Week 4 Cluster Analysis

Theory and Practice

Ying Lin, Ph.D

20 September, 2019

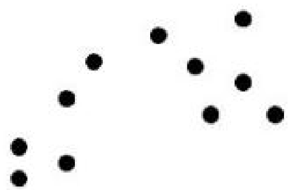
Overview of the Topics

- Basic concepts of cluster analysis
- KMeans algorithm
- Hierarchical Agglomerative Clustering (HAC)
- Evaluation of the clustering outputs
- Hyperparameters of the clustering algorithms
- Visualize clustering output: PCA (Principal Component Analysis)
- R/Python demo of the cluster analysis

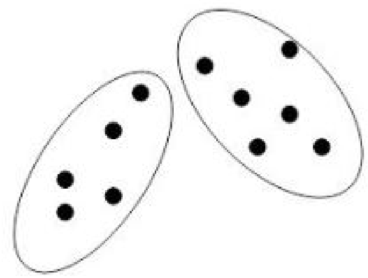
Motivations

- Identify grouping structure of data so that objects within the same group are closer (more similar) to each other while farther (less similar) to those in different groups
  - Various distance/proximity functions
  - intra-cluster distance vs. inter-cluster distance
- Unsupervised learning: used for data exploration
  - Can also be adapted for supervised learning purpose
- Types of cluster analysis
  - Partitional vs. Hierarchical: one point can belong to one or multiple clusters
  - kmeans algorithm vs. HAC (Hierarchical Agglomerative Clustering) algorithm

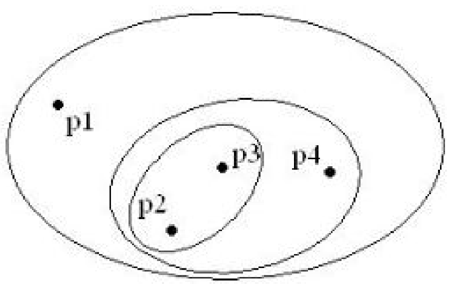
Partitional vs. Hierarchical Clustering



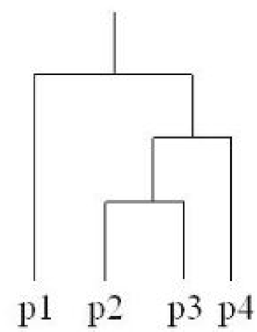
Original Points



A Partitional Clustering



Hierarchical Clustering



Dendrogram

Major Applications of Cluster Analysis

- Data sampling: use centroid as the representative samples
  - Centroid: center of mass which is caculated as the (arithmetic) mean of all the data points in the same cluster (separately for each dimension)
- Marketing segmentation analysis: help marketers segment their customer bases and develop targeted marketing programs
- Insurance: identify groups of motor insurance policy holders with a higher average claim cost
- Microarray analysis for biomedical research: A high throughput technology which allows testing thousand of genes simultaneously in different disease states

Distance Functions

- Clustering is based on . How to measure it?
- Minkowski distance: distance between two vectors is the norm of their difference; L_p norm,

$$d_{minkowski} = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}}$$

- Manhattan distance: L_1 norm

$$d_{manhattan} = \sum_{i=1}^{n} |(x_i - y_i)|$$

- Euclidean distance: L_2 norm

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

6/43

Additional Distance Functions

- Cosine Similarity

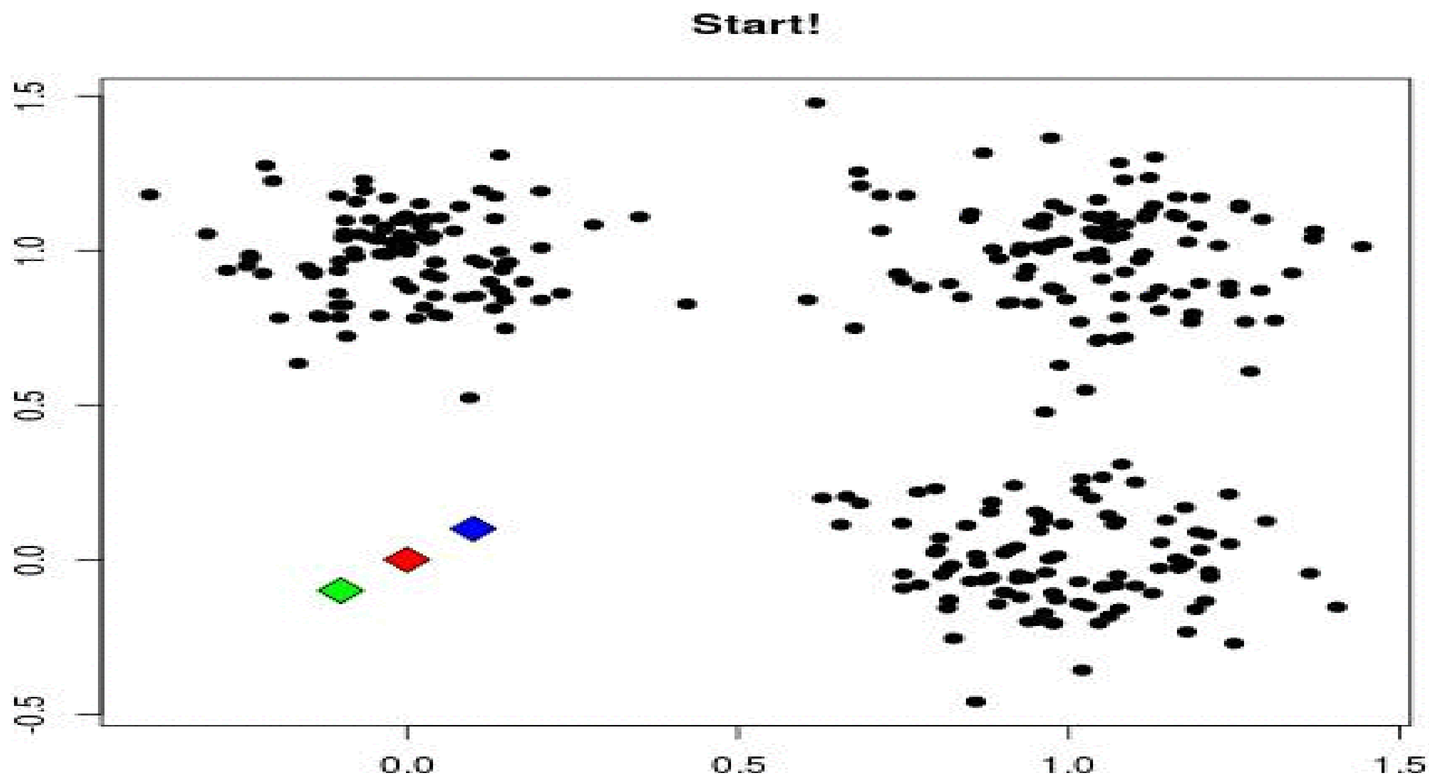
$$cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

  - Magnitude of vectors does not matter but the orientation
  - Used often in measuring similarities between different documents, such as text mining and NLP

Model Preprocessing

- Transformation of categorical attributes into numerical attributes (opposite to association rule mining)
- Data standardization, normalization, rescale
    - Applies to all algorithms based on distance: KNN, SVM, etc.
- Remove or impute missing values
- Detection and removal of noisy data and

Animation of How kmeans Algorithm works



Summary of kmeans

- Kmeans
  - Randomly assign k centroids
  - Assign all data points to their closest centroids
  - Update centroid assignments
  - Repeat the previous two steps until centroids are stable

Cluster Validity Evaluation

- SSE (Sum of Square Error): most common measure

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

  - Cluster cohesion
  - SSE is a monotonically decreasing function of number of clusters. Therefore it can be used to compare cluster performance only for similar number of clusters

11/43

Objective Function for Kmeans Clustering: SSE

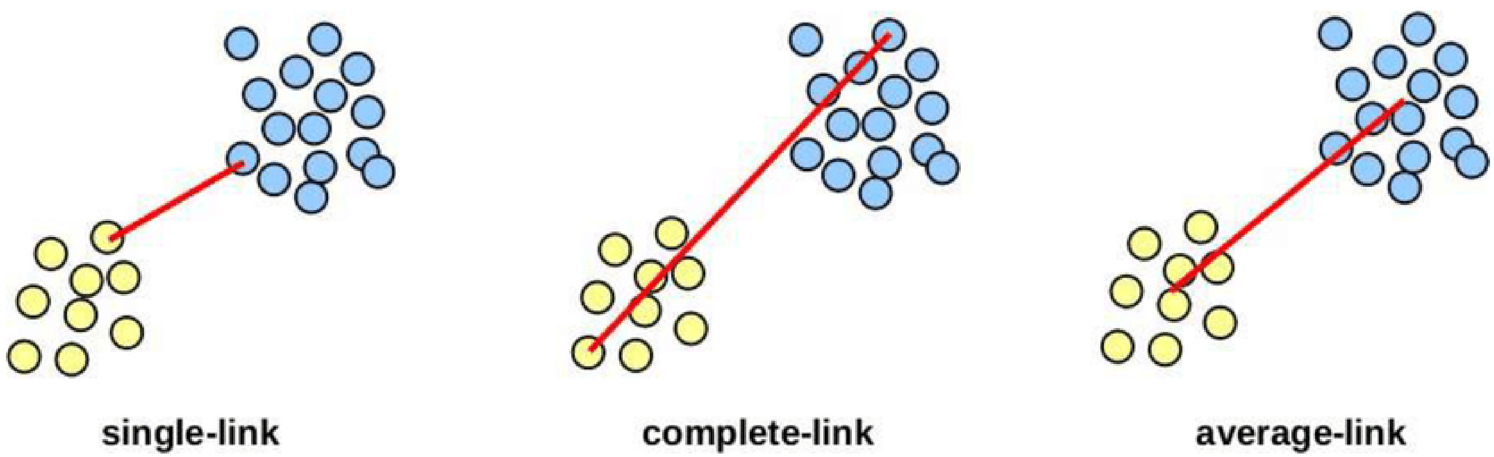
- Also known as loss/cost function
- Goal of Kmeans method is to
  - Minimize SSE of within-cluster variance
  - Same as the sum of Euclidean distance between all data points with their respective cluster centroids
- Therefore no needs to set the distance function for Kmeans method

Hierarchical Agglomerative Clustering

- Treat each data point as a cluster
- Compute the pairwise distance matrix for all clusters
- Merge closer clusters into a larger one and update the distance matrix
- Keep repeating the previous step until there is only one cluster left
- Opposite: HDC (Hierarchical Divisive Clustering)

Defintion of Inter-Cluster Distance

- Single linkage: minimal pairwise distance between data points belonging to different clusters
- Complete linkage: maximal pairwise distance between data points belonging to different clusters
- Average linkage method: average pairwise distance between all data points belonging to different clusters
- Centroid linkage method: pairwise distance between centroids of different clusters



Model Parameters

- Algorithm-specific
  - kmeans
    - Number of clusters: Elbow method to estimate
    - Initial choice of cluster centers (centroid)
      - Kmeans method is guaranteed to converge but not guaranteed to converge to global optimial
    - Maximal number of repeats
  - Hierarchical clustering
    - Distance function between data points
    - Definition of intercluster distance: single vs. complete vs. average vs. centroid linkage
    - Number of clusters to output (needed after clustering)

Comparison Between Kmeans and HAC (1)

- Objective function
  - Kmeans: sum of squared difference between data points and their respective centroid (within SS)
  - HAC: no objective function, greedy algorithm
- Parameter setting
  - Kmeans: need to specify number of clusters prior to running aglorithm
  - HAC: no need to choose number of clusters a priori; can choose after fact
- Performance
  - Kmeans: Fast with linear time complexity $O(N)$ (N: number of data points)
  - HAC: Slow with quadratic complexity $O(N^2)$
  - Hybrid approach: run Kmeans first to reduce dataset size and then HAC to cluster

Comparison Between Kmeans and HAC (2)

- Structure of clusters
  - Kmeans: works best with sphere clusters with similar size
  - HAC: works with clusters of different size and shape
    - Depending on inter-cluster distance definition
    - Single-linkage method: clusters of different size; prone to outliers
    - Complete-linkage method: clusters of similar size; prone to outliers
    - Average/Centroid linkage: resist outliers
  - Both methods couldn’t deal well with clusters of different densities: DBScan
- Both methods mostly work with numerical attributes only (contrary to association rule mining)

Principal Component Analysis (PCA): Cluster Visualization

- Dimension reduction: only pick those first few columns to reduce number of attributes (unrelated to each out)
- High dimensional data isualize: plot inlower dimensional space defined by principal components (PC1 and PC2)
- Benefits
  - Breakdown multicolinearity among original attributes and the new reconstructed dataset (with PCs as columns) have all the columns un-correlated
    - Loading factors gives us direction upon which the projected data points vary the most

Demo Dataset: USArrests

- Crime dataset:
  - Arrests per 100,000 residents for assault, murder, and rape in each of the 50 US states in 1973
  - Percent of the population living in urban areas

```
library(datasets)
str(USArrests)
```

```
## '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 25.8 ...
```

```
row.names(USArrests)
```

```
##  [1] "Alabama"        "Alaska"         "Arizona"        "Arkansas"
##  [5] "California"     "Colorado"       "Connecticut"    "Delaware"
##  [9] "Florida"        "Georgia"        "Hawaii"         "Idaho"
## [13] "Illinois"       "Indiana"        "Iowa"           "Kansas"
```

Data Preprocess

```
sum(!complete.cases(USArrests))
```

```
## [1] 0
```

```
summary(USArrests)
```

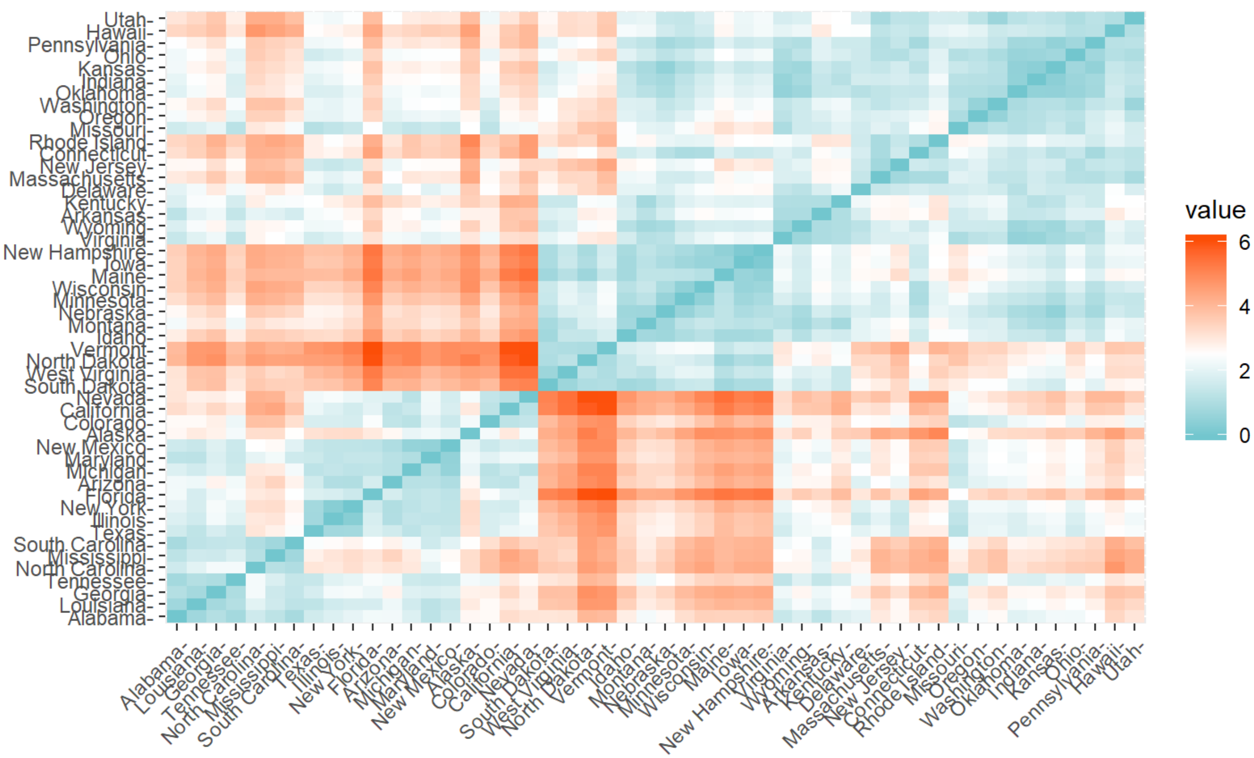
```
##      Murder          Assault         UrbanPop          Rape      
##  Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   : 7.30  
##  1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st Qu.:15.07  
##  Median : 7.250   Median :159.0   Median :66.00   Median :20.10  
##  Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean   :21.23  
##  3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd Qu.:26.18  
##  Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.   :46.00
```

```
df <- na.omit(USArrests)
df <- scale(df, center = T, scale = T)
summary(df)
```

```
##      Murder            Assault            UrbanPop               Rape        
##  Min.   :-1.6044   Min.   :-1.5090   Min.   :-2.31714   Min.   :-1.4874  
##  1st Qu.:-0.8525   1st Qu.:-0.7411   1st Qu.:-0.76271   1st Qu.:-0.6574
```

Distance function and visualization

```
library(factoextra)
distance <- get_dist(df, method = "euclidean")
fviz_dist(distance, gradient = list(low = "#00AFBB", mid = "white", high = "#FC4E07"))
```



kmeans Function

```
km_output <- kmeans(df, centers = 2, nstart = 25, iter.max = 100, algorithm = "Hartigan-Wong")
str(km_output)
```

```
## List of 9
##  $ cluster     : Named int [1:50] 1 1 1 2 1 1 2 2 1 1 ...
##   ..- attr(*, "names")= chr [1:50] "Alabama" "Alaska" "Arizona" "Arkansas" ...
##  $ centers     : num [1:2, 1:4] 1.005 -0.67 1.014 -0.676 0.198 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : chr [1:4] "Murder" "Assault" "UrbanPop" "Rape"
##  $ totss       : num 196
##  $ withinss    : num [1:2] 46.7 56.1
##  $ tot.withinss: num 103
##  $ betweenss   : num 93.1
##  $ size        : int [1:2] 20 30
##  $ iter        : int 1
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

```
names(km_output)
```

22/43

Loss Function: Sum of Square Error

```
km_output$totss
```

```
## [1] 196
```

```
km_output$withinss
```

```
## [1] 46.74796 56.11445
```

```
km_output$betweenss
```

```
## [1] 93.1376
```

```
sum(c(km_output$withinss, km_output$betweenss))
```

```
## [1] 196
```

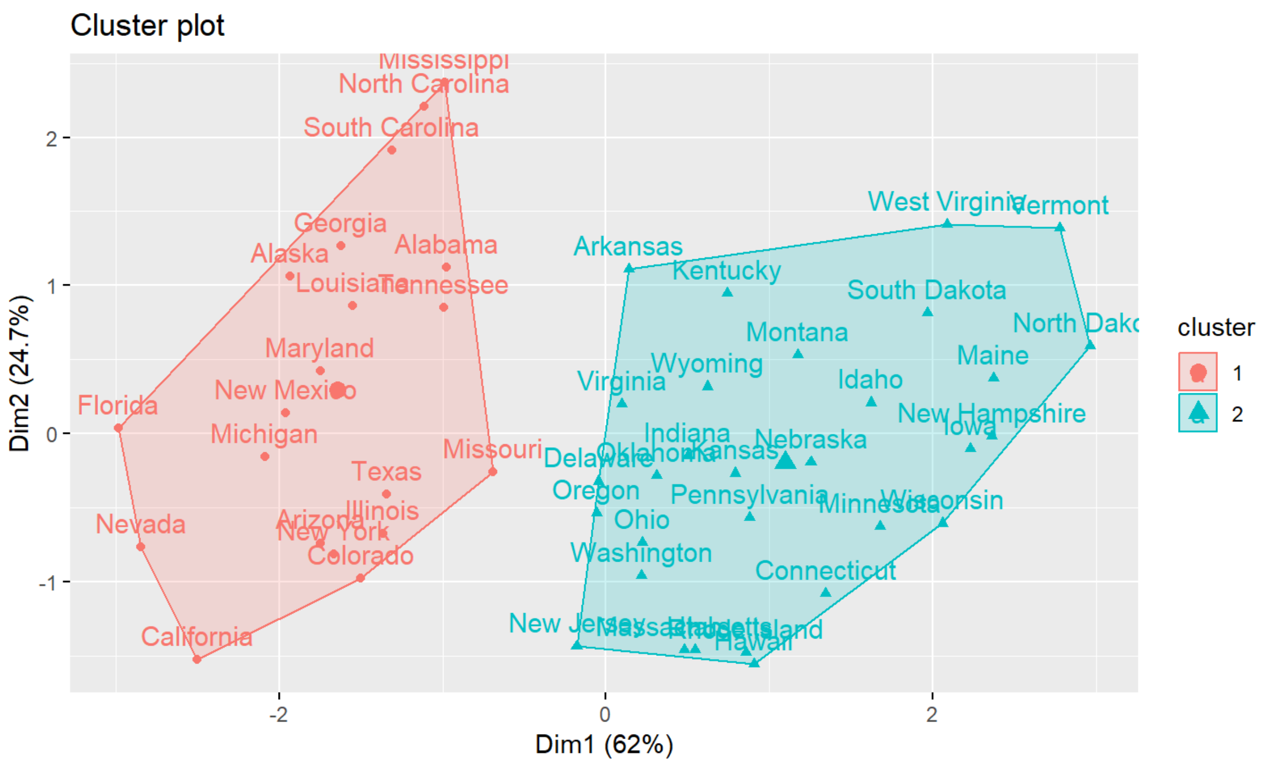
```
cohesion <- sum(km_output$withinss)/km_output$totss
cohesion
```



```
## [1] 0.5248082
```

Visualize Cluster Assignment

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



PCA Analysis

```
pca <- prcomp(USArrests, scale. = T, center = T)
pca$sdev
```

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

```
pca$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
```

```
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4
## Standard deviation     1.5749 0.9949 0.59713 0.41645
## Proportion of Variance 0.6201 0.2474 0.08914 0.04336
## Cumulative Proportion  0.6201 0.8675 0.95664 1.00000
```

PCA Analysis (2)

```
pcs <- as.data.frame(predict(pca, newdata = USArrests))
head(pcs, 5)
```

```
##                   PC1        PC2         PC3        PC4
## Alabama    -0.9756604  1.1220012 -0.43980366  0.1546966
## Alaska     -1.9305379  1.0624269  2.01950027 -0.4341755
## Arizona    -1.7454429 -0.7384595  0.05423025 -0.8262642
## Arkansas    0.1399989  1.1085423  0.11342217 -0.1809736
## California -2.4986128 -1.5274267  0.59254100 -0.3385592
```

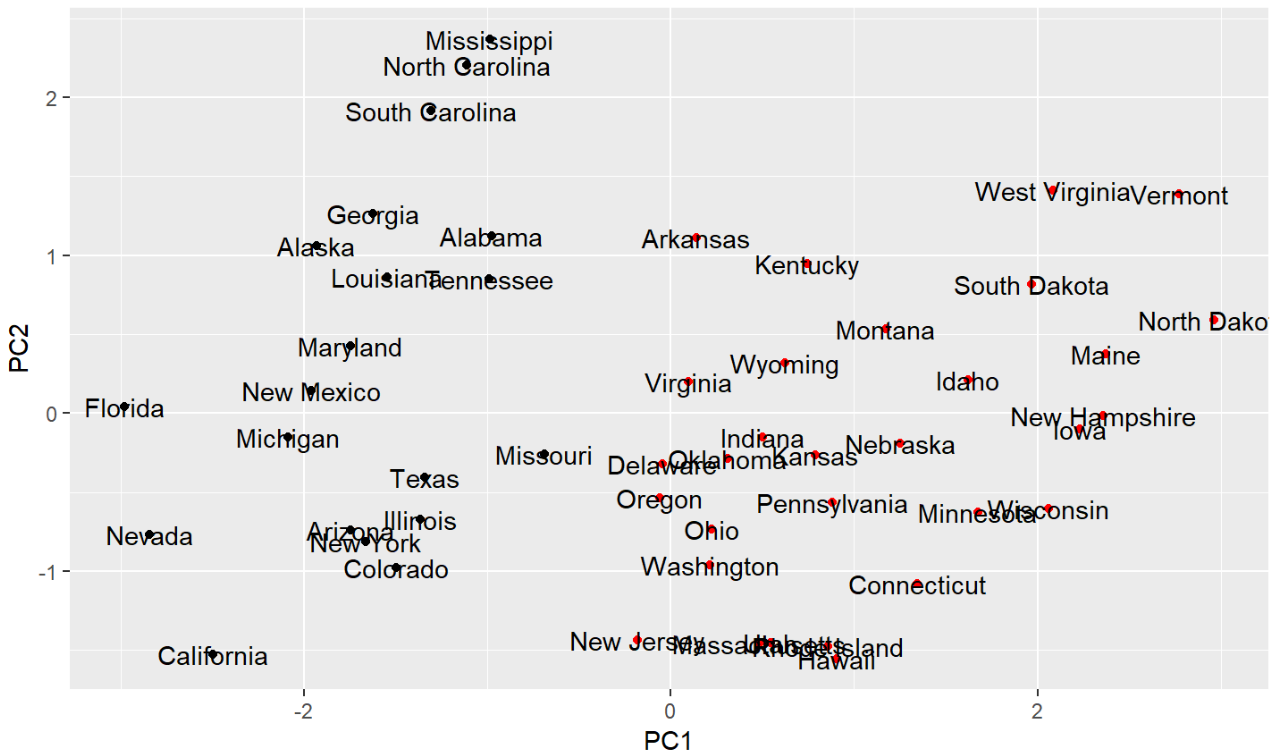
```
cov(pcs) ## covariance is zero
```

```
##               PC1           PC2           PC3           PC4
## PC1  2.480242e+00 -3.812359e-16  1.126674e-16  1.778258e-17
## PC2 -3.812359e-16  9.897652e-01 -2.024956e-16  9.907132e-17
## PC3  1.126674e-16 -2.024956e-16  3.565632e-01 -1.564569e-16
## PC4  1.778258e-17  9.907132e-17 -1.564569e-16  1.734301e-01
```

26/43

Recreate Cluster Assignment with PCA

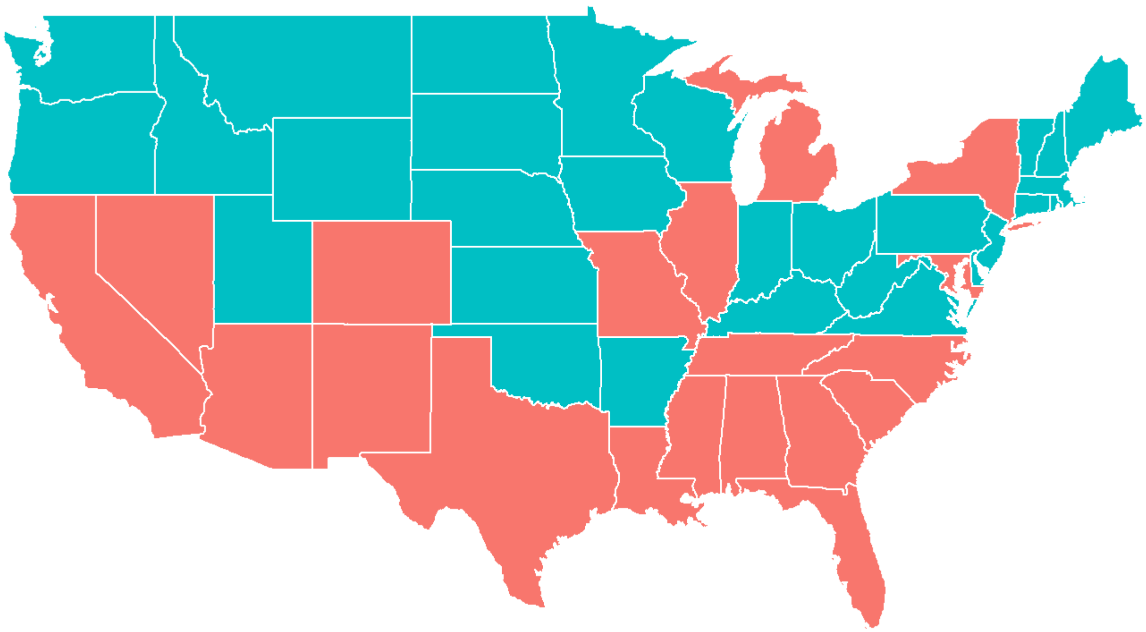
```
cluster <- km_output$cluster
pcs$cluster <- cluster[match(rownames(pcs), names(cluster))]
library(ggplot2)
ggplot(pcs, aes(x = PC1, y = PC2)) +
  geom_point(col=cluster) +
  geom_text(label=rownames(pcs))
```



Visual Cluster Assignment on Map (1)

```
cluster_df <- data.frame(state = tolower(row.names(USArrests)),
                         cluster = unname(km_output$cluster))
library(maps)
states <- map_data("state")
states %>%
  left_join(cluster_df, by = c("region" = "state")) %>%
  ggplot() +
  geom_polygon(aes(x = long, y = lat, fill = as.factor(cluster), group = group),
               color = "white") +
  coord_fixed(1.3) +
  guides(fill = F) +
  theme_bw() +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.border = element_blank(),
        axis.line = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank(),
        axis.title = element_blank())
```

Visual Cluster Assignment on Map (2)



29/43

Elbow method to decide Optimal Number of Clusters (1)

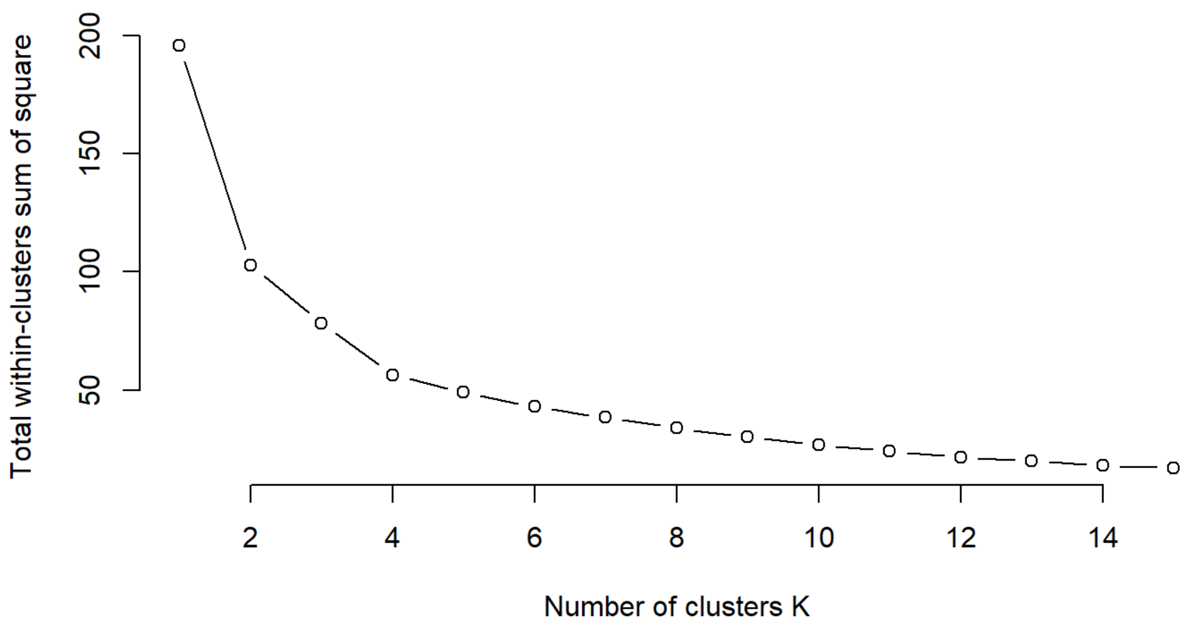
```
set.seed(8)
wss <- function(k){
  return(kmeans(df, k, nstart = 25)$tot.withinss)
}

k_values <- 1:15

wss_values <- purrr::map_dbl(k_values, wss)

plot(x = k_values, y = wss_values,
     type = "b", frame = F,
     xlab = "Number of clusters K",
     ylab = "Total within-clusters sum of square")
```

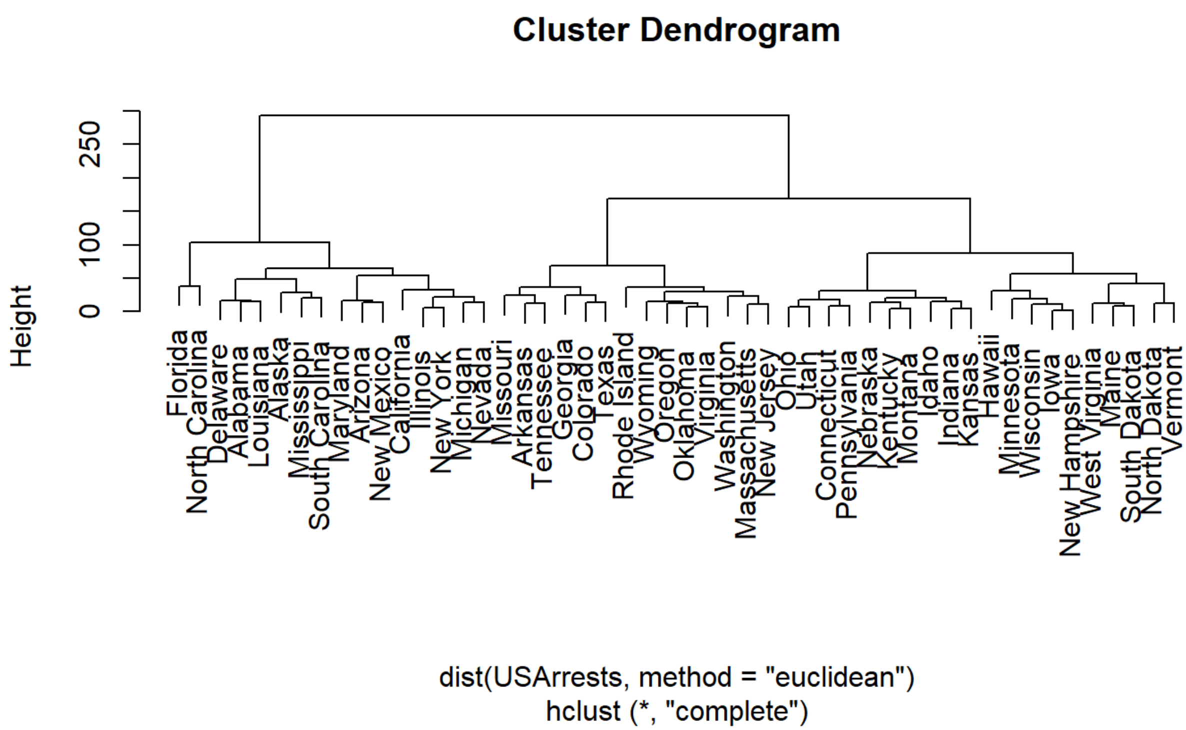
Elbow method to decide Optimal Number of Clusters (2)



31/43

Hierarchical Clustering

```
hac_output <- hclust(dist(USArrests, method = "euclidean"), method = "complete")
plot(hac_output)
```



32/43

Ouput Desirable Number of Clusters After Modeling

```
hac_cut <- cutree(hac_output, 2)

for (i in 1:length(hac_cut)){
  if(hac_cut[i] != km_output$cluster[i]) print(names(hac_cut)[i])
}
```

```
## [1] "Colorado"
## [1] "Delaware"
## [1] "Georgia"
## [1] "Missouri"
## [1] "Tennessee"
## [1] "Texas"
```

Cluster Analysis in Python

Import Libraries for Analysis

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
import seaborn as sns
from scipy.spatial import distance_matrix
from sklearn.cluster import KMeans
```

Data Preprocessing

```
USArrest = r.USArrests
USArrest.isnull().sum(axis=0)
```

```
## Murder      0
## Assault     0
## UrbanPop    0
## Rape        0
## dtype: int64
```

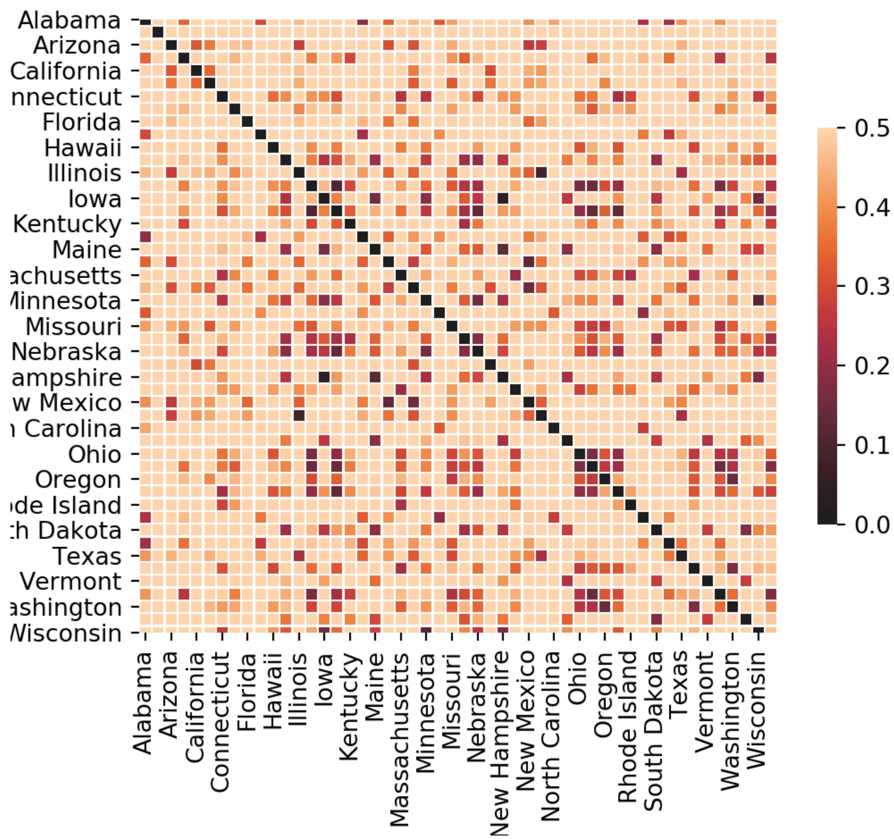
```
min_max_scaler = preprocessing.MinMaxScaler()
np_scaled = min_max_scaler.fit_transform(USArrest)
df_normalized = pd.DataFrame(np_scaled, columns=USArrest.columns, index=USArrest.index)
df_normalized.describe()
```

```
##           Murder   Assault   UrbanPop      Rape
## count  50.000000 50.000000  50.000000 50.000000
## mean    0.420964  0.430685   0.568475  0.360000
## std     0.262380  0.285403   0.245335  0.242025
## min     0.000000  0.000000   0.000000  0.000000
## 25%     0.197289  0.219178   0.381356  0.200904
## 50%     0.388554  0.390411   0.576271  0.330749
```

Visualize the Data (Codes)

```
dist_matrix = pd.DataFrame(distance_matrix(df_normalized.values,
                           df_normalized.values, p=2),
                           index=df_normalized.index,
                           columns=df_normalized.index)
f, ax = plt.subplots(figsize=(11, 9))
sns.heatmap(dist_matrix, vmax=.5, center=0, square=True,
            linewidths=.5, cbar_kws={"shrink": .5})
```

Visualize the Data (Output)



38/43

KMeans Clustering using

```
kmeans = KMeans(n_clusters=2, n_init=25, max_iter=100, random_state=6)
kmeans.fit(df_normalized)
```

```
## KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=100,
##        n_clusters=2, n_init=25, n_jobs=None, precompute_distances='auto',
##        random_state=6, tol=0.0001, verbose=0)
```

```
kmeans.__dict__.keys()
```

```
## dict_keys(['n_clusters', 'init', 'max_iter', 'tol', 'precompute_distances', 'n_init', 'verbo
```

```
kmeans.labels_
```

```
## array([1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1,
##        0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0,
##        0, 0, 0, 0, 0, 0])
```

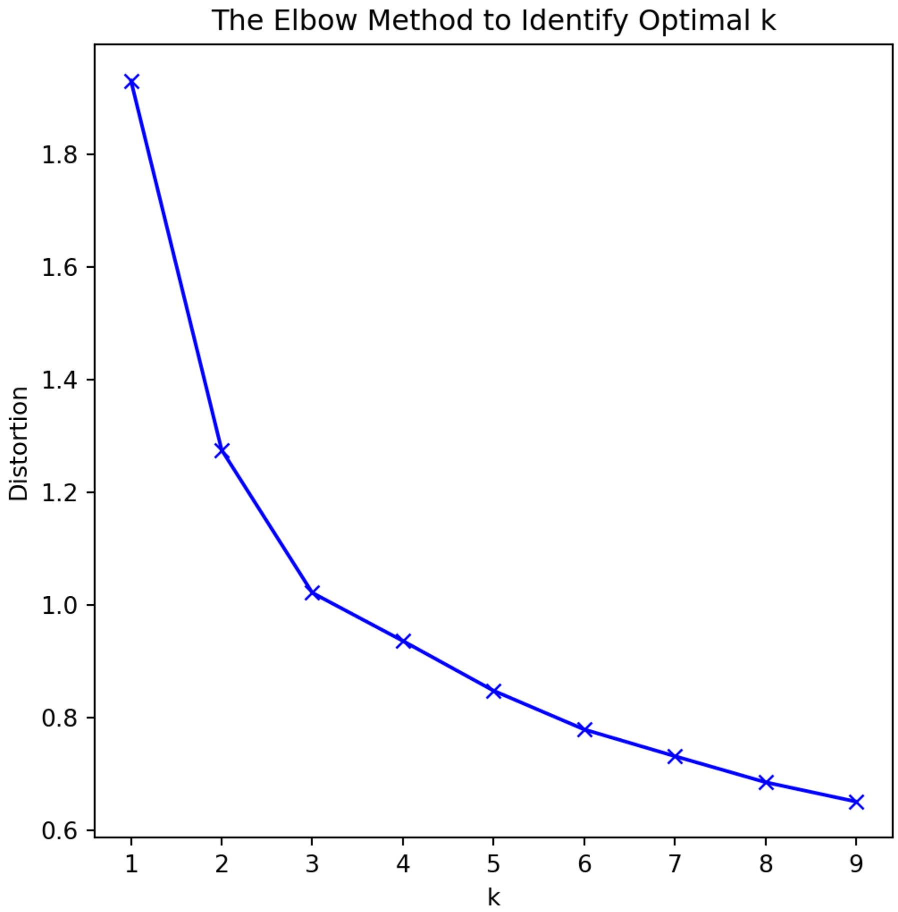
39/43

Elbow Method (Codes)

```
from scipy.spatial.distance import cdist
distortions = []
K = range(1, 10)
for k in K:
    kmeanModel = KMeans(n_clusters=k).fit(dist_matrix)
    kmeanModel.fit(dist_matrix)
    distortions.append(sum(np.min(cdist(dist_matrix, kmeanModel.cluster_centers_, 'euclidean'),

plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method to Identify Optimal k')
plt.show()
```

Elbow Method (Output)



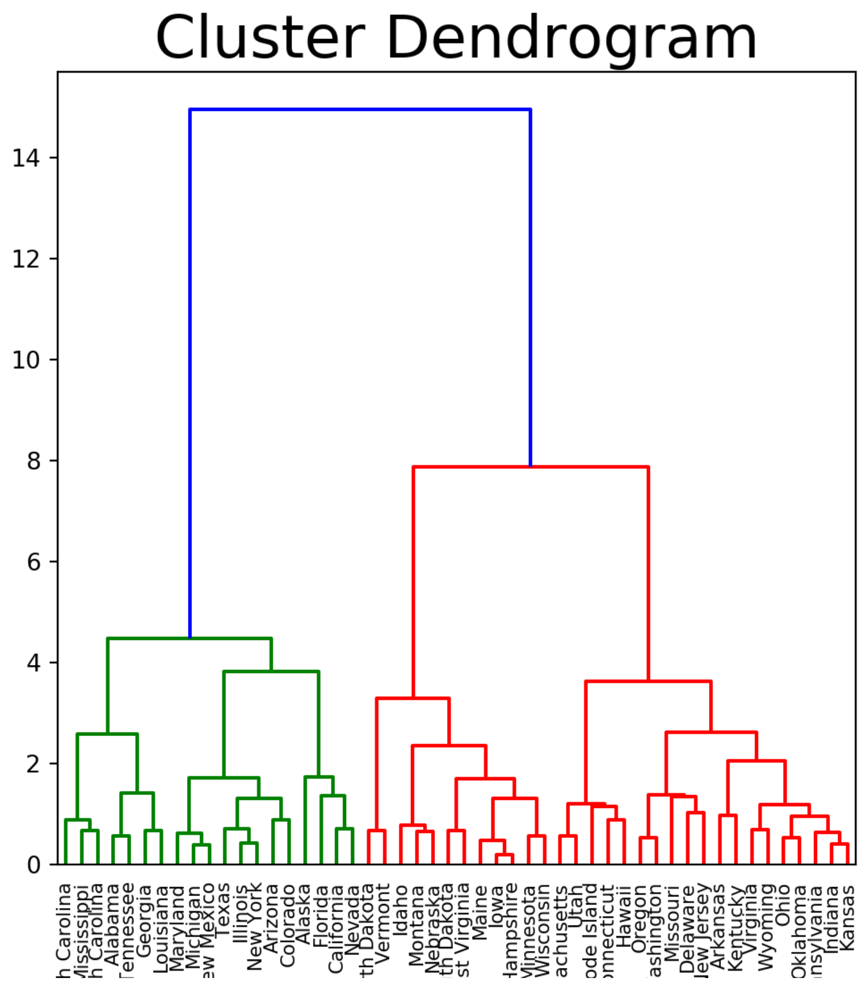
41/43

Create Dendrogram (Codes)

```
from scipy.cluster.hierarchy import ward, dendrogram

linkage_matrix = ward(dist_matrix)

fig, ax = plt.subplots(figsize=(20, 10))
ax.grid(False)
ax.set_title('Cluster Dendrogram', fontsize = 25)
ax = dendrogram(linkage_matrix, orientation='top', labels=dist_matrix.index)
plt.xticks(fontsize=15)
plt.show()
```


Cluster Dendrogram
14
12
10
8
6
4
2
0
h Carolina
Mississippi
h Carolina
Alabama
Tennessee
Georgia
Louisiana
Maryland
Michigan
ew Mexico
Texas
Illinois
New York
Arizona
Colorado
Alaska
Florida
California
Nevada
th Dakota
Vermont
Idaho
Montana
Nebraska
th Dakota
st Virginia
Maine
Iowa
Hampshire
Minnesota
Wisconsin
achusetts
Utah
ode Island
nnecticut
Hawaii
Oregon
ashington
Missouri
Delaware
New Jersey
Arkansas
Kentucky
Virginia
Wyoming
Ohio
Oklahoma
nsylvania
Indiana
Kansas

43/43