c	d
a	0	1	2	3
b	1	0	6	7
c	2	6	0	8
d	3	7	8	0

So, $||(c, d)|| > ||(a, b)||$, e.g., $(8 > 1)$

Supervised and Unsupervised Learning

Supervised: Modeling with **known** outcomes

Supervised and Unsupervised Learning

Supervised: Modeling with **known** outcomes

Unsupervised: Discovery w/ **unknown** outcomes

Supervised Learning

- Linear regression (LS)

Supervised Learning

- Linear regression (LS)
- Generalized linear regression
  (logistic, probit, Poisson, beta)

Supervised Learning

- Linear regression (LS)
- Generalized linear regression
  (logistic, probit, Poisson, beta)
- Bayesian modeling

Supervised Learning

- Linear regression (LS)
- Generalized linear regression (logistic, probit, Poisson, beta)
- Bayesian modeling
- LASSO

Supervised Learning

- Linear regression (LS)
- Generalized linear regression (logistic, probit, Poisson, beta)
- Bayesian modeling
- LASSO
- Neural networks (CNNs)

Supervised Learning

- Linear regression (LS)
- Generalized linear regression
  (logistic, probit, Poisson, beta)
- Bayesian modeling
- LASSO
- Neural networks (CNNs)
- NLP (predict "bot?" 0/1 from matx of word indicators)

Supervised Learning

- Linear regression (LS)
- Generalized linear regression
  (logistic, probit, Poisson, beta)
- Bayesian modeling
- LASSO
- Neural networks (CNNs)
- NLP (predict "bot?" 0/1 from matx of word indicators)
- Decision trees, forests, etc.

Supervised Learning

- Linear regression (LS)
- Generalized linear regression
  (logistic, probit, Poisson, beta)
- Bayesian modeling
- LASSO
- Neural networks (CNNs)
- NLP (predict "bot?" 0/1 from matx of word indicators)
- Decision trees, forests, etc.

Supervised Learning

- Linear regression (LS)
- Generalized linear regression
  (logistic, probit, Poisson, beta)
- Bayesian modeling
- LASSO
- Neural networks (CNNs)
- NLP (predict "bot?" 0/1 from matx of word indicators)
- Decision trees, forests, etc.

If you have $y = f(X)$, it's "supervised".

Unsupervised Learning

- Network analysis (community detection)

Unsupervised Learning

- Network analysis (community detection)
- Principal components analysis (PCA)

Unsupervised Learning

- Network analysis (community detection)
- Principal components analysis (PCA)
- (Some neural networks)

Unsupervised Learning

- Network analysis (community detection)
- Principal components analysis (PCA)
- (Some neural networks)
- NLP (derive topics from documents)

Unsupervised Learning

- Network analysis (community detection)
- Principal components analysis (PCA)
- (Some neural networks)
- NLP (derive topics from documents)
- Clustering algorithms (most)

Clustering is a method of "unsupervised learning" – of trying to derive structure from data when no outcome labels are given.

Clustering is a method of "unsupervised learning" – of trying to derive structure from data when no outcome labels are given.

- Networks: "community detection"

Clustering is a method of "unsupervised learning" – of trying to derive structure from data when no outcome labels are given.

- Networks: "community detection"
  - no predefined community labels

Clustering is a method of "unsupervised learning" – of trying to derive structure from data when no outcome labels are given.

- Networks: "community detection"
  - no predefined community labels
- Roll call voting: party/faction detection

Clustering is a method of "unsupervised learning" – of trying to derive structure from data when no outcome labels are given.

- Networks: "community detection"
  - no predefined community labels
- Roll call voting: party/faction detection
  - no predefined party/faction labels

Clustering is a method of "unsupervised learning" – of trying to derive structure from data when no outcome labels are given.

- Networks: "community detection"
  - no predefined community labels
- Roll call voting: party/faction detection
  - no predefined party/faction labels
  - (e.g., NE's *nonpartisan* "Unicameral")

Clustering is a method of "unsupervised learning" – of trying to derive structure from data when no outcome labels are given.

- Networks: "community detection"
  - no predefined community labels
- Roll call voting: party/faction detection
  - no predefined party/faction labels
  - (e.g., NE's *nonpartisan* "Unicameral")
- Geographic clustering: daily activities

Clustering is a method of "unsupervised learning" – of trying to derive structure from data when no outcome labels are given.

- Networks: "community detection"
  - no predefined community labels
- Roll call voting: party/faction detection
  - no predefined party/faction labels
  - (e.g., NE's *nonpartisan* "Unicameral")
- Geographic clustering: daily activities
  - no "home"/"work"/"leisure" labels

Clustering

- Hierarchical clustering

Clustering

- Hierarchical clustering
  - Find best groups at many levels

Clustering

- Hierarchical clustering
  - Find best groups at many levels
  - Units in one cluster at each level of hierarchy

Clustering

- Hierarchical clustering
  - Find best groups at many levels
  - Units in one cluster at each level of hierarchy
- Partitional clustering

Clustering

- Hierarchical clustering
  - Find best groups at many levels
  - Units in one cluster at each level of hierarchy
- Partitional clustering
  - Find splits in full set

Clustering

- Hierarchical clustering
  - Find best groups at many levels
  - Units in one cluster at each level of hierarchy
- Partitional clustering
  - Find splits in full set
  - Units in only one cluster

Hierarchical Clustering

Hierarchical clustering

Consider a set of units open to cluster discovery, $\{a, b, c, d, e\}$

Hierarchical clustering

Consider a set of units open to cluster discovery, $\{a, b, c, d, e\}$

Divisive clustering

- Start at $\{a, b, c, d, e\}$

Hierarchical clustering

Consider a set of units open to cluster discovery, $\{a, b, c, d, e\}$

Divisive clustering

- Start at $\{a, b, c, d, e\}$
- Find $\{a, b, d\}$ and $\{c, e\}$

Hierarchical clustering

Consider a set of units open to cluster discovery, $\{a, b, c, d, e\}$

Divisive clustering

- Start at $\{a, b, c, d, e\}$
- Find $\{a, b, d\}$ and $\{c, e\}$
- Find $\{a, b\}$, $\{d\}$, and $\{c, e\}$

Hierarchical clustering

Consider a set of units open to cluster discovery, $\{a, b, c, d, e\}$

Divisive clustering

- Start at $\{a, b, c, d, e\}$
- Find $\{a, b, d\}$ and $\{c, e\}$
- Find $\{a, b\}$, $\{d\}$, and $\{c, e\}$
- ...

Hierarchical clustering

Consider a set of units open to cluster discovery, $\{a, b, c, d, e\}$

Divisive clustering

- Start at $\{a, b, c, d, e\}$
- Find $\{a, b, d\}$ and $\{c, e\}$
- Find $\{a, b\}$, $\{d\}$, and $\{c, e\}$
- ...
- End at $\{a\}$, $\{b\}$, $\{c\}$, $\{d\}$, $\{e\}$

Hierarchical clustering

Consider a set of units open to cluster discovery, $\{a, b, c, d, e\}$

Divisive clustering

- Start at $\{a, b, c, d, e\}$
- Find $\{a, b, d\}$ and $\{c, e\}$
- Find $\{a, b\}$, $\{d\}$, and $\{c, e\}$
- ...
- End at $\{a\}$, $\{b\}$, $\{c\}$, $\{d\}$, $\{e\}$

Hierarchical clustering

Consider a set of units open to cluster discovery, $\{a, b, c, d, e\}$

Divisive clustering

- Start at $\{a, b, c, d, e\}$
- Find $\{a, b, d\}$ and $\{c, e\}$
- Find $\{a, b\}$, $\{d\}$, and $\{c, e\}$
- ...
- End at $\{a\}$, $\{b\}$, $\{c\}$, $\{d\}$, $\{e\}$

Divisive clustering is "top-down"

Hierarchical clustering

Consider a set of units open to cluster discovery, $\{a, b, c, d, e\}$

Hierarchical clustering

Consider a set of units open to cluster discovery, $\{a, b, c, d, e\}$

Agglomerative clustering

- Start at $\{a\}$, $\{b\}$, $\{c\}$, $\{d\}$, $\{e\}$

Hierarchical clustering

Consider a set of units open to cluster discovery, $\{a, b, c, d, e\}$

Agglomerative clustering

- Start at $\{a\}$, $\{b\}$, $\{c\}$, $\{d\}$, $\{e\}$
- Find $\{a, b\}$ and leave $\{c\}$, $\{d\}$, $\{e\}$

Hierarchical clustering

Consider a set of units open to cluster discovery, $\{a, b, c, d, e\}$

Agglomerative clustering

- Start at $\{a\}$, $\{b\}$, $\{c\}$, $\{d\}$, $\{e\}$
- Find $\{a, b\}$ and leave $\{c\}$, $\{d\}$, $\{e\}$
- Find $\{a, b\}$, $\{c, e\}$, and leave $\{d\}$

Hierarchical clustering

Consider a set of units open to cluster discovery, $\{a, b, c, d, e\}$

Agglomerative clustering

- Start at $\{a\}$, $\{b\}$, $\{c\}$, $\{d\}$, $\{e\}$
- Find $\{a, b\}$ and leave $\{c\}$, $\{d\}$, $\{e\}$
- Find $\{a, b\}$, $\{c, e\}$, and leave $\{d\}$
- ...

Hierarchical clustering

Consider a set of units open to cluster discovery, $\{a, b, c, d, e\}$

Agglomerative clustering

- Start at $\{a\}$, $\{b\}$, $\{c\}$, $\{d\}$, $\{e\}$
- Find $\{a, b\}$ and leave $\{c\}$, $\{d\}$, $\{e\}$
- Find $\{a, b\}$, $\{c, e\}$, and leave $\{d\}$
- ...
- End at $\{a, b, c, d, e\}$

Hierarchical clustering

Consider a set of units open to cluster discovery, $\{a, b, c, d, e\}$

Agglomerative clustering

- Start at $\{a\}$, $\{b\}$, $\{c\}$, $\{d\}$, $\{e\}$
- Find $\{a, b\}$ and leave $\{c\}$, $\{d\}$, $\{e\}$
- Find $\{a, b\}$, $\{c, e\}$, and leave $\{d\}$
- ...
- End at $\{a, b, c, d, e\}$

Hierarchical clustering

Consider a set of units open to cluster discovery, $\{a, b, c, d, e\}$

Agglomerative clustering

- Start at $\{a\}$, $\{b\}$, $\{c\}$, $\{d\}$, $\{e\}$
- Find $\{a, b\}$ and leave $\{c\}$, $\{d\}$, $\{e\}$
- Find $\{a, b\}$, $\{c, e\}$, and leave $\{d\}$
- ...
- End at $\{a, b, c, d, e\}$

Agglomerative clustering is "bottom-up"

Each unit is in a clustering at every level.

Agglomerative Hierarchical Clustering

Complete linkage clustering: greedily create clusters

Agglomerative Hierarchical Clustering

Complete linkage clustering: greedily create clusters

1. Merge closest pair (into, say, $\{a, b\}$)

Agglomerative Hierarchical Clustering

Complete linkage clustering: greedily create clusters

1. Merge closest pair (into, say, $\{a, b\}$)
2. Update `dist`: delete rows/cols for a, b; add row for $\{a, b\}$ (with max)
3. Merge closest "pair"
4. Update `dist`
5. ...

Agglomerative Hierarchical Complete Linkage Clusters

	a	b	c	d
a	0	1	2	3
b	1	0	6	7
c	2	6	0	8
d	3	7	8	0

Agglomerative Hierarchical Complete Linkage Clusters

	a	b	c	d
a	0	1	2	3
b	1	0	6	7
c	2	6	0	8
d	3	7	8	0

	(a,b)	c	d
(a,b)	0	6	7
c	6	0	8
d	7	8	0

Agglomerative Hierarchical Complete Linkage Clusters

	a	b	c	d
a	0	1	2	3
b	1	0	6	7
c	2	6	0	8
d	3	7	8	0

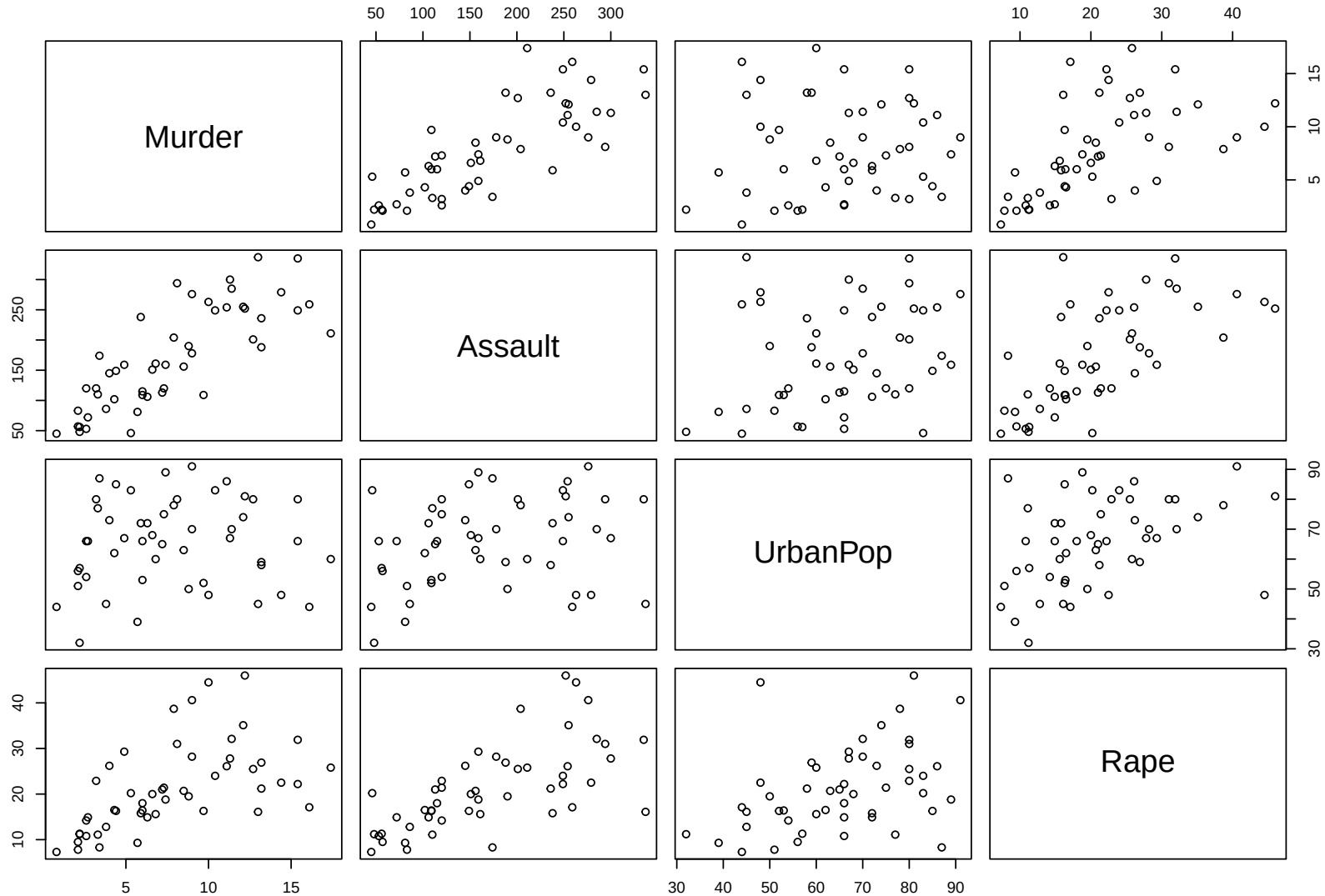
	(a,b)	c	d
(a,b)	0	6	7
c	6	0	8
d	7	8	0

(“friends of friends”)

Agglomerative Hierarchical Clustering

Agglomerative Hierarchical Clustering

```
pairs(USArrests)
```



Agglomerative Hierarchical Clustering

```
d <- dist(USArrests, method = "euclidean")
d |> round(0)
```

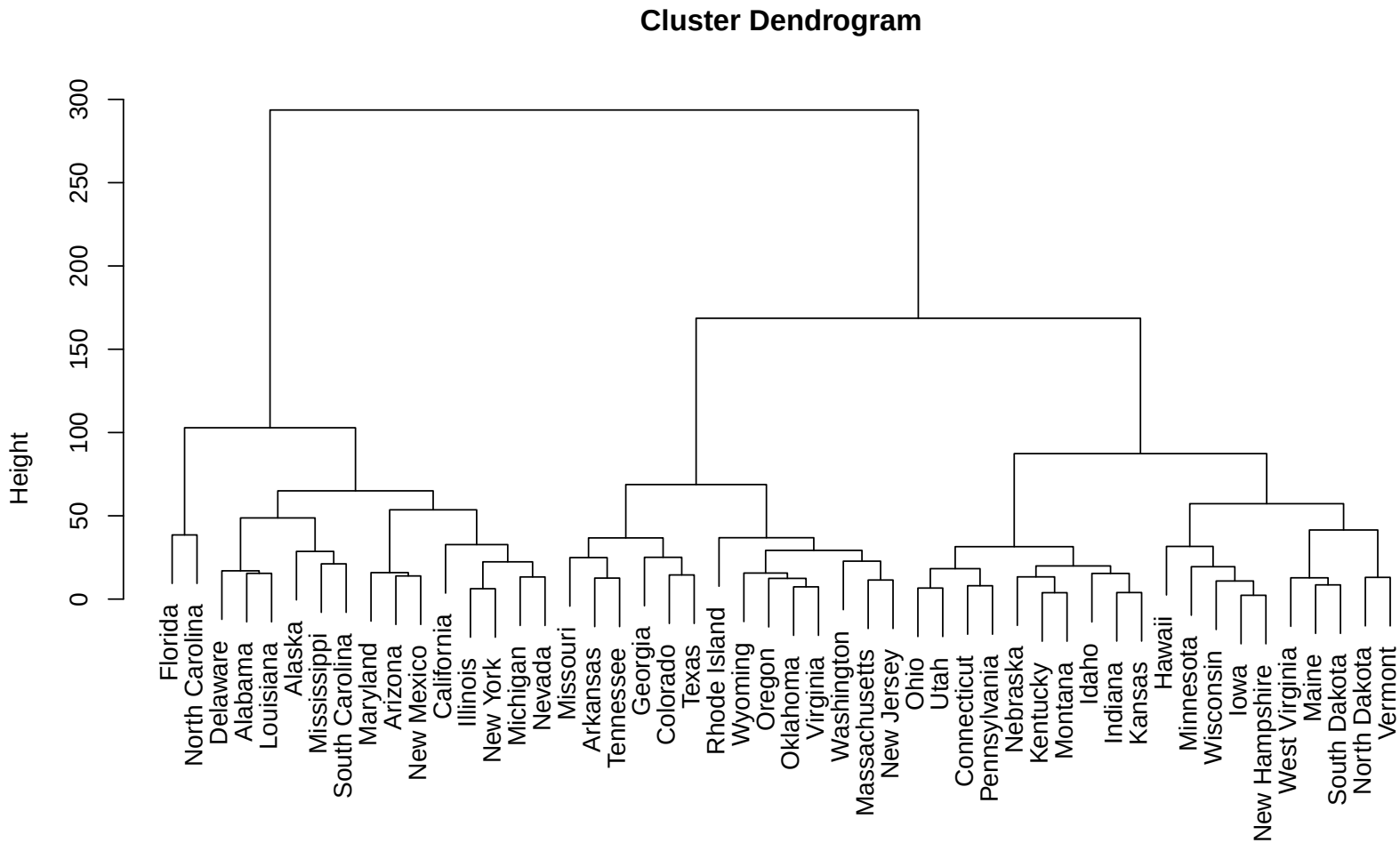
```
##                Alabama Alaska Arizona Arkansas Californi
## Alaska              37
## Arizona             63     47
## Arkansas            47     77     109
## California          56     45      23       98
## Colorado            42     66      90       37         7
## Connecticut        128    159     185       85        16
## Delaware            17     45      59       53         4
## Florida            102     80      42      149         6
## Georgia             26     57      86       26         7
## Hawaii             192    221     248      148        23
## Idaho              117    146     177       71        16
## Illinois            28     43      46       68         3
## Indiana            123    153     182       78        16
## Iowa               181    210     240      135        22
```

Agglomerative Hierarchical Clustering

```
hag <- hclust(d, method = "complete")
plot(hag)
```

Agglomerative Hierarchical Clustering

```
hag <- hclust(d, method = "complete")
plot(hag)
```



Recluster w/ Alternative Distance Metric: Mahalanobis

Recluster w/ Alternative Distance Metric: Mahalanobis

First, define function to make distance matrix:

```
mahal <- function(q, vc){
  storage <- matrix(NA, nrow(q), nrow(q))
  q <- as.matrix(q)
  for(i in 1:nrow(q)){
    storage[row(q), i] <- mahalanobis(x = q,
                                      center=q[i, ],
                                      cov = vc)
  }
  return(sqrt(storage))
}
```

Recluster w/ Alternative Distance Metric: Mahalanobis

```
d_mah <- mahal(USArrests, cov(USArrests))
row.names(d_mah) <- colnames(d_mah) <- row.names(USArrests)
d_mah <- as.dist(d_mah)
d_mah |> round(1)
```

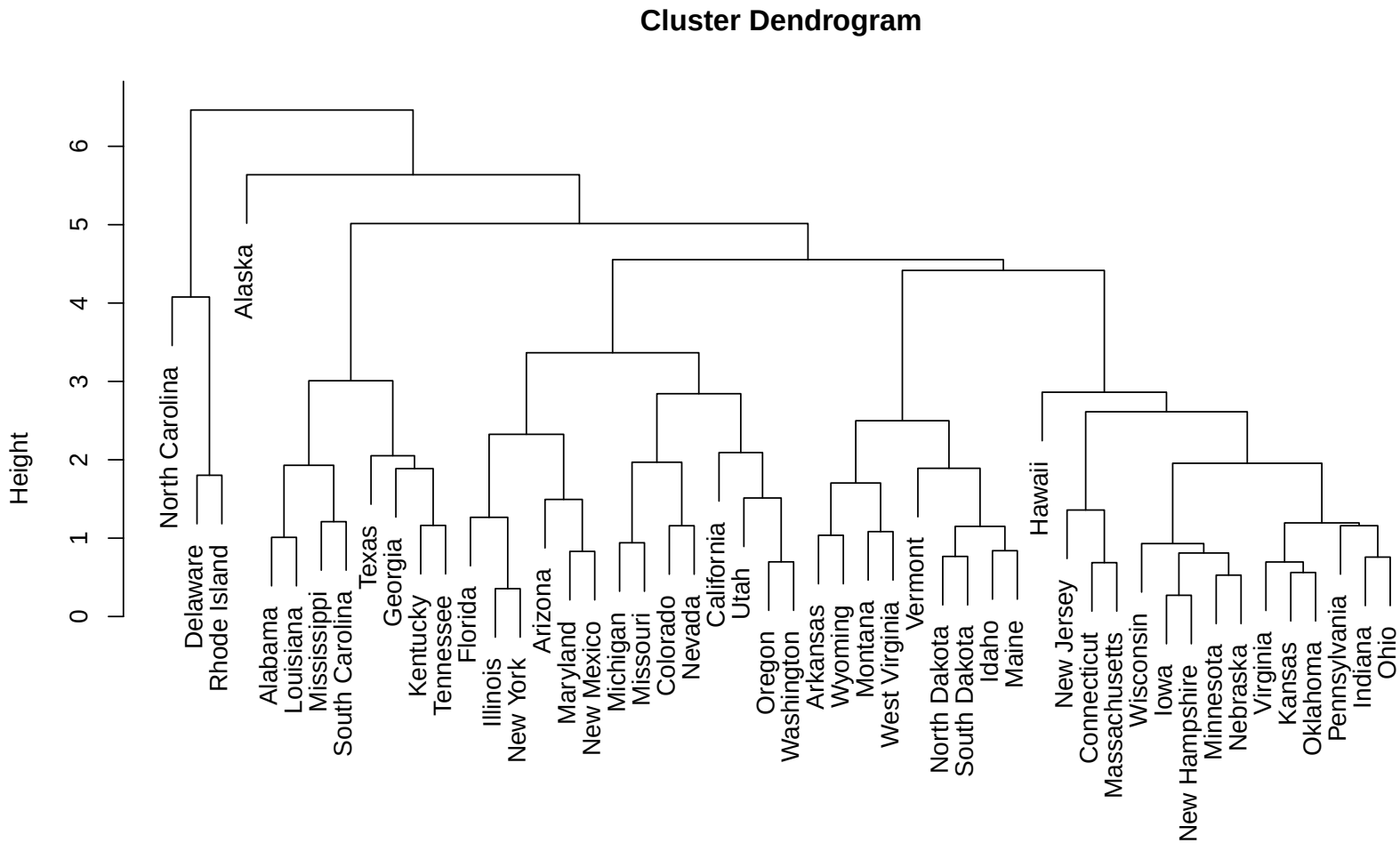
```
##                Alabama Alaska Arizona Arkansas Californi
## Alaska             4.4
## Arizona            3.2    3.9
## Arkansas           1.4    3.5     2.7
## California         3.5    3.6     1.7      3.3
## Colorado           3.3    2.8     2.6      2.9       1.
## Connecticut        2.8    5.4     2.9      2.6       3.
## Delaware           3.0    5.0     1.7      2.6       3.
## Florida            1.8    4.6     2.3      2.5       2.
## Georgia            2.2    5.4     5.0      3.3       4.
## Hawaii             3.5    5.6     4.5      3.8       3.
## Idaho              2.7    3.8     2.5      1.5       3.
## Illinois           2.0    4.9     2.1      2.4       2.
```

Recluster w/ Alternative Distance Metric: Mahalanobis

```
hag <- hclust(d_mah, method = "complete")
plot(hag)
```

Recluster w/ Alternative Distance Metric: Mahalanobis

```
hag <- hclust(d_mah, method = "complete")
plot(hag)
```



Agglomerative Hierarchical Clustering

- Small differences in `Assault` get over-weighted if `euclidean` dist

Agglomerative Hierarchical Clustering

- Small differences in `Assault` get over-weighted if `euclidean` dist
- (`Assault` has large variance, and cor w/ `UrbanPop`)

Agglomerative Hierarchical Clustering

- Small differences in `Assault` get over-weighted if `euclidean` dist
- (`Assault` has large variance, and cor w/ `UrbanPop`)
- Mahalanobis dist allows vars on different scales to all contribute.

Agglomerative Hierarchical Clustering

- Small differences in `Assault` get over-weighted if `euclidean` dist
- (`Assault` has large variance, and cor w/ `UrbanPop`)
- Mahalanobis dist allows vars on different scales to all contribute.
- E.g., `euclidean` clusters

Agglomerative Hierarchical Clustering

- Small differences in `Assault` get over-weighted if `euclidean` dist
- (`Assault` has large variance, and cor w/ `UrbanPop`)
- Mahalanobis dist allows vars on different scales to all contribute.
- E.g., `euclidean` clusters

Agglomerative Hierarchical Clustering

- Small differences in `Assault` get over-weighted if `euclidean` dist
- (`Assault` has large variance, and cor w/ `UrbanPop`)
- Mahalanobis dist allows vars on different scales to all contribute.
- E.g., `euclidean` clusters

```
##      Murder Assault UrbanPop Rape
## Ohio    7.3     120       75 21.4
## Utah    3.2     120       80 22.9
```

Agglomerative Hierarchical Clustering

- Small differences in `Assault` get over-weighted if `euclidean` dist
- (`Assault` has large variance, and cor w/ `UrbanPop`)
- Mahalanobis dist allows vars on different scales to all contribute.
- E.g., `euclidean` clusters

```
##      Murder Assault UrbanPop Rape
## Ohio    7.3     120       75 21.4
## Utah    3.2     120       80 22.9
```

versus

```
##         Murder Assault UrbanPop Rape
## Indiana    7.2     113       65 21.0
## Ohio       7.3     120       75 21.4
```

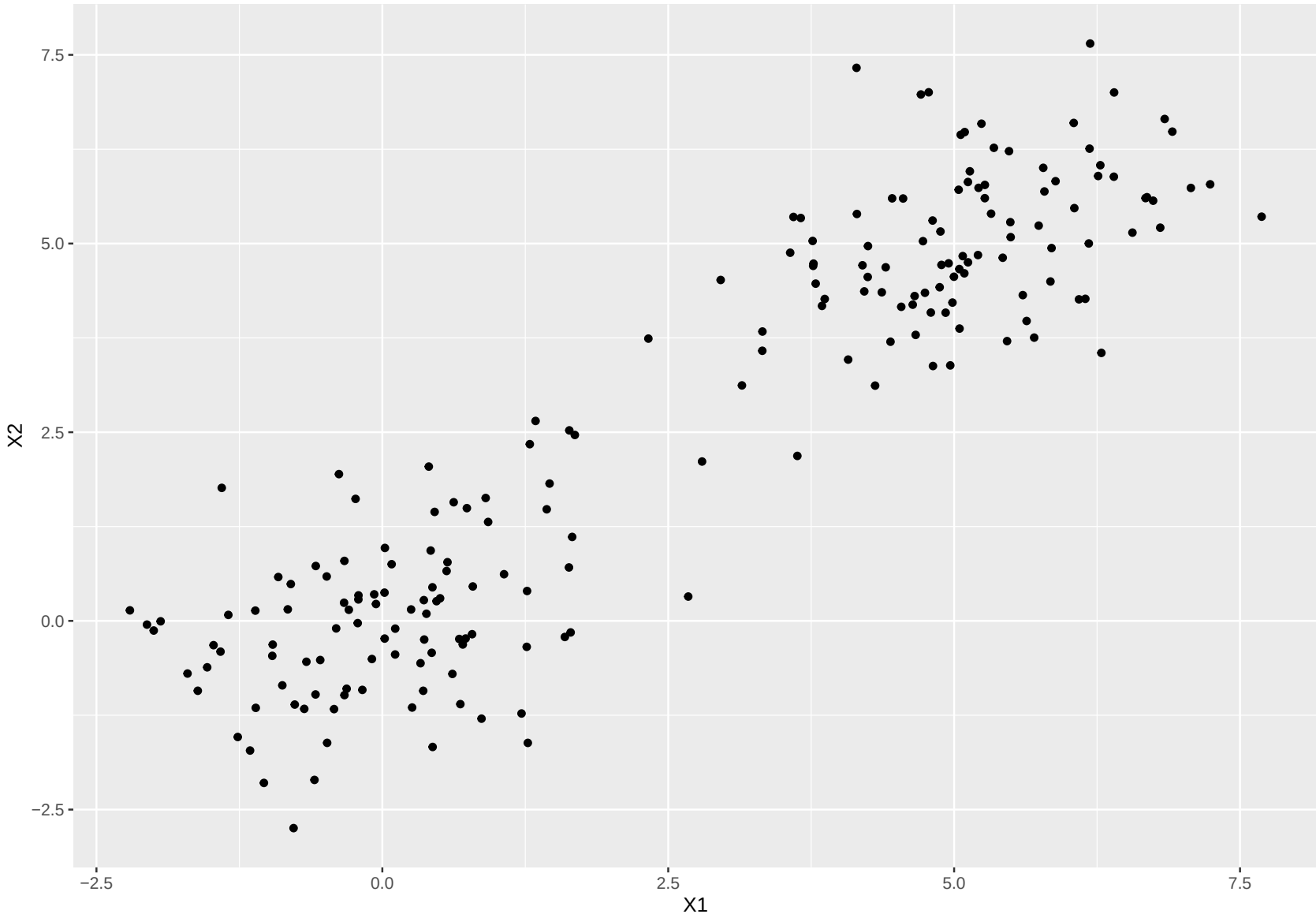
Partitional Clustering

k-means

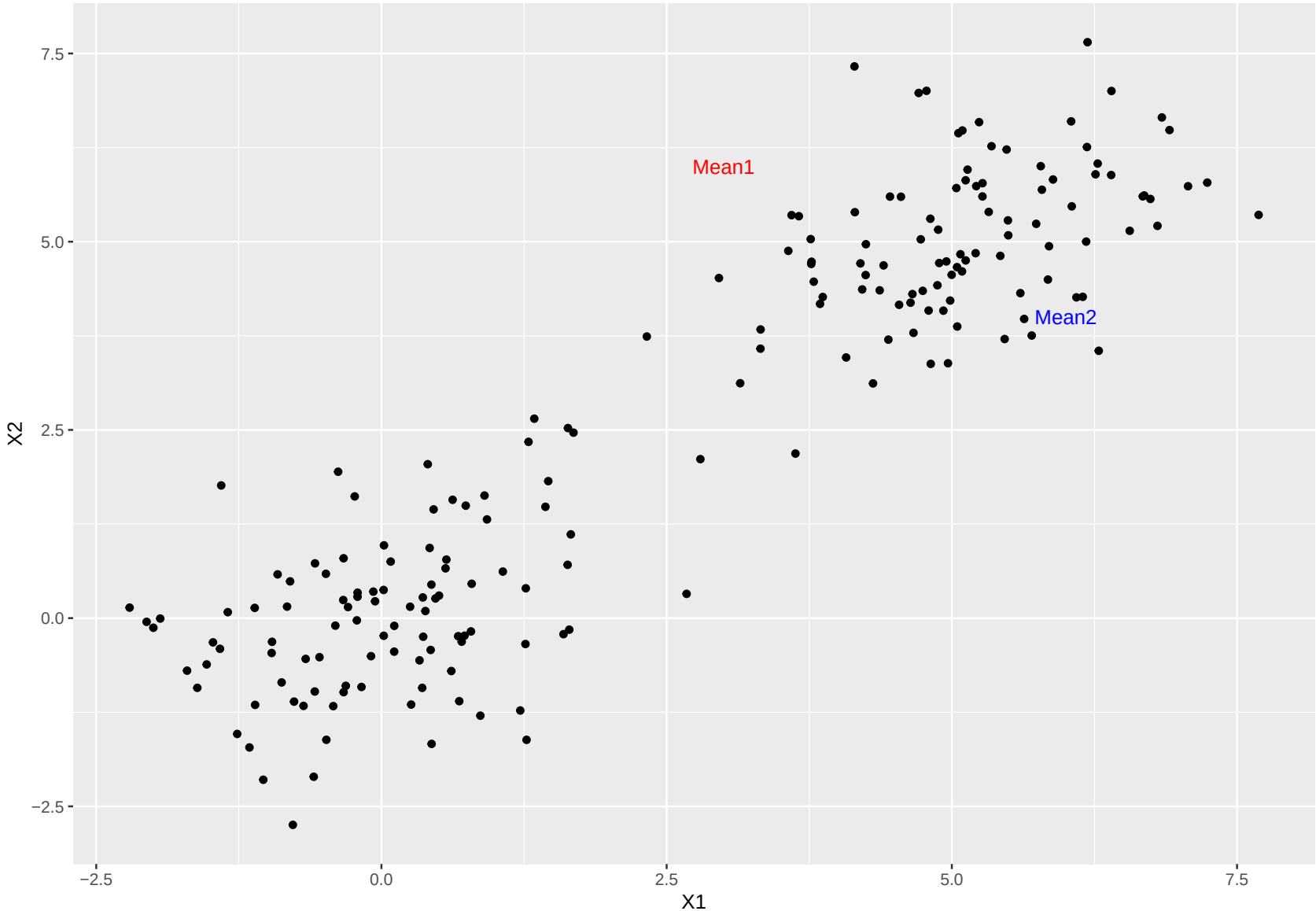
0. Standardize variables, or ensure comparable scales.
   (A diff of 1 unit on $x1$ should $\approx$ a diff of 1 unit on $x2$.)
1. Choose k, the number of clusters to identify.
2. Select the location of a center for each cluster.
3. Assign each observation to the cluster defined by the center closest to it.
4. Relocate each cluster's center to the mean of the observations currently in that cluster.
5. Repeat 3. and 4. until no observations gets assigned to a new cluster.

Suppose we have a set of points measured in a two-dimensional space, with `X1` and `X2` on comparable scales.

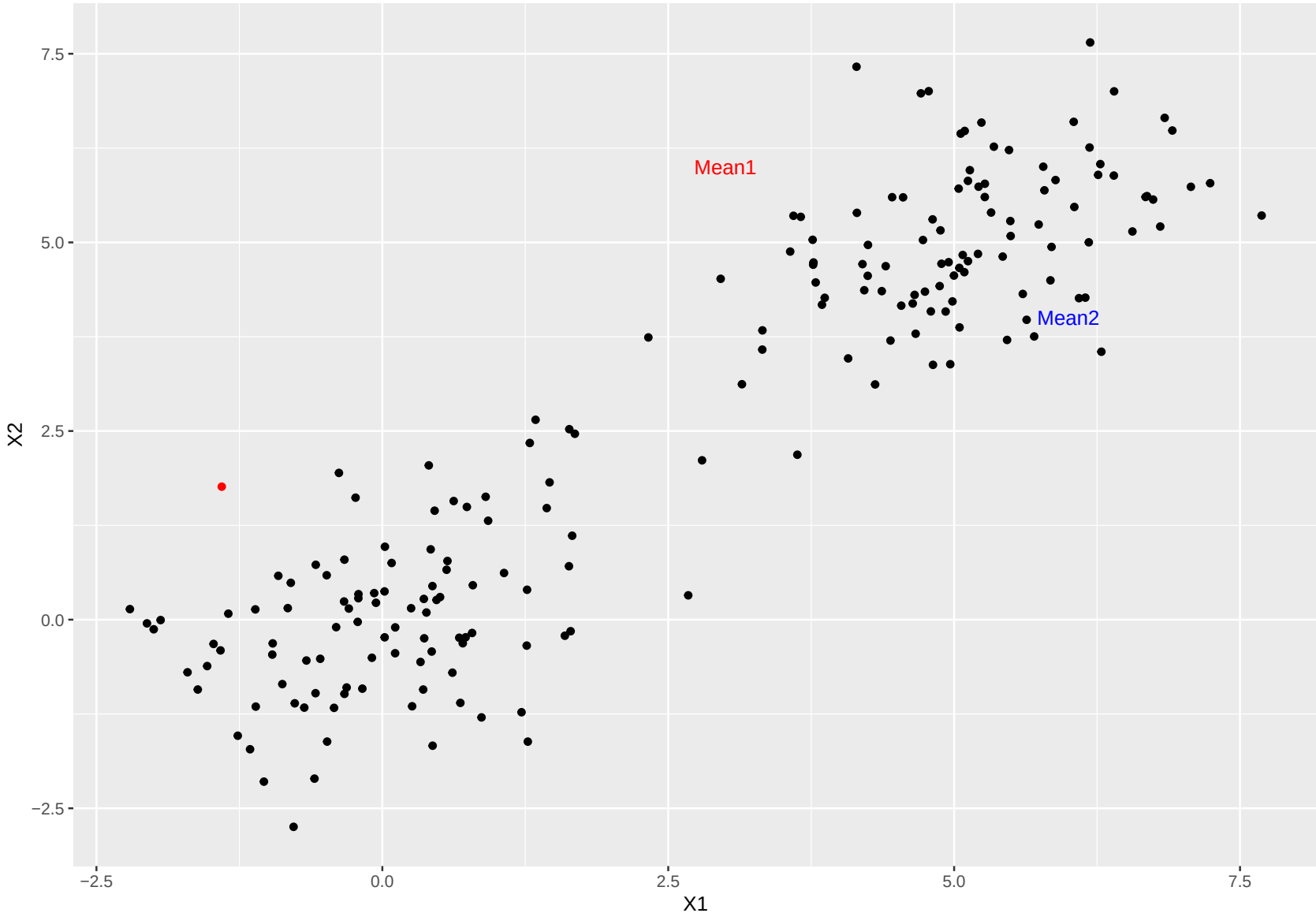
We will find $k = 2$ clusters. Ideas?



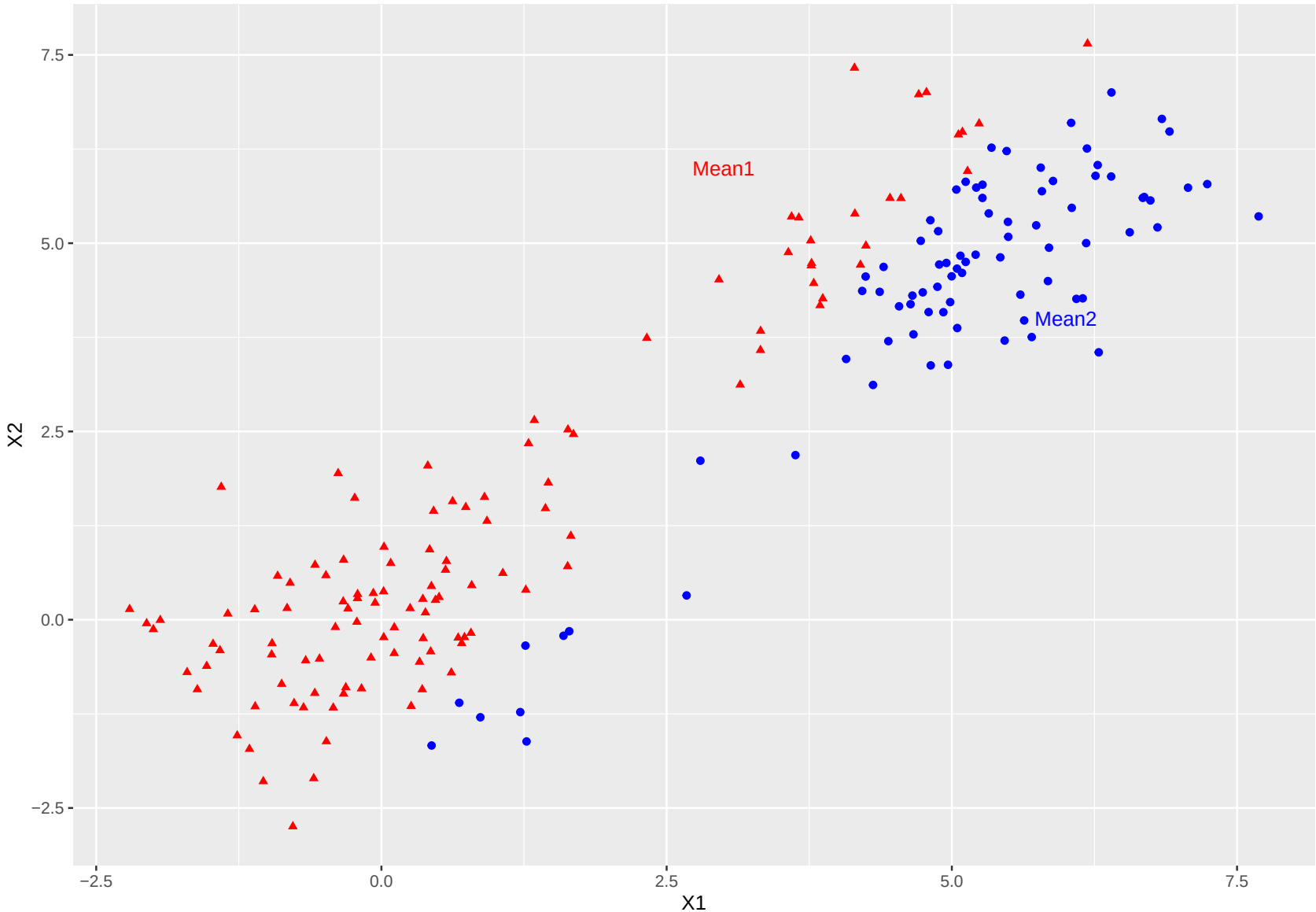
2. Let's randomly select two centroids:



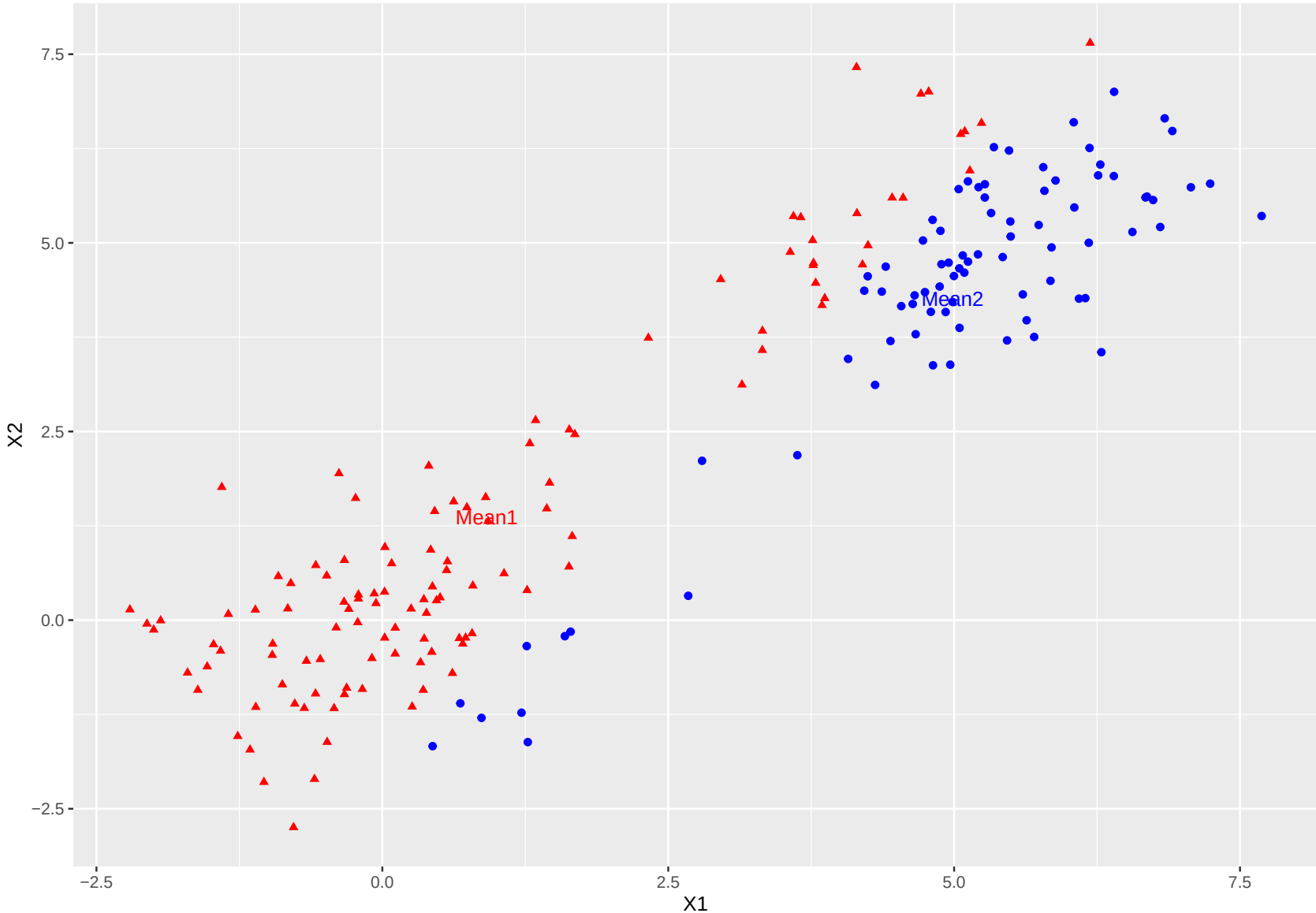
Where “should” this point go?



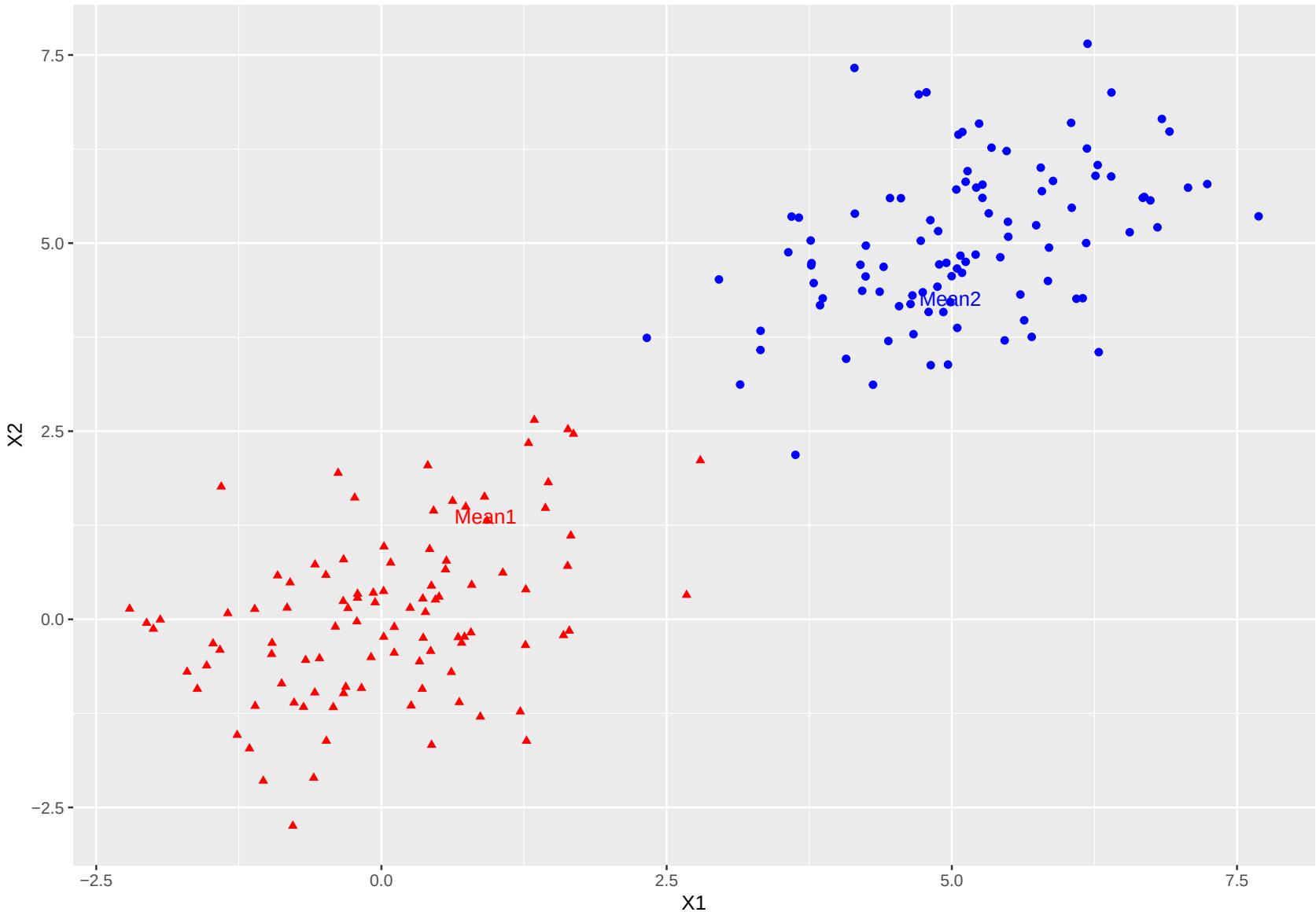
3. Find (Euclidean) distance between each point and the centroids; assign each point to closer centroid:



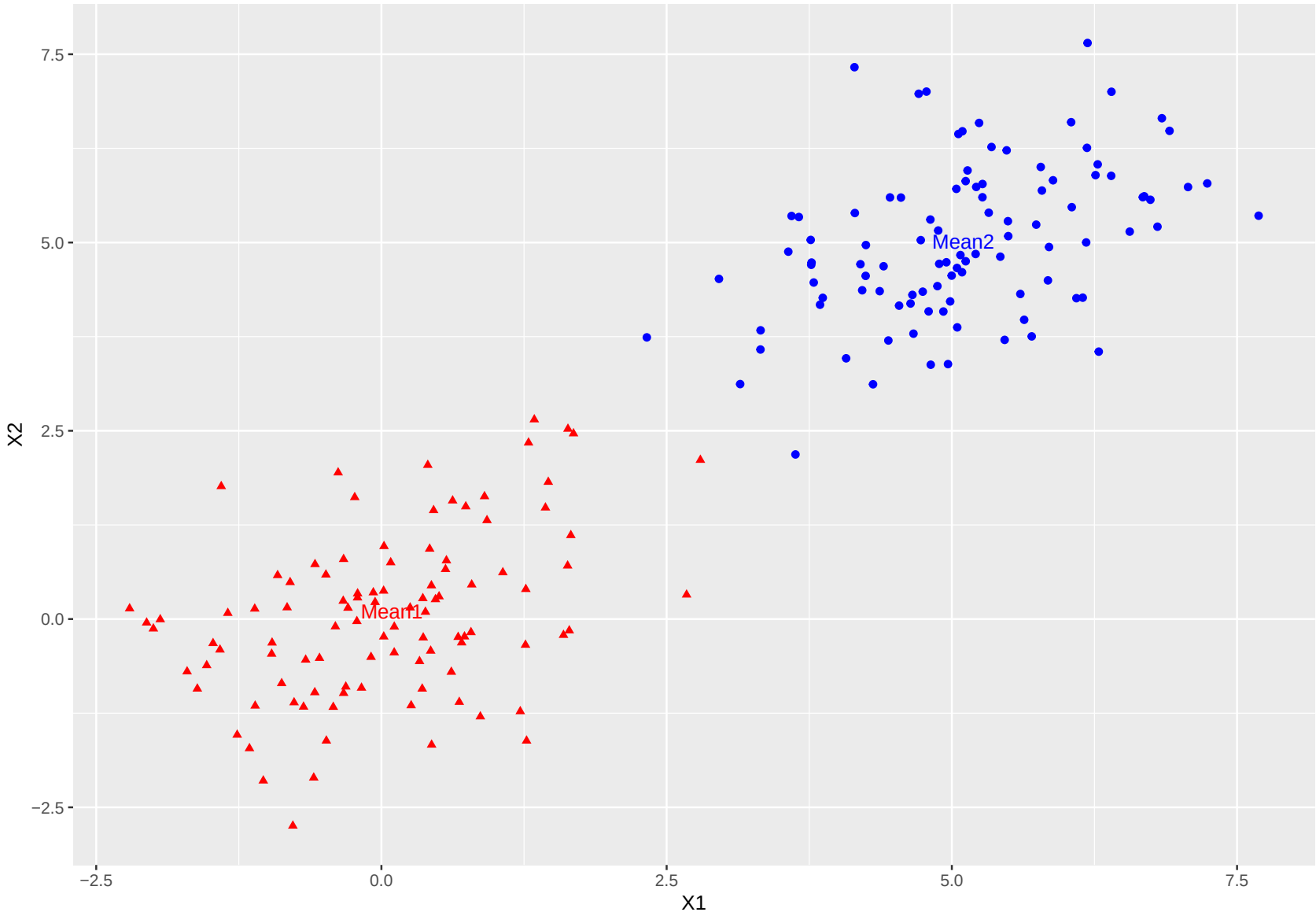
4. Relocate centroids to the mean (`X1`, `X2`) value for each cluster:



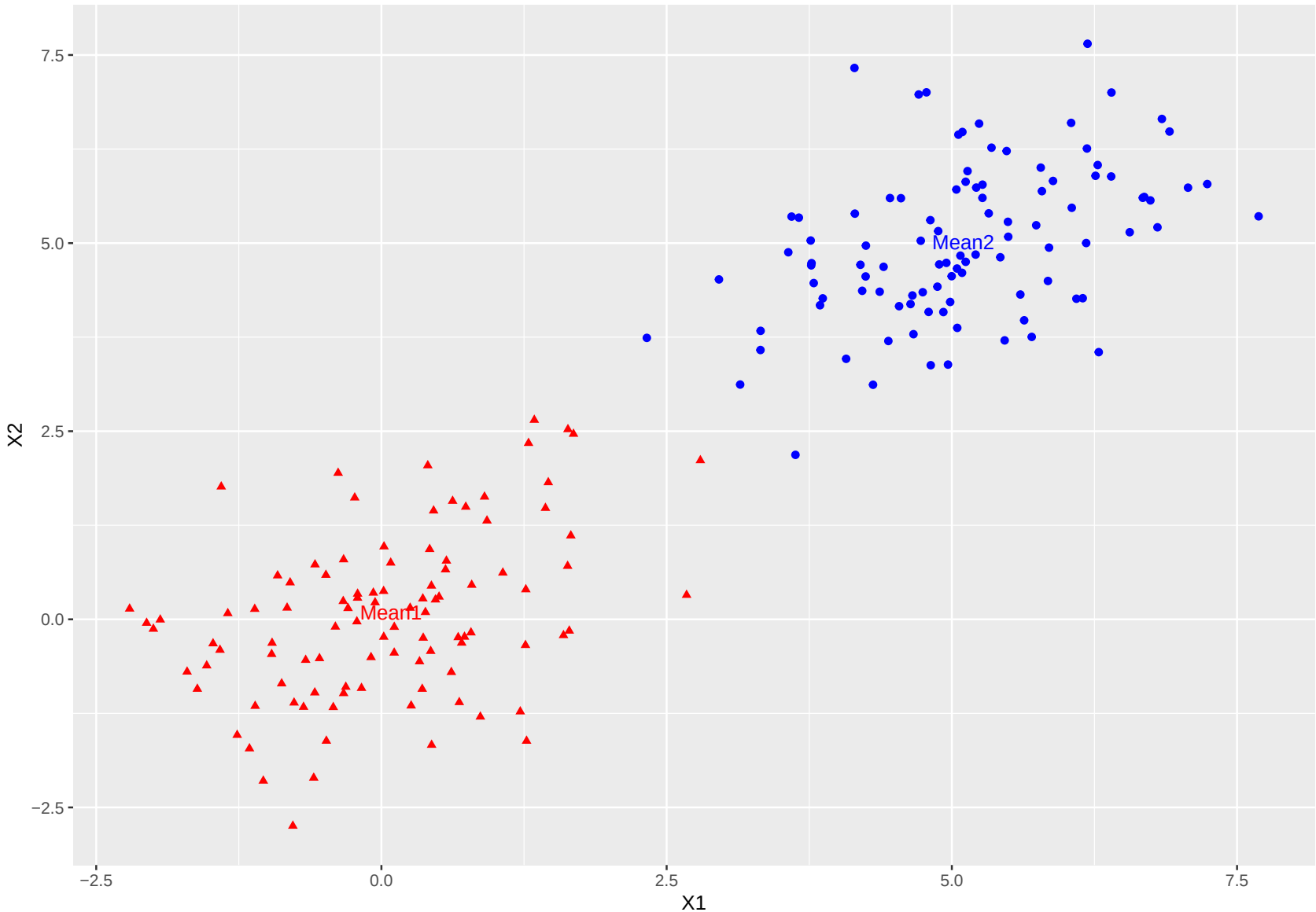
3. Reassign each observation to the closer centroid:



4. Recalculate the centroid locations . . .

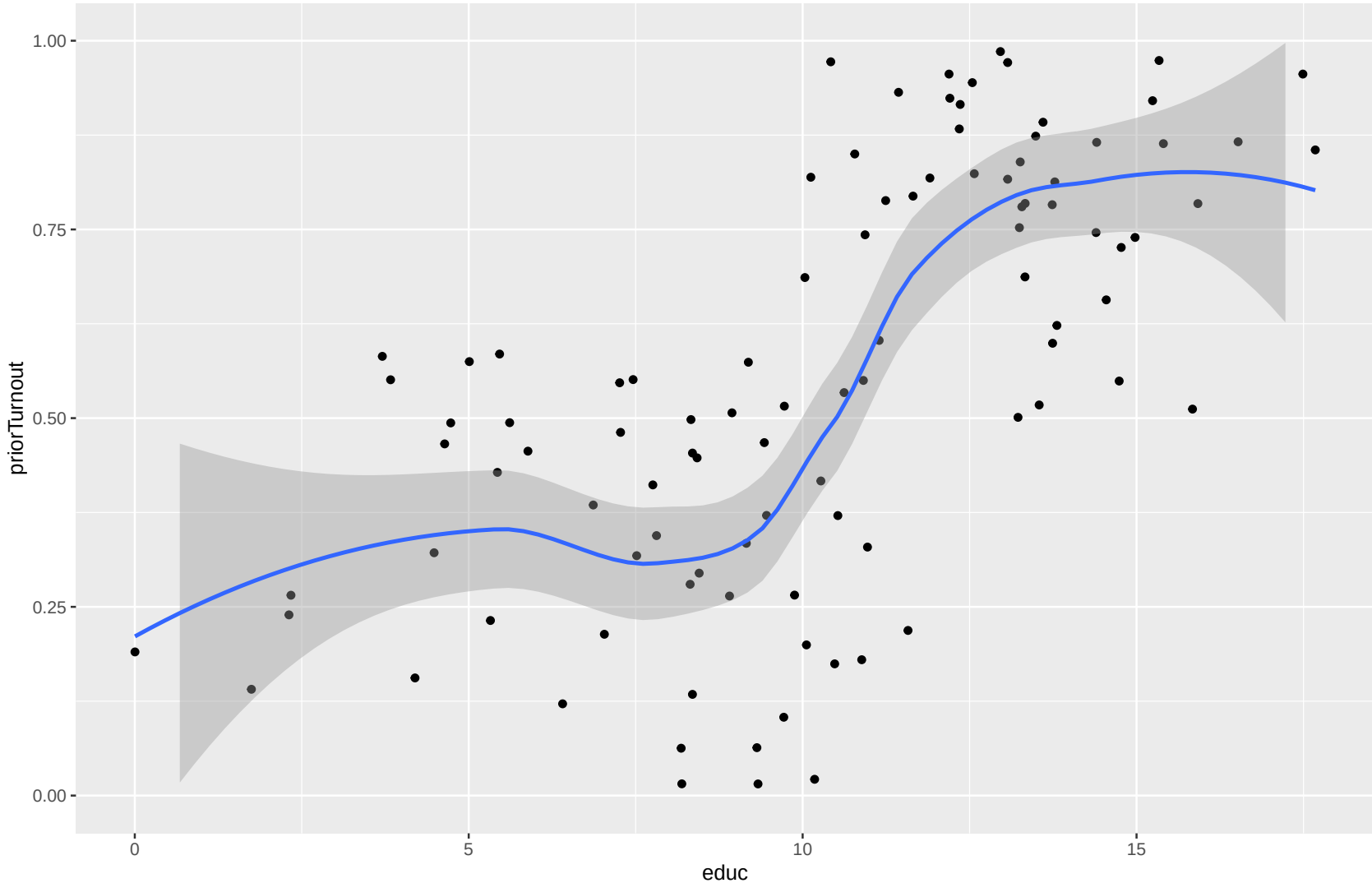


3. Reassign points to clusters . . .



Standardization: Why, before clustering?

Suppose we have prior turnout $[0, 1]$ and education (yrs):



```r
k2 <- kmeans(df2, centers = 2)
names(k2)
```

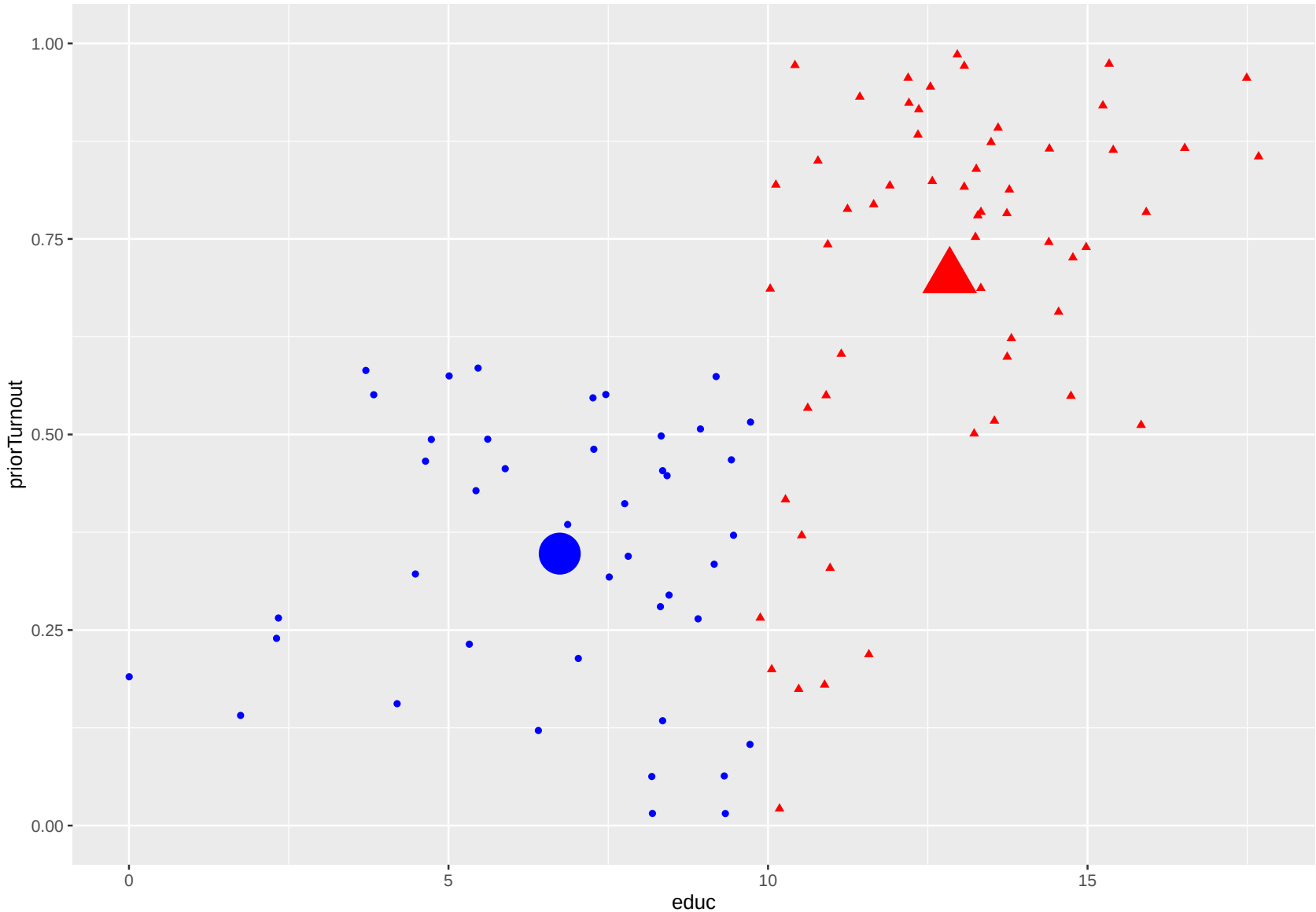
```
## [1] "cluster"      "centers"      "totss"        "withi
## [6] "betweenss"    "size"         "iter"         "ifaul
```

```r
table(k2$cluster)
```

```
##
##  1  2
## 57 43
```

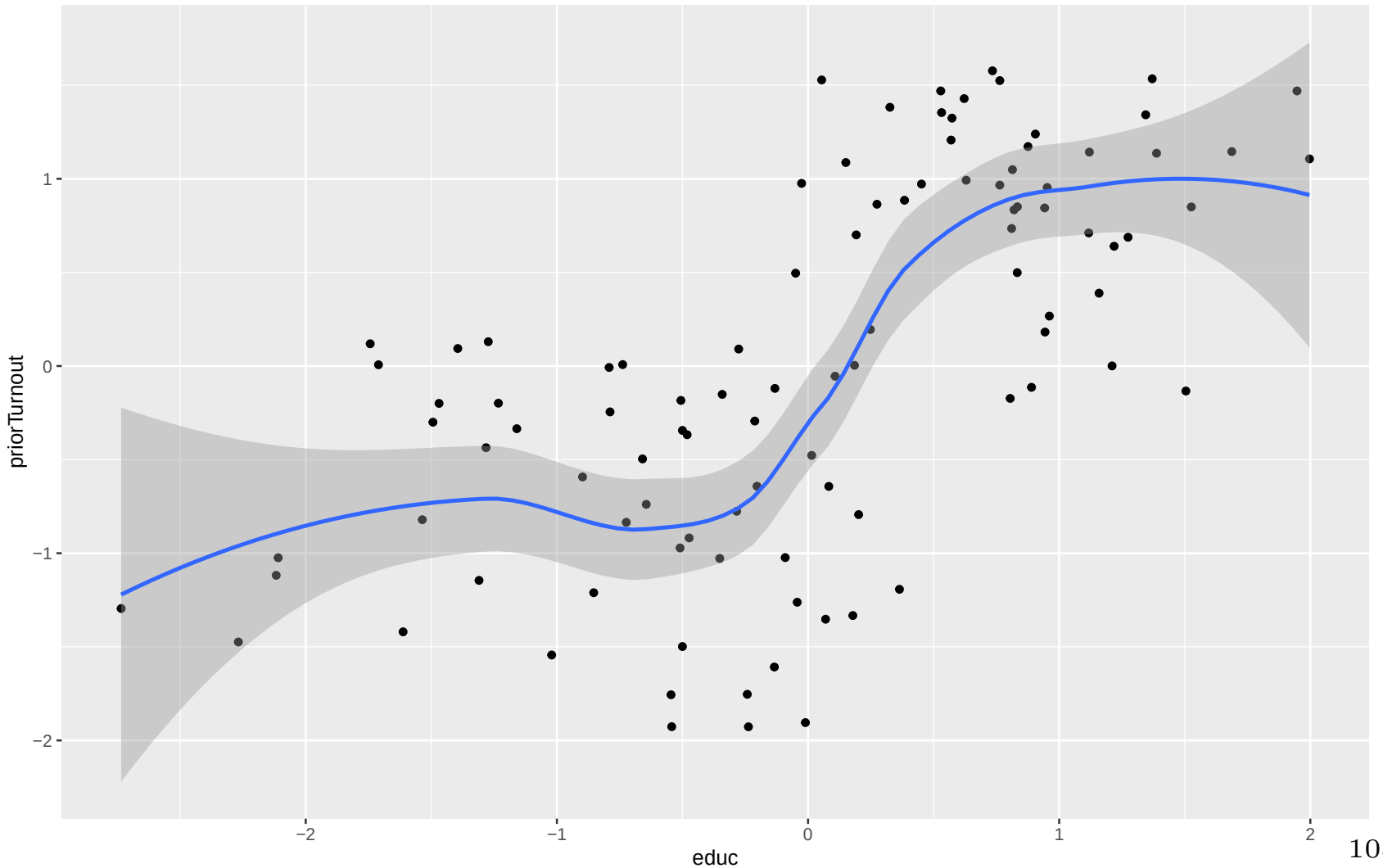
```r
k2$centers
```

```
##        educ priorTurnout
## 1 12.841805    0.7007953
## 2  6.740813    0.3477128
```

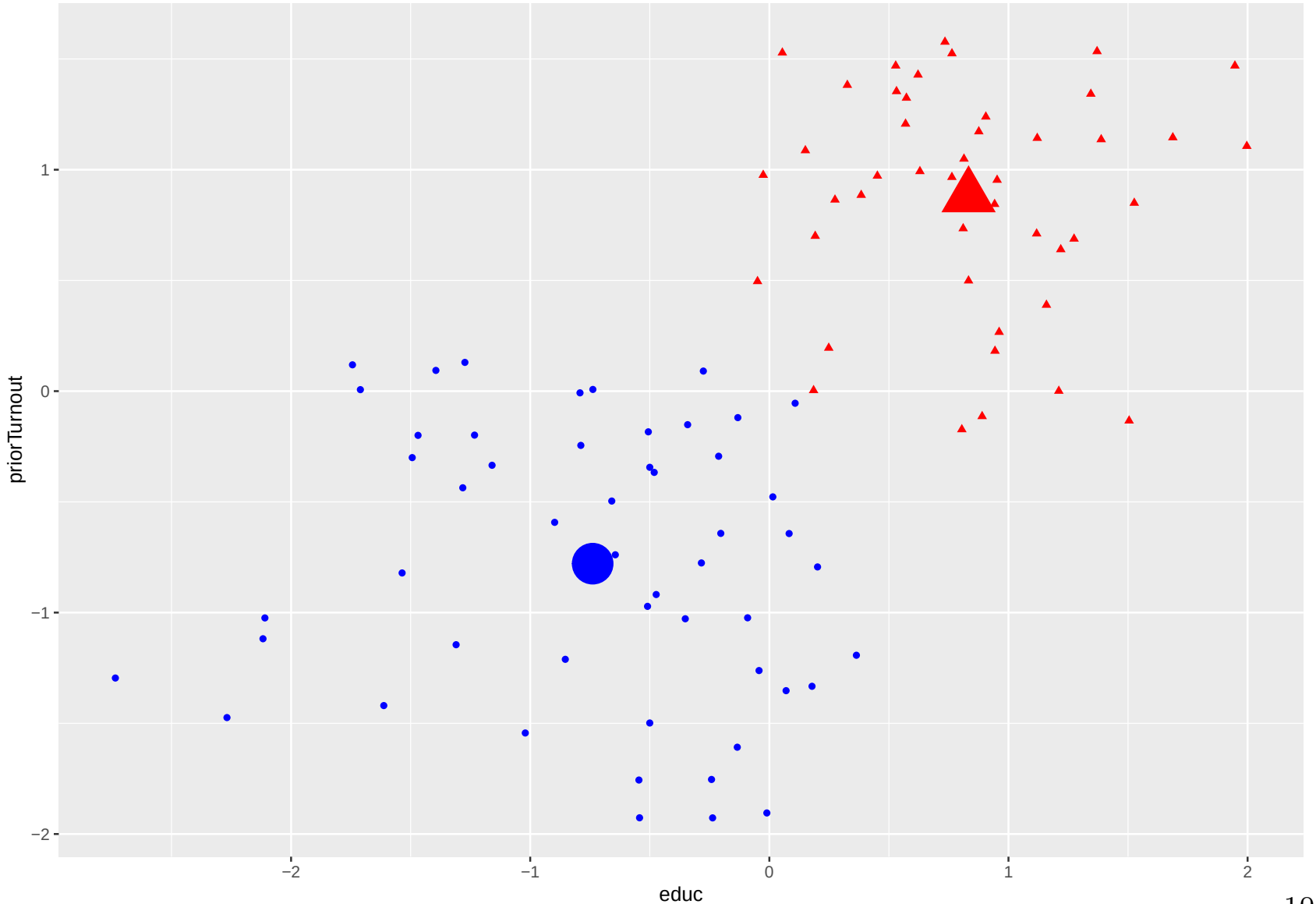

1.00
0.75
0.50
0.25
0.00
priorTurnout
0
5
10
15
educ

Standardized:

```
df2.standard <- data.frame(scale(df2))
ggplot(df2.standard, aes(educ, priorTurnout)) +
  geom_point() + geom_smooth()
```



```
##          educ priorTurnout
## 1 -0.7388515   -0.7791249
## 2  0.8331730    0.8785876
```



Selecting k

How do we select k?

Quick quiz:

Given $y = (4, 2, 5, 9, 10)$, calculate

$$\sum_{i=3}^{4} y_i$$

.

How do we select k?

Total within-cluster sum of squares:

$$TSS_{\text{Within}} = \sum_{(\text{Clusters})} \sum_{(\text{Obs in Cluster}_i)} (\text{Dist}(\text{Obs to Cluster Mean}))^2$$

How do we select k?

Total within-cluster sum of squares:

$$
\begin{aligned}
TSS_{\text{Within}} &= \sum_{(\text{Clusters})} \sum_{(\text{Obs in Cluster}_i)} (\text{Dist}(\text{Obs to Cluster Mean}))^2 \\
&= \sum_{i=1}^{k} \sum_{x_j \in C_i} (\text{Distance}(x_j \text{ to } \mu_i))^2 \\
&= \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2
\end{aligned}
$$

How do we select k?

Total within-cluster sum of squares:

$$
\begin{aligned}
TSS_{\text{Within}} &= \sum_{(\text{Clusters})} \sum_{(\text{Obs in Cluster}_i)} (\text{Dist}(\text{Obs to Cluster Mean}))^2 \\
&= \sum_{i=1}^{k} \sum_{x_j \in C_i} (\text{Distance}(x_j \text{ to } \mu_i))^2 \\
&= \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2
\end{aligned}
$$

- TSS_{Within} should *always* decrease with more clusters

How do we select k?

Total within-cluster sum of squares:

$$
\begin{aligned}
TSS_{\text{Within}} &= \sum_{(\text{Clusters})} \sum_{(\text{Obs in Cluster}_i)} (\text{Dist}(\text{Obs to Cluster Mean}))^2 \\
&= \sum_{i=1}^{k} \sum_{x_j \in C_i} (\text{Distance}(x_j \text{ to } \mu_i))^2 \\
&= \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2
\end{aligned}
$$

- TSS_{Within} should *always* decrease with more clusters
  - (But what's the point of 200 obs, 200 clusters?)

How do we select k?

Total within-cluster sum of squares:

$$
\begin{aligned}
TSS_{\text{Within}} &= \sum_{(\text{Clusters})} \sum_{(\text{Obs in Cluster}_i)} (\text{Dist}(\text{Obs to Cluster Mean}))^2 \\
&= \sum_{i=1}^{k} \sum_{x_j \in C_i} (\text{Distance}(x_j \text{ to } \mu_i))^2 \\
&= \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2
\end{aligned}
$$

- TSS_{Within} should *always* decrease with more clusters
  - (But what's the point of 200 obs, 200 clusters?)
- Select k s.t. additional cluster yields $\downarrow\downarrow\downarrow$ in TSS_{Within}

How do we select k?

```
max_k <- 10 # maximum number clusters to test
tot_within_ss <- vector("numeric", length = max_k - 1)

for(idx_k in 2:max_k){

  k_out <- kmeans(df2.standard, centers = idx_k)
  tot_within_ss[idx_k - 1] <- k_out$tot.withinss

}
```

How do we select k?

```
max_k <- 10 # maximum number clusters to test
tot_within_ss <- vector("numeric", length = max_k - 1)

for(idx_k in 2:max_k){

  k_out <- kmeans(df2.standard, centers = idx_k)
  tot_within_ss[idx_k - 1] <- k_out$tot.withinss

}
```

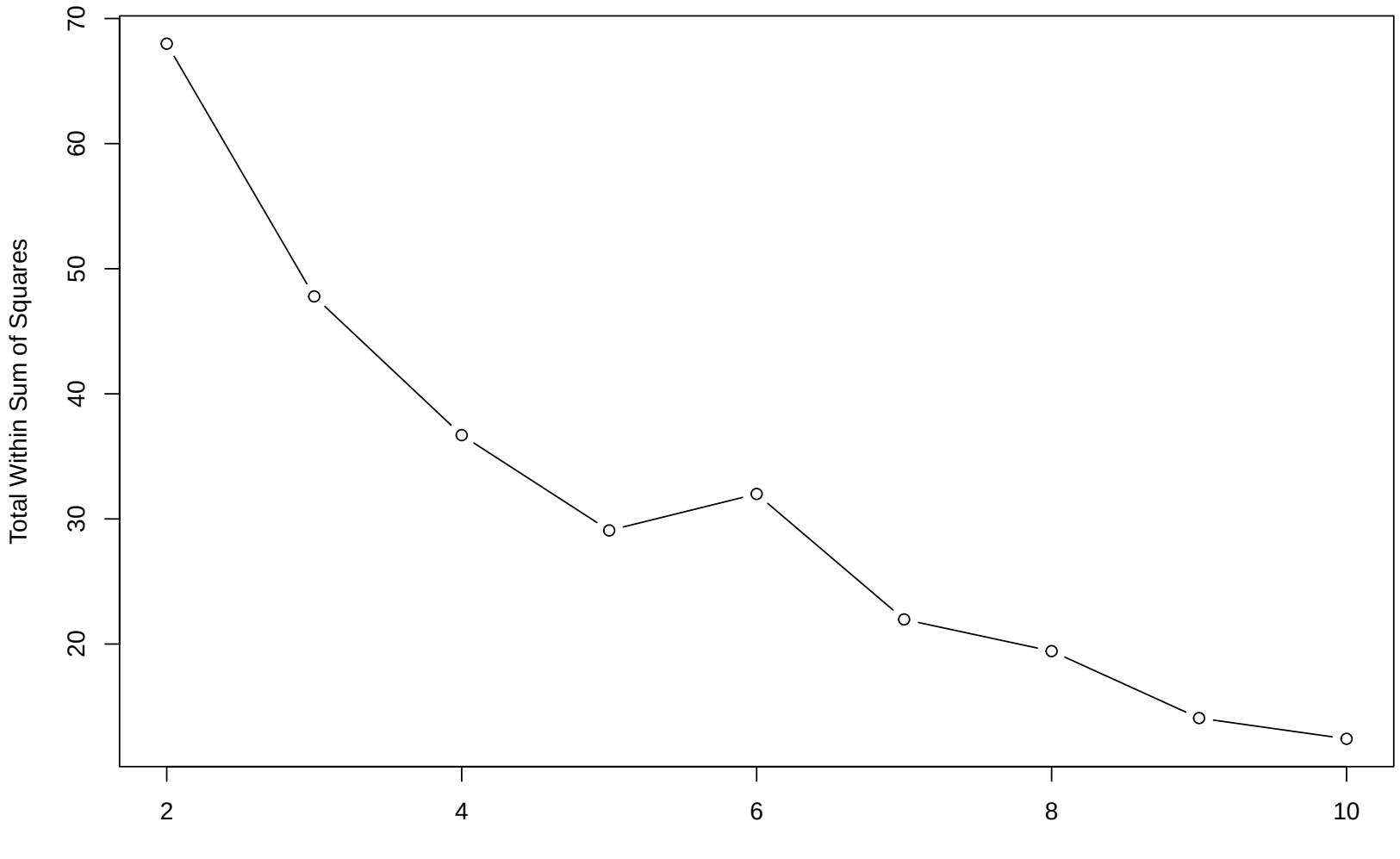
Output:

```
tot_within_ss |> round(1)
```

```
## [1] 68.0 47.8 36.7 29.1 32.0 22.0 19.4 14.1 12.4
```

A Scree Plot

```
plot(2:max_k, tot_within_ss, xlab = "Number of Clusters, k"
     ylab = "Total Within Sum of Squares", type = "b")
```



How do we select k?

```
max_k <- 10 # maximum clusters to test
tot_within_ss <- vector("numeric", length = max_k - 1)

for(idx_k in 2:max_k){

  k_out <- kmeans(df2.standard,
                  centers = idx_k,
                  nstart = 10)

  tot_within_ss[idx_k - 1] <- k_out$tot.withinss
}
```

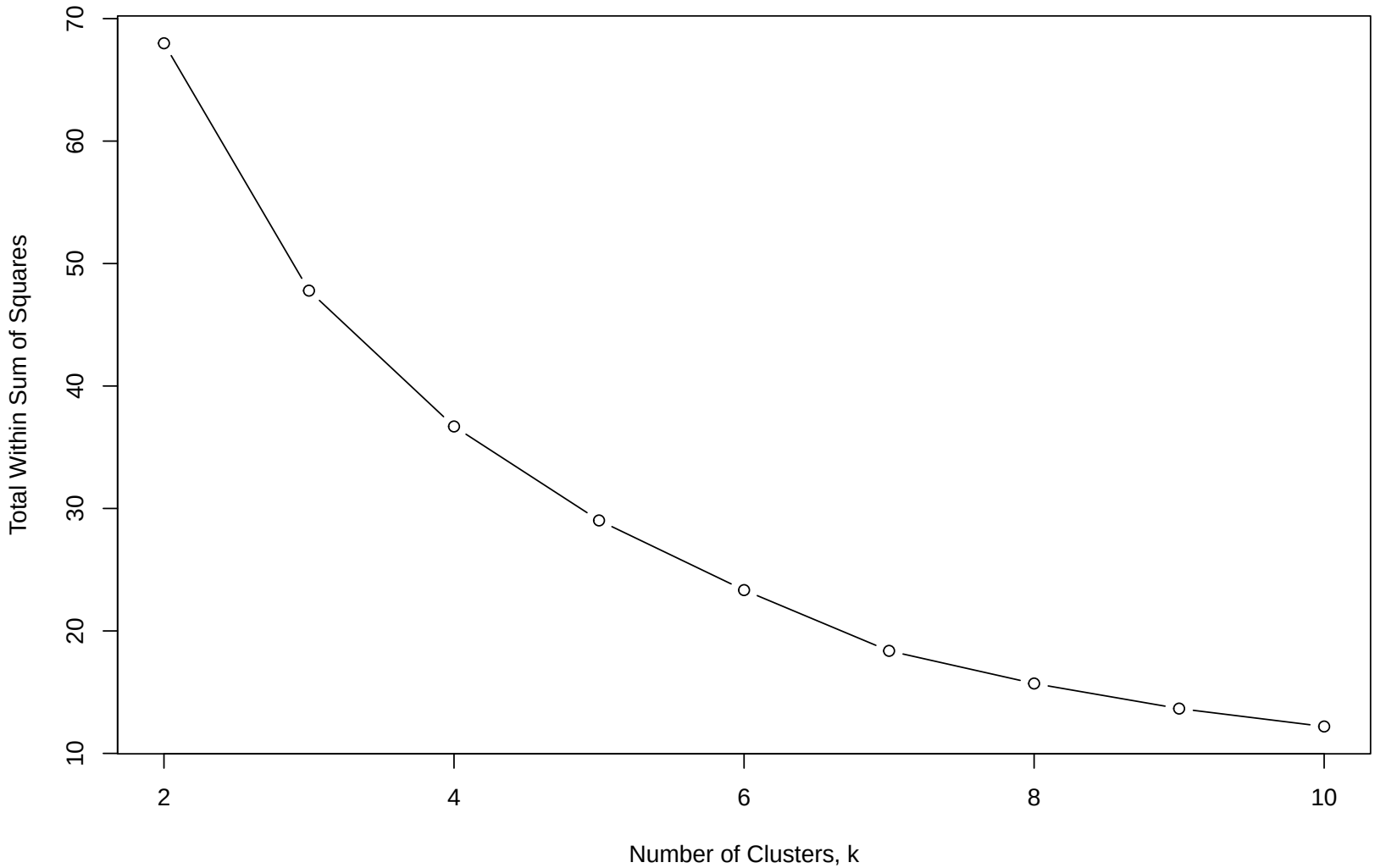
How do we select k?

```
max_k <- 10 # maximum clusters to test
tot_within_ss <- vector("numeric", length = max_k - 1)

for(idx_k in 2:max_k){

  k_out <- kmeans(df2.standard,
                  centers = idx_k,
                  nstart = 10)

  tot_within_ss[idx_k - 1] <- k_out$tot.withinss
}
```

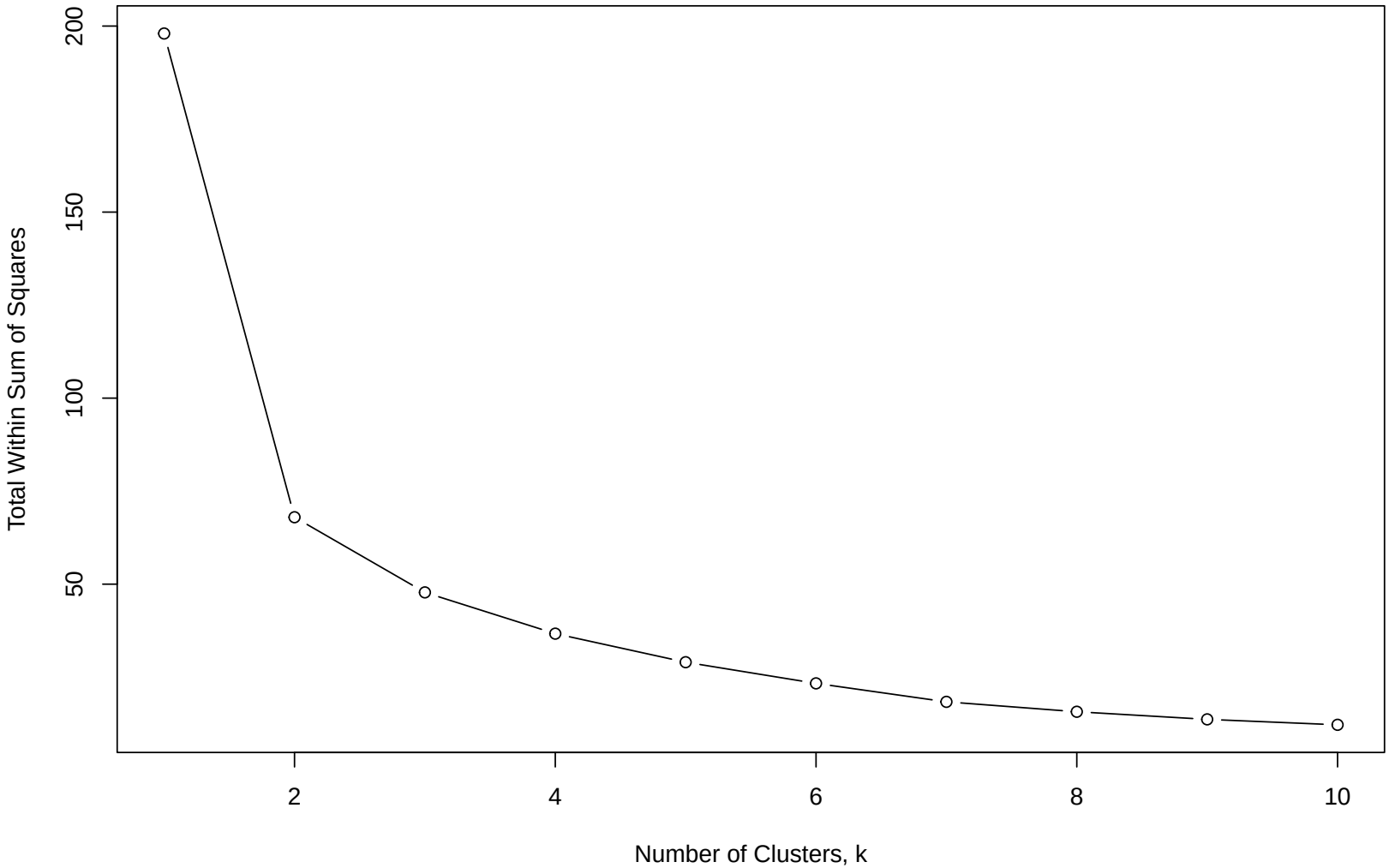
```
tot_within_ss |> round(1)
```

```
## [1] 68.0 47.8 36.7 29.0 23.3 18.4 15.7 13.7 12.2
```

A Better Scree Plot



An Even Better Scree Plot



Other applications: Geolocations

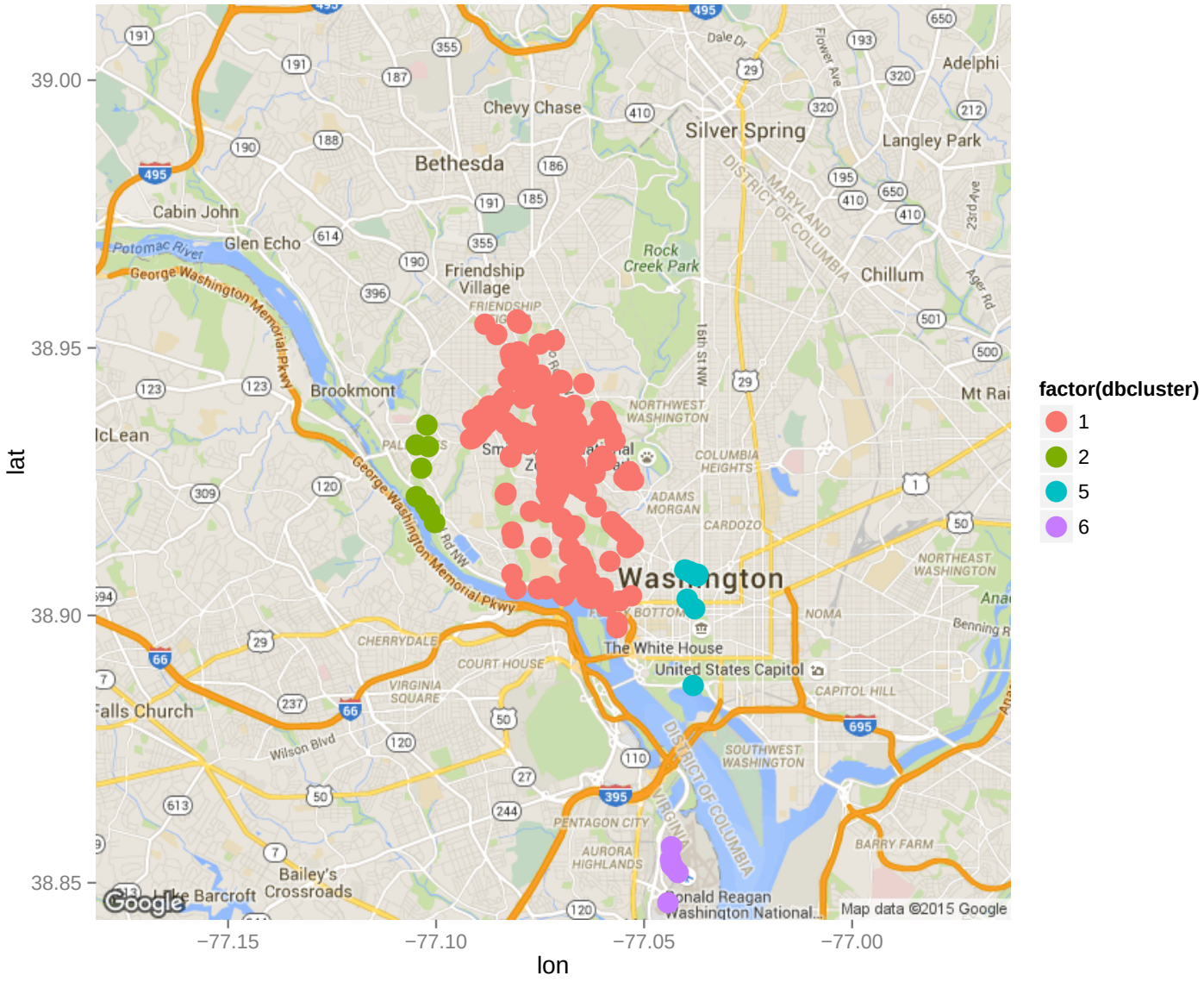


Figure 1: Clusters of Geolocations

Other applications: Regimes

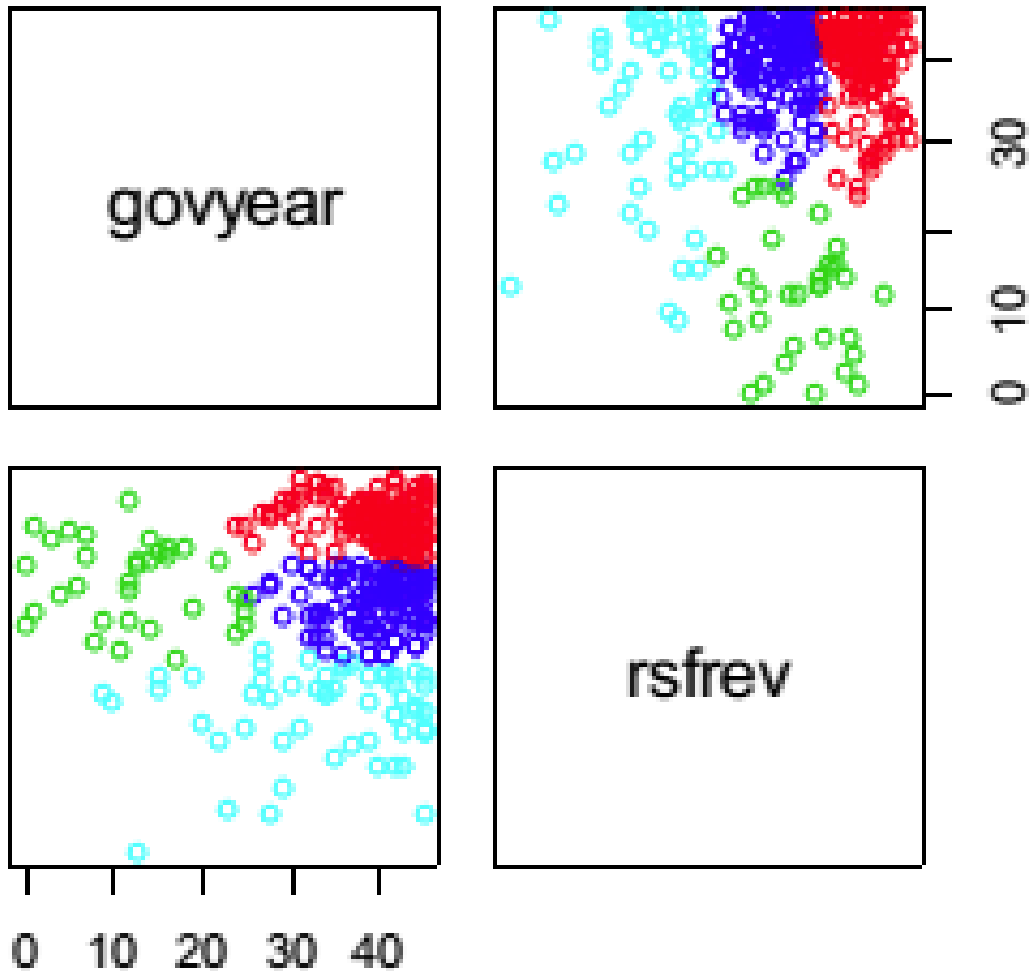


Figure 2: Comparative Regime Types

Other applications: Senate Speeches

Other applications: Senate Speeches

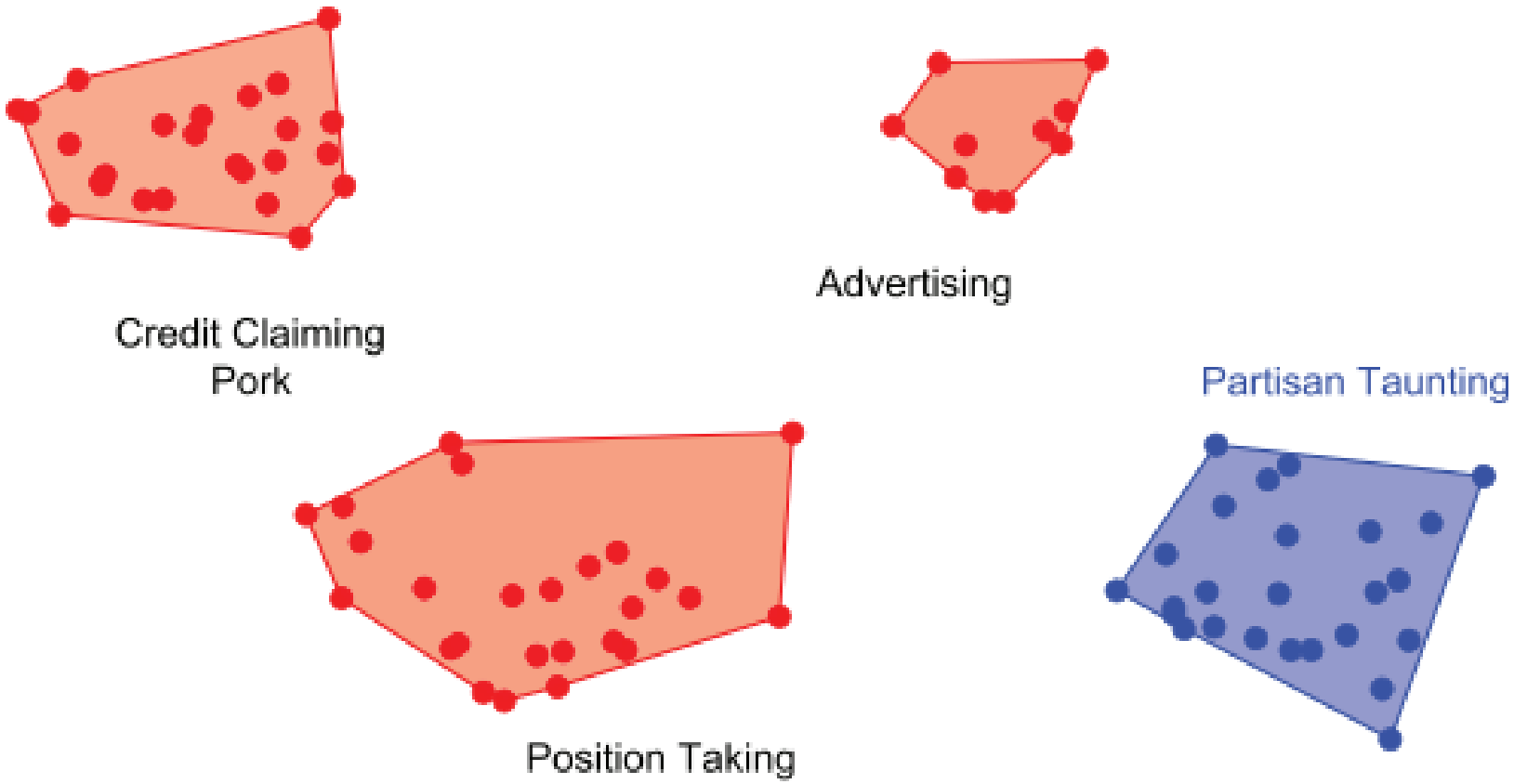


Figure 3: Senate Speeches

DBSCAN (1996)

Density-based clustering

DBSCAN (1996)

Density-based clustering

1. Find each point's neighbors

DBSCAN (1996)

Density-based clustering

1. Find each point's neighbors
2. ID *core* points with enough neighbors

DBSCAN (1996)

Density-based clustering

1. Find each point's neighbors
2. ID *core* points with enough neighbors
3. Connect nearby core points

DBSCAN (1996)

Density-based clustering

1. Find each point's neighbors
2. ID *core* points with enough neighbors
3. Connect nearby core points
4. Assign non-core points to near clusters (or noise)

Other applications:

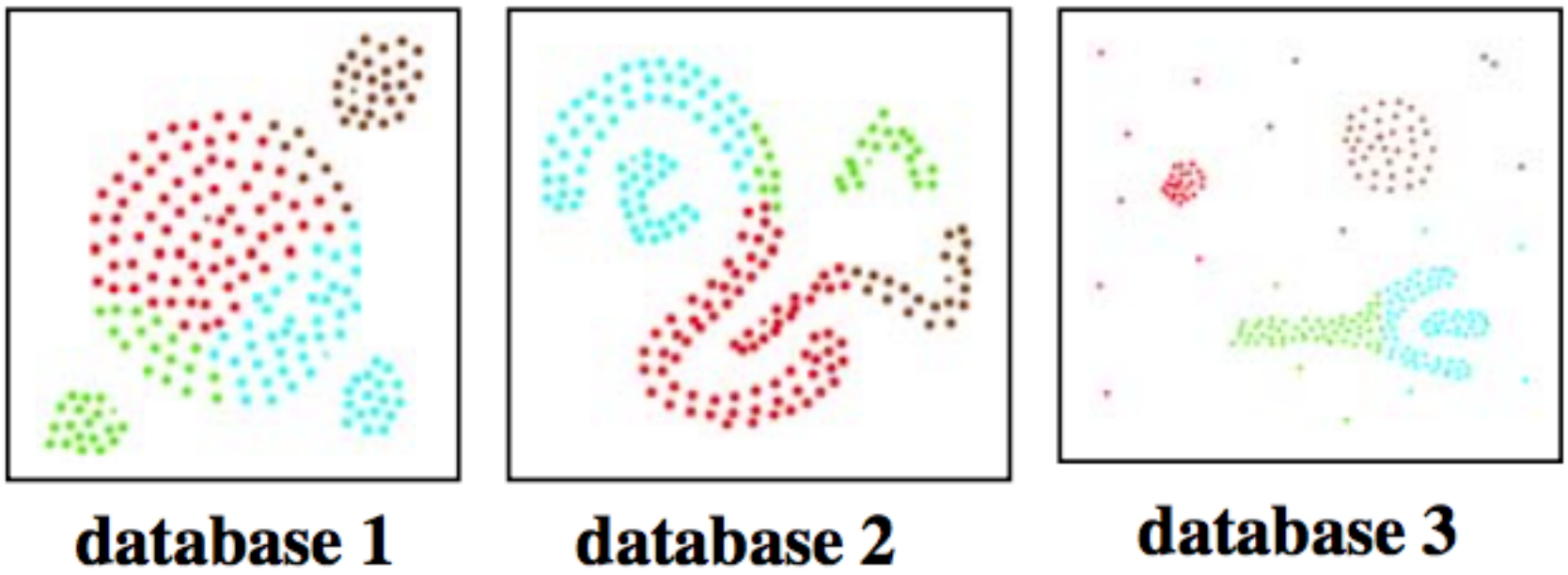


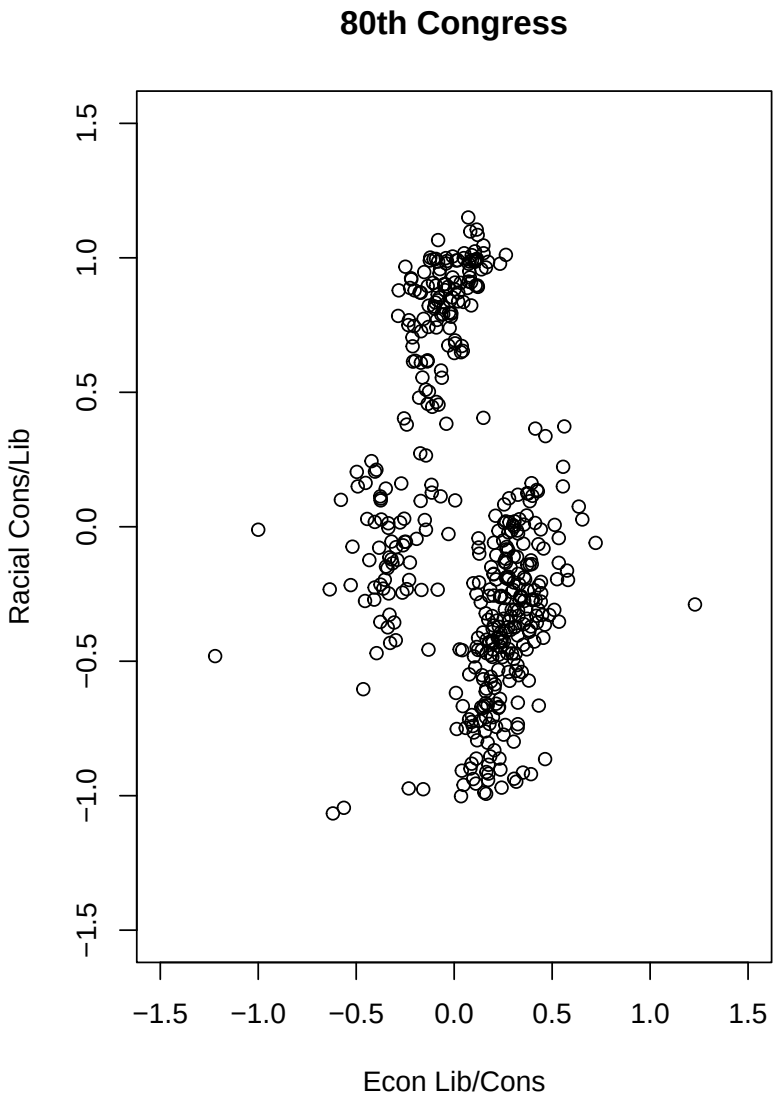
Figure 4: CLARANS: Not Great!

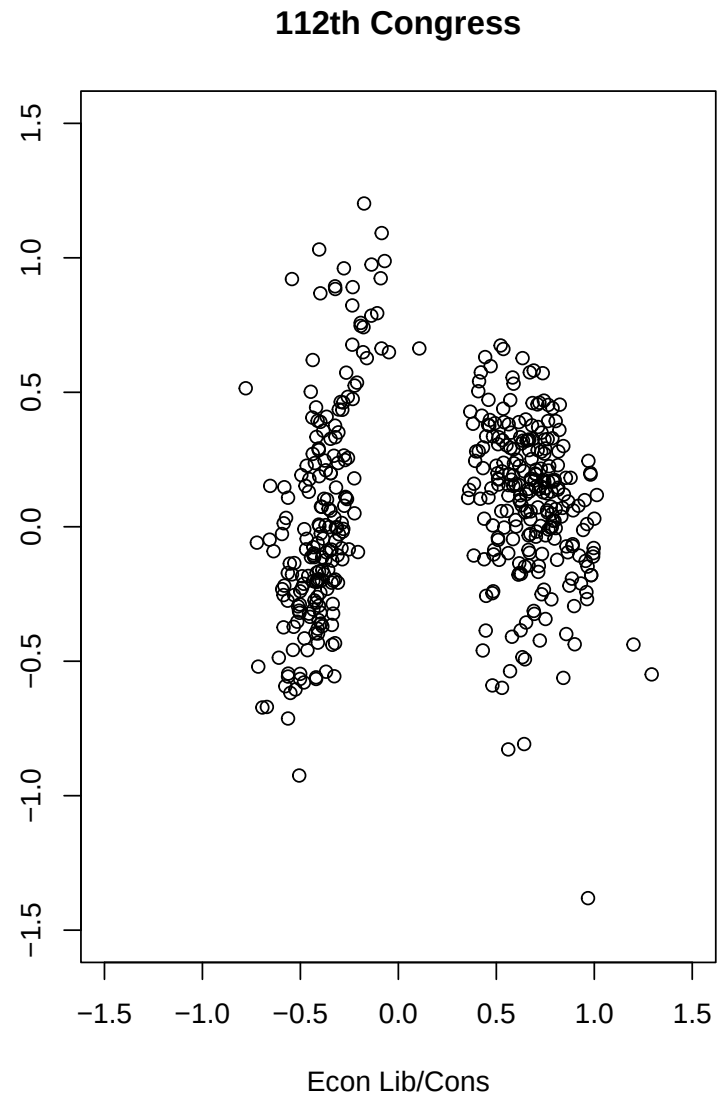
Other applications:

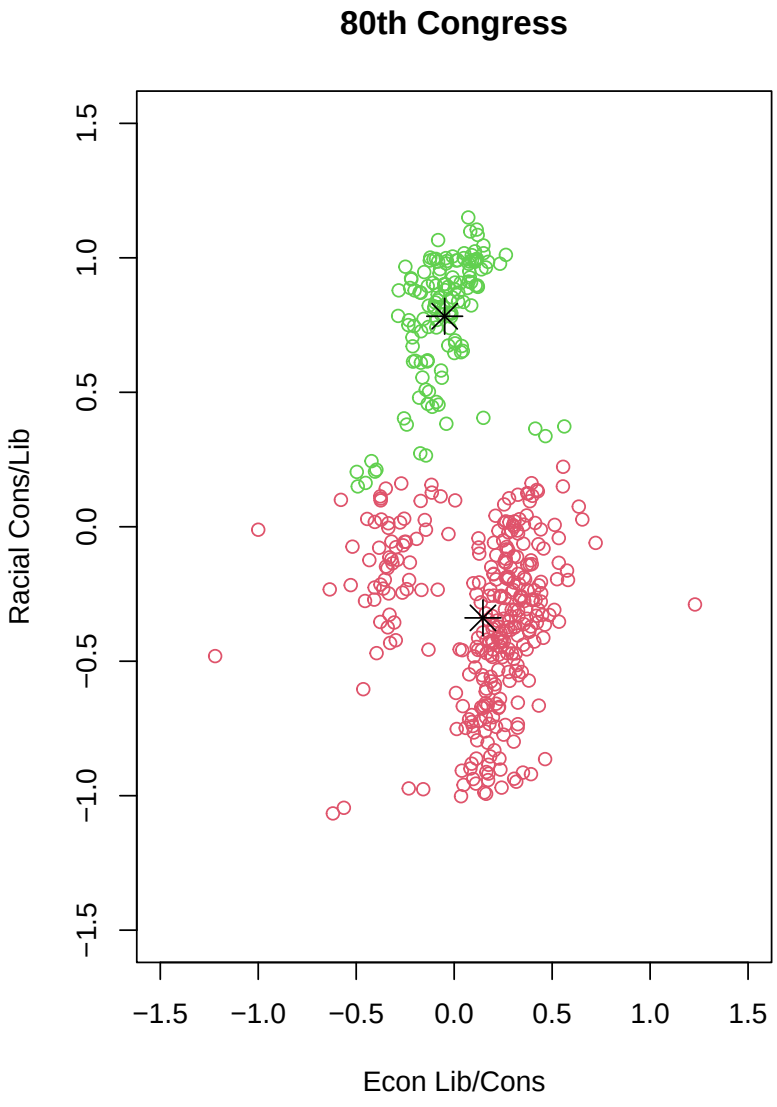
Figure 5: DBSCAN: Great!

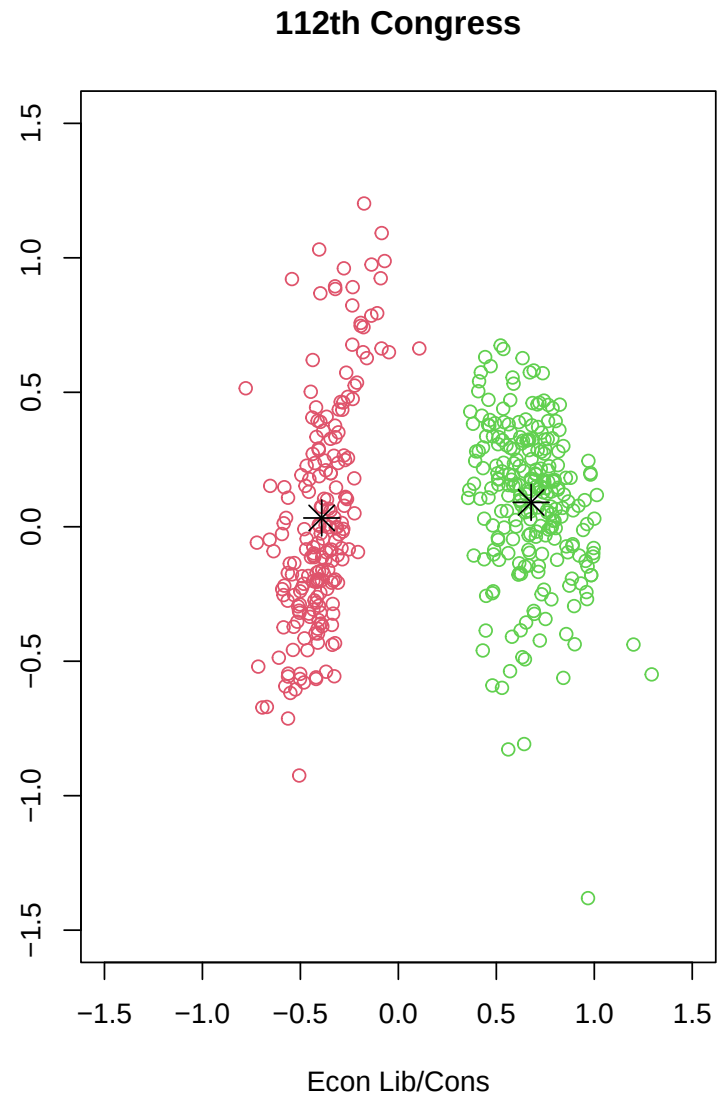
Congress Clusters

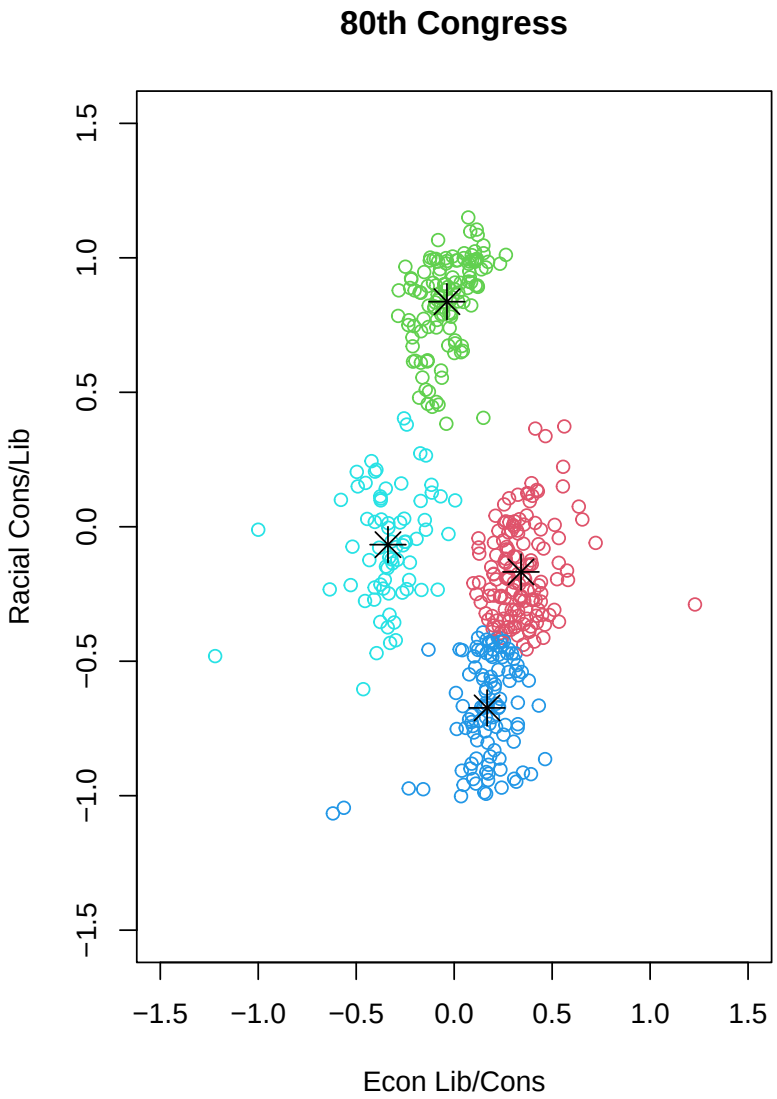
```
congress.url <- "https://tinyurl.com/2kcf3p9r"
congress <- read_csv(congress.url)
dwnom80 <- cbind(congress$dwnom1[congress$congress == 80],
                 congress$dwnom2[congress$congress == 80])
dwnom112 <- cbind(congress$dwnom1[congress$congress == 112]
                  congress$dwnom2[congress$congress == 112]

k80two.out <- kmeans(dwnom80, centers = 2)
k112two.out <- kmeans(dwnom112, centers = 2)

k80four.out <- kmeans(dwnom80, centers = 4)
k112four.out <- kmeans(dwnom112, centers = 4)

lim <- c(-1.5, 1.5)
xlab <- "Econ Lib/Cons"
ylab <- "Racial Cons/Lib"
```

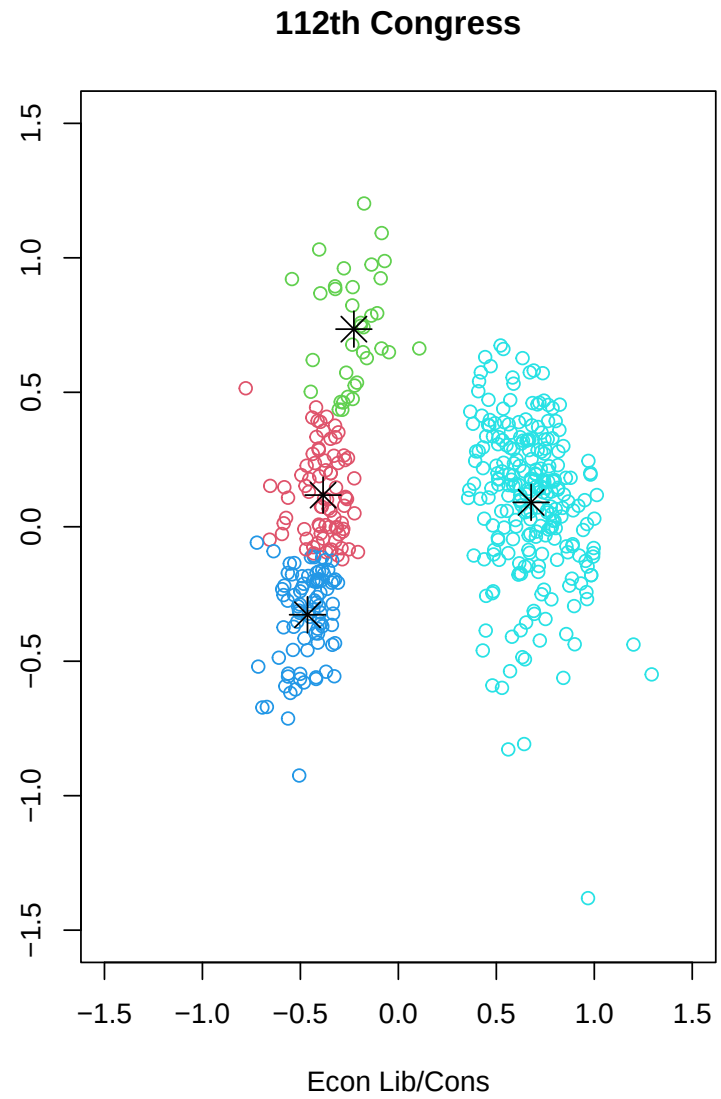

80th Congress
Racial Cons/Lib
Econ Lib/Cons
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


112th Congress
Econ Lib/Cons
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


80th Congress
Racial Cons/Lib
Econ Lib/Cons
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


112th Congress
Econ Lib/Cons
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


80th Congress
Racial Cons/Lib
Econ Lib/Cons
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


112th Congress
Econ Lib/Cons
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

Exercise

Exercise

1. Checkout CRAN Task View for Clustering:

https://cran.r-project.org/web/views/Cluster.html

2. Discover clusters in your final project data!

(Use 2 predictors to visualise; more predictors to discover higher-dim clusters.)