Clustering - V

Prof. Dan A. Simovici

UMB
1. The Kernelized Version of $k$-Means

2. Kernelized Clustering in R

3. The PAM Algorithm

4. Examining a Data Frame in R

5. Executing PAM in R
Let $X = \{x_1, \ldots, x_n\}$ be a set of vectors in $\mathbb{R}^m$ and let $\phi : \mathbb{R}^m \rightarrow \mathbb{R}^p$ be a function, where $\mathbb{R}^p$ is a high-dimensional space referred to as the feature space.

Since $\mathbb{R}^p$ may have a much higher dimension than $m$, the computation of the inner product $\phi(x_i)'\phi(x_j)$ can be costly. Therefore, it is interesting to examine transformations of the form $\phi : \mathbb{R}^m \rightarrow \mathbb{R}^p$ for which there exists a symmetric function $K : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$ such that

$$\phi(x_i)'\phi(x_j) = K(x_i, x_j)$$

for $x_i, x_j \in X$. Thus, a kernel function facilitates the computation of the inner product $\phi(x_i)'\phi(x_j)$ in the high-dimensional space.
Recall:

**Definition**

A function $K : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$ is a $\phi$-kernel if $\phi(x_i)'\phi(x_j) = K(x_i, x_j)$ for all $x_i, x_j \in X$.

A function $K : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$ is a *kernel* if there exists a function $\phi : \mathbb{R}^m \rightarrow \mathbb{R}^p$ such that $K$ is a $\phi$-kernel.
Example

Let $K : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$ be the function defined as

$$K(x, y) = (x' y + a)^d$$

for $x, y \in \mathbb{R}^m$. We can write

$$K(x, y) = (x' y + a)^d = (x_1 y_1 + \cdots + x_m y_m + \sqrt{a} \sqrt{a})^d$$

$$= \sum_{n_1 + \cdots + n_{m+1} = d} \binom{d}{n_1 \cdots n_m n_{m+1}} x_1^{n_1} y_1^{n_1} \cdots x_m^{n_m} y_m^{n_m} a^{n_{m+1} \over 2} a^{n_{m+1} \over 2}$$

$$= (\phi(x), \phi(y)),$$

where

$$\phi(x) = \left( \ldots, \sqrt{\binom{d}{n_1 \cdots n_m n_{m+1}}} x_1^{n_1} \cdots x_m^{n_m} a^{n_{m+1} \over 2}, \ldots \right).$$
Example cont’d

For each monomial that is a component of \( \phi(x) \) we have \( \sum_{j=1}^{m+1} n_j = d \) and \( n_j \geq 0 \) for \( 1 \leq j \leq m + 1 \).

When \( m = 2 \) and \( d = 2 \) we have

\[
K(x, y) = (x_1y_1 + x_2y_2 + a)^2 = x_1^2y_1^2 + a^2 + 2x_1y_1x_2y_2 + 2ax_1y_1 + 2ax_2y_2
\]

where \( \phi : \mathbb{R}^2 \rightarrow \mathbb{R}^6 \) is defined as

\[
\phi(x) = \begin{pmatrix}
x_1^2 \\
x_2^2 \\
\sqrt{2x_1x_2} \\
\sqrt{2ax_1} \\
\sqrt{2ax_2} \\
a
\end{pmatrix}
\]

for \( x \in \mathbb{R}^2 \).
Example

Let $K : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$ be the radial basis function defined as:

$$K(x, y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}$$

for $x, y \in \mathbb{R}^m$.

The linear space $\ell^2(\mathbb{R})$ is an infinite-dimensional linear space that consists of infinite sequences $s = (s_0, s_1, \ldots) \in \text{Seq}(\mathbb{R})$ such that $\sum_{n \in \mathbb{N}} s_n^2$ is finite. We shall prove that $K$ corresponds to a feature function $\phi : \mathbb{R}^m \rightarrow \ell^2(\mathbb{R})$. 
Example cont’d

We have:

\[ K(x, y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}} = e^{-\frac{\|x\|^2 + \|y\|^2 - 2(x, y)}{2\sigma^2}} = e^{-\frac{\|x\|^2}{2\sigma^2}} \cdot e^{-\frac{\|y\|^2}{2\sigma^2}} \cdot e^{\frac{(x, y)}{\sigma^2}} \]

Taking into account that \( e^{\frac{(x, y)}{\sigma^2}} = \sum_{j=0}^{\infty} \frac{1}{j!} \frac{(x, y)^j}{\sigma^{2j}} \) we can write:

\[ e^{\frac{(x, y)}{\sigma^2}} \cdot e^{-\frac{\|x\|^2}{2\sigma^2}} \cdot e^{-\frac{\|y\|^2}{2\sigma^2}} = \sum_{j=0}^{\infty} \frac{(x, y)^j}{j! \sigma^{2j}} \cdot e^{-\frac{\|x\|^2}{2\sigma^2}} \cdot e^{-\frac{\|y\|^2}{2\sigma^2}} = \sum_{j=0}^{\infty} \left( \frac{e^{-\frac{\|x\|^2}{2j\sigma^2}}}{\sigma \sqrt{j!} \frac{1}{j}} \frac{e^{-\frac{\|y\|^2}{2j\sigma^2}}}{\sigma \sqrt{j!} \frac{1}{j}} (x, y) \right)^j = (\phi(x), \phi(y)). \]
The function $\phi$ is given by:

$$
\phi(x) = \left( \ldots, \frac{e^{-\frac{||x||^2}{2j\sigma^2}}}{\sigma^j \sqrt{j!}^j} \left( \begin{array}{c} j \\ n_1, \ldots, n_k \end{array} \right)^{\frac{1}{2}} x_1^{n_1} \cdots x_k^{n_k}, \ldots \right).
$$

In this formula $j$ varies in $\mathbb{N}$ and $n_1 + \cdots + n_k = j$. 
The Kernelized Version of $k$-Means

**Theorem**

Let function $\phi : \mathbb{R}^m \rightarrow \mathbb{R}^p$ be a function, $c^*$ be the centroid of the set $\phi(X) \subseteq \mathbb{R}^p$, and let $sse^*(X)$ be the sum of square errors of $\phi(X)$ in $\mathbb{R}^p$. We have:

$$sse^*(X) = \sum_{j=1}^{n} K(x_j, x_j) - nK(c^*, c^*).$$

Furthermore, if $\sigma = \{C_1, \ldots, C_m\}$ is a partition of $X$ and $c_k^*$ is the centroid of $\phi(C_k)$ for $1 \leq k \leq m$, define the sum of the squared errors of $\sigma$ in $\mathbb{R}^p$ as

$$sse^*(\sigma) = \sum_{\ell=1}^{m} sse^*(C_\ell).$$

Then, we have:

$$sse^*(\sigma) = \sum_{j=1}^{n} K(x_j, x_j) - \sum_{\ell=1}^{m} |C_\ell|K(c_\ell^*, c_\ell^*).$$
Proof

The definition of $\text{sse}^*$ implies:

$$\text{sse}^*(X) = \sum_{j=1}^{n} \| \phi(x_j) - c^* \|^2,$$

where $c^* = \frac{1}{n} \sum_{j=1}^{n} \phi(x_j)$. Therefore,

$$\text{sse}^*(X) = \sum_{j=1}^{n} (\phi(x_j) - c^*)'(\phi(x_j) - c^*)$$

$$= \sum_{j=1}^{n} \phi(x_j)'\phi(x_j) - 2 \sum_{j=1}^{n} \phi(x_j)'c^* + n(c^*)'c^*$$

$$= \sum_{j=1}^{n} K(x_j, x_j) - nK(c^*, c^*)$$
Proof cont’d

For the second equality note that

$$\text{sse}^*(C_\ell) = \sum_{x_j \in C_\ell} K(x_j, x_j) - |C_\ell| K(c^*_\ell, c^*_\ell).$$

This allows us to write

$$\sum_{\ell=1}^{m} \text{sse}^*(C_\ell)$$

$$= \sum_{\ell=1}^{m} \left( \sum_{x_j \in C_\ell} K(x_j, x_j) - |C_\ell| K(c^*_\ell, c^*_\ell) \right)$$

$$= \sum_{j=1}^{n} K(x_j, x_j) - \sum_{\ell=1}^{m} |C_\ell| K(c^*_\ell, c^*_\ell)$$
The Kernelized Version of \(k\)-Means

The distance in the feature space between \(\phi(x_j)\) and the centroid \(c_i^*\) of the set \(C_i\) can be expressed using the kernel \(K\).

**Theorem**

*The distance in the feature space between \(\phi(x_j)\) and the centroid \(c_i^*\) of the set \(C_i\) can be expressed using the kernel \(K\).*
Proof

We have:

\[
\| \phi(x_j) - c_i^* \|^2 = \| \phi(x_j) \|^2 - 2\phi(x_j)'c_i^* + \| c_i^* \|^2 \\
= K(x_j, x_j) - \frac{2}{|C_i|} \sum_{x_p \in C_i} \phi(x_j)'\phi(x_p) \\
+ \frac{1}{|C_i|^2} \sum_{x_p \in C_i} \sum_{x_q \in C_i} K(x_p, x_q) \\
= K(x_j, x_j) - \frac{2}{|C_i|} \sum_{x_p \in C_i} K(x_j, x_p) \\
+ \frac{1}{|C_i|^2} \sum_{x_p \in C_i} \sum_{x_q \in C_i} K(x_p, x_q).
\]
The theorem shows that the distance from the image of a point to the centroid of the cluster in the feature space can be computed via the kernel. The point \( x_j \) is assigned to the cluster \( C_i \), where

\[
i = \arg\min_i \| \phi(x_j) - c_i^* \|^2
\]

\[
= \arg\min_i \left( K(x_j, x_j) - \frac{2}{|C_i|} \sum_{x_p \in C_i} K(x_j, x_p) + \frac{1}{|C_i|^2} \sum_{x_p \in C_i} \sum_{x_q \in C_i} K(x_p, x_q) \right)
\]

\[
= \arg\min_i \left( \frac{1}{|C_i|^2} \sum_{x_p \in C_i} \sum_{x_q \in C_i} + K(x_p, x_q) - \frac{2}{|C_i|} \sum_{x_p \in C_i} K(x_j, x_p) \right),
\]

because the term \( K(x_j, x_j) \) is the same for every \( i \). The kernelized version of the \( k \)-means algorithm proceeds along the same line as the standard \( k \)-means except that the assignment on points to centroids is done in the feature space rather than the original space.
The `kernlab` package contains the function `kkmeans` that implements kernel $k$-means using a variety of possible kernels: the radial base kernel, `rbfdot`, the polynomial kernel `polydot`, and many others. After loading this package using

```r
> library(kernlab)
```

we can build a clustering of the `iris` database excluding the attribute `Species` using

```r
> c1 <- kkmeans(as.matrix(iris[, -5]), centers=3)
```
This call to function \texttt{kkmeans} is using the \texttt{rbfdot} kernel automatic estimation of $\sigma$ and results in the clustering \texttt{c1} described below:

\begin{verbatim}
> c1
Spectral Clustering object of class "specc"

          Cluster memberships:
          
2 3 2 3 2 2 3 2 3 2 3 2 3 2 2 2 2 2 2 2 2 2 3 2 ...

Gaussian Radial Basis kernel function.
Hyperparameter : sigma = 1.15460641776431

Centers:

\begin{verbatim}
[1,]  6.262000  2.872000  4.906000  1.6760000
[2,]  5.151351  3.556757  1.502703  0.2594595
[3,]  4.592308  3.061538  1.346154  0.2076923
\end{verbatim}

Cluster size:

\begin{verbatim}
[1] 100 37 13
\end{verbatim}
\end{verbatim}

Note that the resulting clusters are quite unbalanced containing 100, 37, and 13 specimens and, thus, quite distance from the "natural" clusters that have 50 entries each.
A better result can be obtained with the polydot kernel:

```r
> c2 <- kkmeans(as.matrix(iris[, -5]), kernel = "polydot",
               kpar = list(degree = 2), centers = 3)
```

The resulting clustering is

```r
> c2
Spectral Clustering object of class "specc"

Cluster memberships:

3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 ...
```

Polynomial kernel function.

Hyperparameters: degree = 2 scale = 1 offset = 1

Centers:

```
[1,] 6.853846  3.076923  5.715385  2.053846
[2,] 5.991667  2.795833  4.562500  1.512500
[3,] 5.104762  3.244444  1.933333  0.431746
```
Cluster size:
[1] 39 48 63

Within-cluster sum of squares:
[1] 1182.046 1121.498 1566.437

Note that the sizes of the clusters are closer to the sizes of the reference clusters; also the values of the within-clusters sums of squares are more balanced.
Similar results can be obtained with the polydot kernel of degree 3.
Another algorithm, named PAM (an acronym of “Partition Around Medoids”) developed by Kaufman and Rousseeuw, also requires as an input parameter the number $k$ of clusters to be extracted. The $k$ clusters are determined based on a representative object from each cluster, called the *medoid* of the cluster. The medoid of a cluster is one of the objects that have a most central position in the cluster.
Objects that are tentatively defined as medoids are placed into a set $S$ of \textit{selected objects}. If $O$ is the set of objects then the set $U = O - S$ is the set of \textit{unselected objects}.

The goal of the algorithm is to minimize the average dissimilarity of objects to their closest selected object. Equivalently, we can minimize the sum of the dissimilarities between object and their closest selected object.
The PAM Algorithm

The algorithm has two phases:

1. In the first phase, BUILD, a collection of \( k \) objects are selected for an initial set \( S \).

2. In the second phase, SWAP, one tries to improve the quality of the clustering by exchanging selected objects with unselected objects.
For each object \( p \) we maintain two numbers:

- \( D_p \), the dissimilarity between \( p \) and the closest object in \( S \), and
- \( E_p \), the dissimilarity between \( p \) and the second closest object in \( S \).

These numbers must be updated every time when the sets \( S \) and \( U \) change. Note that \( D_j \leq E_j \) and that we have \( p \in S \) if and only if \( D_p = 0 \).
PAM begins with a set of objects $O$, where $|O| = n$, a dissimilarity $n \times n$ matrix $D$, and a prescribed number of clusters $k$. The $d_{ij}$ entry of the matrix $D$ is the dissimilarity $d(o_i, o_j)$ between the objects $o_i$ and $o_j$. 
The algorithm has two phases:

- the *building phase*, and
- the *swapping phase*. 
The BUILD phase

1. Initialize $S$ by adding to it an object for which the sum of the distances to all other objects is minimal.

2. Consider an object $i \in U$ as a candidate for inclusion into the set of selected objects.

3. For an object $j \in U$ compute $D_j$, the dissimilarity between $j$ and the closest object in $S$.

4. If $D_j > d(i, j)$ object $j$ will contribute to the decision to select object $i$ (because the quality of the clustering may benefit); let $C_{ji} = \max\{D_j - d(j, i), 0\}$.

5. Compute the total gain obtained by adding $i$ to $S$ as $g_i = \sum_{j \in U} C_{ji}$.

6. Choose that object $i$ that maximizes $g_i$; let $S := S \cup \{i\}$ and $U = U - \{i\}$.

These steps are performed until $k$ objects have been selected.
The PAM Algorithm

$S$

$D_p$

$p$

$E_p$

$i$

$d(i,j)$

$j$

$E_j$

$D_j$
The second phase, SWAP, attempts to improve the set of selected objects and, therefore, to improve the quality of the clustering. This is done by considering all pairs \((i, h) \in S \times U\) and consists of computing the effect \(T_{ih}\) on the sum of dissimilarities between objects and the closest selected object caused by swapping \(i\) and \(h\), that is, by transferring \(i\) from \(S\) to \(U\) and transferring \(h\) to from \(U\) to \(S\).

The computation of \(T_{ih}\) involves the computation of the contribution \(K_{jih}\) of each object \(j \in U\) to the swap of \(i\) and \(h\).
$K_{jih}$ is computed taking into account the following cases:

(a) if $d(j, i) > D_j$ and $d(j, h) > D_j$ (which means that there is $\ell \in S$ such that $d(j, h) > d(j, \ell)$) then $K_{jih} = 0$;

(b) if $d(j, i) = D_j$, then two cases occur:
   
   (b)-(i) if $d(j, h) < E_j$, where $E_j$ is the dissimilarity between $j$ and the second closest selected object, then $K_{jih} = d(j, h) - d(j, i)$; note that $K_{jih}$ can be either positive or negative.
   
   (b)-(ii) if $d(j, h) \geq E_j$, then $K_{jih} = E_j - D_j$; in this case $K_{jih} > 0$.

(c) if $d(j, i) > D_j$ and $d(j, h) < D_j$, then

$$K_{jih} = d(j, h) - D_j.$$
Computation of $K_{jih}$

Cases (a) and (b)-(i)

$$K_{jik} = 0$$

\[
\begin{array}{c}
D_j \\
d(j, i) \\
d(j, h)
\end{array}
\]

\[
\begin{array}{c}
D_j \\
(a)
\end{array}
\]

\[
\begin{array}{c}
d(j, i) = D_j \\
d(j, h) \\
K_{jih} = d(j, h) - d(j, i)
\end{array}
\]

(b)-(i)
Computation of $K_{jih}$

Cases (b)-(ii) and (c)

\[ d(i, j) = D_j \quad \quad E_j \quad \quad d(j, h) \]

\[ K_{jik} = E_j - D_j \]

(b)-(ii)

\[ d(j, h) \quad D_j \quad d(j, i) \quad K_{jih} = d(j, h) - D_j \]

(c)
The PAM Algorithm

- Compute the total result of the swap as
  \[ T_{ih} = \sum \{ K_{ji} \mid j \in U \} \].

- Select a pair \((i, h) \in S \times U\) that minimizes \(T_{ih}\).

- If \(T_{ih} < 0\) the swap is carried out, \(D_p\) and \(E_p\) are updated for every object \(p\), and we return at Step 1. If \(\min T_{ih} > 0\), the value of the objective cannot be decreased and the algorithm halts. Of course, this happens when all values of \(T_{ih}\) are positive and this is precisely the halting condition of the algorithm.
Specialized Functions

Functions that allow visualization of a data frame:
- `dim()` shows the dimensions of the data frame by row and column
- `str()` shows the structure of the data frame
- `summary()` provides summary statistics on the columns
- `colnames()` shows the name of each column in the data frame
- `head()` shows the first 6 rows of the data frame
- `tail()` shows the last 6 rows of the data frame
- `View()` shows a spreadsheet-like display of the entire data frame
Example

We demonstrate the use of these functions on the data frame `USArrests`:

```r
> d <- USArrests
> dim(d)
[1] 50 4
> str(d)
'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 72 80 60 ...
$ Rape : num 21.2 44.5 31 19.5 40.6 38.7 11.1 15.8 31.9 25.8 ...
```
Example

> summary(d)

<table>
<thead>
<tr>
<th>Murder</th>
<th>Assault</th>
<th>UrbanPop</th>
<th>Rape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.  : 0.800</td>
<td>Min.  : 45.0</td>
<td>Min.  : 32.00</td>
<td>Min.  : 7.30</td>
</tr>
<tr>
<td>1st Qu.: 4.075</td>
<td>1st Qu.:109.0</td>
<td>1st Qu.:54.50</td>
<td>1st Qu.:15.07</td>
</tr>
<tr>
<td>Median : 7.250</td>
<td>Median :159.0</td>
<td>Median :66.00</td>
<td>Median :20.10</td>
</tr>
<tr>
<td>Mean  : 7.788</td>
<td>Mean  :170.8</td>
<td>Mean  :65.54</td>
<td>Mean  :21.23</td>
</tr>
<tr>
<td>3rd Qu.:11.250</td>
<td>3rd Qu.:249.0</td>
<td>3rd Qu.:77.75</td>
<td>3rd Qu.:26.18</td>
</tr>
<tr>
<td>Max.  :17.400</td>
<td>Max.  :337.0</td>
<td>Max.  :91.00</td>
<td>Max.  :46.00</td>
</tr>
</tbody>
</table>

> colnames(d)

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

```r
> head(d)

<table>
<thead>
<tr>
<th></th>
<th>Murder</th>
<th>Assault</th>
<th>UrbanPop</th>
<th>Rape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alabama</td>
<td>13.2</td>
<td>236</td>
<td>58</td>
<td>21.2</td>
</tr>
<tr>
<td>Alaska</td>
<td>10.0</td>
<td>263</td>
<td>48</td>
<td>44.5</td>
</tr>
<tr>
<td>Arizona</td>
<td>8.1</td>
<td>294</td>
<td>80</td>
<td>31.0</td>
</tr>
<tr>
<td>Arkansas</td>
<td>8.8</td>
<td>190</td>
<td>50</td>
<td>19.5</td>
</tr>
<tr>
<td>California</td>
<td>9.0</td>
<td>276</td>
<td>91</td>
<td>40.6</td>
</tr>
<tr>
<td>Colorado</td>
<td>7.9</td>
<td>204</td>
<td>78</td>
<td>38.7</td>
</tr>
</tbody>
</table>
```
Example

```r
> tail(d)

<table>
<thead>
<tr>
<th>State</th>
<th>Murder</th>
<th>Assault</th>
<th>UrbanPop</th>
<th>Rape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vermont</td>
<td>2.2</td>
<td>48</td>
<td>32</td>
<td>11.2</td>
</tr>
<tr>
<td>Virginia</td>
<td>8.5</td>
<td>156</td>
<td>63</td>
<td>20.7</td>
</tr>
<tr>
<td>Washington</td>
<td>4.0</td>
<td>145</td>
<td>73</td>
<td>26.2</td>
</tr>
<tr>
<td>West Virginia</td>
<td>5.7</td>
<td>81</td>
<td>39</td>
<td>9.3</td>
</tr>
<tr>
<td>Wisconsin</td>
<td>2.6</td>
<td>53</td>
<td>66</td>
<td>10.8</td>
</tr>
<tr>
<td>Wyoming</td>
<td>6.8</td>
<td>161</td>
<td>60</td>
<td>15.6</td>
</tr>
</tbody>
</table>
We need to install the following packages:

cluster
factoextra
ggplot2
ggsignif
A clustering of states that consists of two clusters can be obtained writing

```r
pam.res <- pam(d,2)
```

where `d` is the data frame that contains data from txUSArrests.
A call to `print(pam.res)` results in

**Medoids:**

<table>
<thead>
<tr>
<th>ID</th>
<th>Murder</th>
<th>Assault</th>
<th>UrbanPop</th>
<th>Rape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Michigan</td>
<td>22</td>
<td>12.1</td>
<td>255</td>
<td>74</td>
</tr>
<tr>
<td>Kansas</td>
<td>16</td>
<td>6.0</td>
<td>115</td>
<td>66</td>
</tr>
</tbody>
</table>

**Clustering vector:**

<table>
<thead>
<tr>
<th>Alabama</th>
<th>Alaska</th>
<th>Arizona</th>
<th>Arkansas</th>
<th>California</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Colorado</td>
<td>Connecticut</td>
<td>Delaware</td>
<td>Florida</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Hawaii</td>
<td>Idaho</td>
<td>Illinois</td>
<td>Indiana</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Kansas</td>
<td>Kentucky</td>
<td>Louisiana</td>
<td>Maine</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Massachusetts</td>
<td>Michigan</td>
<td>Minnesota</td>
<td>Mississippi</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Montana</td>
<td>Nebraska</td>
<td>Nevada</td>
<td>New Hampshire</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>New Mexico</td>
<td>New York</td>
<td>North Carolina</td>
<td>North Dakota</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Oklahoma</td>
<td>Oregon</td>
<td>Pennsylvania</td>
<td>Rhode Island</td>
<td>South Carolina</td>
</tr>
</tbody>
</table>

40 / 43
To show the clusters to which the states belong we write:

```r
> dd <- cbind(USArrests, cluster = pam.res$cluster)
> head(dd,n=3)

Murder Assault UrbanPop Rape cluster
Alabama 13.2 236 58 21.2 1
Alaska 10.0 263 48 44.5 1
Arizona 8.1 294 80 31.0 1

> pam.res$medoids

Murder Assault UrbanPop Rape
Michigan 12.1 255 74 35.1
Kansas 6.0 115 66 18.0
```
The clustering can be visualized using the function `fviz_cluster` of the package `factoextra` invoked as

```r
fviz_cluster(pam.res, geom = "point", ellipse.type = "norm")
```

The result is presented in the next slide, where states are represented as points using their first two principal components.
Executing PAM in R