Homework 2

Varun A Bhagat

2018-09-05

```
Sys.setenv(JAVA_HOME='C:\\Program Files\\Java\\jre1.8.0_144')
library(kernlab)
library(kknn)
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:kernlab':
##
##     alpha
```

Question 3.1: Using the same data set (credit_card_data.txt or credit_card_data-headers.txt) as in Question 2.2, use the ksvm or kknn function to find a good classifier:

Part a: Using cross-validation (do this for the k-nearest-neighbors model; SVM is optional);

```
#Load the data
CC_Data = read.table("C:\\Users\\vabhagat\\Downloads\\credit_card_data.txt", header=F, stringsAs
Factors = F)
head(CC_Data)
```

	V1 <int>	V2 <dbl>	V3 <dbl>	V4 <dbl>	V5 <int>	V6 <int>	V7 <int>	V8 <int>	V9 <int>
1	1	30.83	0.000	1.25	1	0	1	1	202
2	0	58.67	4.460	3.04	1	0	6	1	43
3	0	24.50	0.500	1.50	1	1	0	1	280
4	1	27.83	1.540	3.75	1	0	5	0	100
5	1	20.17	5.625	1.71	1	1	0	1	120
6	1	32.08	4.000	2.50	1	1	0	0	360

6 rows | 1-10 of 12 columns

```
set.seed(1)

# Trying train.kknn for leave-one-out cross-validation Kmax=30

CC_Model=train.kknn(V11~.,CC_Data,kmax=30, scale=TRUE)

# Accuracy Matrix
CC_Model_accry=rep(0,30)

# calculate prediction qualities

for (k in 1:30) {
  CC_Model_predicted <- as.integer(fitted(CC_Model)[[k]][1:nrow(CC_Data)] + 0.5)
  CC_Model_accry[k] <- sum(CC_Model_predicted == CC_Data$V11)
}

# show accuracies

CC_Model_accry
```

```
##  [1] 533 533 533 533 557 553 554 555 554 557 557 558 557 557 558 558 558
## [18] 557 556 556 555 554 552 553 553 552 550 548 549 550
```

Part B: splitting the data into training, validation, and test data sets

```
#Load the data
CC_Data = read.table("C:\\Users\\vabhagat\\Downloads\\credit_card_data.txt", header=F, stringsAs
Factors = F)
head(CC_Data)
```

	V1 <int>	V2 <dbl>	V3 <dbl>	V4 <dbl>	V5 <int>	V6 <int>	V7 <int>	V8 <int>	V9 <int>
1	1	30.83	0.000	1.25	1	0	1	1	202
2	0	58.67	4.460	3.04	1	0	6	1	43
3	0	24.50	0.500	1.50	1	1	0	1	280
4	1	27.83	1.540	3.75	1	0	5	0	100
5	1	20.17	5.625	1.71	1	1	0	1	120
6	1	32.08	4.000	2.50	1	1	0	0	360

6 rows | 1-10 of 12 columns

```
set.seed(1)

# Randomly taking 70% to use for training

CC_Data_smpl70 = sample(nrow(CC_Data), round(nrow(CC_Data)*.7))
CC_training_70 = CC_Data[CC_Data_smpl70, ]

# Split the remaing into validation and test
temp = CC_Data[-CC_Data_smpl70, ]
temp2 = sample(nrow(temp), round(nrow(temp)*.5))
cc_validation = temp[temp2, ]
cc_test = temp[-temp2, ]


### Create a function that trains a model with neighbors
train.model = function(neighbors)
          {
              cc_model = kknn(V11 ~., cc_test, cc_validation, k = neighbors, scale = TRUE)

              # store the predictions
              cc_model_fitted = as.matrix(cc_model$fitted.values)

              # round the fitted values
              cc_model_fitted_rounded = as.matrix(lapply(cc_model_fitted[, 1], round))

              # merge rounded that values back into model_fitted
              cc_model_results = data.frame(cc_validation[, 'V11'], cc_model_fitted_rounded)

              colnames(cc_model_results) = c("actual", "predicted")
              return(cc_model_results)
          }
### Create a function that calculates the accuracy based on predictions
eval.model = function(cc_model)
          {
                  results <- vector("list", nrow(cc_model))
                for(i in 1:nrow(cc_model)){
                  results[[i]] <- as.integer(cc_model[i, 'actual'] == cc_model[i, 'predicted'])
                }
                # calculates the percent of correct predictions
                  correct = sum(data.frame(results))
                  accuracy = correct / nrow(cc_model)
                  return(accuracy)
        }

### Train three models using k values of 4, 12, and 15
cc_model_k4 = train.model(4)
cc_model_k12 = train.model(12)
cc_model_k16 = train.model(16)

### Return accuracy for each model
eval.model(cc_model_k4)
```

```
## [1] 0.7244898
```

```
eval.model(cc_model_k12)
```

```
## [1] 0.8571429
```

```
eval.model(cc_model_k16)
```

```
## [1] 0.877551
```

```
### Selecting model_k16 as it has the best results.

### Re-train model_k16 using the entire training dataset and score against the test dataset
cc_model = kknn(V11 ~., CC_training_70, cc_test, k = 16, scale = TRUE)

cc_model_fitted = as.matrix(cc_model$fitted.values)
cc_model_fitted_rounded = as.matrix(lapply(cc_model_fitted[, 1], round))
cc_model_results = data.frame(cc_validation[, 'V11'], cc_model_fitted_rounded)
cc_model_results = data.frame(cc_test[, 'V11'], cc_model_fitted_rounded)
colnames(cc_model_results) = c("actual", "predicted")
model_accuracy = eval.model(cc_model_results)
model_accuracy
```

```
## [1] 0.8367347
```

Question 4.1: Describe a situation or problem from your job, everyday life, current events, etc., for which a clustering model would be appropriate. List some (up to 5) predictors that you might use.

I used to work for a large healthcare firm, clustering was needed to figure the where they should be opening emergerncy care wards. Some of the important predictors were identifying Accident Prone Areas, Average driving distance and distance from nearest emergency clinic.

Question 4.2: The iris data set iris.txt contains 150 data points, each with four predictor variables and one categorical response. The predictors are the width and length of the sepal and petal of flowers and the response is the type of flower. The data is available from the R library datasets and can be accessed with iris once the library is loaded. It is also available at the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Iris (https://archive.ics.uci.edu/ml/datasets/Iris)). The response values are only given to see how well a specific method performed and

should not be used to build the model. Use the R function kmeans to cluster the points as well as possible. Report the best combination of predictors, your suggested value of k, and how well your best clustering predicts flower type.

```
iris_df = read.table("C:\\Users\\vabhagat\\Downloads\\iris.txt", header= TRUE, stringsAsFactors
 = F)
iris_df  = iris_df[,2:6]
head(iris_df)
```

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <chr>
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

6 rows

```
set.seed(33)



# scale the data:

iris_scaled_df = iris_df
for (i in 1:4) { iris_scaled_df[,i] <- (iris_df[,i]-min(iris_df[,i]))/(max(iris_df[,i])-min(iris
_df[,i])) }

# Kmean clustering

iris_cluster_1 = kmeans(iris_scaled_df[,1:4], 2, nstart = 20)
iris_cluster_2 = kmeans(iris_scaled_df[,1:4], 3, nstart = 20)
iris_cluster_3 = kmeans(iris_scaled_df[,1:4], 4, nstart = 20)
iris_cluster_4 = kmeans(iris_scaled_df[,1:4], 5, nstart = 20)

# total distance between data points and cluster centers:

cluster_dist = 0

for (i in 1:nrow(iris_df)) {

  cluster_dist = cluster_dist + dist(rbind(iris_df[i,1:4],iris_cluster_1$centers[iris_cluster_1
$cluster[i],]))
}

cluster_dist[1]
```

```
## [1] 1052.231
```

```
#Comparing clusters with the species.


table(iris_cluster_1$cluster, iris_df$Species)
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         50        50
```

```
table(iris_cluster_2$cluster, iris_df$Species)
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         47        14
##   3      0          3        36
```

```
table(iris_cluster_3$cluster, iris_df$Species)
```

```
##
##     setosa versicolor virginica
##   1      0         27         2
##   2     50          0         0
##   3      0         23        19
##   4      0          0        29
```

```
table(iris_cluster_4$cluster, iris_df$Species)
```

```
##
##     setosa versicolor virginica
##   1      0         27         2
##   2     22          0         0
##   3      0          0        29
##   4      0         23        19
##   5     28          0         0
```

```
# Using Sepal Length Width


iris_cluster_sepal_1 = kmeans(iris_scaled_df[,1:2], 2, nstart = 20)
iris_cluster_sepal_2 = kmeans(iris_scaled_df[,1:2], 3, nstart = 20)
iris_cluster_sepal_3 = kmeans(iris_scaled_df[,1:2], 4, nstart = 20)
iris_cluster_sepal_4 = kmeans(iris_scaled_df[,1:2], 5, nstart = 20)

#Comparing clusters with the species.
table(iris_cluster_sepal_1$cluster, iris_df$Species)
```

```
##
##     setosa versicolor virginica
##   1     50          7         1
##   2      0         43        49
```

```
table(iris_cluster_sepal_2$cluster, iris_df$Species)
```

```
##
##     setosa versicolor virginica
##   1      1         37        16
##   2     49          0         0
##   3      0         13        34
```

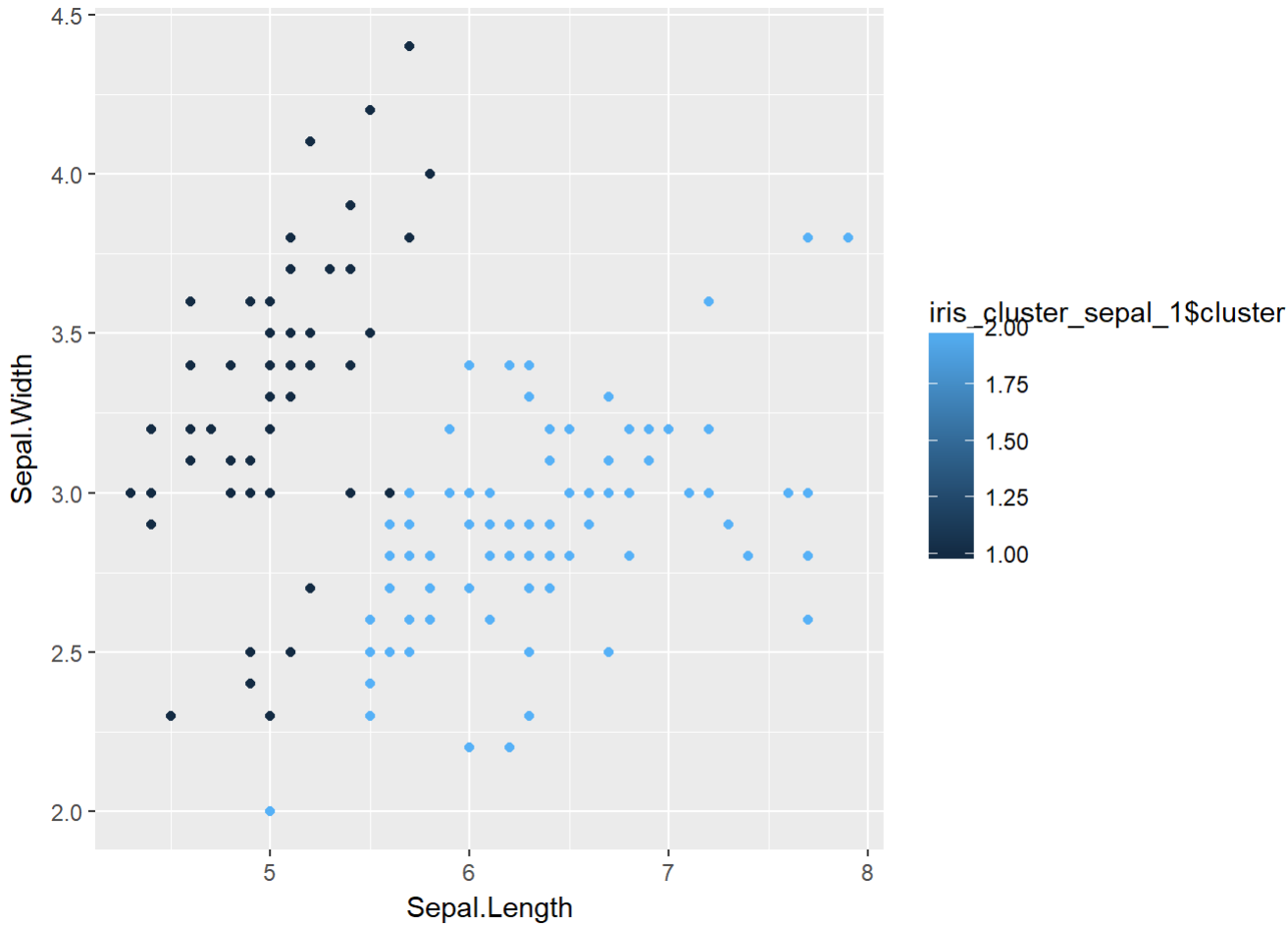
```
table(iris_cluster_sepal_3$cluster, iris_df$Species)
```

```
##
##     setosa versicolor virginica
##   1     33          1         0
##   2      0         10        30
##   3     17          6         1
##   4      0         33        19
```

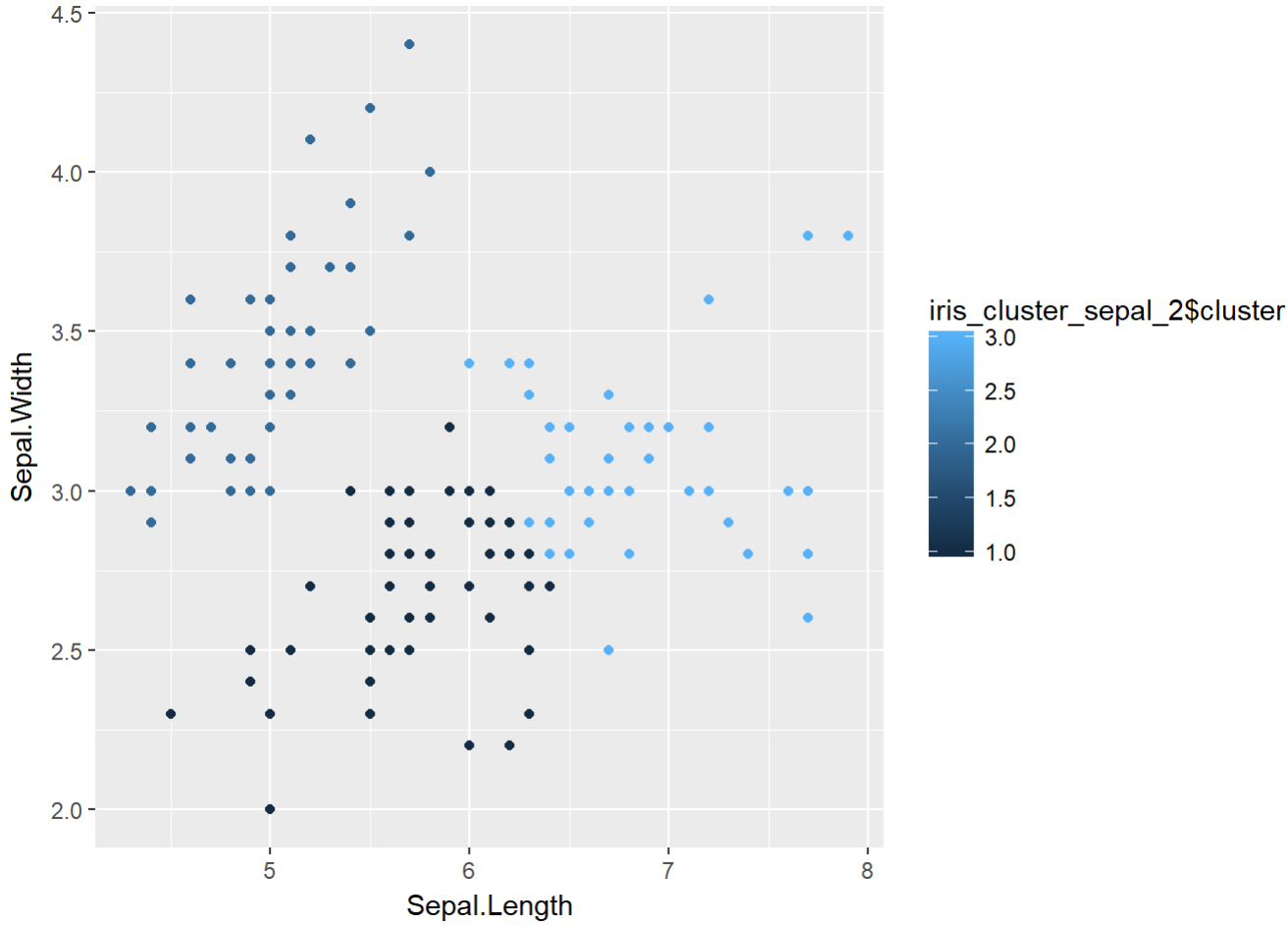
```
table(iris_cluster_sepal_4$cluster, iris_df$Species)
```

```
##
##     setosa versicolor virginica
##   1     35          1         0
##   2      0          2        16
##   3      0         19        25
##   4      1         28         9
##   5     14          0         0
```

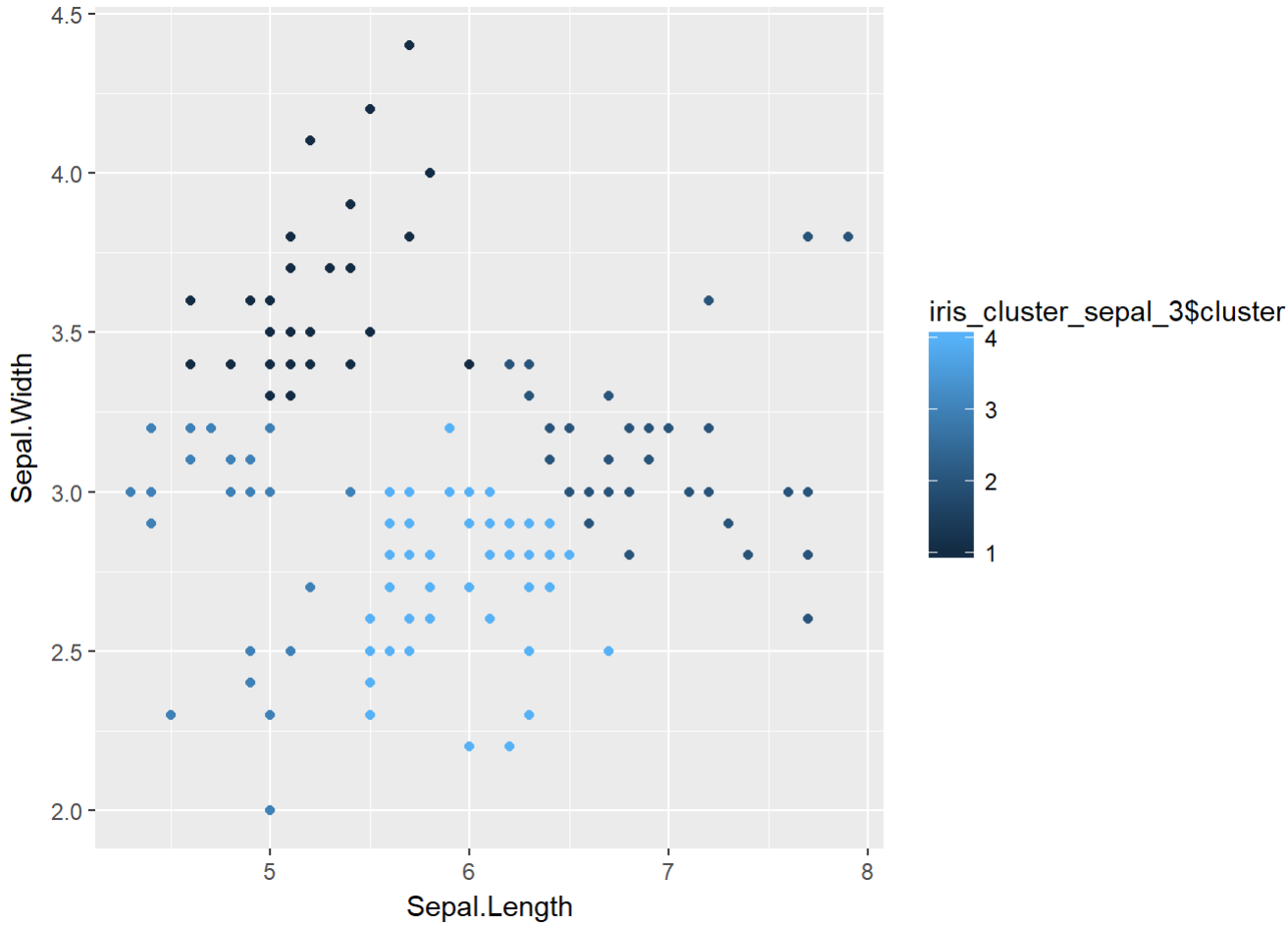
```
# plotting

ggplot(iris, aes(Sepal.Length, Sepal.Width, color = iris_cluster_sepal_1$cluster)) + geom_point
()
```



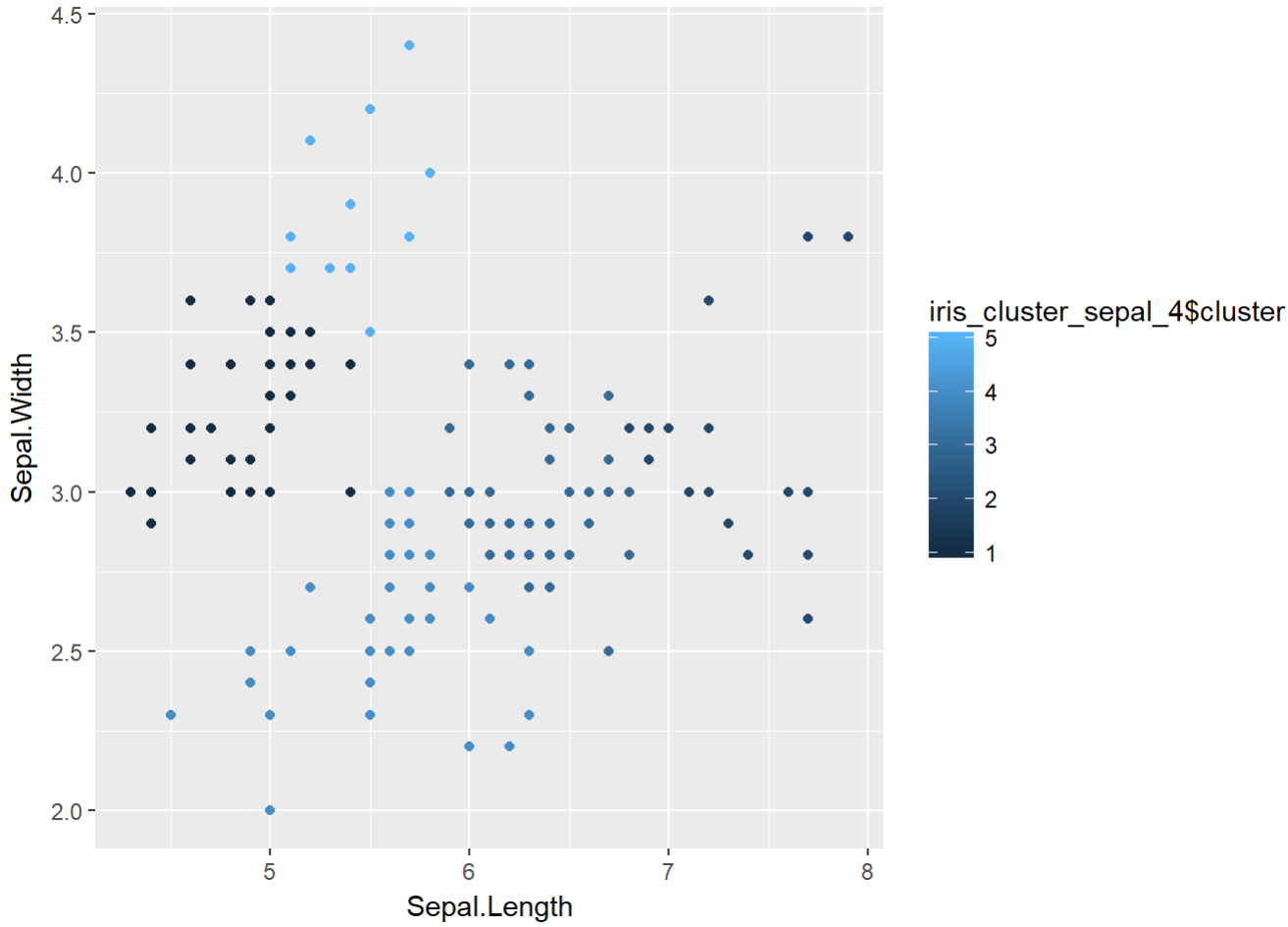
```
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = iris_cluster_sepal_2$cluster)) + geom_point
()
```




```
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = iris_cluster_sepal_3$cluster)) + geom_point
()
```




```
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = iris_cluster_sepal_4$cluster)) + geom_point
()
```




```
# Using Petal Length Width

iris_cluster_petal_1 = kmeans(iris_scaled_df[,3:4], 2, nstart = 20)
iris_cluster_petal_2 = kmeans(iris_scaled_df[,3:4], 3, nstart = 20)
iris_cluster_petal_3 = kmeans(iris_scaled_df[,3:4], 4, nstart = 20)
iris_cluster_petal_4 = kmeans(iris_scaled_df[,3:4], 5, nstart = 20)

#Comparing clusters with the species.
table(iris_cluster_petal_1$cluster, iris_df$Species)
```

```
##
##     setosa versicolor virginica
##   1      0         50        50
##   2     50          0         0
```

```
table(iris_cluster_petal_2$cluster, iris_df$Species)
```

```
##
##     setosa versicolor virginica
##   1      0         48         4
##   2     50          0         0
##   3      0          2        46
```

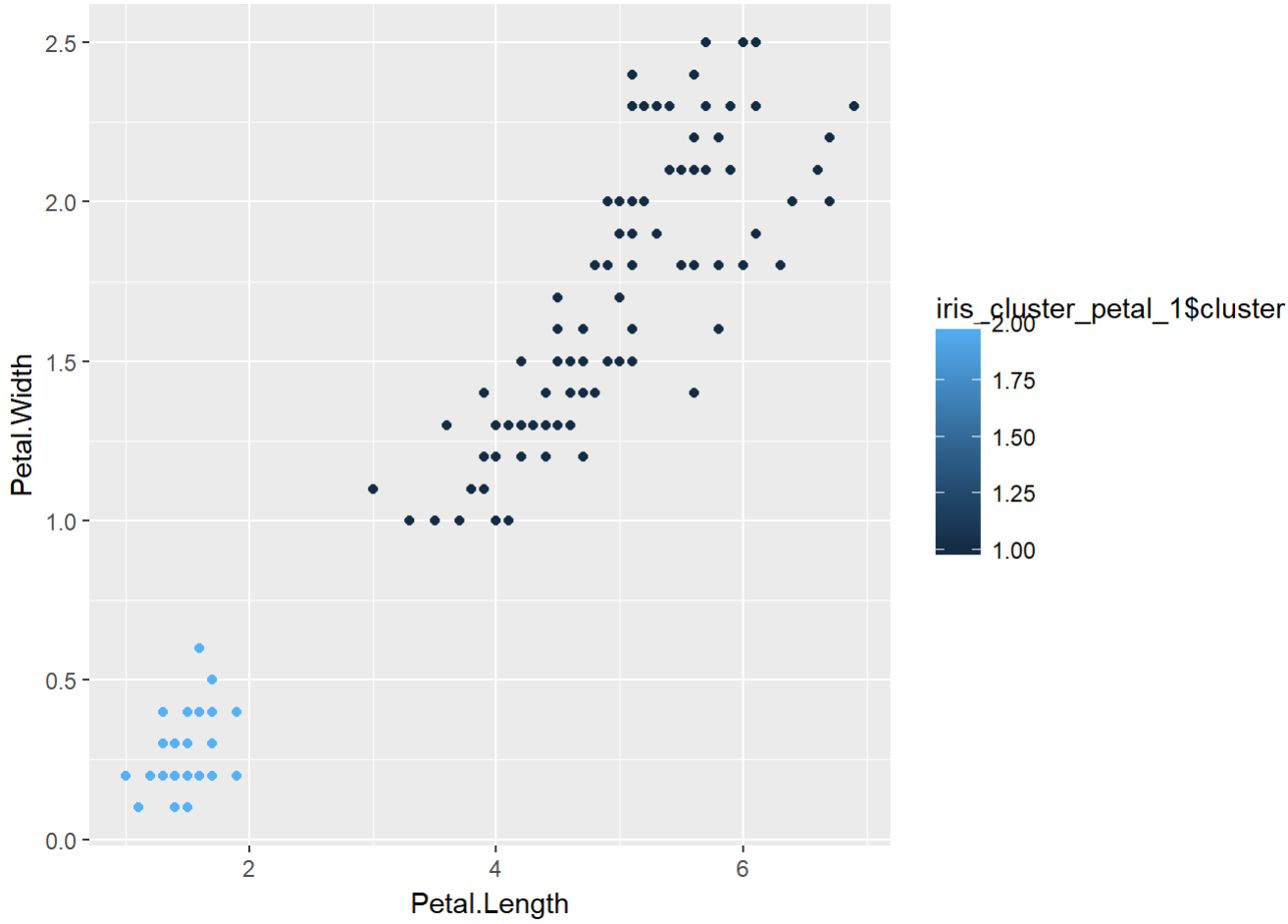
```
table(iris_cluster_petal_3$cluster, iris_df$Species)
```

```
##
##     setosa versicolor virginica
##   1      0         42         0
##   2      0          0        27
##   3     50          0         0
##   4      0          8        23
```

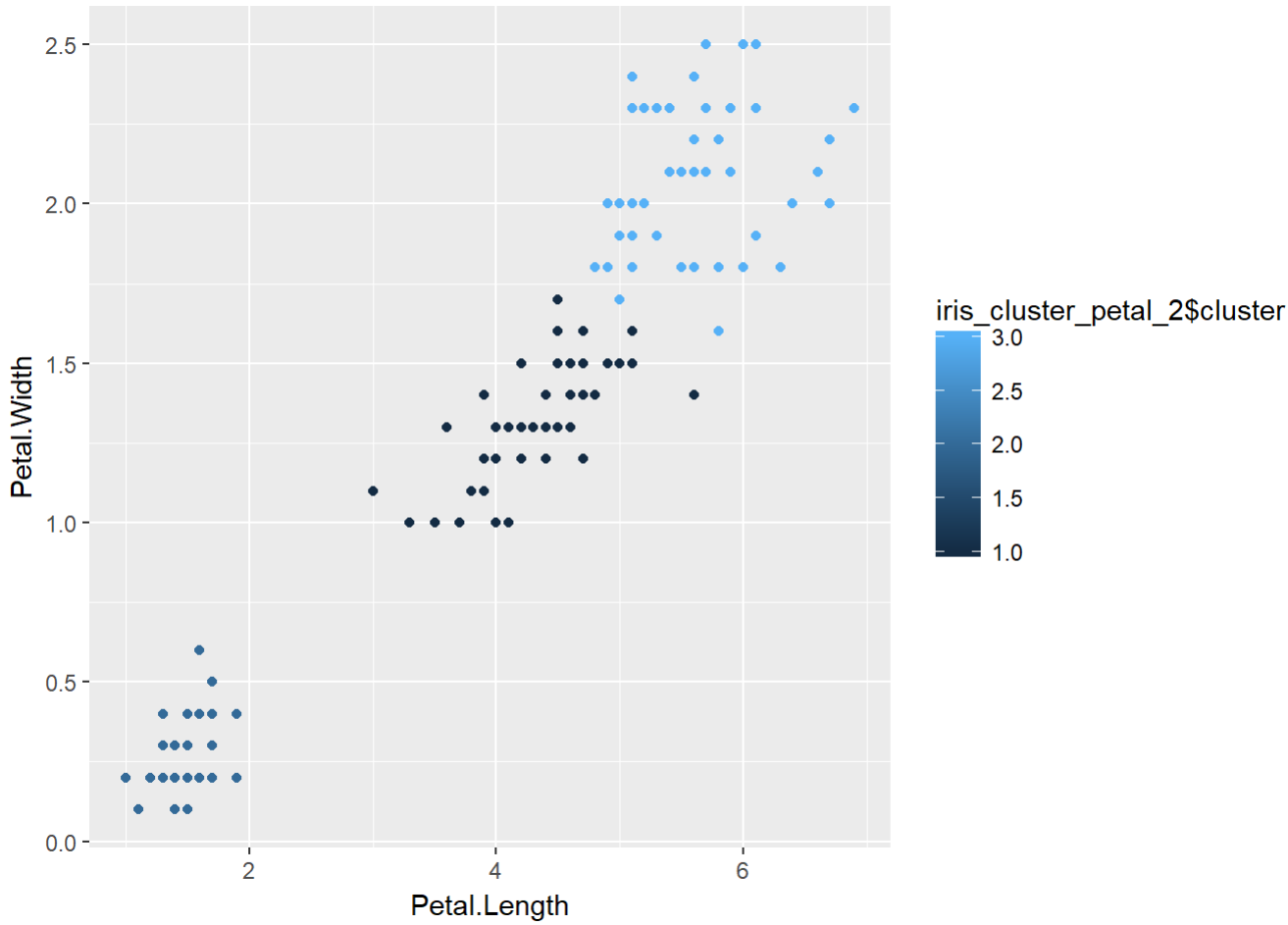
```
table(iris_cluster_petal_4$cluster, iris_df$Species)
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         23         0
##   3      0          0        25
##   4      0          2        21
##   5      0         25         4
```

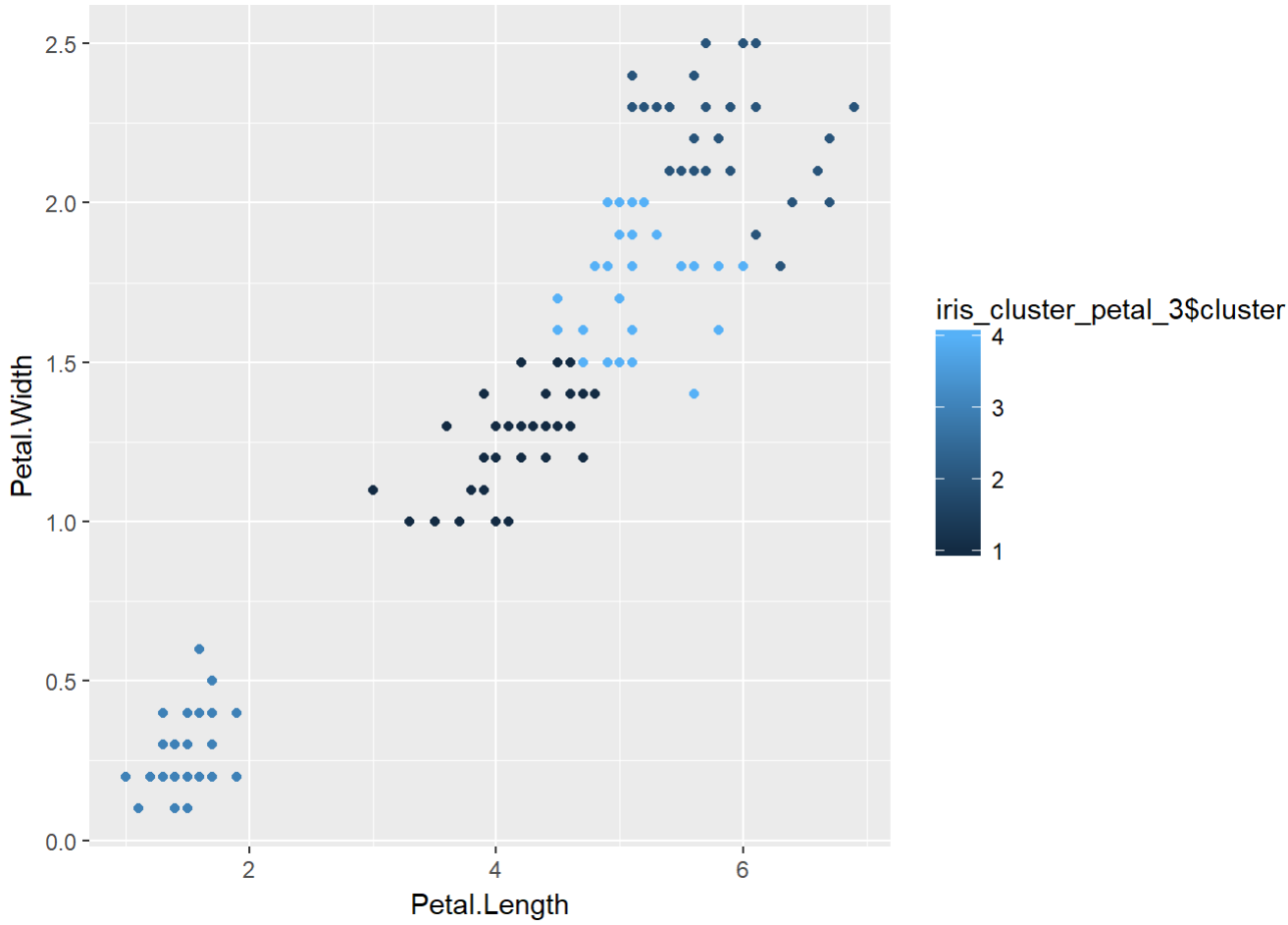
```
# plotting

ggplot(iris, aes(Petal.Length, Petal.Width, color = iris_cluster_petal_1$cluster)) + geom_point
()
```



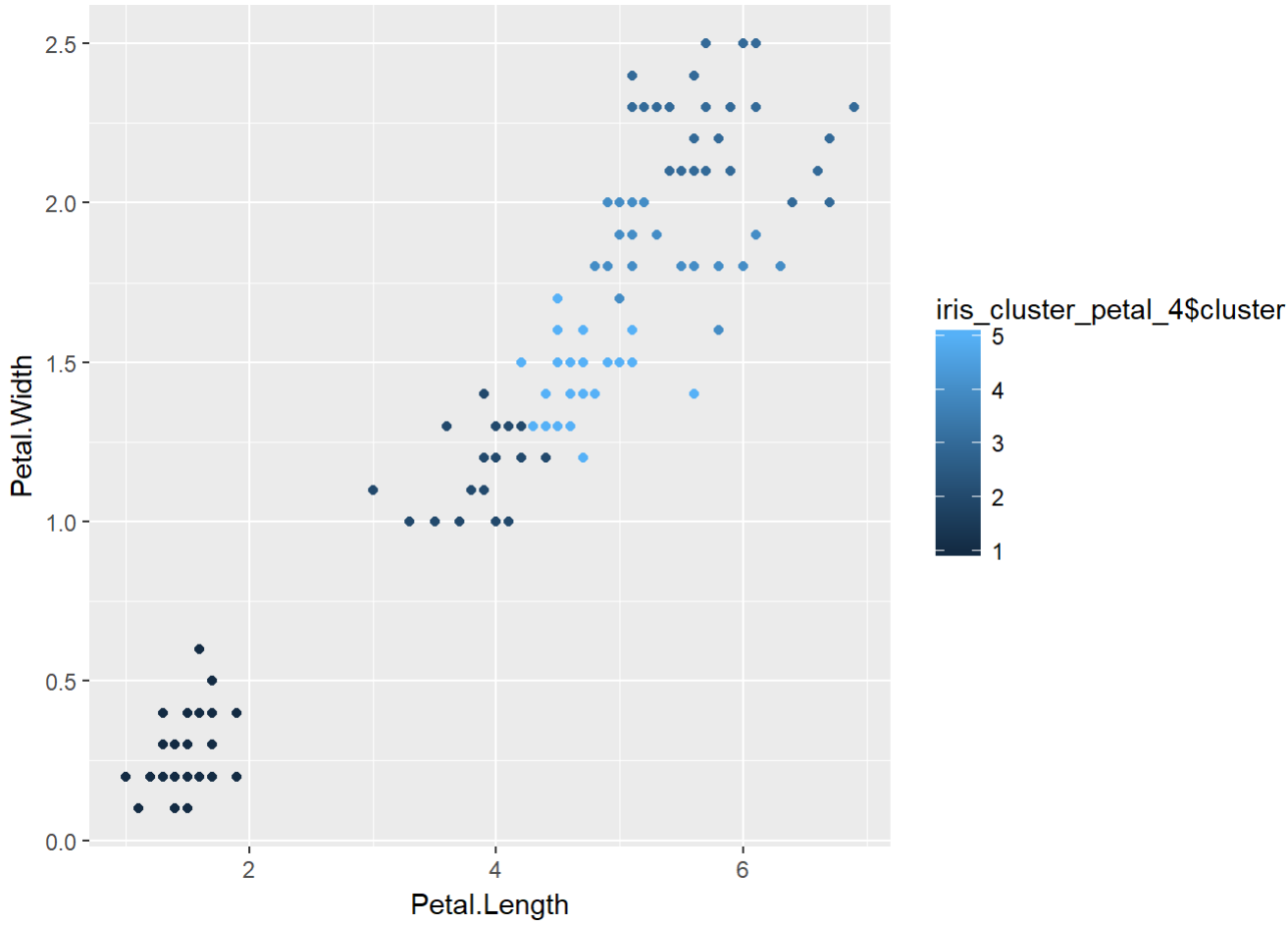
```
ggplot(iris, aes(Petal.Length, Petal.Width, color = iris_cluster_petal_2$cluster)) + geom_point
()
```




```
ggplot(iris, aes(Petal.Length, Petal.Width, color = iris_cluster_petal_3$cluster)) + geom_point
()
```




```
ggplot(iris, aes(Petal.Length, Petal.Width, color = iris_cluster_petal_4$cluster)) + geom_point
()
```


Petal.Width
Petal.Length
iris_cluster_petal_4$cluster
0.0
0.5
1.0
1.5
2.0
2.5
2
4
6
5
4
3
2
1