P8106 Data ScienceII HW5

Yueran Zhang

2023-04-29

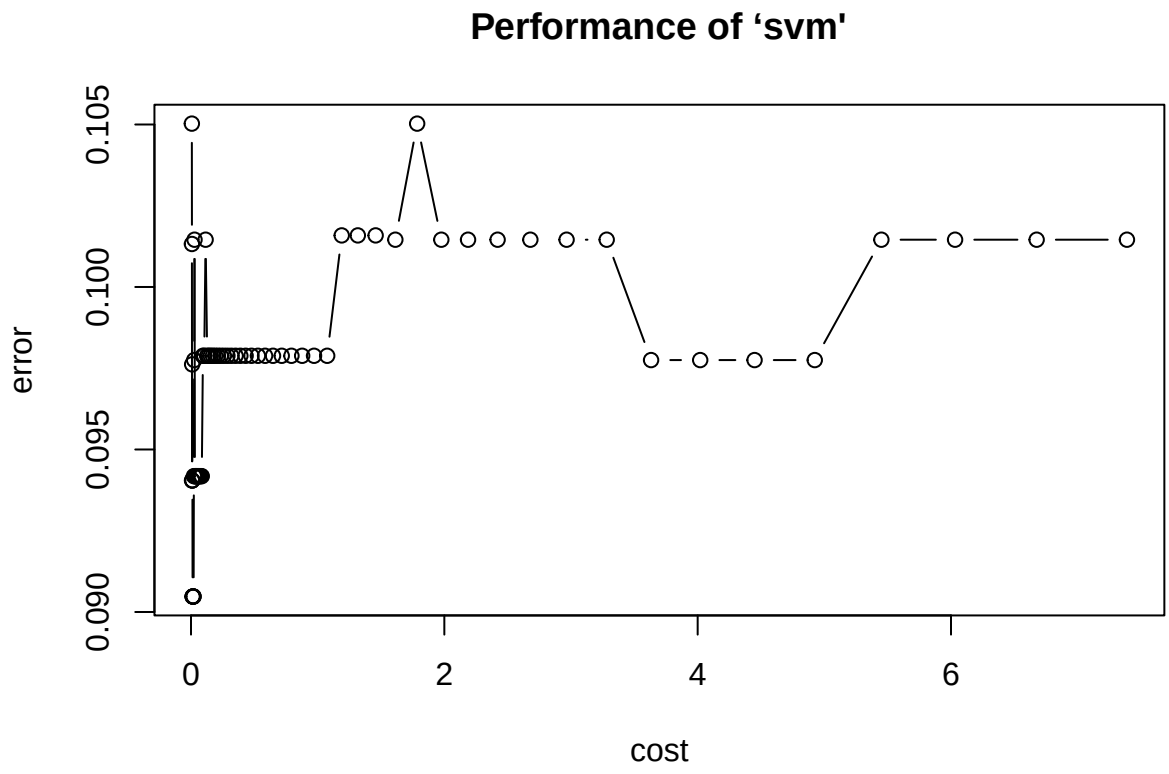
```
library(tidyverse)
library(mlbench)
library(ISLR)
library(caret)
library(e1071)
library(kernlab)
library(factoextra)
library(gridExtra)
library(RColorBrewer)
library(jpeg)
library(knitr)
```

1. In this problem, we will apply support vector machines to predict whether a given car gets high or low gas mileage based on the dataset "auto.csv" (used in Homework 3; see Homework 3 for more details of the dataset). The response variable is mpg cat. The predictors are cylinders, displacement, horsepower, weight, acceleration, year, and origin. Split the dataset into two parts: training data (70%) and test data (30%).

```
# Data Import + Processing

set.seed(123)

auto.data =
  read.csv("./DataSet/auto.csv") %>%
  na.omit() %>%
  mutate(mpg_cat = factor((mpg_cat), levels = c("low", "high")))


RowTrain <- createDataPartition(y = auto.data$mpg_cat,
                                p = 0.7,
                                list = FALSE) # split the dataset into two parts: training data (70%) an
```

Question A - Fit a support vector classifier (linear kernel) to the training data. What are the training and test error rates?

```
set.seed(123)

linear.tune <- tune.svm(mpg_cat ~ . ,
                        data = auto.data[RowTrain,],
                        kernel = "linear",
                        cost = exp(seq(-5,2,len=70)),
                        scale = TRUE)
plot(linear.tune)
```




```
# summary(linear.tune)
linear.tune$best.parameters
```

```
##         cost
## 7 0.01238456
```

```
best.linear <- linear.tune$best.model
summary(best.linear)
```

```
##
## Call:
## best.svm(x = mpg_cat ~ ., data = auto.data[RowTrain, ], cost = exp(seq(-5,
##     2, len = 70)), kernel = "linear", scale = TRUE)
##
```

```
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01238456
##
## Number of Support Vectors:  125
##
##  ( 62 63 )
##
##
## Number of Classes:  2
##
## Levels:
##  low high
```

```
########################
# Training error rates
########################
confusionMatrix(data = best.linear$fitted,
                reference = auto.data$mpg_cat[RowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low  120    7
##       high  18  131
##
##                Accuracy : 0.9094
##                  95% CI : (0.8692, 0.9405)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8188
##
##  Mcnemar's Test P-Value : 0.0455
##
##             Sensitivity : 0.8696
##             Specificity : 0.9493
##          Pos Pred Value : 0.9449
##          Neg Pred Value : 0.8792
##              Prevalence : 0.5000
##          Detection Rate : 0.4348
##    Detection Prevalence : 0.4601
##       Balanced Accuracy : 0.9094
##
##        'Positive' Class : low
##
```

```
########################
# Test error rates
########################
```

```
pred.linear <- predict(best.linear, newdata = auto.data[-RowTrain,])

confusionMatrix(data = pred.linear,
                reference = auto.data$mpg_cat[-RowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   50    3
##       high   8   55
##
##                Accuracy : 0.9052
##                  95% CI : (0.8367, 0.9517)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8103
##
##  Mcnemar's Test P-Value : 0.2278
##
##             Sensitivity : 0.8621
##             Specificity : 0.9483
##          Pos Pred Value : 0.9434
##          Neg Pred Value : 0.8730
##              Prevalence : 0.5000
##          Detection Rate : 0.4310
##    Detection Prevalence : 0.4569
##       Balanced Accuracy : 0.9052
##
##        'Positive' Class : low
##
```

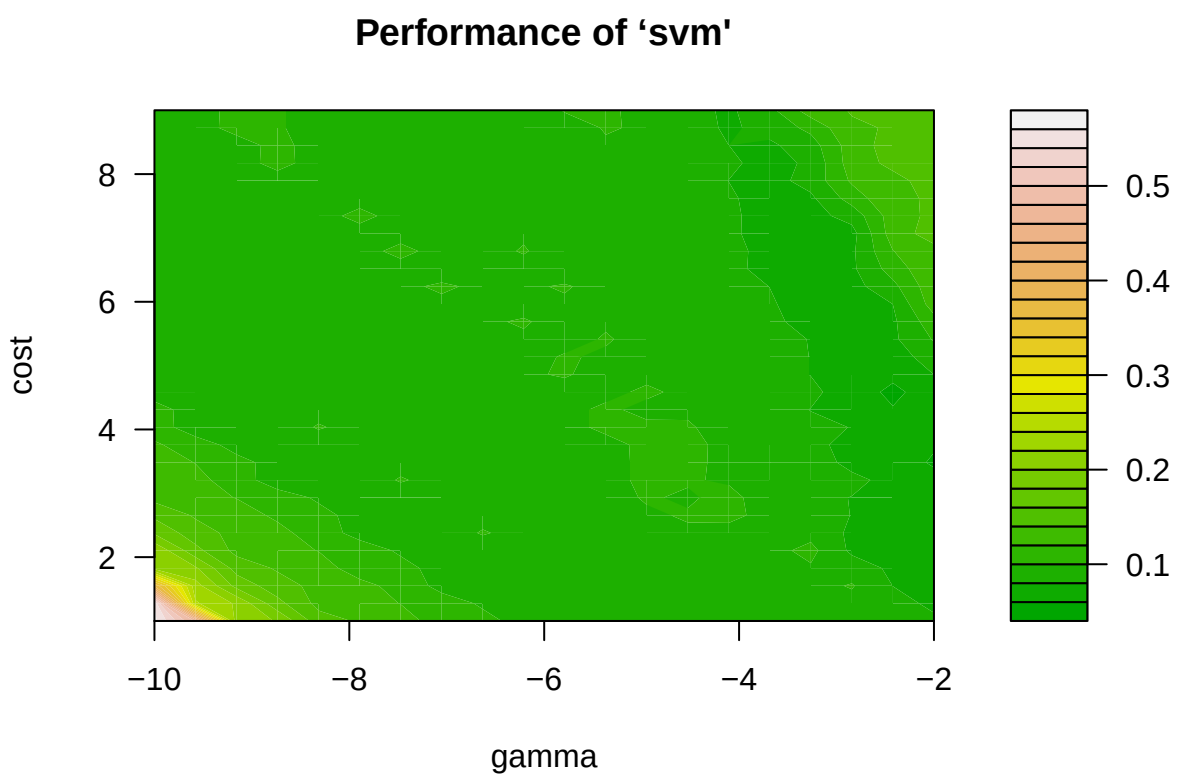
From above output when applying a support vector classifier to the training data,

- For the training data, the accuracy of the fitted support vector classifier reads as `0.9094(90.94%)`, for the given data and observations. If a model will perform at 92.03% accuracy then the error rate will be `1-0.9094 = 9.06%`.
- For the testing data, the accuracy reads as `0.9052(90.52%)`, so the the error rate will be `1-0.9052 = 9.48%`.

Question B - Fit a support vector machine with a radial kernel to the training data. What are the training and test error rates?

```
set.seed(123)

radial.tune <- tune.svm(mpg_cat ~ . ,
                        data = auto.data[RowTrain,],
                        kernel = "radial",
                        cost = exp(seq(1,9,len=30)),
```

```
                          gamma = exp(seq(-10,-2,len=20)))
plot(radial.tune, transform.y = log, transform.x = log,
     color.palette = terrain.colors)
```




```
# summary(radial.tune)
best.radial <- radial.tune$best.model
summary(best.radial)
```

```
##
## Call:
## best.svm(x = mpg_cat ~ ., data = auto.data[RowTrain, ], gamma = exp(seq(-10,
##     -2, len = 20)), cost = exp(seq(1, 9, len = 30)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  98.12154
##
## Number of Support Vectors:  55
##
##  ( 31 24 )
##
##
```

```
## Number of Classes:  2
##
## Levels:
##  low high
```

```
########################
# Training error rates
########################
confusionMatrix(data = best.radial$fitted,
                reference = auto.data$mpg_cat[RowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low  133    4
##       high   5  134
##
##                Accuracy : 0.9674
##                  95% CI : (0.939, 0.985)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9348
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9638
##             Specificity : 0.9710
##          Pos Pred Value : 0.9708
##          Neg Pred Value : 0.9640
##              Prevalence : 0.5000
##          Detection Rate : 0.4819
##    Detection Prevalence : 0.4964
##       Balanced Accuracy : 0.9674
##
##        'Positive' Class : low
##
```

```
########################
# Test error rates
########################
pred.radial <- predict(best.radial, newdata = auto.data[-RowTrain,])

confusionMatrix(data = pred.radial,
                reference = auto.data$mpg_cat[-RowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   54    5
##       high   4   53
```

```
##
##                Accuracy : 0.9224
##                  95% CI : (0.8578, 0.9639)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8448
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9310
##             Specificity : 0.9138
##          Pos Pred Value : 0.9153
##          Neg Pred Value : 0.9298
##              Prevalence : 0.5000
##          Detection Rate : 0.4655
##    Detection Prevalence : 0.5086
##       Balanced Accuracy : 0.9224
##
##        'Positive' Class : low
##
```

From above output when fitting a support vector machine with a radial kernel to the training data,

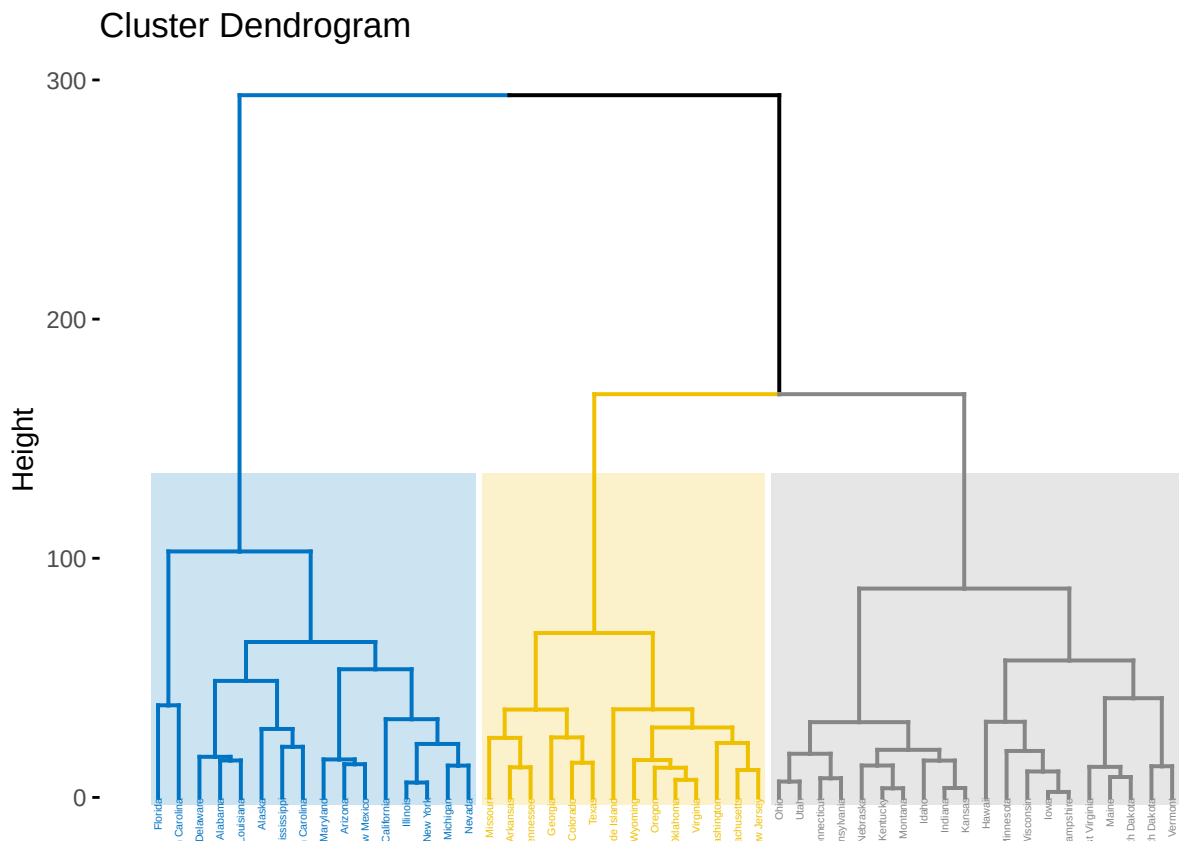
- For the training data, the accuracy of the fitted support vector classifier reads as `0.9674(96.74%)`, for the given data and observations. If a model will perform at 96.74% accuracy then the error rate will be `1-0.9674 = 3.26%`.
- For the testing data, the accuracy reads as `0.9224(92.24%)`, so the the error rate will be `1-0.9224 = 7.76%`.

2. In this problem, we perform hierarchical clustering on the states using the USArrests data in the ISLR package. For each of the 50 states in the United States, the dataset contains the number of arrests per 100,000 residents for each of three crimes: Assault, Murder, and Rape. The dataset also contains the percent of the population in each state living in urban areas, UrbanPop. The four variables will be used as features for clustering.

```
# Data import
USArrests.dat <- USArrests %>%
    na.omit()
```

Question A - Using hierarchical clustering with complete linkage and Euclidean distance, cluster the states. Cut the dendrogram at a height that results in three distinct clusters. Which states belong to which clusters?

```
# Here we use the Euclidean distance and complete linkage.
hc.complete <- hclust(dist(USArrests.dat), method = "complete")

# Using function `fviz_dend()` to visualize the dendrogram
fviz_dend(hc.complete, k = 3, # cut into three distinct clusters
                       cex = 0.3,
                       palette = "jco",
                       color_labels_by_k = TRUE,
                       rect = TRUE, rect_fill = TRUE, rect_border = "jco",
                       labels_track_height = 2.5)
```



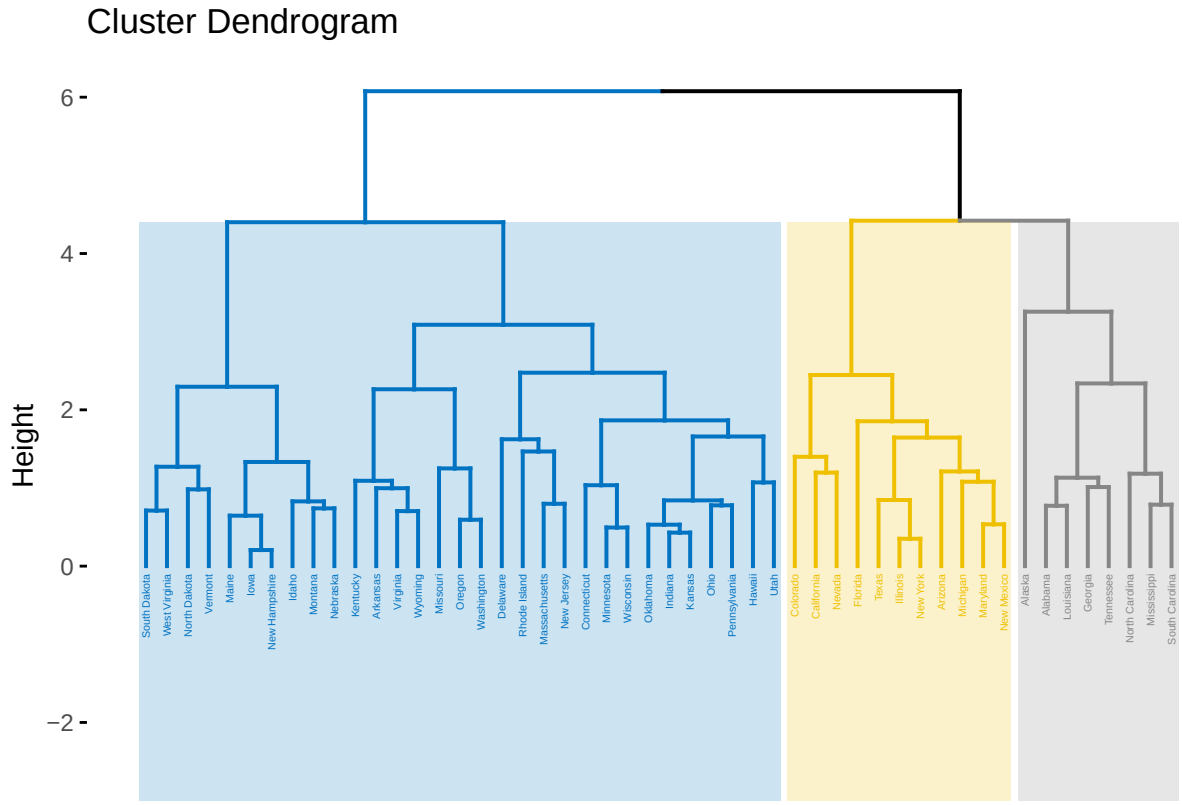

```
ind3.complete <- cutree(hc.complete, 3)

###################################
# The state in each cluster
###################################

col1 <- row.names(USArrests.dat[ind3.complete == 1,])
col2 <- row.names(USArrests.dat[ind3.complete == 2,])
col3 <- row.names(USArrests.dat[ind3.complete == 3,])

# Determine the length of the longest column
max_length <- max(length(col1), length(col2), length(col3))

Cluster1 <- c(col1, rep(" ", max_length - length(col1)))
Cluster2 <- c(col2, rep(" ", max_length - length(col2)))
Cluster3 <- c(col3, rep(" ", max_length - length(col3)))

# Combine the padded columns into a single table
cluster.table <- cbind(Cluster1, Cluster2,Cluster3)

# Print
knitr::kable(cluster.table, format = "simple",caption = "The state in each cluster")
```

Table 1: The state in each cluster

Cluster1	Cluster2	Cluster3
Alabama	Arkansas	Connecticut
Alaska	Colorado	Hawaii
Arizona	Georgia	Idaho
California	Massachusetts	Indiana
Delaware	Missouri	Iowa
Florida	New Jersey	Kansas
Illinois	Oklahoma	Kentucky
Louisiana	Oregon	Maine
Maryland	Rhode Island	Minnesota
Michigan	Tennessee	Montana
Mississippi	Texas	Nebraska
Nevada	Virginia	New Hampshire
New Mexico	Washington	North Dakota
New York	Wyoming	Ohio
North Carolina		Pennsylvania
South Carolina		South Dakota
		Utah
		Vermont
		West Virginia
		Wisconsin

Question B - Hierarchically cluster the states using complete linkage and Euclidean distance, after scaling the variables to have standard deviation one. Does scaling the variables change the clustering results? Why? In your opinion, should the variables be scaled before the inter-observation dissimilarities are computed?

```
#Standardized data
USArrests.sd <- scale(USArrests.dat)

#Fit the Euclidean distance and complete linkage
hc.complete.sd <- hclust(dist(USArrests.sd), method = "complete")

# Using function `fviz_dend()` to visualize the dendrogram
fviz_dend(hc.complete.sd, k = 3, # cut into three distinct clusters
                           cex = 0.3,
                           palette = "jco",
                           color_labels_by_k = TRUE,
                           rect = TRUE, rect_fill = TRUE, rect_border = "jco",

                                  labels_track_height = 2.5)
```

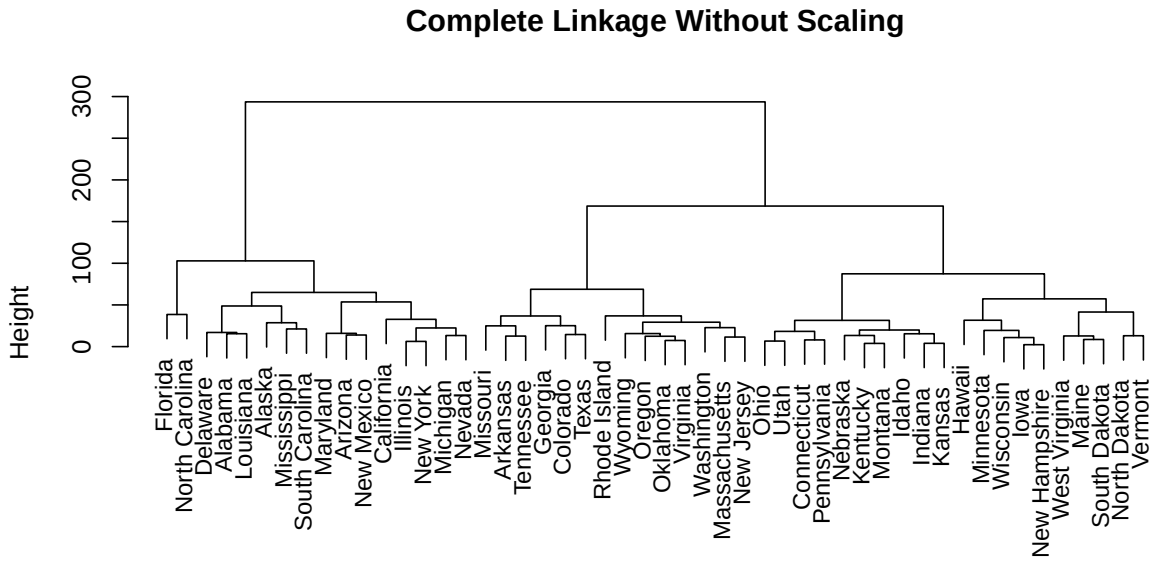



```r
ind3.complete.sd <- cutree(hc.complete.sd, 3)

##############################################
# The state in each cluster with with Scaled Variables
##############################################

col1.sd <- row.names(USArrests.sd[ind3.complete.sd  == 1,])
col2.sd <- row.names(USArrests.sd[ind3.complete.sd  == 2,])
col3.sd <- row.names(USArrests.sd[ind3.complete.sd  == 3,])

# Determine the length of the longest column
max_length <- max(length(col1.sd), length(col2.sd), length(col3.sd))

Cluster1 <- c(col1.sd, rep(" ", max_length - length(col1.sd)))
Cluster2 <- c(col2.sd, rep(" ", max_length - length(col2.sd)))
Cluster3 <- c(col3.sd, rep(" ", max_length - length(col3.sd)))

# Combine the padded columns into a single table
cluster.tibble <- cbind(Cluster1, Cluster2,Cluster3)

# Print
knitr::kable(cluster.tibble, format = "simple", caption = "The state in each cluster with Scaled Variabl
```

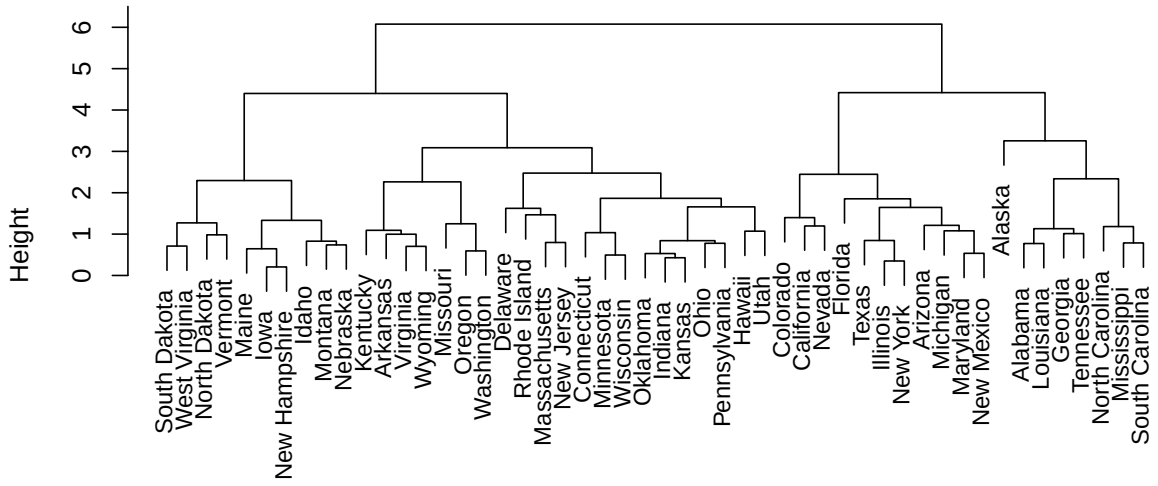
Table 2: The state in each cluster with Scaled Variables

Cluster1	Cluster2	Cluster3
Alabama	Arizona	Arkansas
Alaska	California	Connecticut
Georgia	Colorado	Delaware
Louisiana	Florida	Hawaii
Mississippi	Illinois	Idaho
North Carolina	Maryland	Indiana
South Carolina	Michigan	Iowa
Tennessee	Nevada	Kansas
	New Mexico	Kentucky
	New York	Maine
	Texas	Massachusetts
		Minnesota
		Missouri
		Montana
		Nebraska
		New Hampshire
		New Jersey
		North Dakota
		Ohio
		Oklahoma
		Oregon
		Pennsylvania
		Rhode Island
		South Dakota
		Utah
		Vermont
		Virginia
		Washington
		West Virginia
		Wisconsin
		Wyoming

```
plot(hc.complete,main="Complete Linkage Without Scaling", xlab ="", sub = "", cex = .9)
```




```
plot(hc.complete.sd, main="Complete Linkage with Scaled Variables", xlab = "", sub = "", cex = 0.9)
```



- Scaling the variables impacts the clusters that are obtained, the branch lengths, and the height of the tree. For example, without scaling, Ohio clusters with Utah while with scaling Ohio clusters nearby

Pennsylvanian. In addition, the height of the un-scaled tree is 300 while the height of the scaled tree is 6. Without scaling, we cut the tree at a height of ~150 whereas we cut the scaled tree at a height of ~4 to obtain 3 clusters. In addition, the branch for Alaska (and many other states) is shorter in the scaled tree.

- In this scenario, scaling is more appropriate because `Murder`, `Assault`, and `Rape` all have unites of per 100,000 people while `UrbanPop` is the percentage of the state population that lives in urban areas. Therefore, it is important to scale so that the units of `UrbanPop` has an equal contribution to the hierarchical clustering algorithm as the other variables.
- It is important to scale the variables before computing inter-observation dissimilarities in hierarchical clustering, especially when the variables are measured on different scales, which ensures that each variable has an equal contribution to the clustering process and can help to avoid bias in the clustering results.