p8106_hw5_qz2266

Qing Zhou

2023-05-01

Question 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". The response variable is mpg cat. The predictors are cylinders, displacement, horsepower, weight, acceleration, year, and origin.

```
# Data import and preparation
auto = read.csv("data/auto.csv") %>%
  mutate(
    mpg_cat = as.factor(mpg_cat),
    mpg_cat = fct_relevel(mpg_cat, c("low", "high")),
    year = factor(year),
    origin = as.factor(origin))
```

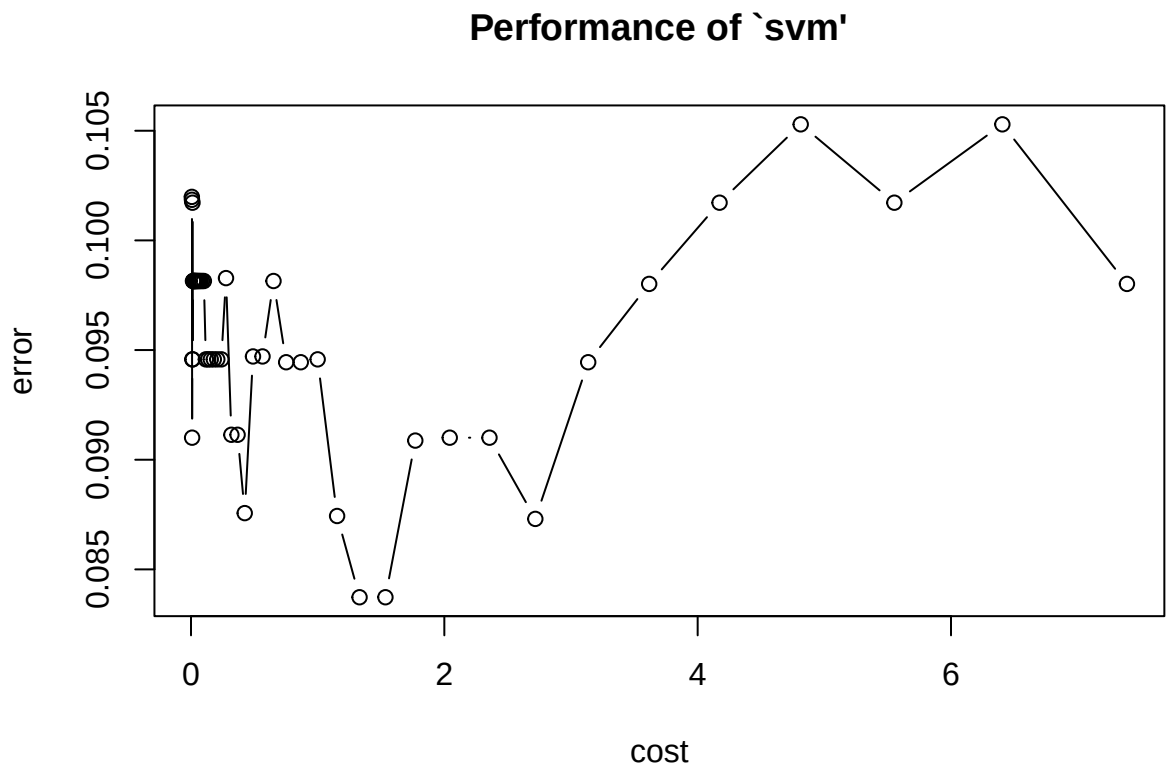
Split the dataset into two parts - training data (70%) and test data (30%).

```
set.seed(1)
# Data partition
trainRows <- createDataPartition(y = auto$mpg_cat, p = 0.7,list = FALSE)
auto_train = auto[trainRows, ]
auto_test = auto[-trainRows, ]
```

Here I mutated `year` as a factor variable, since I don't assume there's a linear relationship between model year and gas millage.

(a) Fit a support vector classifier (linear kernel) to the training data.

```
set.seed(1)
# Fit model
linear.tune <- tune.svm(mpg_cat ~ . ,
                        data = auto_train,
                        kernel = "linear",
                        cost = exp(seq(-5,2,len = 50)),
                        scale = TRUE)
plot(linear.tune)
```




```
# Optimal tuning parameters
linear.tune$best.parameters
```

```
##        cost
## 39 1.535063
```

```
# Extract final model and summarize
best.linear <- linear.tune$best.model
summary(best.linear)
```

```
##
## Call:
## best.svm(x = mpg_cat ~ ., data = auto_train, cost = exp(seq(-5, 2,
##     len = 50)), kernel = "linear", scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1.535063
##
## Number of Support Vectors:  62
##
##  ( 30 32 )
##
```

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

```
# Report training error rate
confusionMatrix(data = best.linear$fitted,
                reference = auto_train$mpg_cat)
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction low high
##       low  129    8
##       high   9  130
##
##                Accuracy : 0.9384
##                  95% CI : (0.9032, 0.9637)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8768
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9348
##             Specificity : 0.9420
##          Pos Pred Value : 0.9416
##          Neg Pred Value : 0.9353
##              Prevalence : 0.5000
##          Detection Rate : 0.4674
##    Detection Prevalence : 0.4964
##       Balanced Accuracy : 0.9384
##
##        'Positive' Class : low
##
```

```
# Report test error rate
pred.linear <- predict(best.linear, newdata = auto_test)
confusionMatrix(data = pred.linear,
                reference = auto_test$mpg_cat)
```

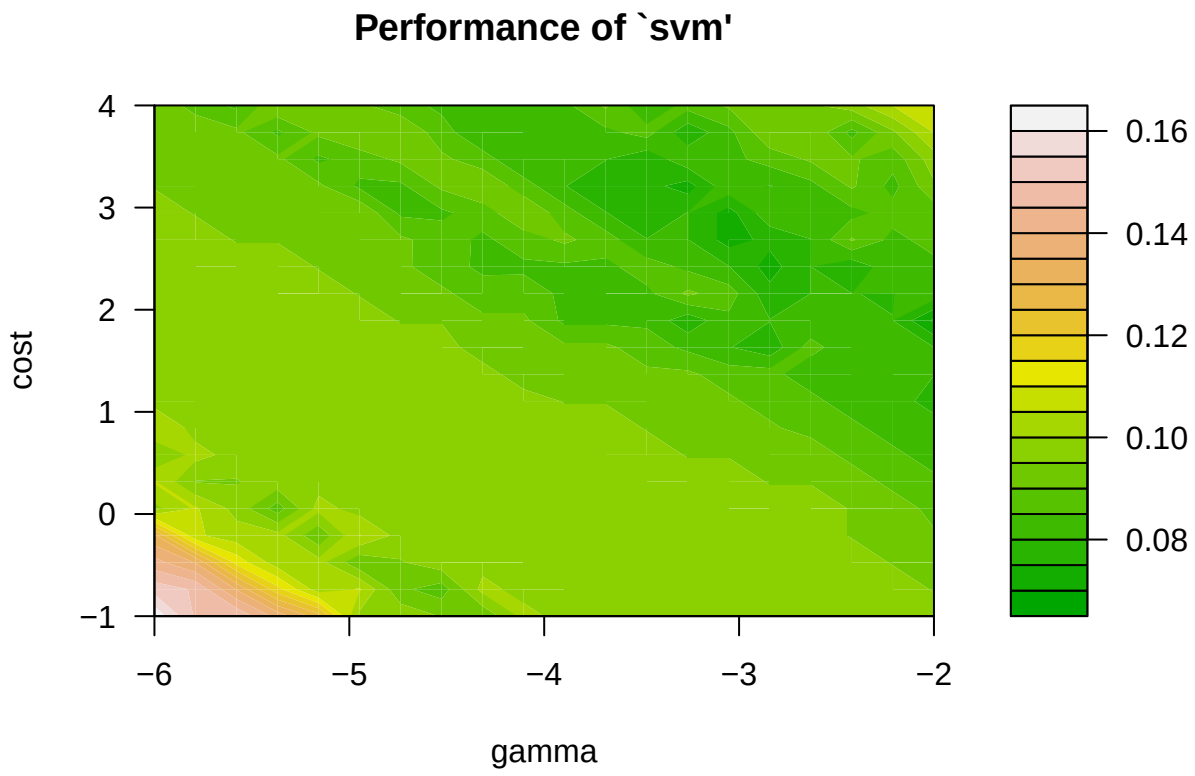
```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction low high
##       low   50    3
##       high   8   55
##
##                Accuracy : 0.9052
##                  95% CI : (0.8367, 0.9517)
##     No Information Rate : 0.5
```

```
##     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
##
```

- With cross-validation, we find the optimal tuning parameter is when cost is 1.535063, which minimizes the error. There are 62 support vectors in the optimal support vector classifier with a linear kernel.
- According to the confusion Matrix above, for the training data, the accuracy of the fitted support vector classifier is 0.9384, so the training error rate is (1-0.9384)x100% = 6.16% .
- According to the confusion Matrix above, the accuracy when applied the model to the test data is 0.9052, so the test error rate is (1-0.9052)x100% = 9.48% .

b) Fit a support vector machine with a radial kernel to the training data.

```
set.seed(1)
# Fit model
radial.tune <- tune.svm(mpg_cat ~ .,
                        data = auto_train,
                        kernel = "radial",
                        cost = exp(seq(-1,4,len = 20)),
                        gamma = exp(seq(-6,-2,len = 20)))
plot(radial.tune, transform.y = log, transform.x = log,
     color.palette = terrain.colors)
```




```
# Optimal tuning parameters
radial.tune$best.parameters
```

```
##         gamma     cost
## 240 0.1353353 6.650798
```

```
# Extract final model and summarize
best.radial <- radial.tune$best.model
summary(best.radial)
```

```
##
## Call:
## best.svm(x = mpg_cat ~ ., data = auto_train, gamma = exp(seq(-6,
##     -2, len = 20)), cost = exp(seq(-1, 4, len = 20)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  6.650798
##
## Number of Support Vectors:  72
##
##  ( 35 37 )
##
```

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

```
# Report training error rate
confusionMatrix(data = best.radial$fitted,
                reference = auto_train$mpg_cat)
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction low high
##       low  134    2
##       high   4  136
##
##                Accuracy : 0.9783
##                  95% CI : (0.9533, 0.992)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9565
##
##  Mcnemar's Test P-Value : 0.6831
##
##             Sensitivity : 0.9710
##             Specificity : 0.9855
##          Pos Pred Value : 0.9853
##          Neg Pred Value : 0.9714
##              Prevalence : 0.5000
##          Detection Rate : 0.4855
##    Detection Prevalence : 0.4928
##       Balanced Accuracy : 0.9783
##
##        'Positive' Class : low
##
```

```
# Report test error rate
pred.radial <- predict(best.radial, newdata = auto_test)
confusionMatrix(data = pred.radial,
                reference = auto_test$mpg_cat)
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction low high
##       low   52    4
##       high   6   54
##
##                Accuracy : 0.9138
##                  95% CI : (0.8472, 0.9579)
##     No Information Rate : 0.5
```

```
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8276
##
##  Mcnemar's Test P-Value : 0.7518
##
##             Sensitivity : 0.8966
##             Specificity : 0.9310
##          Pos Pred Value : 0.9286
##          Neg Pred Value : 0.9000
##              Prevalence : 0.5000
##          Detection Rate : 0.4483
##    Detection Prevalence : 0.4828
##       Balanced Accuracy : 0.9138
##
##        'Positive' Class : low
##
```

- From the summary, we found the optimal tuning parameters, gamma and cost, of the support vector machine are 0.1353353 and 6.650798. There are 72 support vectors in the optimal support vector classifier with a linear kernel.
- We use the best model to determine our training and testing error rates. The training error rate for the support vector machine is (1-0.9783)x100% = 2.17%. The test error rate is (1-0.9138)x100% = 8.62%.

Question 2

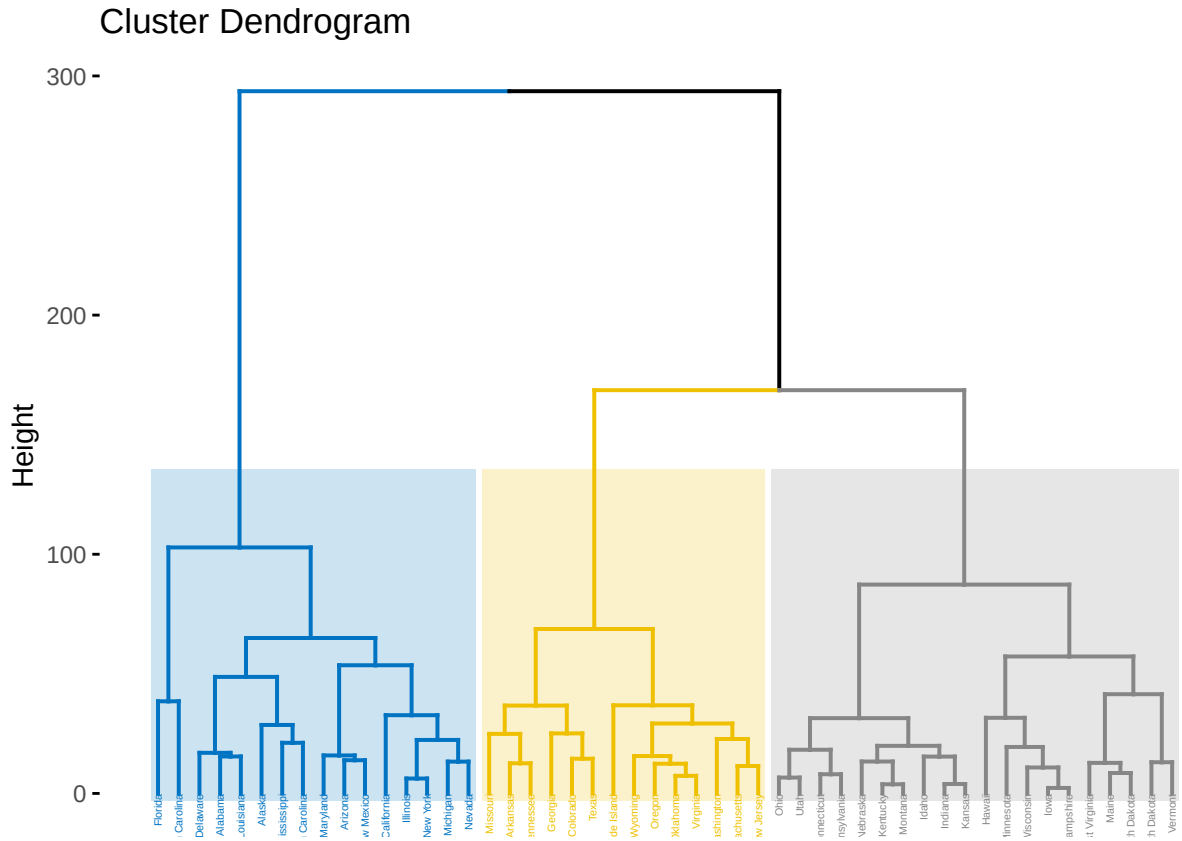
We perform hierarchical clustering on the states using the USArrests data in the ISLR package. For each of the 50 states in the United States, the dataset contains the number of arrests per 100,000 residents for each of three crimes: Assault, Murder, and Rape. The dataset also contains the percent of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

```
# import data
data(USArrests)
arrests_df = USArrests %>%
  as.data.frame() %>%
  janitor::clean_names()
```

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.

```
# compute the 3 clusters of states
hc.complete <- hclust(dist(arrests_df), method = "complete")

# visualize
fviz_dend(hc.complete, k = 3,
          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 states in different clusters
clus1 <- rownames(arrests_df[ind3.complete == 1,]); clus1
```

```
##  [1] "Alabama"        "Alaska"         "Arizona"        "California"
##  [5] "Delaware"       "Florida"        "Illinois"       "Louisiana"
##  [9] "Maryland"       "Michigan"       "Mississippi"    "Nevada"
## [13] "New Mexico"     "New York"       "North Carolina" "South Carolina"
```

```
clus2 <- rownames(arrests_df[ind3.complete == 2,]); clus2
```

```
##  [1] "Arkansas"      "Colorado"      "Georgia"       "Massachusetts"
##  [5] "Missouri"      "New Jersey"    "Oklahoma"      "Oregon"
##  [9] "Rhode Island"  "Tennessee"     "Texas"         "Virginia"
## [13] "Washington"    "Wyoming"
```

```
clus3 <- rownames(arrests_df[ind3.complete == 3,]); clus3
```

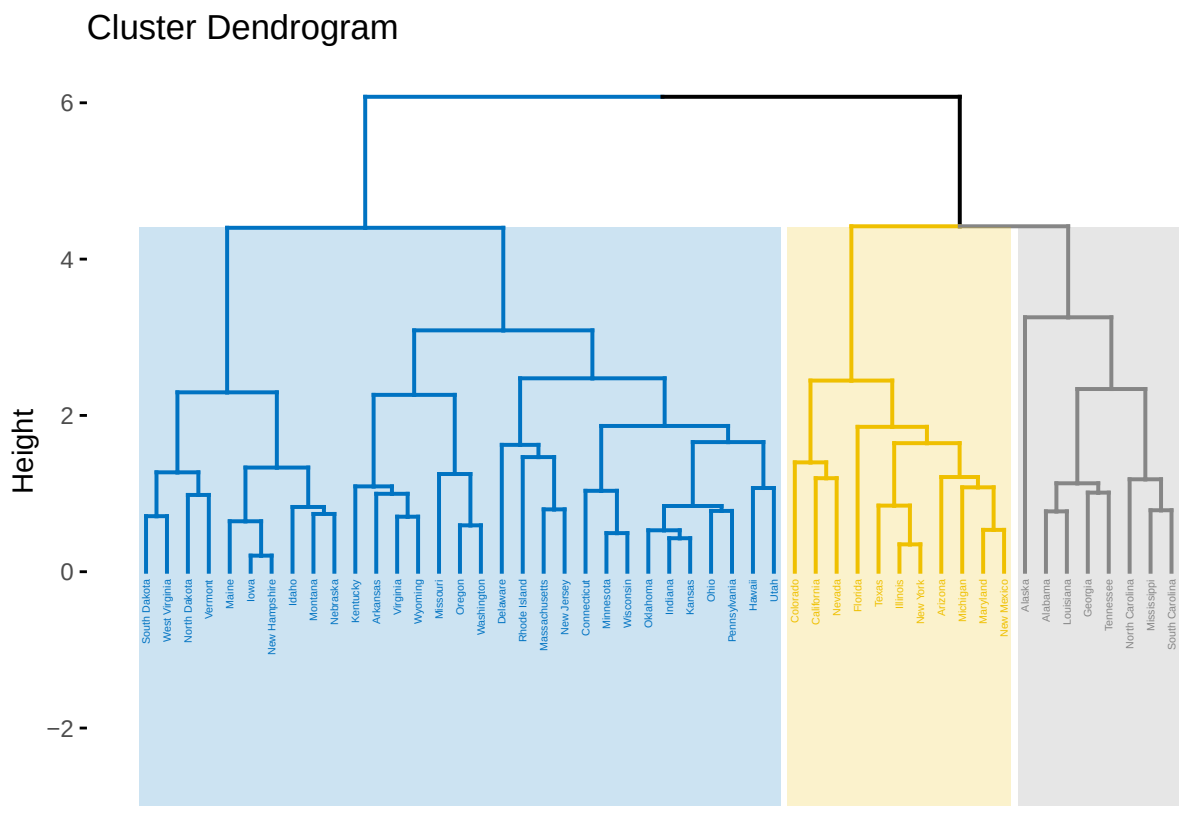
```
##  [1] "Connecticut"   "Hawaii"        "Idaho"         "Indiana"
##  [5] "Iowa"          "Kansas"        "Kentucky"      "Maine"
##  [9] "Minnesota"     "Montana"       "Nebraska"      "New Hampshire"
## [13] "North Dakota"  "Ohio"          "Pennsylvania"  "South Dakota"
## [17] "Utah"          "Vermont"       "West Virginia" "Wisconsin"
```

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

b) Hierarchically cluster the states using complete linkage and Euclidean distance, after scaling the variables to have standard deviation one.

```
arrests_df_scaled = scale(arrests_df)

hc.complete.scaled <- hclust(dist(arrests_df_scaled), method = "complete")
fviz_dend(hc.complete.scaled, k = 3,
          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.scaled <- cutree(hc.complete.scaled, 3)

# The states in different clusters for standardized data
scaled.clus1 <- rownames(arrests_df[ind3.complete.scaled == 1,]); scaled.clus1
```

```
## [1] "Alabama"        "Alaska"         "Georgia"        "Louisiana"
## [5] "Mississippi"    "North Carolina" "South Carolina" "Tennessee"
```

```
scaled.clus2 <- rownames(arrests_df[ind3.complete.scaled == 2,]); scaled.clus2
```

```
##  [1] "Arizona"    "California" "Colorado"   "Florida"    "Illinois"
##  [6] "Maryland"   "Michigan"   "Nevada"     "New Mexico" "New York"
## [11] "Texas"
```

```
scaled.clus3 <- rownames(arrests_df[ind3.complete.scaled == 3,]); scaled.clus3
```

```
##  [1] "Arkansas"      "Connecticut"   "Delaware"      "Hawaii"
##  [5] "Idaho"         "Indiana"       "Iowa"          "Kansas"
##  [9] "Kentucky"      "Maine"         "Massachusetts" "Minnesota"
## [13] "Missouri"      "Montana"       "Nebraska"      "New Hampshire"
## [17] "New Jersey"    "North Dakota"  "Ohio"          "Oklahoma"
## [21] "Oregon"        "Pennsylvania"  "Rhode Island"  "South Dakota"
## [25] "Utah"          "Vermont"       "Virginia"      "Washington"
## [29] "West Virginia" "Wisconsin"     "Wyoming"
```

- The states in cluster 1 include Alabama, Alaska, Georgia, Louisiana, Mississippi, North Carolina, South Carolina, and Tennessee.
- The states in cluster 2 include Arizona, California, Colorado, Florida, Illinois, Maryland, Michigan, Nevada, New Mexico, New York, and Texas.
- The states in cluster 3 include Arkansas, Connecticut, Delaware, Hawaii, Idaho, Indiana, Iowa, Kansas, Kentucky, Maine, 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, and Wyoming.

c). Does scaling the variables change the clustering results? Why? In your opinion, should the variables be scaled before the inter-observation dissimilarities are computed?

```
skimr::skim_without_charts(arrests_df)
```

Table 1: Data summary

Name	arrests_df
Number of rows	50
Number of columns	4
Column type frequency:	
numeric	4

Table 1: Data summary

Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
murder	0	1	7.79	4.36	0.8	4.08	7.25	11.25	17.4
assault	0	1	170.76	83.34	45.0	109.00	159.00	249.00	337.0
urban_pop	0	1	65.54	14.47	32.0	54.50	66.00	77.75	91.0
rape	0	1	21.23	9.37	7.3	15.08	20.10	26.17	46.0

- Yes, scaling the variables does change the clustering results. In the scaled dataset, cluster 3 encompasses a greater number of states than cluster 3 in the non-scaled dataset. Additionally, the remaining two clusters also consist of different states after scaling the variables.
- The reason is, when the variables in a dataset have different scales, the distance metric used in clustering can be dominated by the variables with the largest scales. To avoid bias towards these variables, it can be beneficial to scale all variables to the same scale. This will allow each variable to contribute equally to the distance metric and help prevent the clustering algorithm from being biased towards variables with larger scales. This problem employs the Euclidean distance as the distance metric, but the variable urban_pop (percent of the population in each state living in urban areas) in the dataset has units that are incomparable to the other variables such as murder, and rape (number of arrests per 100,000 residents for each of the three crimes).
- In general, whether scaling the variables or not depends on the specific problem and the nature of the data. As for this problem, in my opinion, the variables should be scaled before the inter-observation dissimilarities are computed. By doing so, we can avoid unfairly assigning more significance to the variables with larger magnitudes or incomparable units, such as urban_pop.