Homework-2

Question 3.1a

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:

(a) using cross-validation (do this for the k-nearest-neighbors model; SVM is optional); and

(b) splitting the data into training, validation, and test data sets (pick either KNN or SVM; the other is optional).

Import Packages

```
library(kernlab)
library(kknn)

set.seed(42069)
```

Load Data

```
data <- read.table("C:/Users/Admin/Desktop/MM/Homework 2/credit_card_data.txt")

head(data)
```

```
##   V1    V2    V3   V4 V5 V6 V7 V8  V9 V10 V11
## 1  1 30.83 0.000 1.25  1  0  1  1 202   0   1
## 2  0 58.67 4.460 3.04  1  0  6  1  43 560   1
## 3  0 24.50 0.500 1.50  1  1  0  1 280 824   1
## 4  1 27.83 1.540 3.75  1  0  5  0 100   3   1
## 5  1 20.17 5.625 1.71  1  1  0  1 120   0   1
## 6  1 32.08 4.000 2.50  1  1  0  0 360   0   1
```

Cross validation data split 80:20

```
train.idx <- sample(nrow(data), size=nrow(data)*0.8)
train <- data[train.idx,]
test <- data[-train.idx,]
```

Leave-one-out cross validation method

```
kmax = 50 # number of k values across validation
kknn.loocv <- train.kknn(V11~.,
                         train,
                         kmax=kmax,
```

```
                          scale=TRUE)
acc.list <- list()
for (k in 1:kmax){
  predicted <- round(fitted(kknn.loocv)[[k]]) # returned predictions
  acc.list[k] <- sum(predicted == train[,11])/nrow(train)*100
}
```

```
## [1] "Highest accuracy achieved in validation: 86.0420650095602 %"
```

```
## [1] "Corresponding k-value validation: 5"
```

Prediction on test set

```
kknn.loocv <- train.kknn(V11~.,
                         train,
                         ks=which.max(acc.list),
                         scale=TRUE)
predicted <- predict(kknn.loocv, test)
acc <- sum(round(predicted) == test[,11])/nrow(test)*100
```

```
## [1] "Accuracy on test set: 82.4427480916031 %"
```

k-fold cross validation method

```
acc.list <- list()
for (k in 1:kmax){
  kknn.kfold <- cv.kknn(V11~.,
                        train,
                        k=k,
                        kcv=5, # 5-fold cross validation
                        scale=TRUE)
  predicted <- as.integer(round(kknn.kfold[[1]][,2]))
  acc <- sum(predicted == train[,11])/nrow(train)*100
  acc.list[k] <- acc
}
```

```
## [1] "Highest accuracy achieved in validation: 86.6156787762906 %"
```

```
## [1] "Corresponding k-value validation: 19"
```

Prediction on test set

```
kknn.kfold <- kknn(V11~.,
                   train,
                   test,
                   k=which.max(acc.list),
                   scale=TRUE)
predicted = predict(kknn.kfold)
acc = sum(round(predicted) == test[,11])/nrow(test)*100
```

```
## [1] "Accuracy on test set: 84.7328244274809 %"
```

Question 3.1b

Import Packages

```
rm(list=ls())
library(kernlab)
library(kknn)

set.seed(42069)
```

Load Data

```
data <- read.table("C:/Users/Admin/Desktop/MM/Homework 2/credit_card_data.txt")

head(data)
```

```
##   V1    V2    V3   V4 V5 V6 V7 V8  V9 V10 V11
## 1  1 30.83 0.000 1.25  1  0  1  1 202   0   1
## 2  0 58.67 4.460 3.04  1  0  6  1  43 560   1
## 3  0 24.50 0.500 1.50  1  1  0  1 280 824   1
## 4  1 27.83 1.540 3.75  1  0  5  0 100   3   1
## 5  1 20.17 5.625 1.71  1  1  0  1 120   0   1
## 6  1 32.08 4.000 2.50  1  1  0  0 360   0   1
```

Train, Valid, Test split 70:20:10

```
train.idx = sample(nrow(data), size=nrow(data)*0.7)
train = data[train.idx,] # train data set 70%

valid.test = data[-train.idx,] # valid & test 30%

valid.idx = sample(nrow(valid.test), size=nrow(valid.test)/3*2)
valid = valid.test[valid.idx,] # valid 20%
test = valid.test[-valid.idx,] # test 10%

train.valid.idx = c(valid.idx, train.idx)
train.valid = data[train.valid.idx,] # train & valid data for testing
```

Selecting kknn model on train and validation set

```
acc.list <- list()
for (k in 1:50){
  kknn.model <- kknn(V11~.,
                     train,
                     valid,
                     k=k,
                     scale=TRUE)
  predicted <- round(fitted(kknn.model))
  acc <- sum(predicted==valid[,11])/nrow(valid)*100
  acc.list[k] <- acc
}
```

```
## [1] "Highest accuracy achieved in validation: 85.4961832061069 %"
```

```
## [1] "Corresponding k-value validation: 12"
```

Prediction on test data

```
kknn.model <- kknn(V11~.,
                   train,
                   test,
                   k=which.max(acc.list),
                   scale=TRUE)
predicted <- predict(kknn.model)
acc <- sum(round(predicted) == test[,11])/nrow(test)*100
```

```
## [1] "Accuracy on test set: 86.3636363636364 %"
```

Selecting ksvm model on train and validation set

```
c <- c(0.001, 0.01, 0.1, 1, 10, 100, 1000)
acc.list <- list()
for (i in 1:length(c)){
  ksvm.model <- ksvm(x=as.matrix(train[,1:10]),
                     y=as.factor(train[,11]),
                     type='C-svc',
                     kernel='rbfdot',
                     C=c[i], # c selection in loop
                     scaled=TRUE)
  predicted <- predict(ksvm.model, valid[,1:10]) # train on valid set
  acc <- sum(predicted==valid[,11])/nrow(valid)*100
  acc.list[i] <- acc
}
```

```
## [1] "Highest accuracy achieved in validation: 86.2595419847328 %"
```

```
## [1] "Corresponding c-value with highest accuracy: 0.1"
```

Prediction on test data

```
ksvm.model <- ksvm(x=as.matrix(train.valid[,1:10]),
                   y=as.factor(train.valid[,11]),
                   type='C-svc',
                   kernel='rbfdot',
                   C=c[which.max(acc.list)],
                   scaled=TRUE)
predicted <- predict(ksvm.model, test[,1:10])
acc <- sum(predicted==test[,11])/nrow(test)*100
```

```
## [1] "Accuracy on test set: 86.3636363636364 %"
```

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.

In marketing analytics, it is easy to build a model to generalize customer behavior. However, not all customers behave the same. In this particular situation of credit card sales for example, model performance will increase significantly if different types of customers are identified first, before the application of any form of purchase propensity model. To cluster customers based on their behavior, we can look at:

1. Customer credit score
2. Spending patterns or habits in different categories of purchase using transactional data
3. Net worth in their savings account
4. Demographic information such as gender, number of family members, supplementary card spend, location, and etc.
5. Customer responses to marketing advertisements, click through rate

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). ***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

Import Packages

```
rm(list=ls())
library(kernlab)
library(ggplot2)
```

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

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

```
set.seed(42069)
```

Load data

```
data = read.table("C:/Users/Admin/Desktop/MM/Homework 2/iris.txt")
head(data)
```

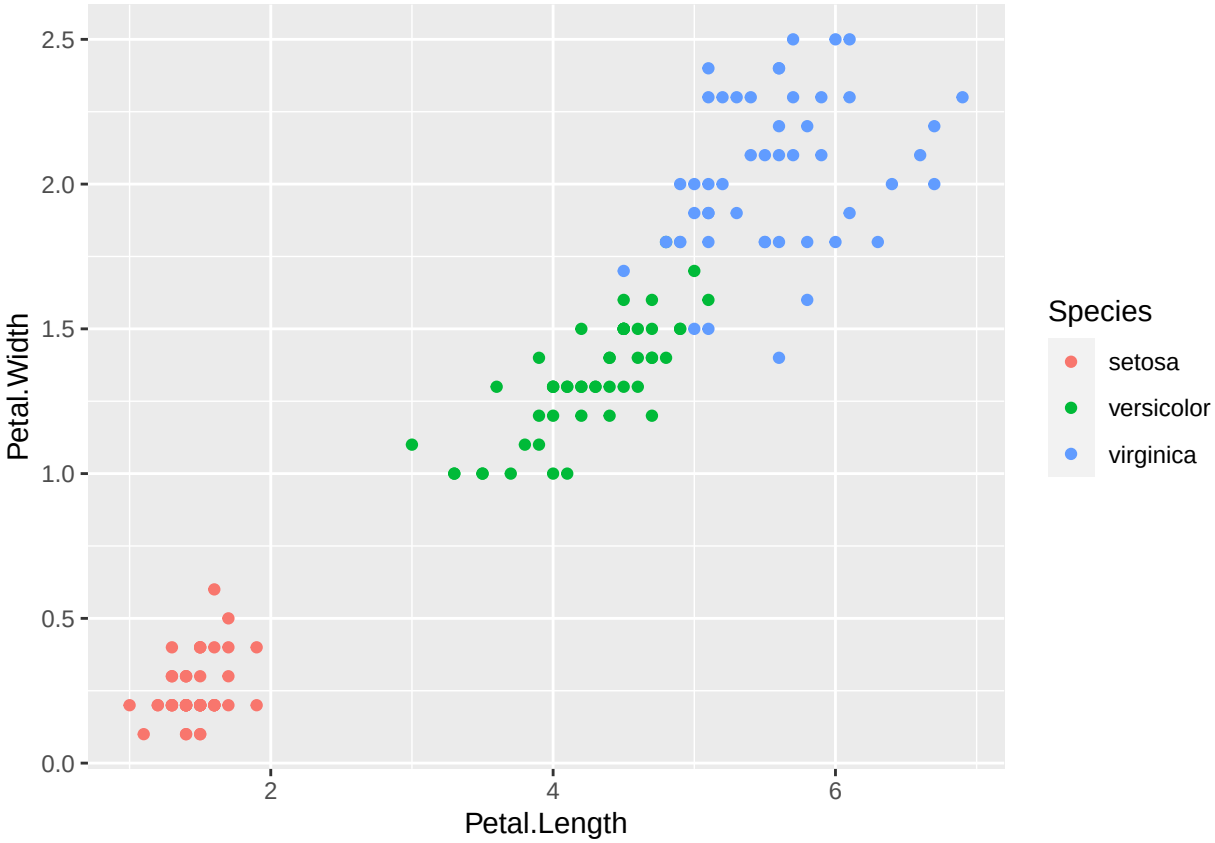
```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5  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

```
table(data[,5]) # tabulate species
```

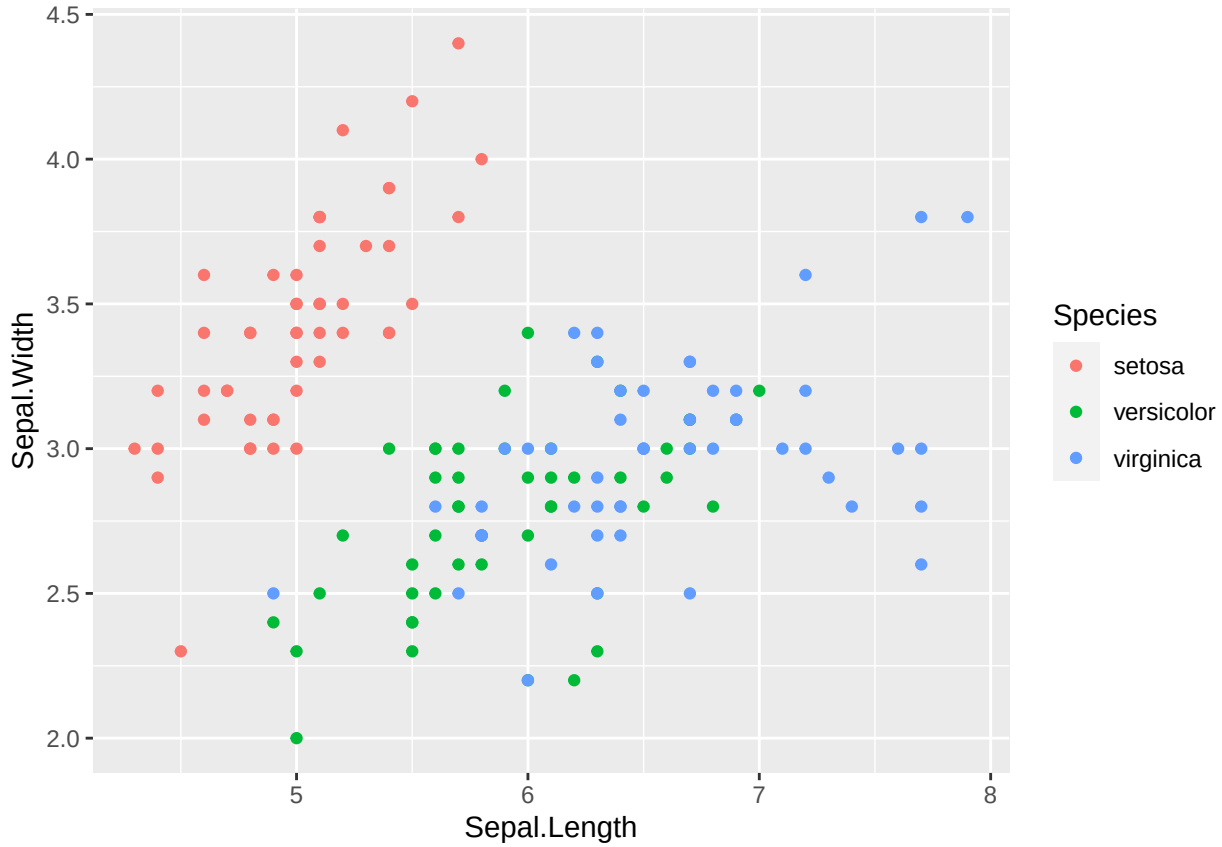
```
##
##     setosa versicolor  virginica
##         50         50         50
```

Visualize data

```
ggplot(data, aes(Petal.Length, Petal.Width, color = Species)) + geom_point()
```




```
ggplot(data, aes(Sepal.Length, Sepal.Width, color = Species)) + geom_point()
```



Scaling data: (x-xmin)/(xmax-xmin)

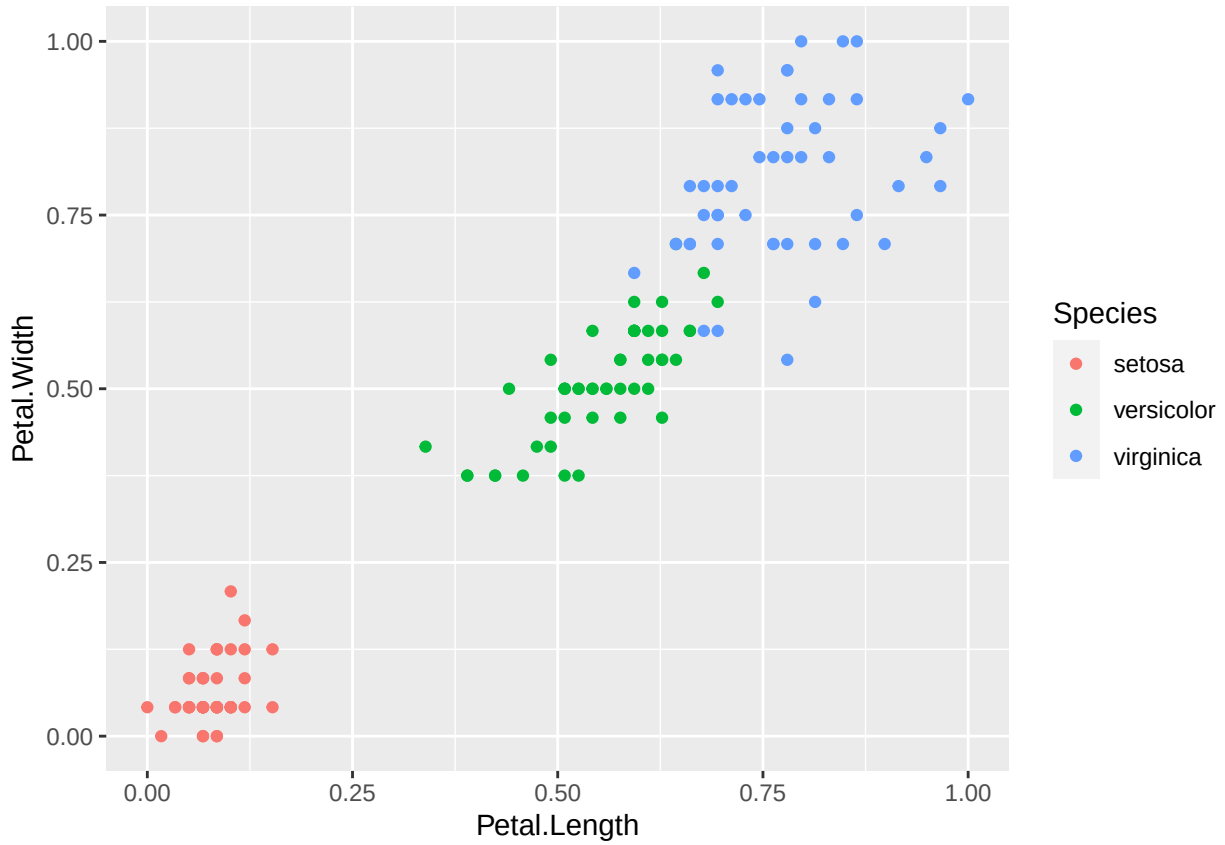
```
data.scaled <- data
for (i in 1:4){
  data.scaled[i] <- (data[i]-min(data[i]))/(max(data[i]-min(data[i])))
}

head(data.scaled)
```

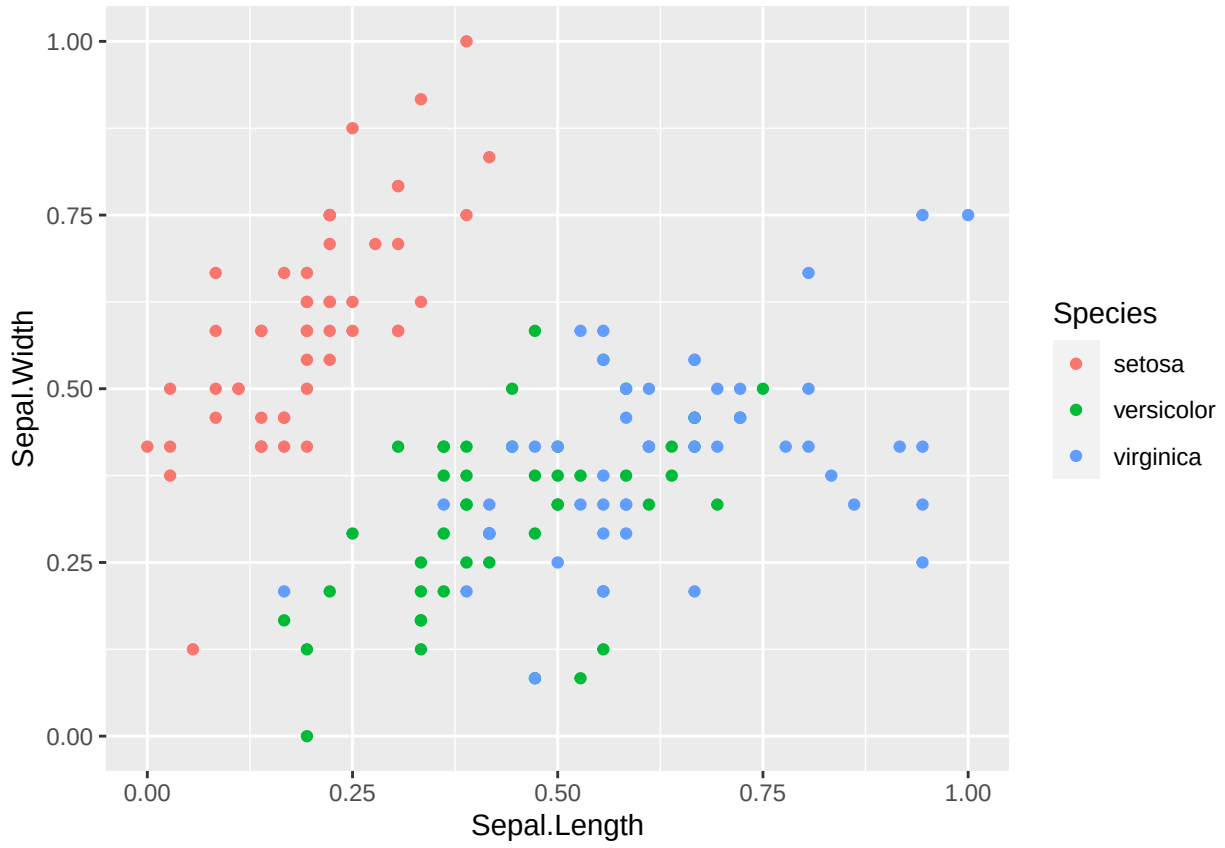
```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1   0.22222222   0.6250000   0.06779661  0.04166667  setosa
## 2   0.16666667   0.4166667   0.06779661  0.04166667  setosa
## 3   0.11111111   0.5000000   0.05084746  0.04166667  setosa
## 4   0.08333333   0.4583333   0.08474576  0.04166667  setosa
## 5   0.19444444   0.6666667   0.06779661  0.04166667  setosa
## 6   0.30555556   0.7916667   0.11864407  0.12500000  setosa
```

Visualize scaled data

```
ggplot(data.scaled, aes(Petal.Length, Petal.Width, color = Species)) + geom_point()
```




```
ggplot(data.scaled, aes(Sepal.Length, Sepal.Width, color = Species)) + geom_point()
```



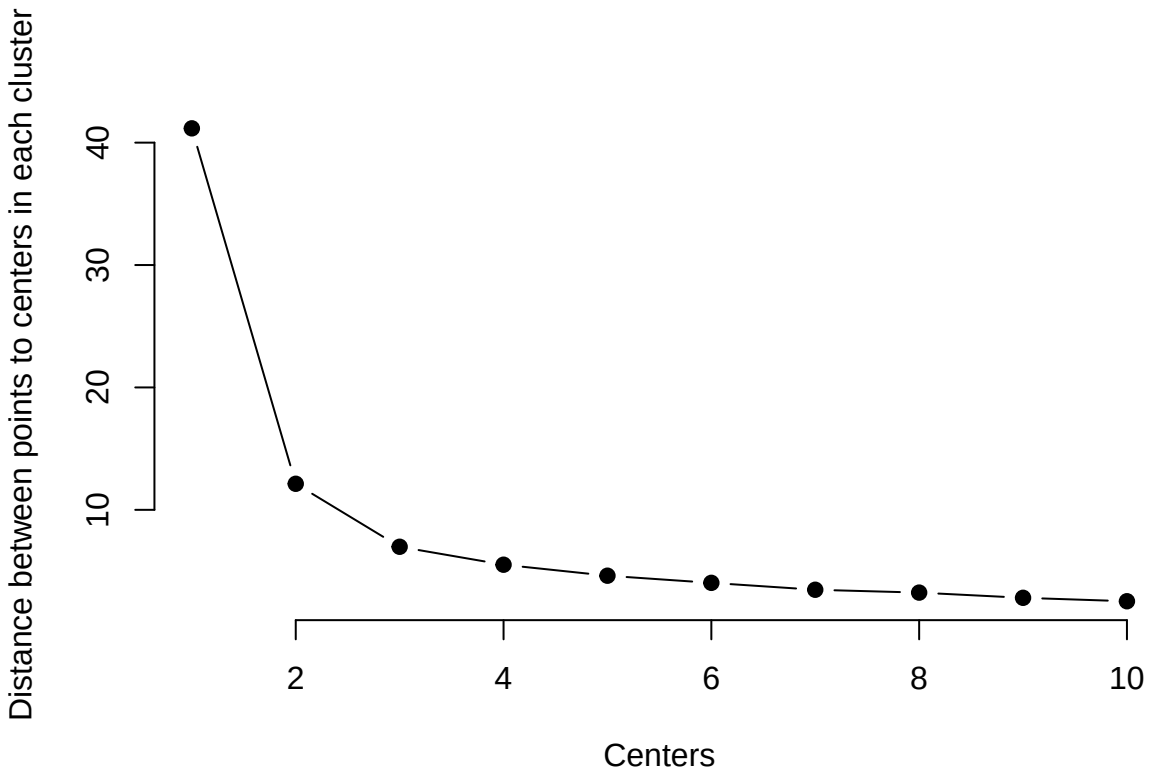
Center selection

```
c <- c(1:10)
dist <- list()

for (i in 1:length(c)){
  model <- kmeans(as.matrix(data.scaled[,1:4]),
                  centers=i,
                  iter.max=10,
                  nstart=10)
  dist[i] <- model$tot.withinss
}
```

Plot elbow chart to find optimal number of clusters

```
plot(c,
     dist,
     pch=19,
     frame=FALSE,
     type='b',
     xlab='Centers',
     ylab='Distance between points to centers in each cluster')
```



Overall distance between points to center begins to plateau at c=3.

Model with 3 centers

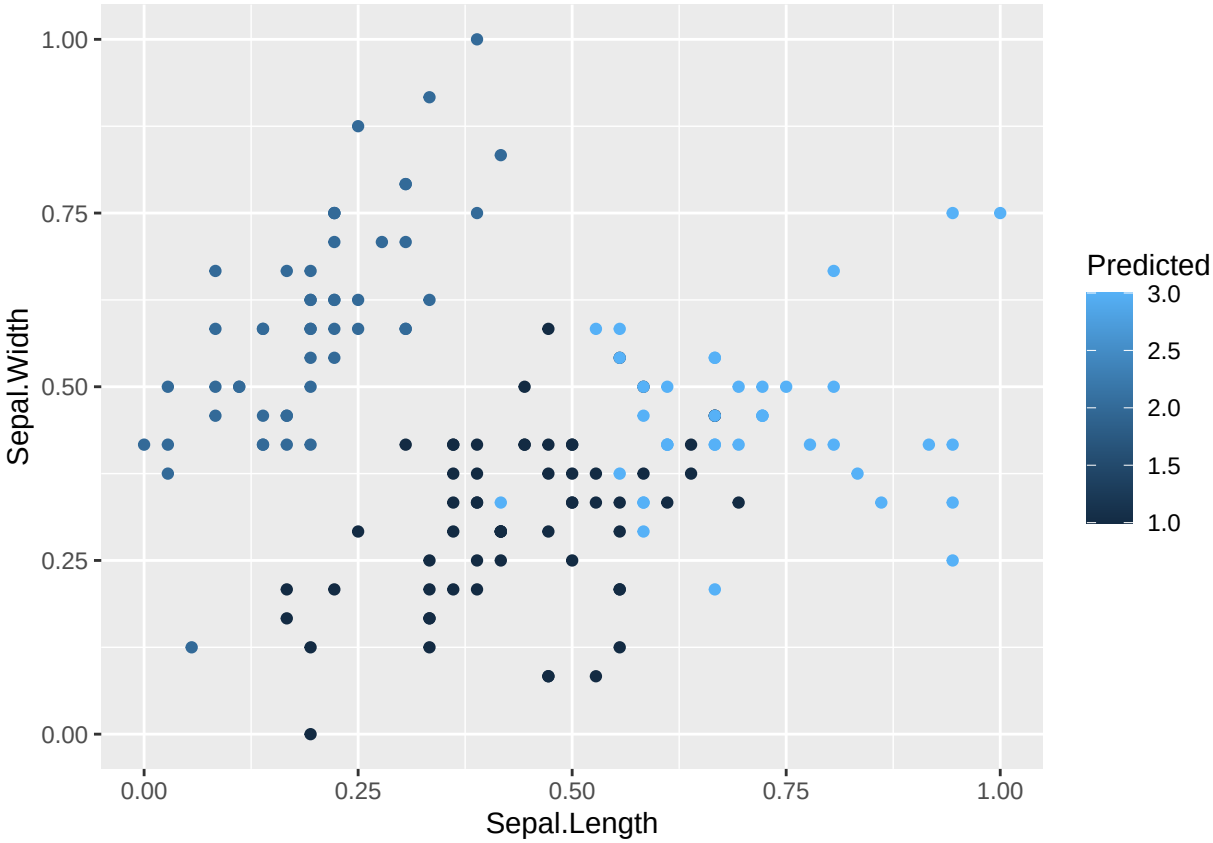
```
model <- kmeans(as.matrix(data.scaled[,1:4]),
                centers=3,
                iter.max=10,
                nstart=10)

data.scaled[6] <- model$cluster
colnames(data.scaled)[6] <- 'Predicted'
```

Visualize prediction

```
ggplot(data.scaled, aes(Petal.Length, Petal.Width, color = Predicted)) + geom_point()
```


```
ggplot(data.scaled, aes(Sepal.Length, Sepal.Width, color = Predicted)) + geom_point()
```


1.00
0.75
0.50
0.25
0.00
Sepal.Width
0.00
0.25
0.50
0.75
1.00
Sepal.Length
Predicted
3.0
2.5
2.0
1.5
1.0