Data Science II Homework 5

Camille Okonkwo

Contents

```r
library(tidymodels)
library(caret)
library(ISLR)
library(kernlab)
library(factoextra)
set.seed(2)
```

Question 1

Background

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%).

```
auto = read_csv("data/auto.csv") |>
  drop_na() |>
  mutate(
    mpg_cat = as.factor(mpg_cat),
    mpg_cat = forcats::fct_relevel(mpg_cat, c("low", "high")),
    cylinders = as.factor(cylinders),
    origin = as.factor(origin)
  )

set.seed(2)

# create a random split of 70% training and 30% test data
data_split = initial_split(data = auto, prop = 0.7)

# partitioned datasets
training_data = training(data_split)
testing_data = testing(data_split)

head(training_data)
```

```
## # A tibble: 6 x 8
##   cylinders displacement horsepower weight acceleration  year origin mpg_cat
##   <fct>            <dbl>      <dbl>  <dbl>        <dbl> <dbl> <fct>  <fct>
## 1 4                   86         64   1875         16.4    81 1      high
## 2 6                  225        100   3651         17.7    76 1      low
## 3 6                  231        165   3445         13.4    78 1      low
## 4 5                  131        103   2830         15.9    78 2      low
## 5 4                   98         65   2380         20.7    81 1      high
## 6 4                   97         75   2155         16.4    76 3      high
```

```
head(testing_data)
```

```
## # A tibble: 6 x 8
##   cylinders displacement horsepower weight acceleration  year origin mpg_cat
##   <fct>            <dbl>      <dbl>  <dbl>        <dbl> <dbl> <fct>  <fct>
## 1 8                  302        140   3449         10.5    70 1      low
## 2 8                  390        190   3850          8.5    70 1      low
## 3 4                  113         95   2372         15      70 3      high
## 4 6                  200         85   2587         16      70 1      low
## 5 4                   97         88   2130         14.5    70 3      high
## 6 4                  107         90   2430         14.5    70 2      high
```

```
# training data
x_1 = model.matrix(mpg_cat ~ ., training_data)[, -1] # matrix of predictors
head(x_1)
```

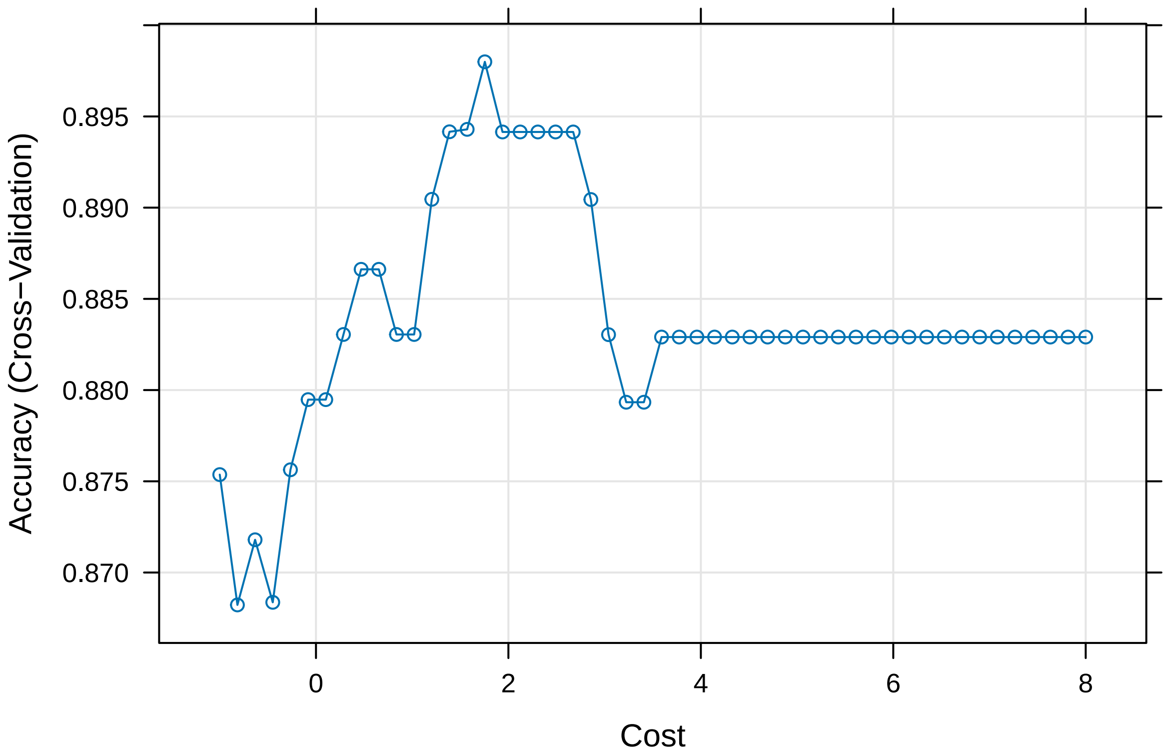
```
##   cylinders4 cylinders5 cylinders6 cylinders8 displacement horsepower weight
```

```
## 1           1          0          0          0          86          64    1875
## 2           0          0          1          0         225         100    3651
## 3           0          0          1          0         231         165    3445
## 4           0          1          0          0         131         103    2830
## 5           1          0          0          0          98          65    2380
## 6           1          0          0          0          97          75    2155
##   acceleration year origin2 origin3
## 1         16.4   81       0       0
## 2         17.7   76       0       0
## 3         13.4   78       0       0
## 4         15.9   78       1       0
## 5         20.7   81       0       0
## 6         16.4   76       0       1
```

```
y_1 = training_data$mpg_cat # vector of response

# testing data
x_2 = model.matrix(mpg_cat ~ .,testing_data)[, -1] # matrix of predictors
y_2 = testing_data$mpg_cat # vector of response
```

(a) Fit a support vector classifier to the training data. What are the training and test error rates?

```
# 10-fold cross-validation
ctrl = trainControl(method = "cv", number = 10)


set.seed(2)

# support vector classifier
svml.fit = train(x_1, y_1,
                 method = "svmLinear",
                 tuneGrid = data.frame(C = exp(seq(-1, 8, len = 50))),
                 trControl = ctrl)

plot(svml.fit, highlight = TRUE, xTrans = log)
```




```
svml.fit$bestTune
```

```
##          C
## 16 5.784038
```

```
# what are the training error rates?
svml.predict = predict(svml.fit,
              newdata = x_1)

confusionMatrix(data = svml.predict,
                reference = y_1,
                )
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction low high
##       low  122    7
##       high  14  131
##
##                Accuracy : 0.9234
##                  95% CI : (0.8852, 0.9519)
##     No Information Rate : 0.5036
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8466
##
##  Mcnemar's Test P-Value : 0.1904
##
##             Sensitivity : 0.8971
##             Specificity : 0.9493
##          Pos Pred Value : 0.9457
##          Neg Pred Value : 0.9034
##              Prevalence : 0.4964
##          Detection Rate : 0.4453
##    Detection Prevalence : 0.4708
##       Balanced Accuracy : 0.9232
##
##        'Positive' Class : low
##
```

```r
# 1 - accuracy
svml_training_error = 1 - 0.9234
svml_training_error
```

```
## [1] 0.0766
```

```r
# test error
svml.test = predict(svml.fit,
              newdata = x_2)

confusionMatrix(data = svml.test,
                reference = y_2,
                )
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   56    2
##       high   4   56
##
##                Accuracy : 0.9492
##                  95% CI : (0.8926, 0.9811)
##     No Information Rate : 0.5085
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8983
##
##  Mcnemar's Test P-Value : 0.6831
```

```
##
##             Sensitivity : 0.9333
##             Specificity : 0.9655
##          Pos Pred Value : 0.9655
##          Neg Pred Value : 0.9333
##              Prevalence : 0.5085
##          Detection Rate : 0.4746
##    Detection Prevalence : 0.4915
##       Balanced Accuracy : 0.9494
##
##        'Positive' Class : low
##
```

```r
# 1 - accuracy
svml_test_error = 1 - 0.9492
svml_test_error
```

```
## [1] 0.0508
```

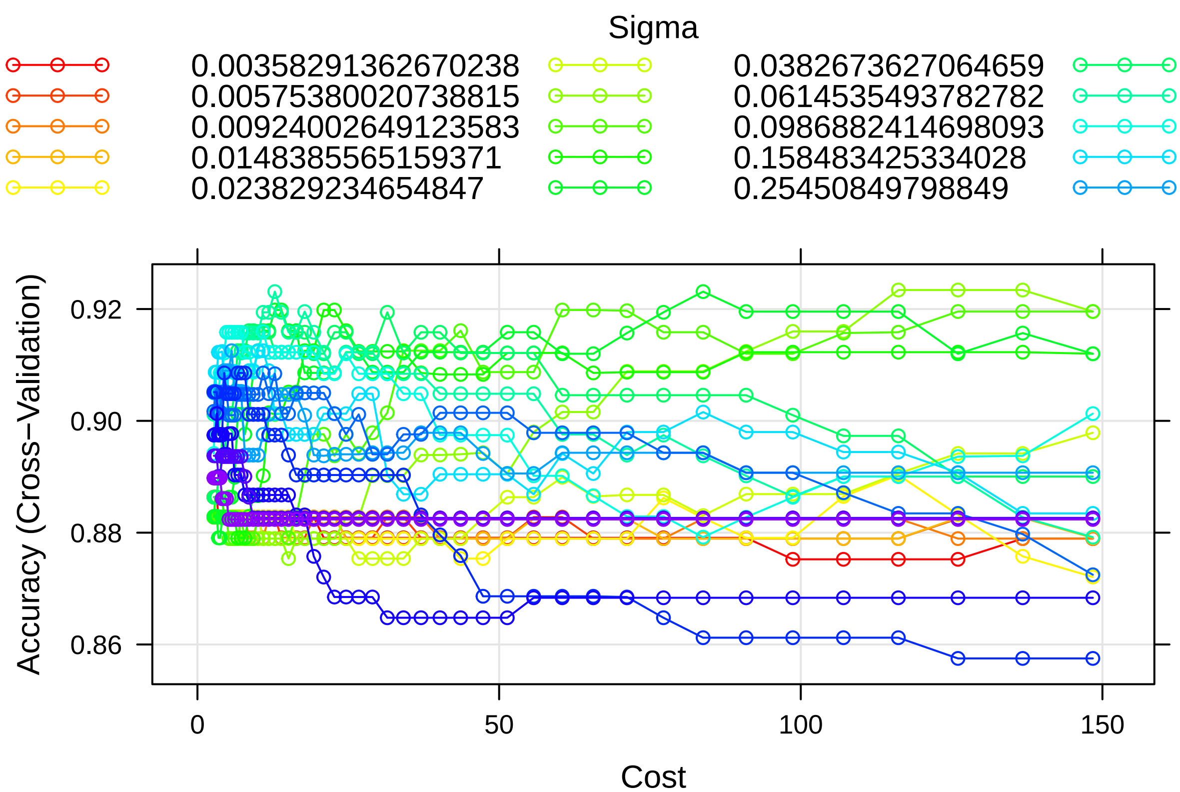
The training error rate is 0.0766, or 7.66%. The testing error rate is 0.0508, or 5.08%.

(b) Fit a support vector machine with a radial kernel to the training data. What are the training and test error rates?

```
svmr.grid = expand.grid(C = exp(seq(1, 5, len = 50)),
                        sigma = exp(seq(-8, 1, len = 20)))

set.seed(2)

# SVM with radial kernel
svmr.fit = train(x_1, y_1,
                 method = "svmRadialSigma",
                 tuneGrid = svmr.grid,
                 trControl = ctrl)

svmr.fit$bestTune
```

```
##         sigma        C
## 927 0.0057538 116.1755
```

```
myCol = rainbow(25)
myPar = list(superpose.symbol = list(col = myCol),
             superpose.line = list(col = myCol))

plot(svmr.fit, highlight = TRUE, par.settings = myPar)
```




```
# training error
svmr.predict = predict(svmr.fit,
                       newdata = x_1)

confusionMatrix(data = svmr.predict,
```

```
                 reference = y_1,
                 )
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low  124    4
##       high  12  134
##
##                Accuracy : 0.9416
##                  95% CI : (0.9069, 0.9663)
##     No Information Rate : 0.5036
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8832
##
##  Mcnemar's Test P-Value : 0.08012
##
##             Sensitivity : 0.9118
##             Specificity : 0.9710
##          Pos Pred Value : 0.9688
##          Neg Pred Value : 0.9178
##              Prevalence : 0.4964
##          Detection Rate : 0.4526
##    Detection Prevalence : 0.4672
##       Balanced Accuracy : 0.9414
##
##        'Positive' Class : low
##
```

```
# 1 - accuracy
svmr_training_error = 1 - 0.9416
svmr_training_error
```

```
## [1] 0.0584
```

```
# test error
svmr.test = predict(svmr.fit,
              newdata = x_2)

confusionMatrix(data = svmr.test,
                reference = y_2,
                )
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   56    1
##       high   4   57
##
##                Accuracy : 0.9576
##                  95% CI : (0.9039, 0.9861)
##     No Information Rate : 0.5085
```

```
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9153
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.9333
##             Specificity : 0.9828
##          Pos Pred Value : 0.9825
##          Neg Pred Value : 0.9344
##              Prevalence : 0.5085
##          Detection Rate : 0.4746
##    Detection Prevalence : 0.4831
##       Balanced Accuracy : 0.9580
##
##        'Positive' Class : low
##
```

```r
# 1 - accuracy
svmr_test_error = 1 - 0.9576
svmr_test_error
```

```
## [1] 0.0424
```

The training error rate is 0.0584, or 5.84%. The testing error rate is 0.0424, or 4.24%.

Question 2

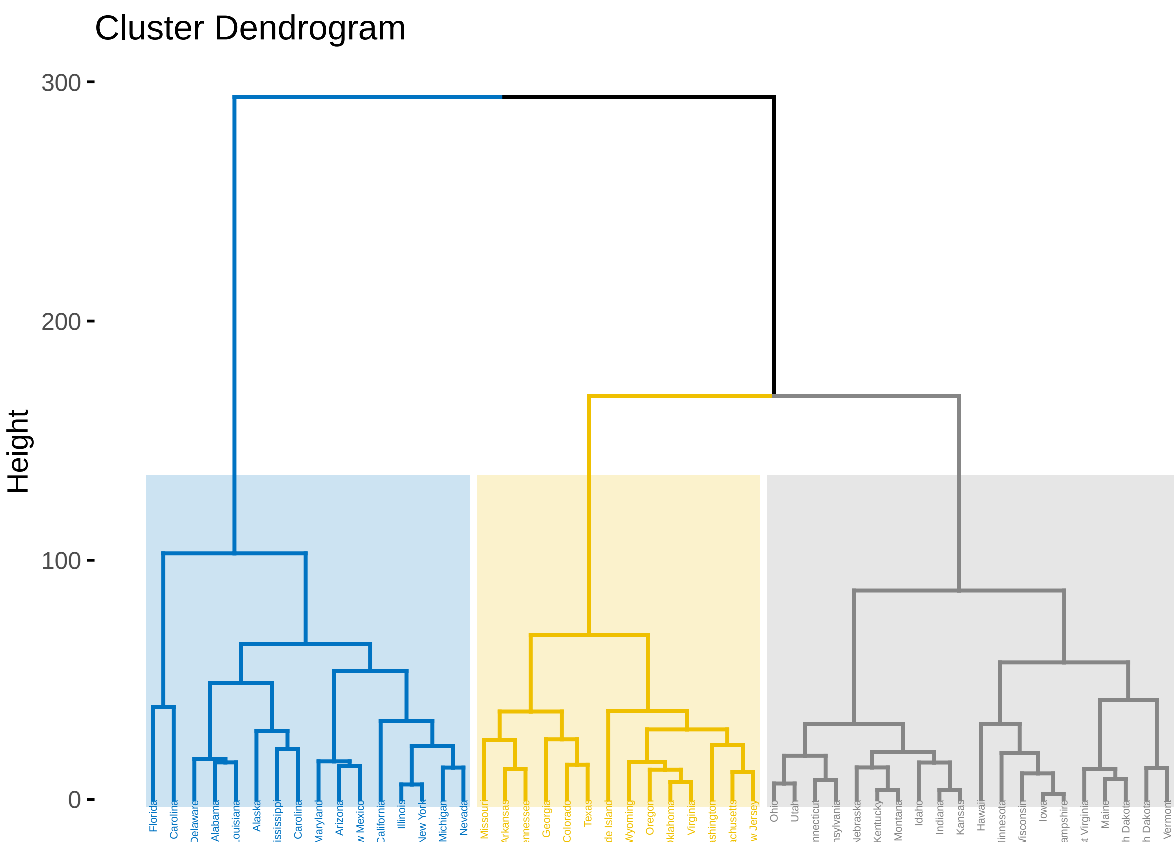
Background

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.

```r
data(USArrests)

us_arrests = na.omit(USArrests)
```

(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?

```r
# complete linkage and euclidean distance
hc.complete = hclust(dist(us_arrests), method = "complete")

# hierarchical clustering
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)
```




```r
us.complete = cutree(hc.complete, 3)

# 1st cluster states
complete_1 = us_arrests[us.complete == 1,]
complete_1
```

```
##                Murder Assault UrbanPop Rape
## Alabama          13.2     236       58 21.2
## Alaska           10.0     263       48 44.5
## Arizona           8.1     294       80 31.0
## California        9.0     276       91 40.6
## Delaware          5.9     238       72 15.8
```

```
## Florida           15.4     335       80 31.9
## Illinois          10.4     249       83 24.0
## Louisiana         15.4     249       66 22.2
## Maryland          11.3     300       67 27.8
## Michigan          12.1     255       74 35.1
## Mississippi       16.1     259       44 17.1
## Nevada            12.2     252       81 46.0
## New Mexico        11.4     285       70 32.1
## New York          11.1     254       86 26.1
## North Carolina    13.0     337       45 16.1
## South Carolina    14.4     279       48 22.5
```

```r
state_names1 = rownames(complete_1)

# 2nd cluster states
complete_2 = us_arrests[us.complete == 2,]
complete_2
```

```
##                Murder Assault UrbanPop Rape
## Arkansas         8.8     190       50 19.5
## Colorado         7.9     204       78 38.7
## Georgia         17.4     211       60 25.8
## Massachusetts    4.4     149       85 16.3
## Missouri         9.0     178       70 28.2
## New Jersey       7.4     159       89 18.8
## Oklahoma         6.6     151       68 20.0
## Oregon           4.9     159       67 29.3
## Rhode Island     3.4     174       87  8.3
## Tennessee       13.2     188       59 26.9
## Texas           12.7     201       80 25.5
## Virginia         8.5     156       63 20.7
## Washington       4.0     145       73 26.2
## Wyoming          6.8     161       60 15.6
```

```r
state_names2 = rownames(complete_2)

# 3rd cluster states
complete_3 = us_arrests[us.complete == 3,]
complete_3
```

```
##                Murder Assault UrbanPop Rape
## Connecticut      3.3     110       77 11.1
## Hawaii           5.3      46       83 20.2
## Idaho            2.6     120       54 14.2
## Indiana          7.2     113       65 21.0
## Iowa             2.2      56       57 11.3
## Kansas           6.0     115       66 18.0
## Kentucky         9.7     109       52 16.3
## Maine            2.1      83       51  7.8
## Minnesota        2.7      72       66 14.9
## Montana          6.0     109       53 16.4
## Nebraska         4.3     102       62 16.5
## New Hampshire    2.1      57       56  9.5
## North Dakota     0.8      45       44  7.3
## Ohio             7.3     120       75 21.4
```

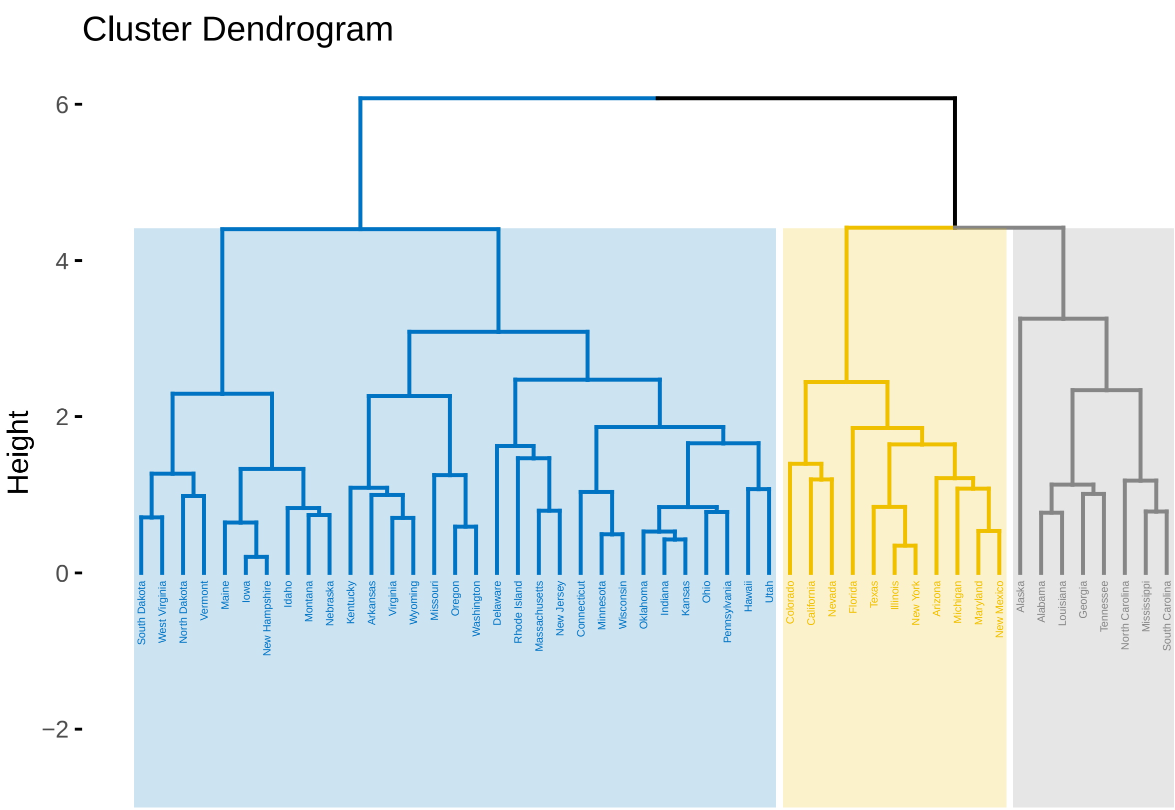
```
## Pennsylvania     6.3     106       72 14.9
## South Dakota     3.8      86       45 12.8
## Utah             3.2     120       80 22.9
## Vermont          2.2      48       32 11.2
## West Virginia    5.7      81       39  9.3
## Wisconsin        2.6      53       66 10.8
```

```
state_names3 = rownames(complete_3)
```

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

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

```r
# scale the variables to have standard deviation one
scaled_arrests = scale(us_arrests)

# complete linkage and Euclidean distance
hc_complete = hclust(dist(scaled_arrests), method = "complete")

# hierarchical clustering dendogram
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)
```




```r
clusters_scaled = cutree(hc_complete, k = 3)

# 1st cluster states
scaled_1 = us_arrests[clusters_scaled == 1,]
scaled_1
```

```
##                Murder Assault UrbanPop Rape
## Alabama          13.2     236       58 21.2
## Alaska           10.0     263       48 44.5
## Georgia          17.4     211       60 25.8
## Louisiana        15.4     249       66 22.2
## Mississippi      16.1     259       44 17.1
```

```
## North Carolina   13.0     337       45 16.1
## South Carolina   14.4     279       48 22.5
## Tennessee        13.2     188       59 26.9
```

```
scaled_names1 = rownames(scaled_1)

# 2nd cluster states
scaled_2 = us_arrests[clusters_scaled == 2,]
scaled_2
```

```
##             Murder Assault UrbanPop Rape
## Arizona        8.1     294       80 31.0
## California     9.0     276       91 40.6
## Colorado       7.9     204       78 38.7
## Florida       15.4     335       80 31.9
## Illinois      10.4     249       83 24.0
## Maryland      11.3     300       67 27.8
## Michigan      12.1     255       74 35.1
## Nevada        12.2     252       81 46.0
## New Mexico    11.4     285       70 32.1
## New York      11.1     254       86 26.1
## Texas         12.7     201       80 25.5
```

```
scaled_names2 = rownames(scaled_2)

# 3rd cluster states
scaled_3 = us_arrests[clusters_scaled == 3,]
scaled_3
```

```
##                Murder Assault UrbanPop Rape
## Arkansas          8.8     190       50 19.5
## Connecticut       3.3     110       77 11.1
## Delaware          5.9     238       72 15.8
## Hawaii            5.3      46       83 20.2
## Idaho             2.6     120       54 14.2
## Indiana           7.2     113       65 21.0
## Iowa              2.2      56       57 11.3
## Kansas            6.0     115       66 18.0
## Kentucky          9.7     109       52 16.3
## Maine             2.1      83       51  7.8
## Massachusetts     4.4     149       85 16.3
## Minnesota         2.7      72       66 14.9
## Missouri          9.0     178       70 28.2
## Montana           6.0     109       53 16.4
## Nebraska          4.3     102       62 16.5
## New Hampshire     2.1      57       56  9.5
## New Jersey        7.4     159       89 18.8
## North Dakota      0.8      45       44  7.3
## Ohio              7.3     120       75 21.4
## Oklahoma          6.6     151       68 20.0
## Oregon            4.9     159       67 29.3
## Pennsylvania      6.3     106       72 14.9
## Rhode Island      3.4     174       87  8.3
## South Dakota      3.8      86       45 12.8
## Utah              3.2     120       80 22.9
```

```
## Vermont          2.2      48       32 11.2
## Virginia         8.5     156       63 20.7
## Washington       4.0     145       73 26.2
## West Virginia    5.7      81       39  9.3
## Wisconsin        2.6      53       66 10.8
## Wyoming          6.8     161       60 15.6
```

```
scaled_names3 = rownames(scaled_3)
```

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

Scaling the variables does change the clustering results. This is because hierarchical clustering methods, specifically methods like complete linkage and Euclidean distance, are sensitive to variable scales and variances. When variables have different scales or variances, those with larger variations can disproportionately influence the distance calculations, potentially leading to clusters that are biased towards these variables.

In my opinion, scaling the variables before computing inter-observation dissimilarities is advantageous. It ensures that each variable contributes equally to the distance calculations, preventing any single variable from dominating the clustering process. This approach typically results in more balanced clusters that better capture the underlying patterns in the data.

It's important to consider, however, the situations where scaling may not be necessary or even counterproductive. For example, if the variables are already on similar scales or if domain knowledge/the research question suggests that scaling is unnecessary for the specific analysis, then clustering without scaling might be more appropriate.