```r
library(ggplot2)
library(class)
## Call the NaiveBayes Classifier Package e1071, which auto calls the Class package ##
library("e1071")

#Train classifier
classifier<-naiveBayes(iris[,1:4], iris[,5])

# evaluate classification
table(predict(classifier, iris[,-5]), iris[,5], dnn=list('predicted','actual'))
```

```
##             actual
## predicted    setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         47         3
##   virginica       0          3        47
```

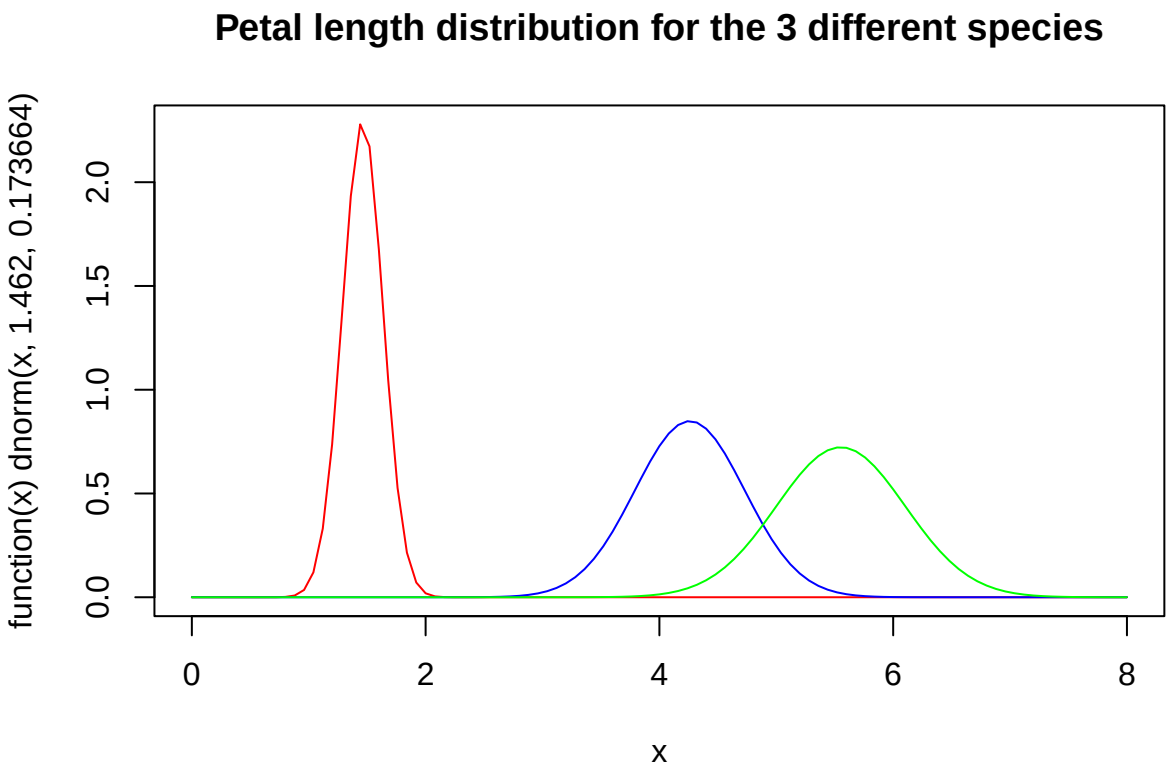
```r
# examine class means and standard deviations for petal length
classifier$tables$Petal.Length
```

```
##             Petal.Length
## iris[, 5]      [,1]      [,2]
##   setosa     1.462 0.1736640
##   versicolor 4.260 0.4699110
##   virginica  5.552 0.5518947
```

```r
# plot normal distributions at the means of the classes
# one class
plot(function(x) dnorm(x, 1.462, 0.1736640), 0, 8, col="red", main="Petal length distribution for the 3

# another class
curve(dnorm(x, 4.260, 0.4699110), add=TRUE, col="blue")

# the final class
curve(dnorm(x, 5.552, 0.5518947 ), add=TRUE, col = "green")
```



EXCERSISE 1

Repeat the naive bayes analysis using the abalone datase

```
# reading in abalone data
abalone <- read.csv(url("https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data
                    sep = ",")
# rename columns
colnames(abalone) <- c("sex", "length", 'diameter', 'height', 'whole_weight', 'shucked_weight', 'viscera
                       'rings' )

#Train classifier
classifier.a<-naiveBayes(abalone[,-9], abalone[,9])

# evaluate classification
table(predict(classifier.a, abalone[,-9]), abalone[,9], dnn=list('predicted','actual'))
```

```
##           actual
## predicted   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
##         1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         3   1   1  13  24  11   3   1   0   0   0   0   0   0   0   0   0   0
##         4   0   0   2  25  33  16   6   1   0   0   0   0   0   0   0   0   0
##         5   0   0   0   7  52  78  56  19  11   7   4   0   1   0   0   0   0
##         6   0   0   0   1  16  96 112  60  33  23   7   5   2   1   0   0   0
```

```
##         7    0   0   0   0   3  47 136 151  96  52  34  22  17   6   6   5   0
##         8    0   0   0   0   0  13  50 140 124  74  51  40  24  20  17   4   5
##         9    0   0   0   0   0   4  22 136 193 156  94  48  56  31  20  17  19
##         10   0   0   0   0   0   1   5  37 113 121  64  40  29  26  32  11   9
##         11   0   0   0   0   0   1   3  23 115 190 216  93  63  31  24  20  15
##         12   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         13   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         14   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         15   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
##         16   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         17   0   0   0   0   0   0   0   0   0   1   1   0   2   2   1   2   4
##         18   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         19   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         20   0   0   0   0   0   0   0   0   0   0   0   2   2   0   1   3   1
##         21   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         22   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         23   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         24   0   0   0   0   0   0   0   1   2   1   4   2   1   1   1   0   0
##         25   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         26   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         27   0   0   0   0   0   0   0   0   2   9  12  14   6   8   1   5   5
##         29   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##           actual
## predicted  18  19  20  21  22  23  24  25  26  27  29
##         1   0   0   0   0   0   0   0   0   0   0   0
##         2   0   0   0   0   0   0   0   0   0   0   0
##         3   0   0   0   0   0   0   0   0   0   0   0
##         4   0   0   0   0   0   0   0   0   0   0   0
##         5   0   0   0   0   0   0   0   0   0   0   0
##         6   0   0   0   0   0   0   0   0   0   0   0
##         7   3   1   0   0   0   0   0   0   0   0   0
##         8   3   2   2   2   1   0   0   0   0   0   0
##         9   7  10   3   2   2   4   0   0   0   0   0
##         10  8   7   9   1   0   1   0   0   1   0   0
##         11 16  12   6   5   1   3   0   0   0   0   1
##         12  0   0   0   0   0   0   0   0   0   0   0
##         13  0   0   0   0   0   0   0   0   0   0   0
##         14  0   0   0   0   0   0   0   0   0   0   0
##         15  0   0   0   0   0   0   0   0   0   0   0
##         16  0   0   0   0   0   0   0   0   0   1   0
##         17  2   0   3   2   2   0   0   1   0   0   0
##         18  0   0   0   0   0   0   0   0   0   0   0
##         19  0   0   0   0   0   0   0   0   0   0   0
##         20  1   0   1   2   0   0   0   0   0   0   0
##         21  0   0   0   0   0   0   0   0   0   0   0
##         22  0   0   0   0   0   0   0   0   0   0   0
##         23  0   0   0   0   0   0   0   0   0   0   0
##         24  1   0   0   0   0   0   2   0   0   0   0
##         25  0   0   0   0   0   0   0   0   0   0   0
##         26  0   0   0   0   0   0   0   0   0   0   0
##         27  1   0   2   0   0   1   0   0   0   1   0
##         29  0   0   0   0   0   0   0   0   0   0   0
```

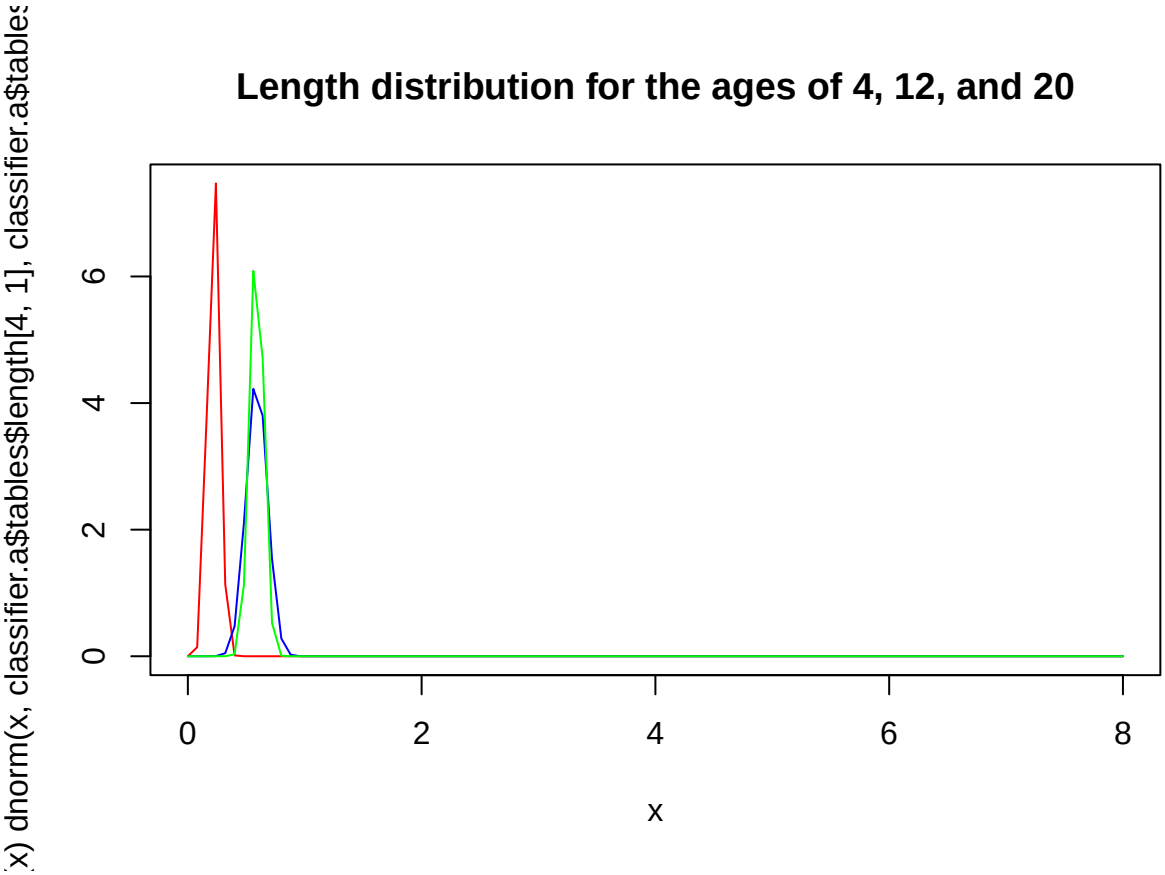
```r
# examine class means and standard deviations for the length
classifier.a$tables$length
```

```
##             length
## abalone[, 9]      [,1]        [,2]
##            1  0.0750000          NA
##            2  0.1500000          NA
##            3  0.1760000 0.034754239
##            4  0.2214912 0.049838712
##            5  0.2857391 0.060658823
##            6  0.3693629 0.075214677
##            7  0.4220332 0.076018024
##            8  0.4987764 0.079221016
##            9  0.5468650 0.082891324
##            10 0.5746293 0.085484290
##            11 0.5993737 0.086864494
##            12 0.5894569 0.089453148
##            13 0.5788916 0.085462505
##            14 0.5801984 0.079890928
##            15 0.5757282 0.069006989
##            16 0.5875373 0.077172441
##            17 0.6010345 0.069627824
##            18 0.5960714 0.072981251
##            19 0.5956250 0.067903513
##            20 0.6036538 0.057437926
##            21 0.6182143 0.075182014
##            22 0.5950000 0.054313902
##            23 0.5872222 0.102868579
##            24 0.6950000 0.007071068
##            25 0.6450000          NA
##            26 0.6000000          NA
##            27 0.6075000 0.081317280
##            29 0.7000000          NA
```

```r
# plot normal distributions at the means of 3 of the classes
# one class
plot(function(x) dnorm(x, classifier.a$tables$length[4,1], classifier.a$tables$length[4,2]), 0, 8, col=

# another class
curve(dnorm(x, classifier.a$tables$length[12,1], classifier.a$tables$length[12, 2]), add=TRUE, col="blu

# the final class
curve(dnorm(x, classifier.a$tables$length[12,1], classifier.a$tables$length[20, 2]), add=TRUE, col = "g
```



Try 3 different subsets of features not just all features at once

```
# Feature subset 1:

#Train classifier
classifier.a.s1<-naiveBayes(abalone[, c(-9, -1, -3, -4)], abalone[,9])

# evaluate classification
table(predict(classifier.a.s1, abalone[,-9]), abalone[,9], dnn=list('predicted','actual'))
```

```
##           actual
## predicted   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
##         1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         3   1   1  13  26  11   2   1   0   0   0   0   0   0   0   0   0   0
##         4   0   0   2  23  37  22   6   3   0   0   0   0   0   0   0   0   0
##         5   0   0   0   8  51  79  64  18  16   8   5   1   1   0   0   0   0
##         6   0   0   0   0  12  93 120  67  35  32  10   8   3   1   1   0   0
##         7   0   0   0   0   4  46 115 132  88  47  31  20  19   7   5   6   1
##         8   0   0   0   0   0  11  65 183 156  93  66  54  33  26  23   6   7
##         9   0   0   0   0   0   4  13 109 179 147  85  38  45  28  23  17  17
##         10  0   0   0   0   0   1   5  40 124 135  89  51  40  27  24  13   9
##         11  0   0   0   0   0   1   2  16  89 166 190  82  56  31  25  19  17
##         12  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         13  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         14  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

```
##        15   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##        16   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##        17   0   0   0   0   0   0   0   0   0   0   1   1   0   0   0   4   1
##        18   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##        19   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##        20   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##        21   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##        22   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##        23   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##        24   0   0   0   0   0   0   0   0   0   3   3   2   2   2   2   1   2
##        25   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##        26   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##        27   0   0   0   0   0   0   0   0   2   3   7  10   4   4   0   1   4
##        29   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##          actual
## predicted  18  19  20  21  22  23  24  25  26  27  29
##        1    0   0   0   0   0   0   0   0   0   0   0
##        2    0   0   0   0   0   0   0   0   0   0   0
##        3    0   0   0   0   0   0   0   0   0   0   0
##        4    0   0   0   0   0   0   0   0   0   0   0
##        5    0   0   0   0   0   0   0   0   0   0   0
##        6    0   0   0   0   0   0   0   0   0   0   0
##        7    3   1   0   0   0   0   0   0   0   0   0
##        8    3   4   2   2   2   2   0   0   0   0   0
##        9   10  12   4   2   1   2   0   0   0   0   0
##        10   6   5   9   1   0   2   0   0   1   0   0
##        11  16  10   9   7   3   2   0   1   0   1   0
##        12   0   0   0   0   0   0   0   0   0   0   0
##        13   0   0   0   0   0   0   0   0   0   0   0
##        14   0   0   0   0   0   0   0   0   0   0   0
##        15   0   0   0   0   0   0   0   0   0   0   0
##        16   0   0   0   1   0   0   0   0   0   0   0
##        17   2   0   0   1   0   0   0   0   0   0   0
##        18   0   0   0   0   0   0   0   0   0   0   0
##        19   0   0   0   0   0   0   0   0   0   0   0
##        20   0   0   0   0   0   0   0   0   0   0   0
##        21   0   0   0   0   0   0   0   0   0   0   0
##        22   0   0   0   0   0   0   0   0   0   0   0
##        23   0   0   0   0   0   0   0   0   0   0   0
##        24   1   0   1   0   0   0   2   0   0   0   1
##        25   0   0   0   0   0   0   0   0   0   0   0
##        26   0   0   0   0   0   0   0   0   0   0   0
##        27   1   0   1   0   0   1   0   0   0   1   0
##        29   0   0   0   0   0   0   0   0   0   0   0
```

```
# examine class means and standard deviations for the length
classifier.a.s1$tables$length
```

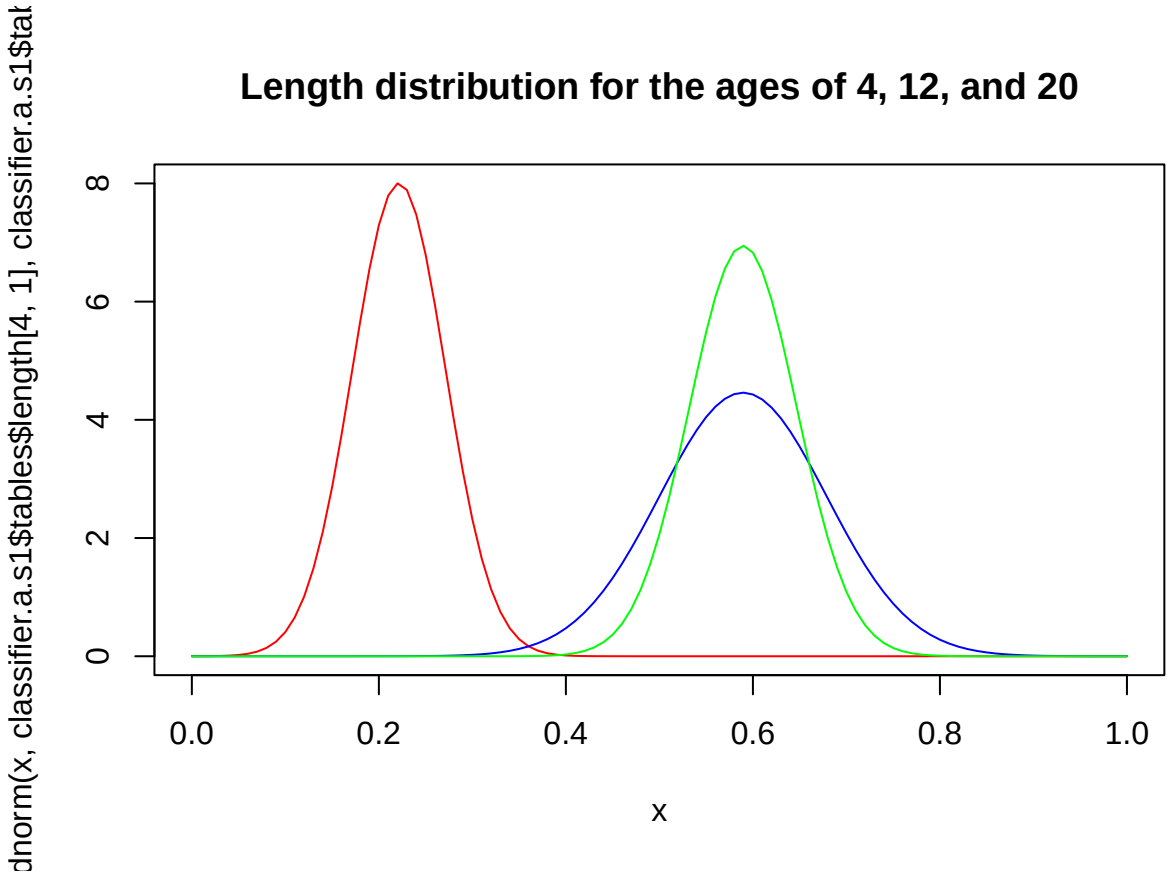
```
##             length
## abalone[, 9]      [,1]        [,2]
##            1  0.0750000          NA
##            2  0.1500000          NA
##            3  0.1760000 0.034754239
##            4  0.2214912 0.049838712
##            5  0.2857391 0.060658823
```

```
##           6  0.3693629 0.075214677
##           7  0.4220332 0.076018024
##           8  0.4987764 0.079221016
##           9  0.5468650 0.082891324
##           10 0.5746293 0.085484290
##           11 0.5993737 0.086864494
##           12 0.5894569 0.089453148
##           13 0.5788916 0.085462505
##           14 0.5801984 0.079890928
##           15 0.5757282 0.069006989
##           16 0.5875373 0.077172441
##           17 0.6010345 0.069627824
##           18 0.5960714 0.072981251
##           19 0.5956250 0.067903513
##           20 0.6036538 0.057437926
##           21 0.6182143 0.075182014
##           22 0.5950000 0.054313902
##           23 0.5872222 0.102868579
##           24 0.6950000 0.007071068
##           25 0.6450000          NA
##           26 0.6000000          NA
##           27 0.6075000 0.081317280
##           29 0.7000000          NA
```

```
# plot normal distributions at the means of 3 of the classes
# one class
plot(function(x) dnorm(x, classifier.a.s1$tables$length[4,1], classifier.a.s1$tables$length[4,2]), 0, 1

# another class
curve(dnorm(x, classifier.a.s1$tables$length[12,1], classifier.a.s1$tables$length[12, 2]), add=TRUE, col

# the final class
curve(dnorm(x, classifier.a.s1$tables$length[12,1], classifier.a.s1$tables$length[20, 2]), add=TRUE, col
```




```r
# Feature subset 2:

#Train classifier
classifier.a.s2<-naiveBayes(abalone[, c(-9, -8, -7, -6)], abalone[,9])

# evaluate classification
table(predict(classifier.a.s2, abalone[,-9]), abalone[,9], dnn=list('predicted','actual'))
```

```
##           actual
## predicted   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
##         1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         3   0   1  12  19  10   2   1   0   0   0   0   0   0   0   0   0   0
##         4   1   0   3  26  25  10   2   0   0   0   0   0   0   0   0   0   0
##         5   0   0   0  11  58  70  41  16   7   3   3   0   1   0   0   0   0
##         6   0   0   0   1  18  92 111  37  20  15   6   3   1   1   0   0   0
##         7   0   0   0   0   4  66 146 161 104  53  33  19  12   5   3   4   0
##         8   0   0   0   0   0   8  39 120  80  49  26  16  20   8   8   3   2
##         9   0   0   0   0   0   9  42 156 227 179 111  72  61  43  31  17  21
##         10  0   0   0   0   0   1   6  50 120 123  65  43  28  27  29  14   9
##         11  0   0   0   0   0   1   3  27 129 211 238 111  78  40  31  29  25
##         12  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         13  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         14  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         15  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         16  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

```
##        17   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##        18   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##        19   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##        20   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##        21   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##        22   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##        23   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##        24   0   0   0   0   0   0   0   1   2   1   4   2   2   1   1   0   0
##        25   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##        26   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##        27   0   0   0   0   0   0   0   0   0   0   0   1   1   0   1   0   0   1
##        29   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##          actual
## predicted  18  19  20  21  22  23  24  25  26  27  29
##        1    0   0   0   0   0   0   0   0   0   0   0
##        2    0   0   0   0   0   0   0   0   0   0   0
##        3    0   0   0   0   0   0   0   0   0   0   0
##        4    0   0   0   0   0   0   0   0   0   0   0
##        5    0   0   0   0   0   0   0   0   0   0   0
##        6    0   0   0   0   0   0   0   0   0   0   0
##        7    2   1   0   0   0   0   0   0   0   0   0
##        8    3   1   1   0   0   0   0   0   0   0   0
##        9    6  10   6   4   2   4   0   0   0   0   0
##        10   8   7   6   2   1   1   0   0   0   1   0
##        11  22  13  13   8   3   4   0   1   1   1   1
##        12   0   0   0   0   0   0   0   0   0   0   0
##        13   0   0   0   0   0   0   0   0   0   0   0
##        14   0   0   0   0   0   0   0   0   0   0   0
##        15   0   0   0   0   0   0   0   0   0   0   0
##        16   0   0   0   0   0   0   0   0   0   0   0
##        17   0   0   0   0   0   0   0   0   0   0   0
##        18   0   0   0   0   0   0   0   0   0   0   0
##        19   0   0   0   0   0   0   0   0   0   0   0
##        20   0   0   0   0   0   0   0   0   0   0   0
##        21   0   0   0   0   0   0   0   0   0   0   0
##        22   0   0   0   0   0   0   0   0   0   0   0
##        23   0   0   0   0   0   0   0   0   0   0   0
##        24   1   0   0   0   0   0   2   0   0   0   0
##        25   0   0   0   0   0   0   0   0   0   0   0
##        26   0   0   0   0   0   0   0   0   0   0   0
##        27   0   0   0   0   0   0   0   0   0   0   0
##        29   0   0   0   0   0   0   0   0   0   0   0
```

```
# examine class means and standard deviations for the length
classifier.a.s2$tables$length
```

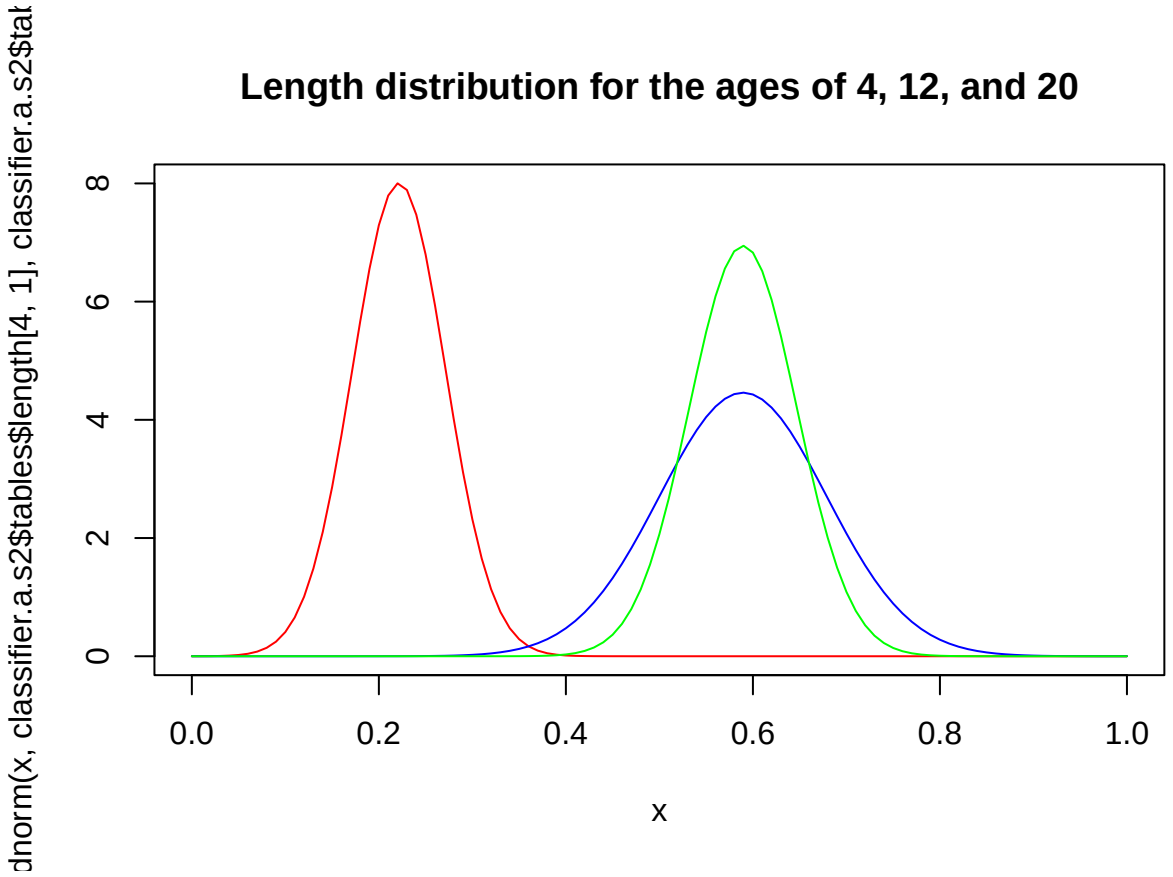
```
##             length
## abalone[, 9]      [,1]        [,2]
##            1  0.0750000          NA
##            2  0.1500000          NA
##            3  0.1760000 0.034754239
##            4  0.2214912 0.049838712
##            5  0.2857391 0.060658823
##            6  0.3693629 0.075214677
##            7  0.4220332 0.076018024
```

```
##           8  0.4987764 0.079221016
##           9  0.5468650 0.082891324
##           10 0.5746293 0.085484290
##           11 0.5993737 0.086864494
##           12 0.5894569 0.089453148
##           13 0.5788916 0.085462505
##           14 0.5801984 0.079890928
##           15 0.5757282 0.069006989
##           16 0.5875373 0.077172441
##           17 0.6010345 0.069627824
##           18 0.5960714 0.072981251
##           19 0.5956250 0.067903513
##           20 0.6036538 0.057437926
##           21 0.6182143 0.075182014
##           22 0.5950000 0.054313902
##           23 0.5872222 0.102868579
##           24 0.6950000 0.007071068
##           25 0.6450000          NA
##           26 0.6000000          NA
##           27 0.6075000 0.081317280
##           29 0.7000000          NA
```

```r
# plot normal distributions at the means of 3 of the classes
# one class
plot(function(x) dnorm(x, classifier.a.s2$tables$length[4,1], classifier.a.s2$tables$length[4,2]), 0, 1

# another class
curve(dnorm(x, classifier.a.s2$tables$length[12,1], classifier.a.s2$tables$length[12, 2]), add=TRUE, col

# the final class
curve(dnorm(x, classifier.a.s2$tables$length[12,1], classifier.a.s2$tables$length[20, 2]), add=TRUE, col
```




```
## Feature subset 3:

#Train classifier
classifier.a.s3<-naiveBayes(abalone[, c(-9, -8, -7, -6)], abalone[,9])

# evaluate classification
table(predict(classifier.a.s3, abalone[,-9]), abalone[,9], dnn=list('predicted','actual'))
```

```
##           actual
## predicted   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
##         1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         3   0   1  12  19  10   2   1   0   0   0   0   0   0   0   0   0   0
##         4   1   0   3  26  25  10   2   0   0   0   0   0   0   0   0   0   0
##         5   0   0   0  11  58  70  41  16   7   3   3   0   1   0   0   0   0
##         6   0   0   0   1  18  92 111  37  20  15   6   3   1   1   0   0   0
##         7   0   0   0   0   4  66 146 161 104  53  33  19  12   5   3   4   0
##         8   0   0   0   0   0   8  39 120  80  49  26  16  20   8   8   3   2
##         9   0   0   0   0   0   9  42 156 227 179 111  72  61  43  31  17  21
##         10  0   0   0   0   0   1   6  50 120 123  65  43  28  27  29  14   9
##         11  0   0   0   0   0   1   3  27 129 211 238 111  78  40  31  29  25
##         12  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         13  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         14  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         15  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         16  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

```
##         17   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         18   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         19   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         20   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         21   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         22   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         23   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         24   0   0   0   0   0   0   0   1   2   1   4   2   2   1   1   0   0
##         25   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         26   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         27   0   0   0   0   0   0   0   0   0   0   1   1   0   1   0   0   1
##         29   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##           actual
## predicted  18  19  20  21  22  23  24  25  26  27  29
##         1   0   0   0   0   0   0   0   0   0   0   0
##         2   0   0   0   0   0   0   0   0   0   0   0
##         3   0   0   0   0   0   0   0   0   0   0   0
##         4   0   0   0   0   0   0   0   0   0   0   0
##         5   0   0   0   0   0   0   0   0   0   0   0
##         6   0   0   0   0   0   0   0   0   0   0   0
##         7   2   1   0   0   0   0   0   0   0   0   0
##         8   3   1   1   0   0   0   0   0   0   0   0
##         9   6  10   6   4   2   4   0   0   0   0   0
##         10  8   7   6   2   1   1   0   0   0   1   0
##         11 22  13  13   8   3   4   0   1   1   1   1
##         12  0   0   0   0   0   0   0   0   0   0   0
##         13  0   0   0   0   0   0   0   0   0   0   0
##         14  0   0   0   0   0   0   0   0   0   0   0
##         15  0   0   0   0   0   0   0   0   0   0   0
##         16  0   0   0   0   0   0   0   0   0   0   0
##         17  0   0   0   0   0   0   0   0   0   0   0
##         18  0   0   0   0   0   0   0   0   0   0   0
##         19  0   0   0   0   0   0   0   0   0   0   0
##         20  0   0   0   0   0   0   0   0   0   0   0
##         21  0   0   0   0   0   0   0   0   0   0   0
##         22  0   0   0   0   0   0   0   0   0   0   0
##         23  0   0   0   0   0   0   0   0   0   0   0
##         24  1   0   0   0   0   0   2   0   0   0   0
##         25  0   0   0   0   0   0   0   0   0   0   0
##         26  0   0   0   0   0   0   0   0   0   0   0
##         27  0   0   0   0   0   0   0   0   0   0   0
##         29  0   0   0   0   0   0   0   0   0   0   0
```

```
# examine class means and standard deviations for the length
classifier.a.s3$tables$length
```

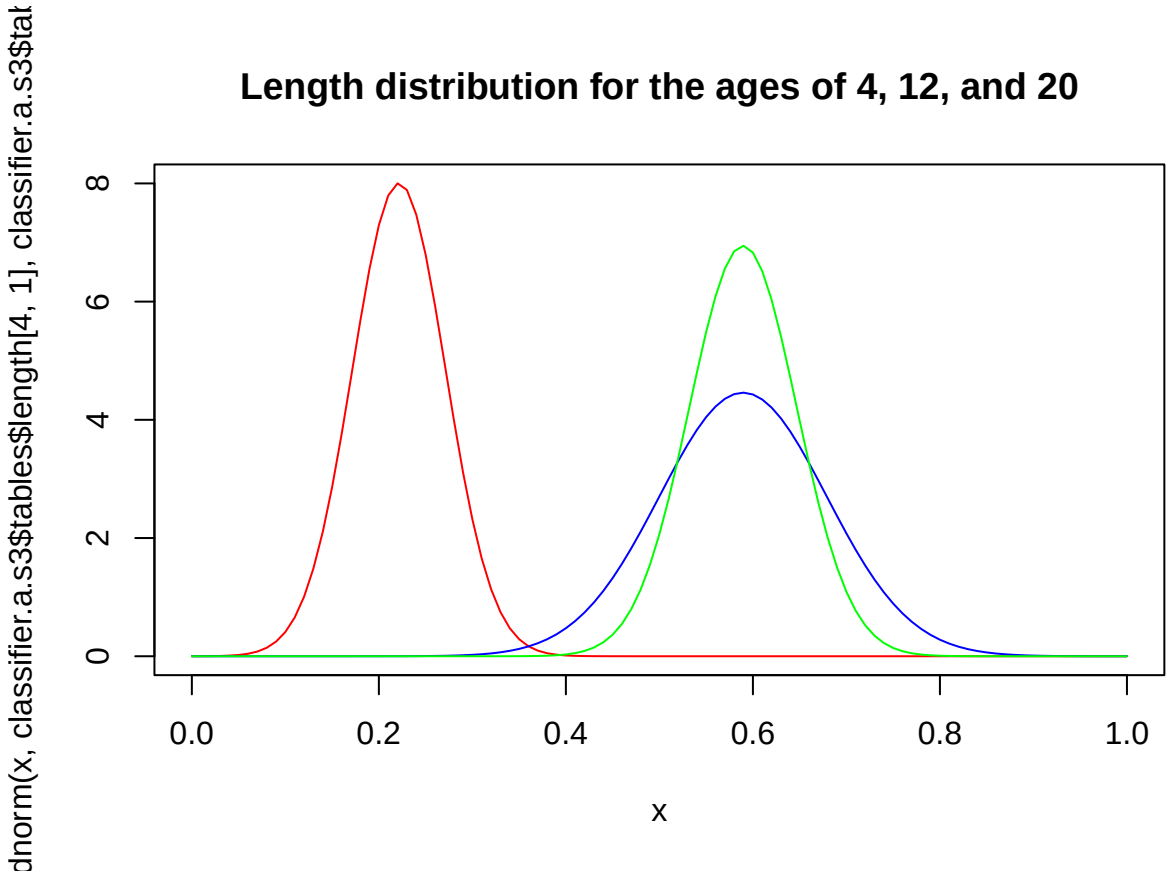
```
##             length
## abalone[, 9]      [,1]        [,2]
##            1  0.0750000          NA
##            2  0.1500000          NA
##            3  0.1760000 0.034754239
##            4  0.2214912 0.049838712
##            5  0.2857391 0.060658823
##            6  0.3693629 0.075214677
##            7  0.4220332 0.076018024
```

```
##           8  0.4987764 0.079221016
##           9  0.5468650 0.082891324
##           10 0.5746293 0.085484290
##           11 0.5993737 0.086864494
##           12 0.5894569 0.089453148
##           13 0.5788916 0.085462505
##           14 0.5801984 0.079890928
##           15 0.5757282 0.069006989
##           16 0.5875373 0.077172441
##           17 0.6010345 0.069627824
##           18 0.5960714 0.072981251
##           19 0.5956250 0.067903513
##           20 0.6036538 0.057437926
##           21 0.6182143 0.075182014
##           22 0.5950000 0.054313902
##           23 0.5872222 0.102868579
##           24 0.6950000 0.007071068
##           25 0.6450000          NA
##           26 0.6000000          NA
##           27 0.6075000 0.081317280
##           29 0.7000000          NA
```

```r
# plot normal distributions at the means of 3 of the classes
# one class
plot(function(x) dnorm(x, classifier.a.s3$tables$length[4,1], classifier.a.s3$tables$length[4,2]), 0, 1

# another class
curve(dnorm(x, classifier.a.s3$tables$length[12,1], classifier.a.s3$tables$length[12, 2]), add=TRUE, col

# the final class
curve(dnorm(x, classifier.a.s3$tables$length[12,1], classifier.a.s3$tables$length[20, 2]), add=TRUE, col
```




```r
# Feature subset 3:

#Train classifier
classifier.a.s3<-naiveBayes(abalone[, c(-9, -5, -1, -8)], abalone[,9])

# evaluate classification
table(predict(classifier.a.s3, abalone[,-9]), abalone[,9], dnn=list('predicted','actual'))
```

```
##           actual
## predicted   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
##         1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         3   0   1  12  22  10   1   1   0   0   0   0   0   0   0   0   0   0
##         4   1   0   3  25  31  17   5   2   0   0   0   0   0   0   0   0   0
##         5   0   0   0   9  52  74  42  14   9   4   5   0   1   0   0   0   0
##         6   0   0   0   1  18  89 117  56  36  27   8   6   2   1   0   0   0
##         7   0   0   0   0   4  60 143 146  95  53  33  22  17   8   6   5   2
##         8   0   0   0   0   0  11  43 125  95  55  43  37  28  18  18   7   4
##         9   0   0   0   0   0   5  32 157 222 176 102  54  54  39  25  15  18
##         10  0   0   0   0   0   1   5  43 111 112  67  40  38  22  24  15  11
##         11  0   0   0   0   0   1   3  24 119 205 225 105  61  36  29  25  23
##         12  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         13  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         14  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         15  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         16  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

```
##         17   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         18   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         19   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         20   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         21   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         22   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         23   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         24   0   0   0   0   0   0   0   1   2   1   3   2   1   1   1   0   0
##         25   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         26   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         27   0   0   0   0   0   0   0   0   0   1   1   1   1   1   0   0   0
##         29   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##           actual
## predicted  18  19  20  21  22  23  24  25  26  27  29
##         1   0   0   0   0   0   0   0   0   0   0   0
##         2   0   0   0   0   0   0   0   0   0   0   0
##         3   0   0   0   0   0   0   0   0   0   0   0
##         4   0   0   0   0   0   0   0   0   0   0   0
##         5   0   0   0   0   0   0   0   0   0   0   0
##         6   0   0   0   0   0   0   0   0   0   0   0
##         7   3   1   0   0   0   0   0   0   0   0   0
##         8   3   3   2   2   1   3   0   0   0   0   0
##         9   9  11   8   3   2   2   0   0   0   0   0
##         10  7   6   6   1   0   1   0   1   1   1   0
##         11 20  11  10   8   3   3   0   0   0   1   1
##         12  0   0   0   0   0   0   0   0   0   0   0
##         13  0   0   0   0   0   0   0   0   0   0   0
##         14  0   0   0   0   0   0   0   0   0   0   0
##         15  0   0   0   0   0   0   0   0   0   0   0
##         16  0   0   0   0   0   0   0   0   0   0   0
##         17  0   0   0   0   0   0   0   0   0   0   0
##         18  0   0   0   0   0   0   0   0   0   0   0
##         19  0   0   0   0   0   0   0   0   0   0   0
##         20  0   0   0   0   0   0   0   0   0   0   0
##         21  0   0   0   0   0   0   0   0   0   0   0
##         22  0   0   0   0   0   0   0   0   0   0   0
##         23  0   0   0   0   0   0   0   0   0   0   0
##         24  0   0   0   0   0   0   2   0   0   0   0
##         25  0   0   0   0   0   0   0   0   0   0   0
##         26  0   0   0   0   0   0   0   0   0   0   0
##         27  0   0   0   0   0   0   0   0   0   0   0
##         29  0   0   0   0   0   0   0   0   0   0   0
```

```
# examine class means and standard deviations for the length
classifier.a.s3$tables$length
```

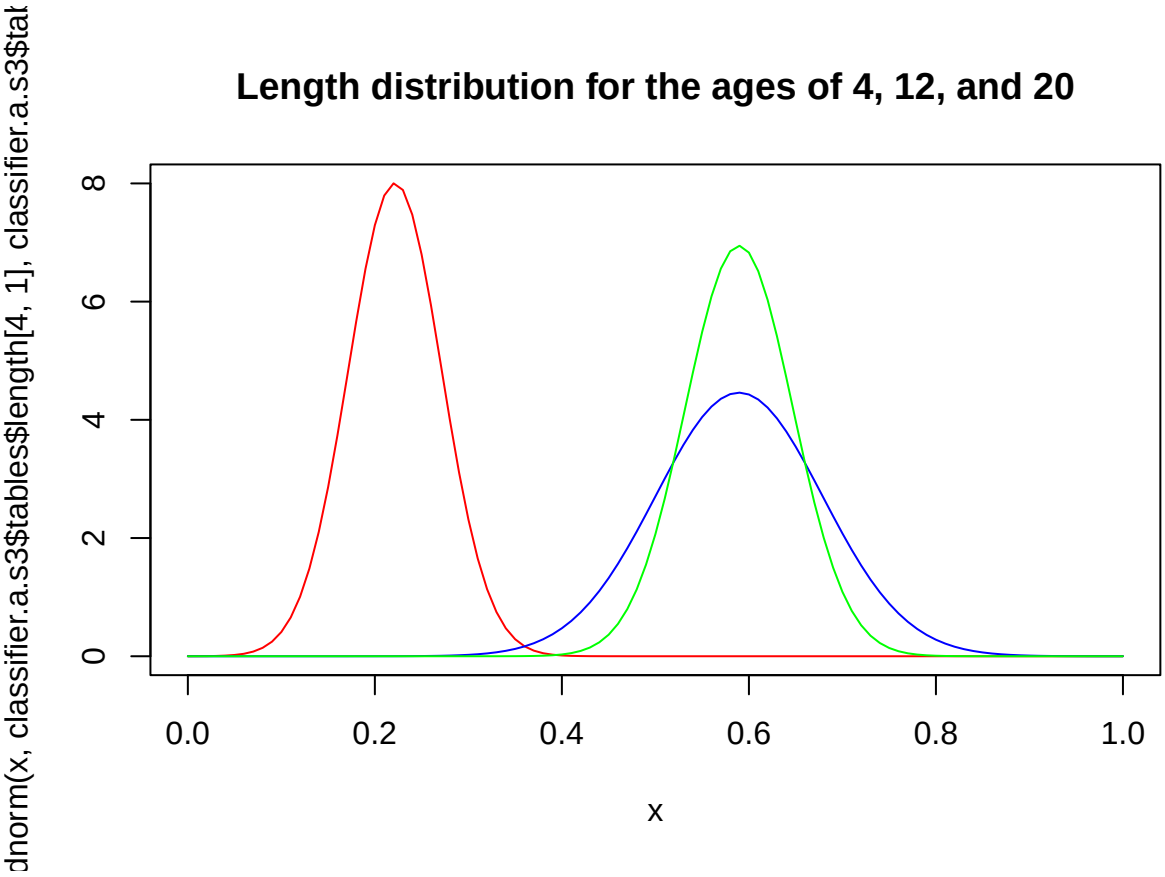
```
##              length
## abalone[, 9]      [,1]        [,2]
##            1  0.0750000          NA
##            2  0.1500000          NA
##            3  0.1760000 0.034754239
##            4  0.2214912 0.049838712
##            5  0.2857391 0.060658823
##            6  0.3693629 0.075214677
##            7  0.4220332 0.076018024
```

```
##           8  0.4987764 0.079221016
##           9  0.5468650 0.082891324
##           10 0.5746293 0.085484290
##           11 0.5993737 0.086864494
##           12 0.5894569 0.089453148
##           13 0.5788916 0.085462505
##           14 0.5801984 0.079890928
##           15 0.5757282 0.069006989
##           16 0.5875373 0.077172441
##           17 0.6010345 0.069627824
##           18 0.5960714 0.072981251
##           19 0.5956250 0.067903513
##           20 0.6036538 0.057437926
##           21 0.6182143 0.075182014
##           22 0.5950000 0.054313902
##           23 0.5872222 0.102868579
##           24 0.6950000 0.007071068
##           25 0.6450000          NA
##           26 0.6000000          NA
##           27 0.6075000 0.081317280
##           29 0.7000000          NA
```

```r
# plot normal distributions at the means of 3 of the classes
# one class
plot(function(x) dnorm(x, classifier.a.s3$tables$length[4,1], classifier.a.s3$tables$length[4,2]), 0, 1

# another class
curve(dnorm(x, classifier.a.s3$tables$length[12,1], classifier.a.s3$tables$length[12, 2]), add=TRUE, col

# the final class
curve(dnorm(x, classifier.a.s3$tables$length[12,1], classifier.a.s3$tables$length[20, 2]), add=TRUE, col
```



Compare the models using contingency tables

```
# Subset 1
table(predict(classifier.a.s1, abalone[,-9]), abalone[,9], dnn=list('predicted','actual'))
```

```
##           actual
## predicted   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
##         1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         3   1   1  13  26  11   2   1   0   0   0   0   0   0   0   0   0   0
##         4   0   0   2  23  37  22   6   3   0   0   0   0   0   0   0   0   0
##         5   0   0   0   8  51  79  64  18  16   8   5   1   1   0   0   0   0
##         6   0   0   0   0  12  93 120  67  35  32  10   8   3   1   1   0   0
##         7   0   0   0   0   4  46 115 132  88  47  31  20  19   7   5   6   1
##         8   0   0   0   0   0  11  65 183 156  93  66  54  33  26  23   6   7
##         9   0   0   0   0   0   4  13 109 179 147  85  38  45  28  23  17  17
##         10  0   0   0   0   0   1   5  40 124 135  89  51  40  27  24  13   9
##         11  0   0   0   0   0   1   2  16  89 166 190  82  56  31  25  19  17
##         12  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         13  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         14  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         15  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         16  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         17  0   0   0   0   0   0   0   0   0   0   1   1   0   0   0   4   1
##         18  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         19  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

```
##         20   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         21   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         22   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         23   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         24   0   0   0   0   0   0   0   0   0   3   3   2   2   2   2   1   2
##         25   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         26   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         27   0   0   0   0   0   0   0   0   2   3   7  10   4   4   0   1   4
##         29   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##           actual
## predicted  18  19  20  21  22  23  24  25  26  27  29
##         1   0   0   0   0   0   0   0   0   0   0   0
##         2   0   0   0   0   0   0   0   0   0   0   0
##         3   0   0   0   0   0   0   0   0   0   0   0
##         4   0   0   0   0   0   0   0   0   0   0   0
##         5   0   0   0   0   0   0   0   0   0   0   0
##         6   0   0   0   0   0   0   0   0   0   0   0
##         7   3   1   0   0   0   0   0   0   0   0   0
##         8   3   4   2   2   2   2   0   0   0   0   0
##         9  10  12   4   2   1   2   0   0   0   0   0
##         10  6   5   9   1   0   2   0   0   1   0   0
##         11 16  10   9   7   3   2   0   1   0   1   0
##         12  0   0   0   0   0   0   0   0   0   0   0
##         13  0   0   0   0   0   0   0   0   0   0   0
##         14  0   0   0   0   0   0   0   0   0   0   0
##         15  0   0   0   0   0   0   0   0   0   0   0
##         16  0   0   0   1   0   0   0   0   0   0   0
##         17  2   0   0   1   0   0   0   0   0   0   0
##         18  0   0   0   0   0   0   0   0   0   0   0
##         19  0   0   0   0   0   0   0   0   0   0   0
##         20  0   0   0   0   0   0   0   0   0   0   0
##         21  0   0   0   0   0   0   0   0   0   0   0
##         22  0   0   0   0   0   0   0   0   0   0   0
##         23  0   0   0   0   0   0   0   0   0   0   0
##         24  1   0   1   0   0   0   2   0   0   0   1
##         25  0   0   0   0   0   0   0   0   0   0   0
##         26  0   0   0   0   0   0   0   0   0   0   0
##         27  1   0   1   0   0   1   0   0   0   1   0
##         29  0   0   0   0   0   0   0   0   0   0   0
```

```
# Subset 2
table(predict(classifier.a.s2, abalone[,-9]), abalone[,9], dnn=list('predicted','actual'))
```

```
##           actual
## predicted   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
##         1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         3   0   1  12  19  10   2   1   0   0   0   0   0   0   0   0   0   0
##         4   1   0   3  26  25  10   2   0   0   0   0   0   0   0   0   0   0
##         5   0   0   0  11  58  70  41  16   7   3   3   0   1   0   0   0   0
##         6   0   0   0   1  18  92 111  37  20  15   6   3   1   1   0   0   0
##         7   0   0   0   0   4  66 146 161 104  53  33  19  12   5   3   4   0
##         8   0   0   0   0   0   8  39 120  80  49  26  16  20   8   8   3   2
##         9   0   0   0   0   0   9  42 156 227 179 111  72  61  43  31  17  21
##         10  0   0   0   0   0   1   6  50 120 123  65  43  28  27  29  14   9
```

```
##         11   0   0   0   0   0   1   3  27 129 211 238 111  78  40  31  29  25
##         12   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         13   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         14   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         15   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         16   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         17   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         18   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         19   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         20   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         21   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         22   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         23   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         24   0   0   0   0   0   0   0   1   2   1   4   2   2   1   1   0   0
##         25   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         26   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         27   0   0   0   0   0   0   0   0   0   0   1   1   0   1   0   0   1
##         29   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##           actual
## predicted  18  19  20  21  22  23  24  25  26  27  29
##         1   0   0   0   0   0   0   0   0   0   0   0
##         2   0   0   0   0   0   0   0   0   0   0   0
##         3   0   0   0   0   0   0   0   0   0   0   0
##         4   0   0   0   0   0   0   0   0   0   0   0
##         5   0   0   0   0   0   0   0   0   0   0   0
##         6   0   0   0   0   0   0   0   0   0   0   0
##         7   2   1   0   0   0   0   0   0   0   0   0
##         8   3   1   1   0   0   0   0   0   0   0   0
##         9   6  10   6   4   2   4   0   0   0   0   0
##         10  8   7   6   2   1   1   0   0   0   1   0
##         11 22  13  13   8   3   4   0   1   1   1   1
##         12  0   0   0   0   0   0   0   0   0   0   0
##         13  0   0   0   0   0   0   0   0   0   0   0
##         14  0   0   0   0   0   0   0   0   0   0   0
##         15  0   0   0   0   0   0   0   0   0   0   0
##         16  0   0   0   0   0   0   0   0   0   0   0
##         17  0   0   0   0   0   0   0   0   0   0   0
##         18  0   0   0   0   0   0   0   0   0   0   0
##         19  0   0   0   0   0   0   0   0   0   0   0
##         20  0   0   0   0   0   0   0   0   0   0   0
##         21  0   0   0   0   0   0   0   0   0   0   0
##         22  0   0   0   0   0   0   0   0   0   0   0
##         23  0   0   0   0   0   0   0   0   0   0   0
##         24  1   0   0   0   0   0   2   0   0   0   0
##         25  0   0   0   0   0   0   0   0   0   0   0
##         26  0   0   0   0   0   0   0   0   0   0   0
##         27  0   0   0   0   0   0   0   0   0   0   0
##         29  0   0   0   0   0   0   0   0   0   0   0
```

```
# Subset 3
table(predict(classifier.a.s3, abalone[,-9]), abalone[,9], dnn=list('predicted','actual'))
```

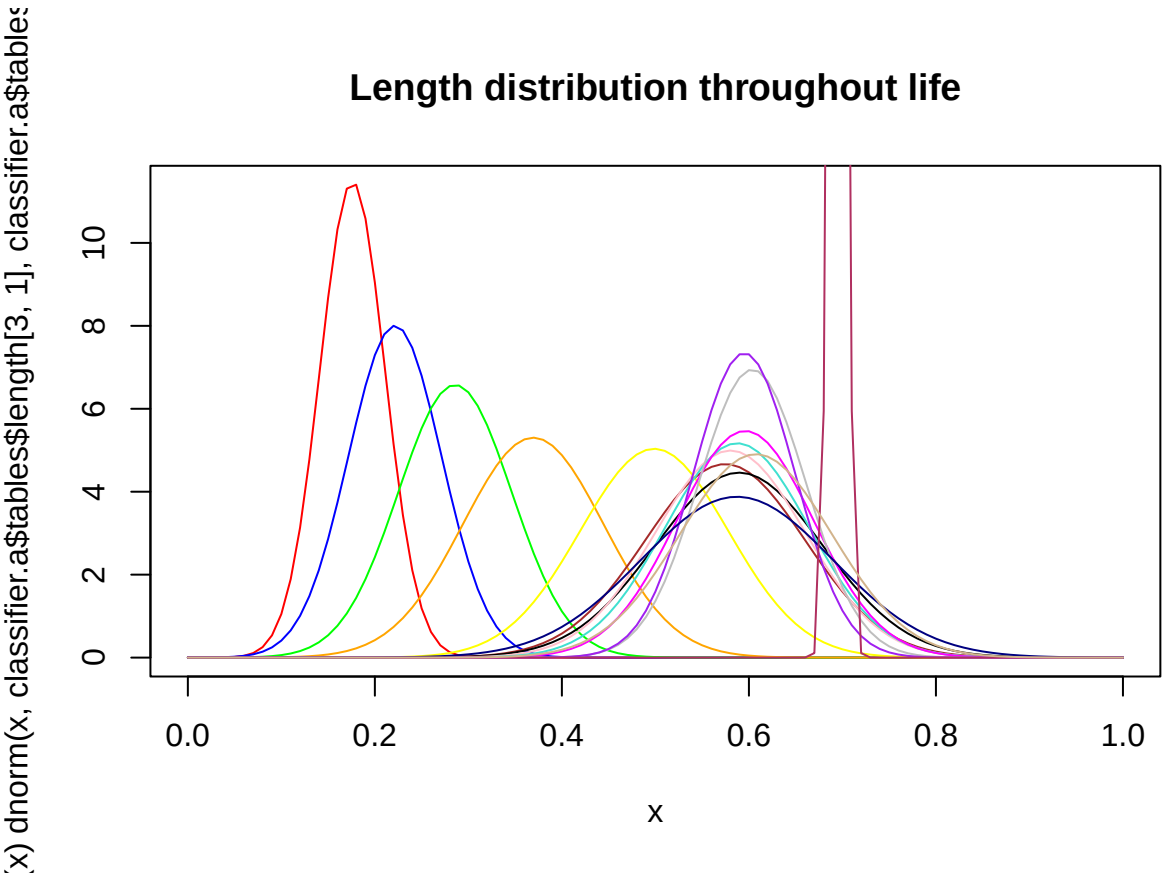
```
##           actual
## predicted   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
##         1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

```
##         2    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##         3    0    1   12   22   10    1    1    0    0    0    0    0    0    0    0    0    0
##         4    1    0    3   25   31   17    5    2    0    0    0    0    0    0    0    0    0
##         5    0    0    0    9   52   74   42   14    9    4    5    0    1    0    0    0    0
##         6    0    0    0    1   18   89  117   56   36   27    8    6    2    1    0    0    0
##         7    0    0    0    0    4   60  143  146   95   53   33   22   17    8    6    5    2
##         8    0    0    0    0    0   11   43  125   95   55   43   37   28   18   18    7    4
##         9    0    0    0    0    0    5   32  157  222  176  102   54   54   39   25   15   18
##         10   0    0    0    0    0    1    5   43  111  112   67   40   38   22   24   15   11
##         11   0    0    0    0    0    1    3   24  119  205  225  105   61   36   29   25   23
##         12   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##         13   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##         14   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##         15   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##         16   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##         17   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##         18   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##         19   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##         20   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##         21   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##         22   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##         23   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##         24   0    0    0    0    0    0    0    1    2    1    3    2    1    1    1    0    0
##         25   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##         26   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##         27   0    0    0    0    0    0    0    0    0    0    1    1    1    1    1    0    0
##         29   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##           actual
## predicted  18  19  20  21  22  23  24  25  26  27  29
##         1   0   0   0   0   0   0   0   0   0   0   0
##         2   0   0   0   0   0   0   0   0   0   0   0
##         3   0   0   0   0   0   0   0   0   0   0   0
##         4   0   0   0   0   0   0   0   0   0   0   0
##         5   0   0   0   0   0   0   0   0   0   0   0
##         6   0   0   0   0   0   0   0   0   0   0   0
##         7   3   1   0   0   0   0   0   0   0   0   0
##         8   3   3   2   2   1   3   0   0   0   0   0
##         9   9  11   8   3   2   2   0   0   0   0   0
##         10  7   6   6   1   0   1   0   1   1   1   0
##         11 20  11  10   8   3   3   0   0   0   1   1
##         12  0   0   0   0   0   0   0   0   0   0   0
##         13  0   0   0   0   0   0   0   0   0   0   0
##         14  0   0   0   0   0   0   0   0   0   0   0
##         15  0   0   0   0   0   0   0   0   0   0   0
##         16  0   0   0   0   0   0   0   0   0   0   0
##         17  0   0   0   0   0   0   0   0   0   0   0
##         18  0   0   0   0   0   0   0   0   0   0   0
##         19  0   0   0   0   0   0   0   0   0   0   0
##         20  0   0   0   0   0   0   0   0   0   0   0
##         21  0   0   0   0   0   0   0   0   0   0   0
##         22  0   0   0   0   0   0   0   0   0   0   0
##         23  0   0   0   0   0   0   0   0   0   0   0
##         24  0   0   0   0   0   0   2   0   0   0   0
##         25  0   0   0   0   0   0   0   0   0   0   0
```

```
##         26   0   0   0   0   0   0   0   0   0   0   0
##         27   0   0   0   0   0   0   0   0   0   0   0
##         29   0   0   0   0   0   0   0   0   0   0   0
```

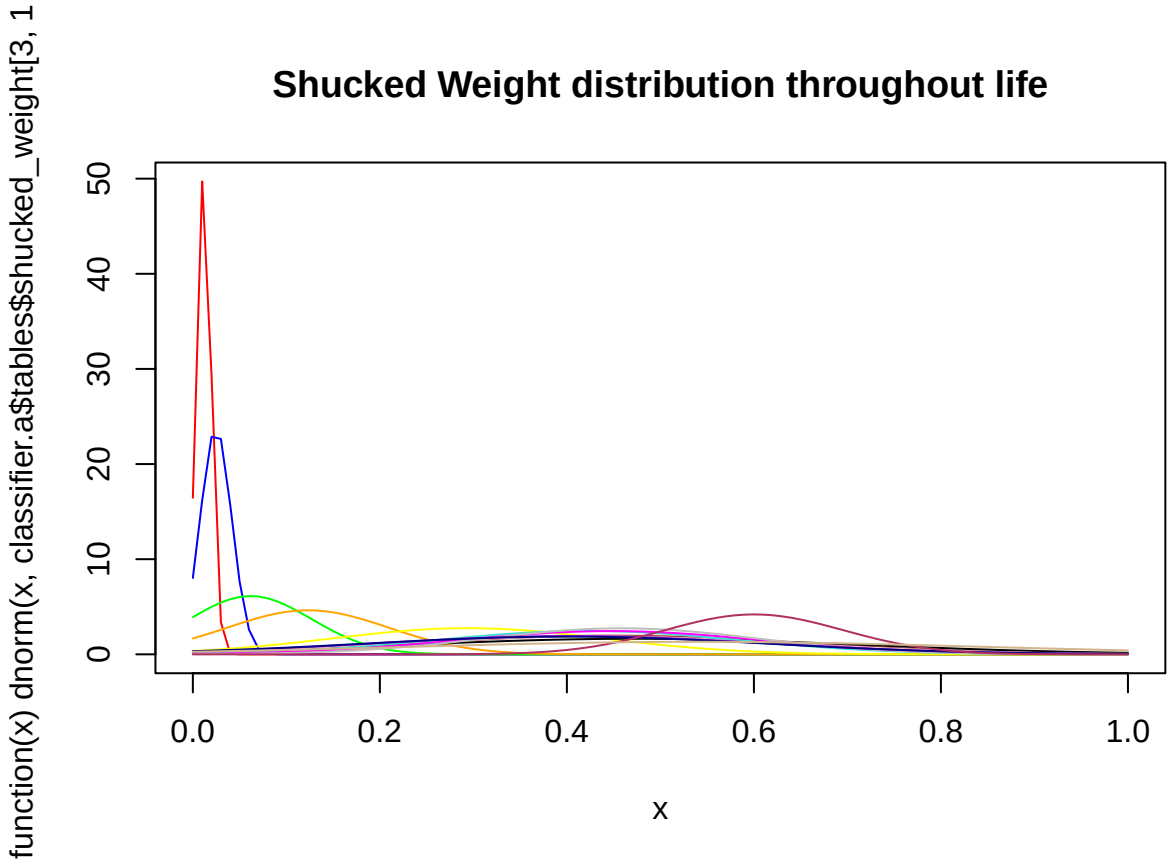
Plot the distribution of classes along 3 different features

```
# Length
plot(function(x) dnorm(x, classifier.a$tables$length[3,1], classifier.a$tables$length[3,2]), 0, 1, col=
curve(dnorm(x, classifier.a$tables$length[4,1], classifier.a$tables$length[4, 2]), add=TRUE, col="blue"
curve(dnorm(x, classifier.a$tables$length[5,1], classifier.a$tables$length[5, 2]), add=TRUE, col = "gre
curve(dnorm(x, classifier.a$tables$length[6,1], classifier.a$tables$length[6, 2]), add=TRUE, col="orang
curve(dnorm(x, classifier.a$tables$length[8,1], classifier.a$tables$length[8, 2]), add=TRUE, col = "yel
curve(dnorm(x, classifier.a$tables$length[10,1], classifier.a$tables$length[10, 2]), add=TRUE, col="bro
curve(dnorm(x, classifier.a$tables$length[12,1], classifier.a$tables$length[12, 2]), add=TRUE, col = "b
curve(dnorm(x, classifier.a$tables$length[14,1], classifier.a$tables$length[14, 2]), add=TRUE, col="pin
curve(dnorm(x, classifier.a$tables$length[16,1], classifier.a$tables$length[16, 2]), add=TRUE, col = "t
curve(dnorm(x, classifier.a$tables$length[18,1], classifier.a$tables$length[18, 2]), add=TRUE, col="mag
curve(dnorm(x, classifier.a$tables$length[20,1], classifier.a$tables$length[20, 2]), add=TRUE, col = "g
curve(dnorm(x, classifier.a$tables$length[22,1], classifier.a$tables$length[22, 2]), add=TRUE, col="pur
curve(dnorm(x, classifier.a$tables$length[23,1], classifier.a$tables$length[23, 2]), add=TRUE, col = "n
curve(dnorm(x, classifier.a$tables$length[24,1], classifier.a$tables$length[24, 2]), add=TRUE, col="mar
curve(dnorm(x, classifier.a$tables$length[27,1], classifier.a$tables$length[27, 2]), add=TRUE, col = "t
```



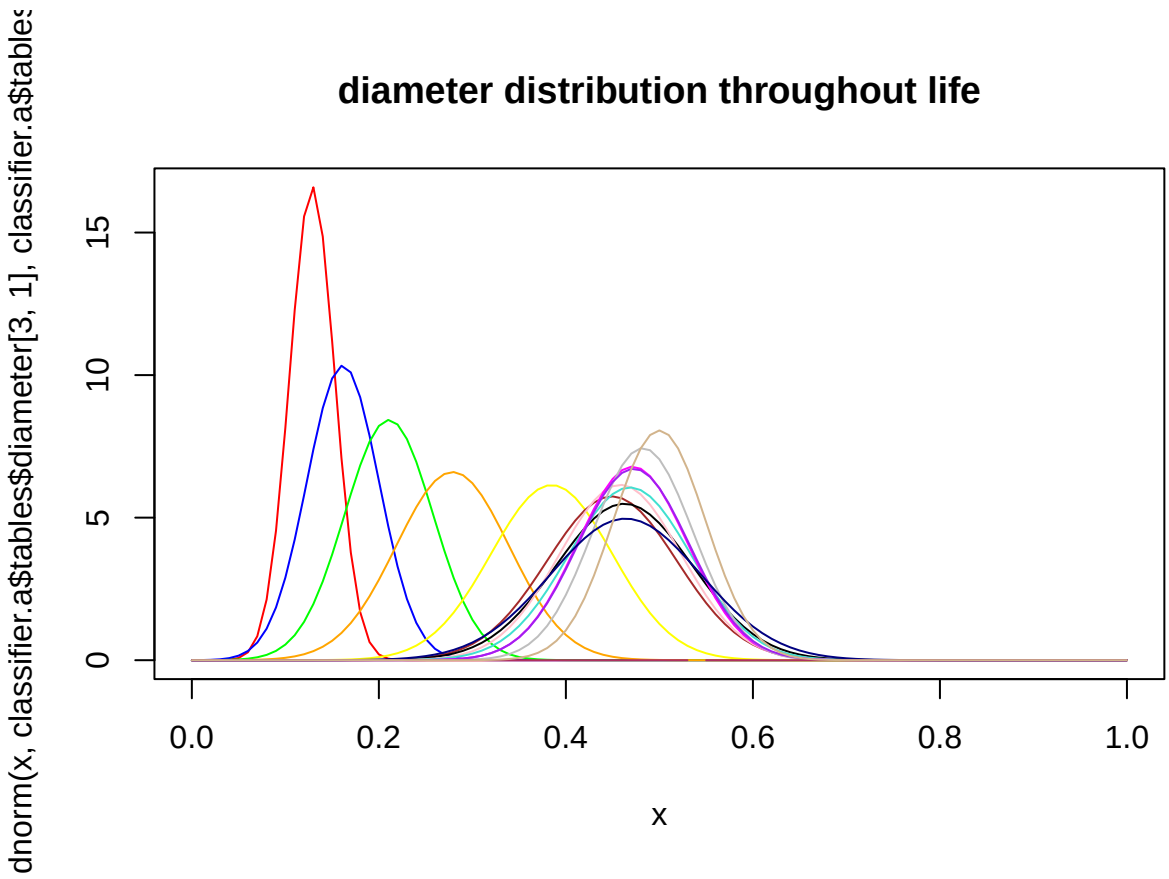

```
# shucked weight
plot(function(x) dnorm(x, classifier.a$tables$shucked_weight[3,1], classifier.a$tables$shucked_weight[3
curve(dnorm(x, classifier.a$tables$shucked_weight[4,1], classifier.a$tables$shucked_weight[4, 2]), add=
```

```
curve(dnorm(x, classifier.a$tables$shucked_weight[5,1], classifier.a$tables$shucked_weight[5, 2]), add=
curve(dnorm(x, classifier.a$tables$shucked_weight[6,1], classifier.a$tables$shucked_weight[6, 2]), add=
curve(dnorm(x, classifier.a$tables$shucked_weight[8,1], classifier.a$tables$shucked_weight[8, 2]), add=
curve(dnorm(x, classifier.a$tables$shucked_weight[10,1], classifier.a$tables$shucked_weight[10, 2]), ad
curve(dnorm(x, classifier.a$tables$shucked_weight[12,1], classifier.a$tables$shucked_weight[12, 2]), ad
curve(dnorm(x, classifier.a$tables$shucked_weight[14,1], classifier.a$tables$shucked_weight[14, 2]), ad
curve(dnorm(x, classifier.a$tables$shucked_weight[16,1], classifier.a$tables$shucked_weight[16, 2]), ad
curve(dnorm(x, classifier.a$tables$shucked_weight[18,1], classifier.a$tables$shucked_weight[18, 2]), ad
curve(dnorm(x, classifier.a$tables$shucked_weight[20,1], classifier.a$tables$shucked_weight[20, 2]), ad
curve(dnorm(x, classifier.a$tables$shucked_weight[22,1], classifier.a$tables$shucked_weight[22, 2]), ad
curve(dnorm(x, classifier.a$tables$shucked_weight[23,1], classifier.a$tables$shucked_weight[23, 2]), ad
curve(dnorm(x, classifier.a$tables$shucked_weight[24,1], classifier.a$tables$shucked_weight[24, 2]), ad
curve(dnorm(x, classifier.a$tables$shucked_weight[27,1], classifier.a$tables$shucked_weight[27, 2]), ad
```




```
# diameter
plot(function(x) dnorm(x, classifier.a$tables$diameter[3,1], classifier.a$tables$diameter[3,2]), 0, 1, 
curve(dnorm(x, classifier.a$tables$diameter[4,1], classifier.a$tables$diameter[4, 2]), add=TRUE, col="bl
curve(dnorm(x, classifier.a$tables$diameter[5,1], classifier.a$tables$diameter[5, 2]), add=TRUE, col = 
curve(dnorm(x, classifier.a$tables$diameter[6,1], classifier.a$tables$diameter[6, 2]), add=TRUE, col="or
curve(dnorm(x, classifier.a$tables$diameter[8,1], classifier.a$tables$diameter[8, 2]), add=TRUE, col = 
curve(dnorm(x, classifier.a$tables$diameter[10,1], classifier.a$tables$diameter[10, 2]), add=TRUE, col=
curve(dnorm(x, classifier.a$tables$diameter[12,1], classifier.a$tables$diameter[12, 2]), add=TRUE, col =
curve(dnorm(x, classifier.a$tables$diameter[14,1], classifier.a$tables$diameter[14, 2]), add=TRUE, col=
curve(dnorm(x, classifier.a$tables$diameter[16,1], classifier.a$tables$diameter[16, 2]), add=TRUE, col =
curve(dnorm(x, classifier.a$tables$diameter[18,1], classifier.a$tables$diameter[18, 2]), add=TRUE, col=
curve(dnorm(x, classifier.a$tables$diameter[20,1], classifier.a$tables$diameter[20, 2]), add=TRUE, col =
```

```
curve(dnorm(x, classifier.a$tables$diameter[22,1], classifier.a$tables$diameter[22, 2]), add=TRUE, col=
curve(dnorm(x, classifier.a$tables$diameter[23,1], classifier.a$tables$diameter[23, 2]), add=TRUE, col =
curve(dnorm(x, classifier.a$tables$diameter[24,1], classifier.a$tables$diameter[24, 2]), add=TRUE, col=
curve(dnorm(x, classifier.a$tables$diameter[27,1], classifier.a$tables$diameter[27, 2]), add=TRUE, col =
```



PROVIDED CODE

```
# read dataset
abalone <- read.csv(url("https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data
                    sep = ",")
# rename columns
colnames(abalone) <- c("sex", "length", 'diameter', 'height', 'whole_weight', 'shucked_wieght', 'viscera
                       'rings' )

# add new column abalone$age.group with 3 values based on the number of rings
abalone$age.group <- cut(abalone$rings, br=c(0,8,11,35), labels = c("young", 'adult', 'old'))
# drop the sex column (categorical variable)
abalone.norm <- abalone[,-1]
# optionally normalize
normalize <- function(x) {return ((x - min(x)) / (max(x) - min(x))) }
abalone.norm[1:7] <- as.data.frame(lapply(abalone.norm[1:7], normalize))

# sample 2924 from 4177 (~70%)
s_abalone <- sample(4177,2924)
```

```r
## create train & test sets based on sampled indexes
abalone.norm.train <-abalone.norm[s_abalone,]
abalone.norm.test <-abalone.norm[-s_abalone,]

sqrt(2924)
```

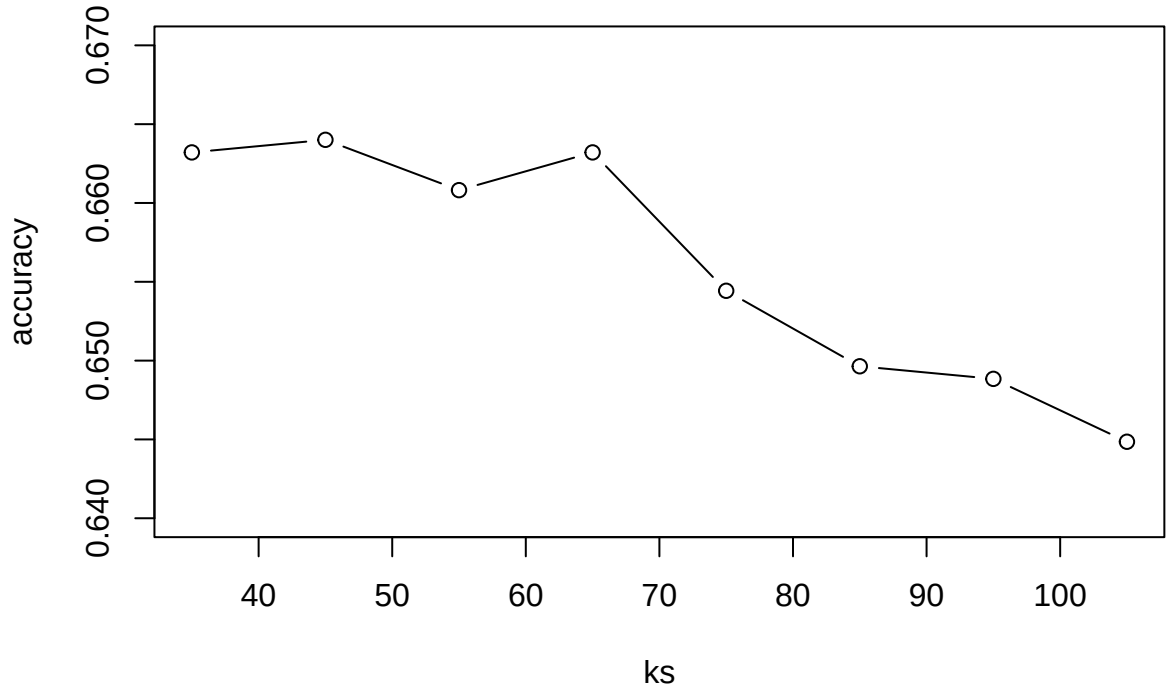
```
## [1] 54.07402
```

```r
k = 55
# k = 80
# train model & predict
KNNpred <- knn(train = abalone.norm.train[1:7], test = abalone.norm.test[1:7], cl = abalone.norm.train$a
# create contingency table/ confusion matrix
contingency.table <- table(KNNpred,abalone.norm.test$age.group)

contingency.matrix = as.matrix(contingency.table)
sum(diag(contingency.matrix))/length(abalone.norm.test$age.group)
```

```
## [1] 0.660814
```

```r
accuracy <- c()
ks <- c(35,45,55,65,75,85,95,105)
for (k in ks) {
  KNNpred <- knn(train = abalone.norm.train[1:7], test = abalone.norm.test[1:7], cl = abalone.norm.trai
  cm = as.matrix(table(Actual=KNNpred, Predicted = abalone.norm.test$age.group, dnn=list('predicted','a
  accuracy <- c(accuracy,sum(diag(cm))/length(abalone.norm.test$age.group))
}
plot(ks,accuracy,type = "b", ylim = c(0.64,0.67))
```



EXCERSISE 2

Repeat the kNN analysis using the iris dataset

```
# optionally normalize
iris.norm <- iris
iris.norm[1:4] <- as.data.frame(lapply(iris.norm[1:4], normalize))

# sample 2924 from 4177 (~70%)
s_iris <- sample(nrow(iris.norm), 105)
## create train & test sets based on sampled indexes
iris.norm.train <-iris.norm[s_iris,]
iris.norm.test <-iris.norm[-s_iris,]

sqrt(105)
```

```
## [1] 10.24695
```

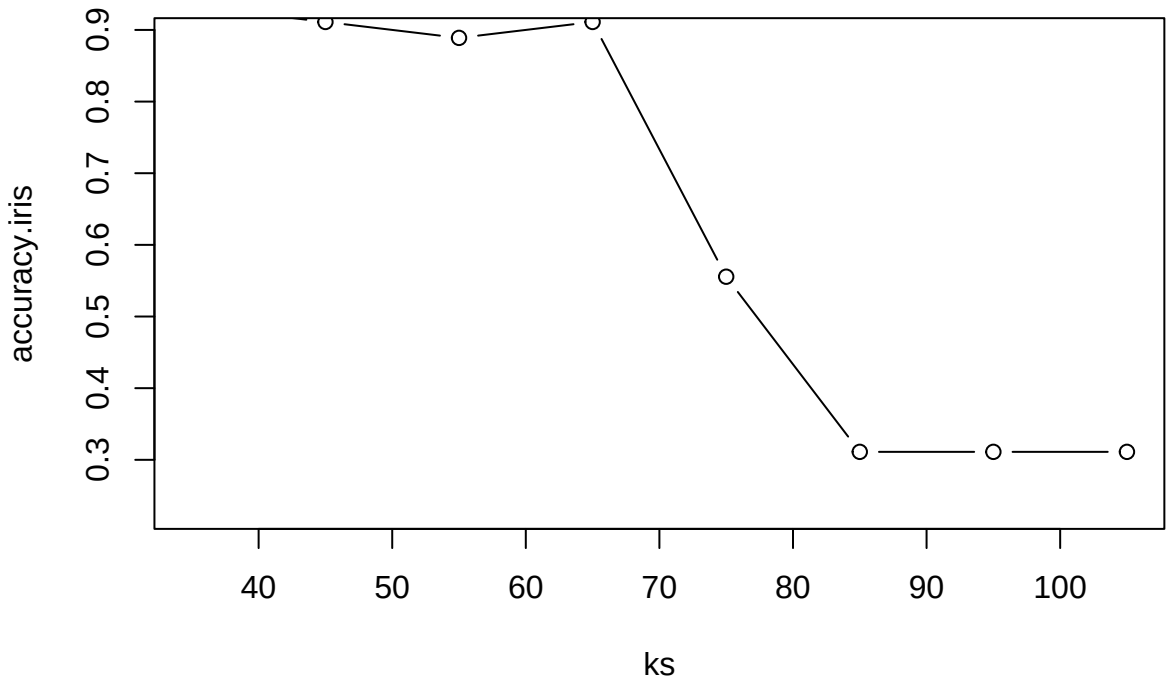
```
k = 10
# k = 80
# train model & predict
KNNpred.iris <- knn(train = iris.norm.train[,1:4], test = iris.norm.test[,1:4], cl = iris.norm.train$Sp
# create contingency table/ confusion matrix
contingency.table.iris <- table(KNNpred.iris,iris.norm.test$Species)

contingency.matrix.iris = as.matrix(contingency.table.iris)
```

```r
sum(diag(contingency.matrix.iris))/length(iris.norm.test$Species)
```

```
## [1] 0.9555556
```

```r
accuracy.iris <- c()
ks <- c(35,45,55,65,75,85,95,105)
for (k in ks) {
  KNNpred.iris <- knn(train = iris.norm.train[1:4], test = iris.norm.test[1:4], cl = iris.norm.train$Sp
  cm.iris = as.matrix(table(Actual=KNNpred.iris, Predicted = iris.norm.test$Species, dnn=list('predicte
  accuracy.iris <- c(accuracy.iris,sum(diag(cm.iris))/length(iris.norm.test$Species))
}
plot(ks,accuracy.iris,type = "b", ylim = c(0.23,0.89))
```



Try 2 diferent subsets of features

```r
# Subset 1
abalone.norm[1:7] <- as.data.frame(lapply(abalone.norm[1:7], normalize))

# sample 2924 from 4177 (~70%)
s_abalone <- sample(4177,2924)
## create train & test sets based on sampled indexes
abalone.norm.train <-abalone.norm[s_abalone,]
abalone.norm.test <-abalone.norm[-s_abalone,]

sqrt(2924)
```

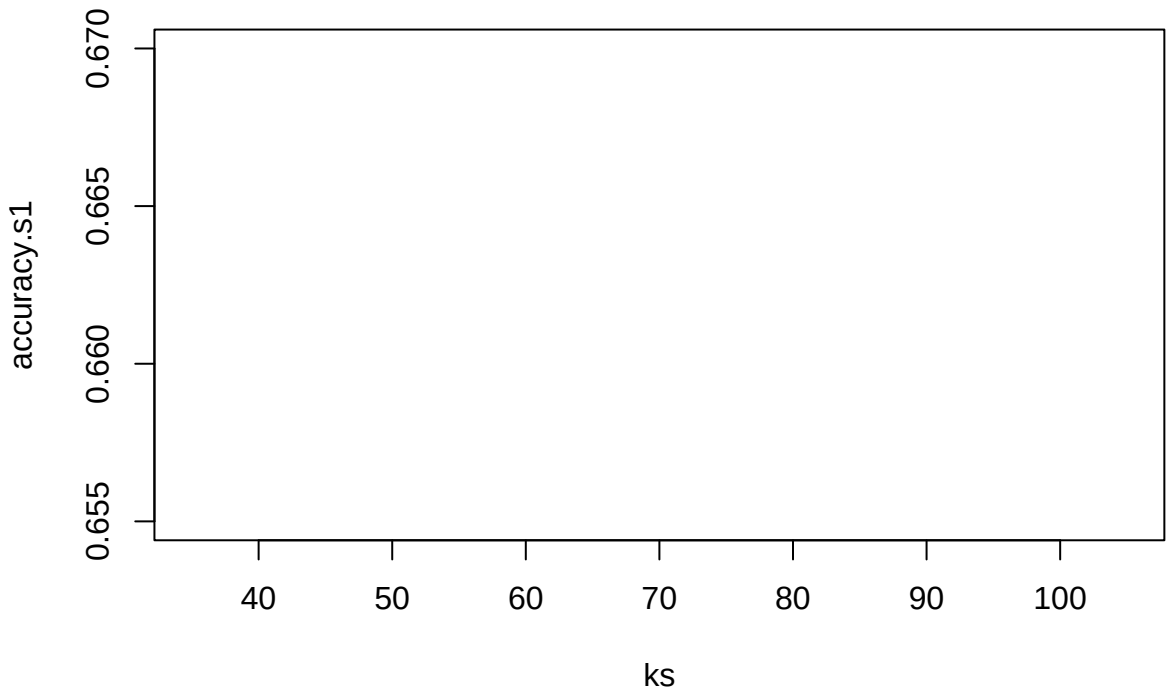
```
## [1] 54.07402
```

```
k = 55
# k = 80
# train model & predict
KNNpred.s1 <- knn(train = abalone.norm.train[c(1, 3, 5, 7)], test = abalone.norm.test[c(1, 3, 5, 7)], c
# create contingency table/ confusion matrix
contingency.table.s1 <- table(KNNpred.s1,abalone.norm.test$age.group)

contingency.matrix.s1 = as.matrix(contingency.table.s1)
sum(diag(contingency.matrix.s1))/length(abalone.norm.test$age.group)
```

```
## [1] 0.6967279
```

```
accuracy.s1 <- c()
ks <- c(35,45,55,65,75,85,95,105)
for (k in ks) {
  KNNpred.s1 <- knn(train = abalone.norm.train[c(1, 3, 5, 7)], test = abalone.norm.test[c(1, 3, 5, 7)],
  cm.s1 = as.matrix(table(Actual=KNNpred.s1, Predicted = abalone.norm.test$age.group, dnn=list('predict
  accuracy.s1 <- c(accuracy.s1,sum(diag(cm.s1))/length(abalone.norm.test$age.group))
}
plot(ks,accuracy.s1,type = "b", ylim = c(0.655,0.67))
```




```
## Subset 2
abalone.norm[1:7] <- as.data.frame(lapply(abalone.norm[1:7], normalize))

# sample 2924 from 4177 (~70%)
s_abalone <- sample(4177,2924)
```

```
## create train & test sets based on sampled indexes
abalone.norm.train <-abalone.norm[s_abalone,]
abalone.norm.test <-abalone.norm[-s_abalone,]

sqrt(2924)
```

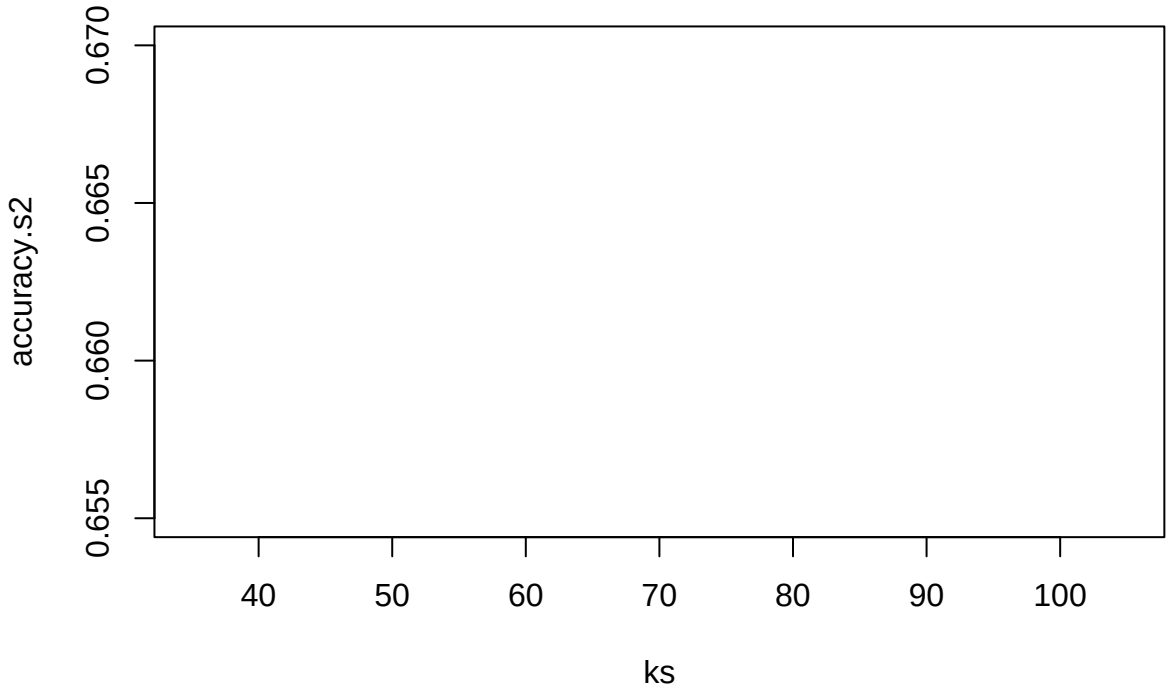
```
## [1] 54.07402
```

```
k = 55
# k = 80
# train model & predict
KNNpred.s2 <- knn(train = abalone.norm.train[c(2, 4, 6)], test = abalone.norm.test[c(2, 4, 6)], cl = aba
# create contingency table/ confusion matrix
contingency.table.s2 <- table(KNNpred.s2,abalone.norm.test$age.group)

contingency.matrix.s2 = as.matrix(contingency.table.s2)
sum(diag(contingency.matrix.s2))/length(abalone.norm.test$age.group)
```

```
## [1] 0.5881883
```

```
accuracy.s2 <- c()
ks <- c(35,45,55,65,75,85,95,105)
for (k in ks) {
  KNNpred.s2 <- knn(train = abalone.norm.train[c(2, 4, 6)], test = abalone.norm.test[c(2, 4, 6)], cl = a
  cm.s2 = as.matrix(table(Actual=KNNpred.s2, Predicted = abalone.norm.test$age.group, dnn=list('predicte
  accuracy.s2 <- c(accuracy.s2,sum(diag(cm.s2))/length(abalone.norm.test$age.group))
}
plot(ks,accuracy.s2,type = "b", ylim = c(0.655,0.67))
```




```
##         Compare models using contingency tables and accuracy plots          ##
# Subset 1
table(knn(train = abalone.norm.train[c(2, 4, 6)], test = abalone.norm[c(2, 4, 6)], cl = abalone.norm.tra
, abalone.norm[,9], dnn=list('predicted','actual'))
```

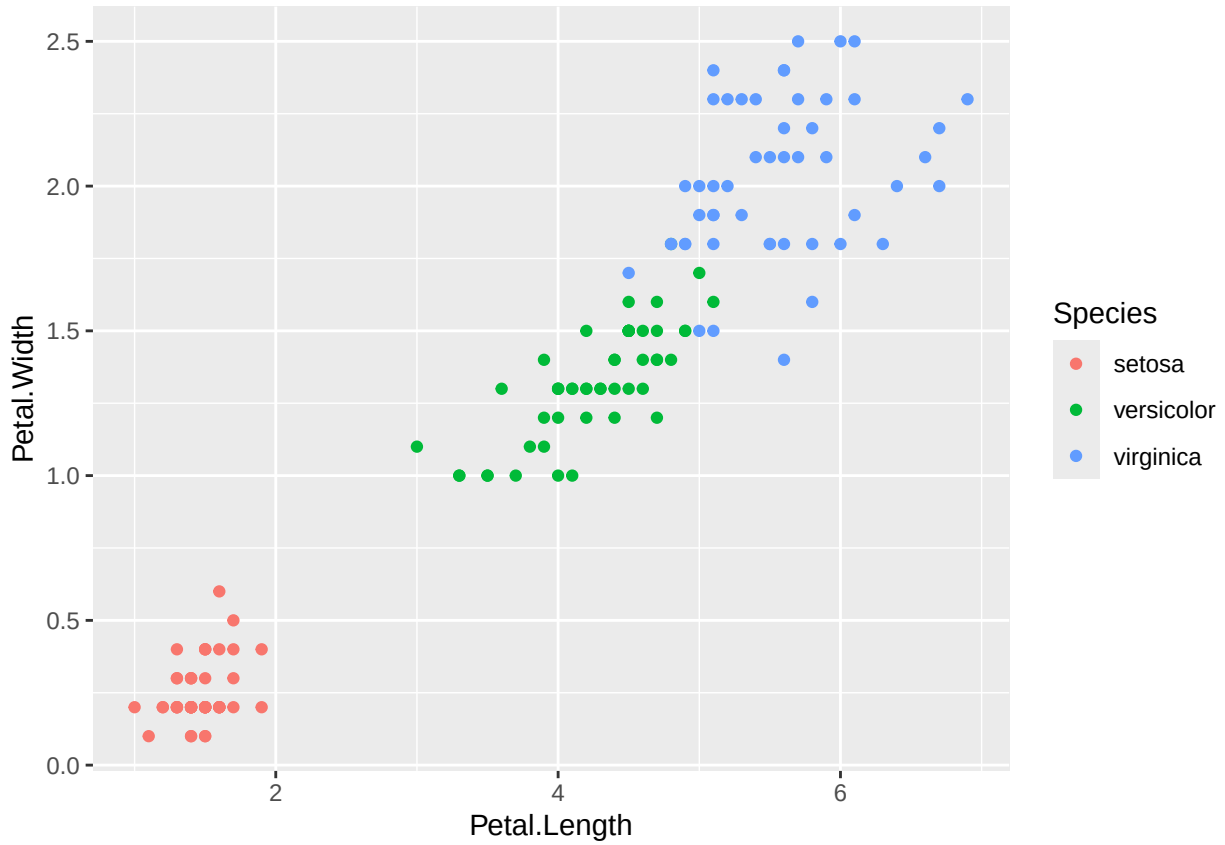
```
##           actual
## predicted young adult  old
##     young  1022   308   96
##     adult   373  1452  794
##     old      12    50   70
```

```
# Subset 2
table(knn(train = abalone.norm.train[c(1, 3, 5, 7)], test = abalone.norm[c(1, 3, 5, 7)], cl = abalone.no
      , abalone.norm[,9], dnn=list('predicted','actual'))
```

```
##           actual
## predicted young adult  old
##     young  1035   260   52
##     adult   361  1458  588
##     old      11    92  320
```

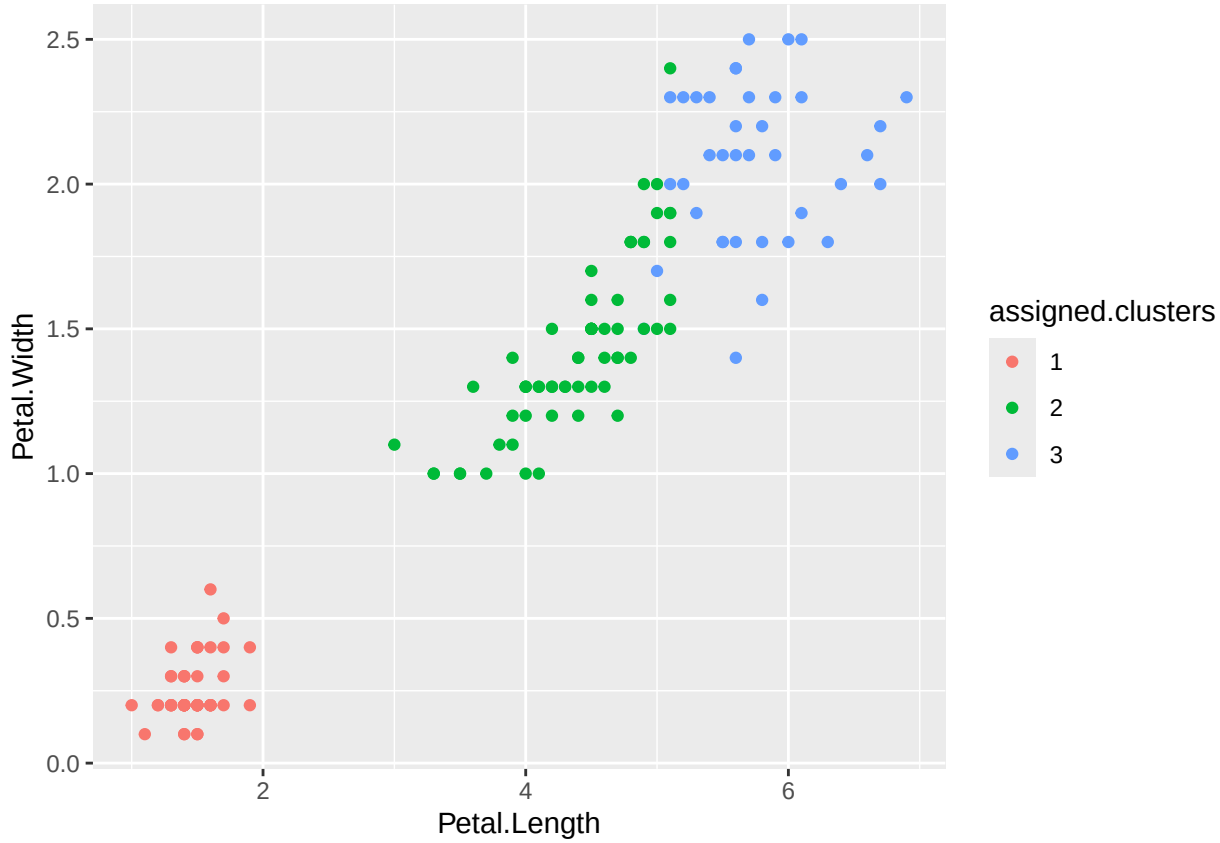
PROVIDED CODE

```
# Plot iris petal length vs. petal width, color by species
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = Species)) +
  geom_point()
```

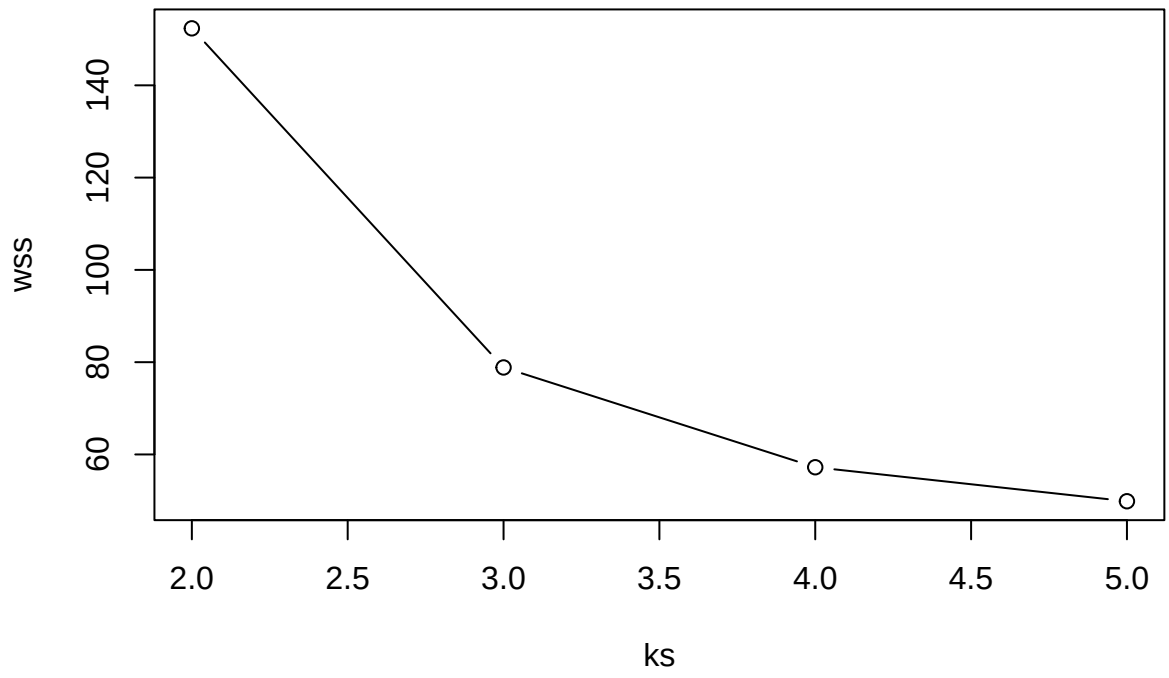



```
# set seed for random number generator
set.seed(123)

# run k-means
iris.km <- kmeans(iris[,-5], centers = 3)
assigned.clusters <- as.factor(iris.km$cluster)
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = assigned.clusters)) +
  geom_point()
```




```
wss <- c()
ks <- c(2,3,4,5)
for (k in ks) {
  iris.km <- kmeans(iris[,-5], centers = k)
  wss <- c(wss,iris.km$tot.withinss)
}
plot(ks,wss,type = "b")
```



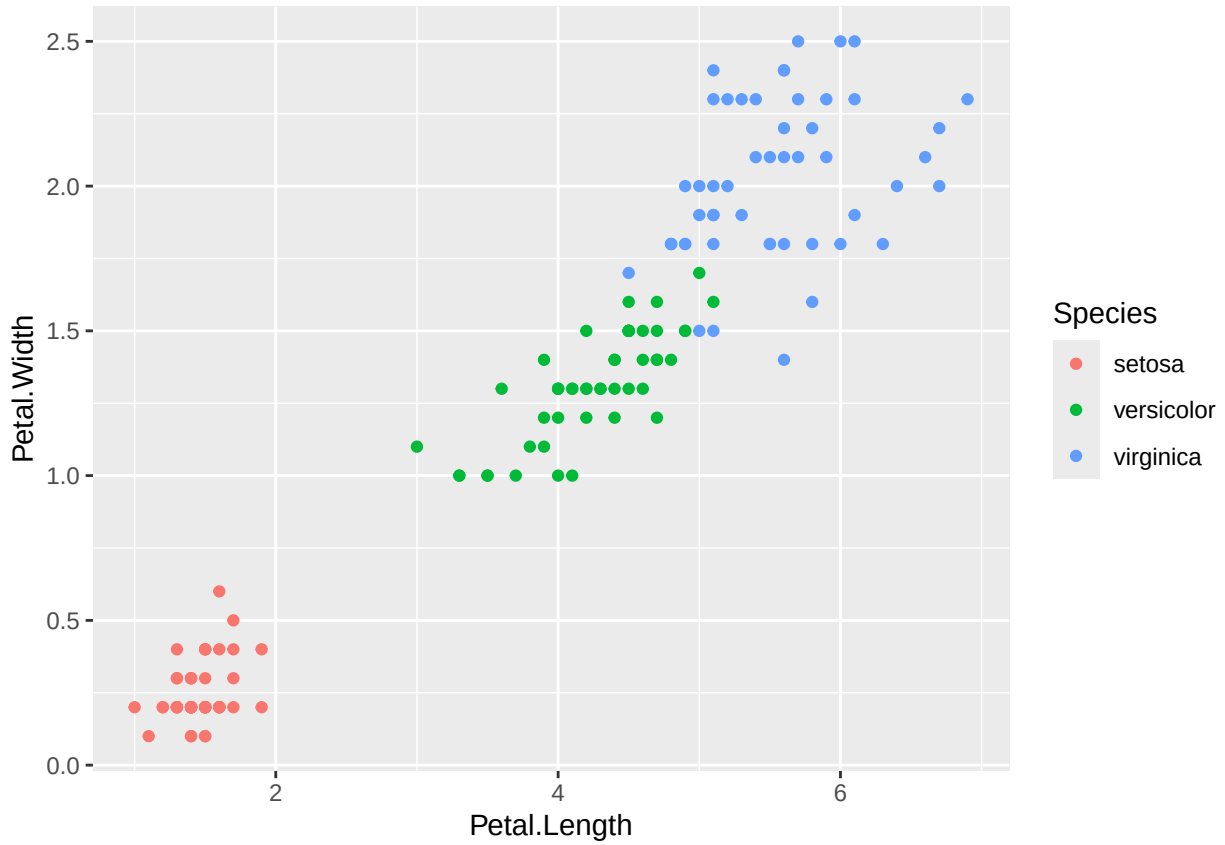

```
labeled.clusters <- as.character(assigned.clusters)
labeled.clusters[labeled.clusters==1] <- "setosa"
labeled.clusters[labeled.clusters==2] <- "versivolor"
labeled.clusters[labeled.clusters==3] <- "virginica"
table(labeled.clusters, iris[,5])
```

```
##
## labeled.clusters setosa versicolor virginica
##       setosa         50          0         0
##       versivolor      0         48        14
##       virginica       0          2        36
```

EXCERSISE 3

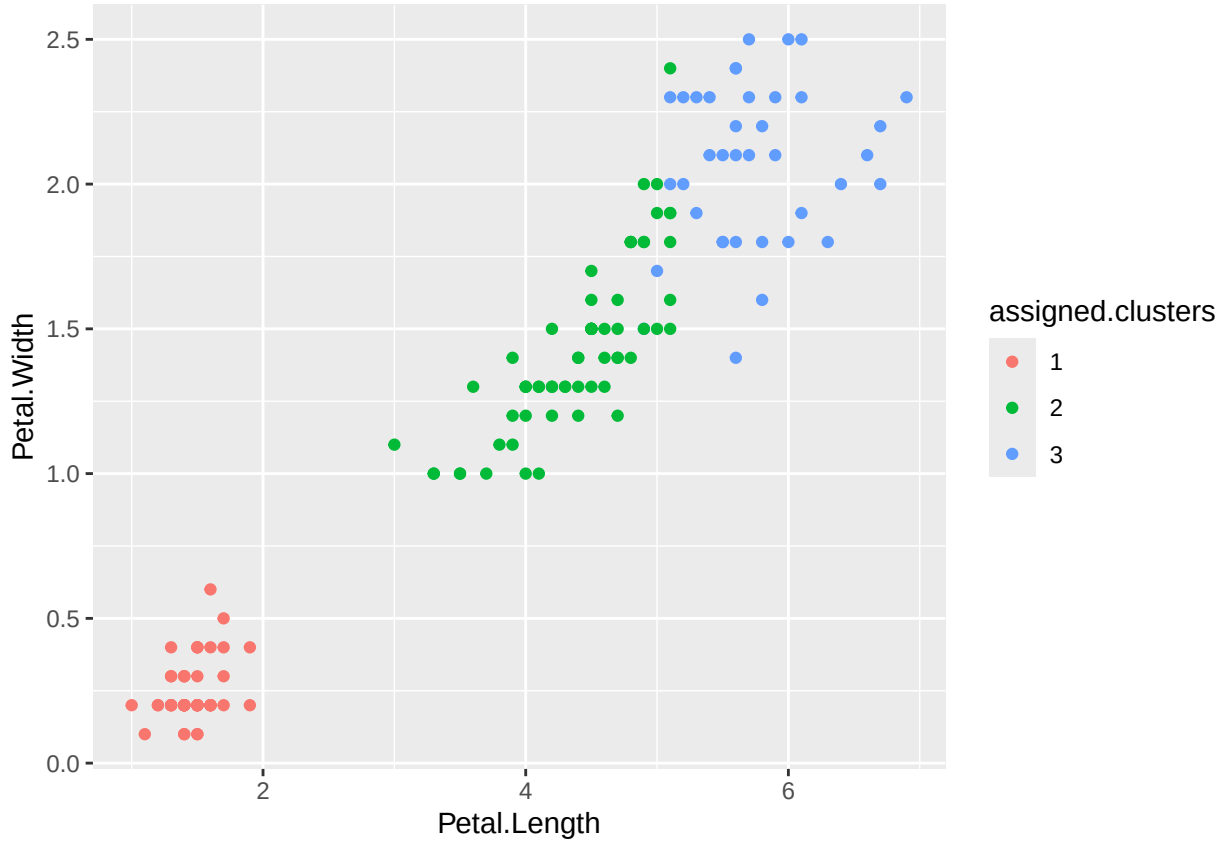
Run K Means Analysis using the abalone & iris datasets

```
# iris dataset
# Plot iris petal length vs. petal width, color by species
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = Species)) +
  geom_point()
```

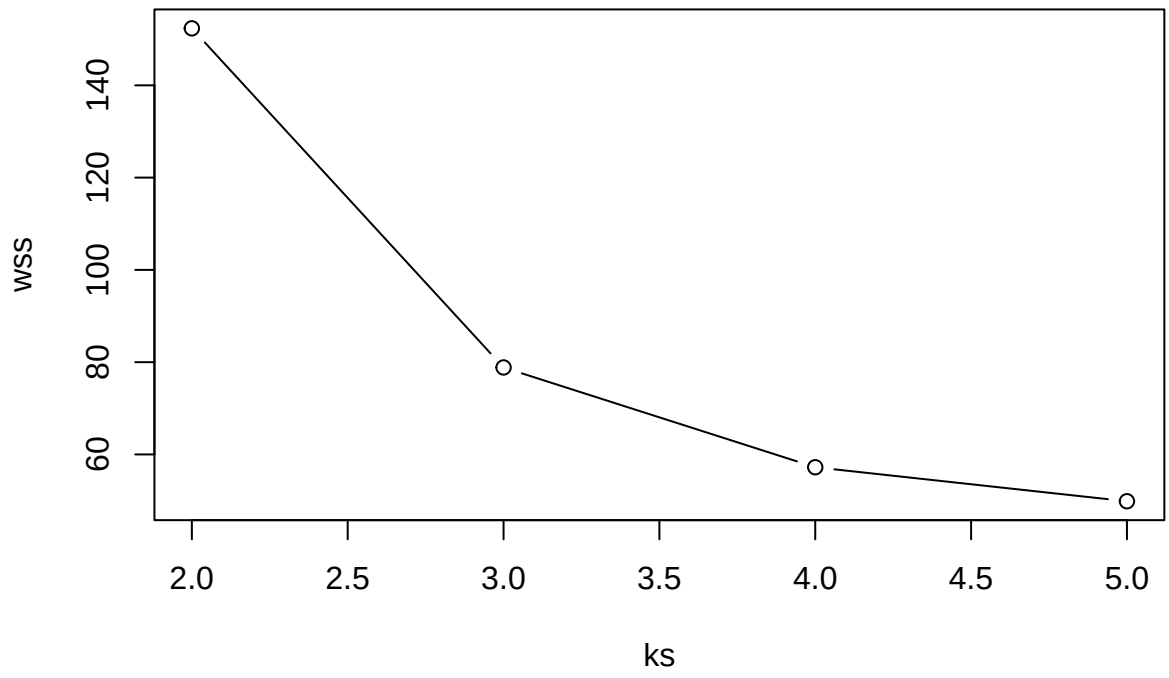



```
# set seed for random number generator
set.seed(123)

# run k-means
iris.km <- kmeans(iris[,-5], centers = 3)
assigned.clusters.iris <- as.factor(iris.km$cluster)
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = assigned.clusters)) +
  geom_point()
```



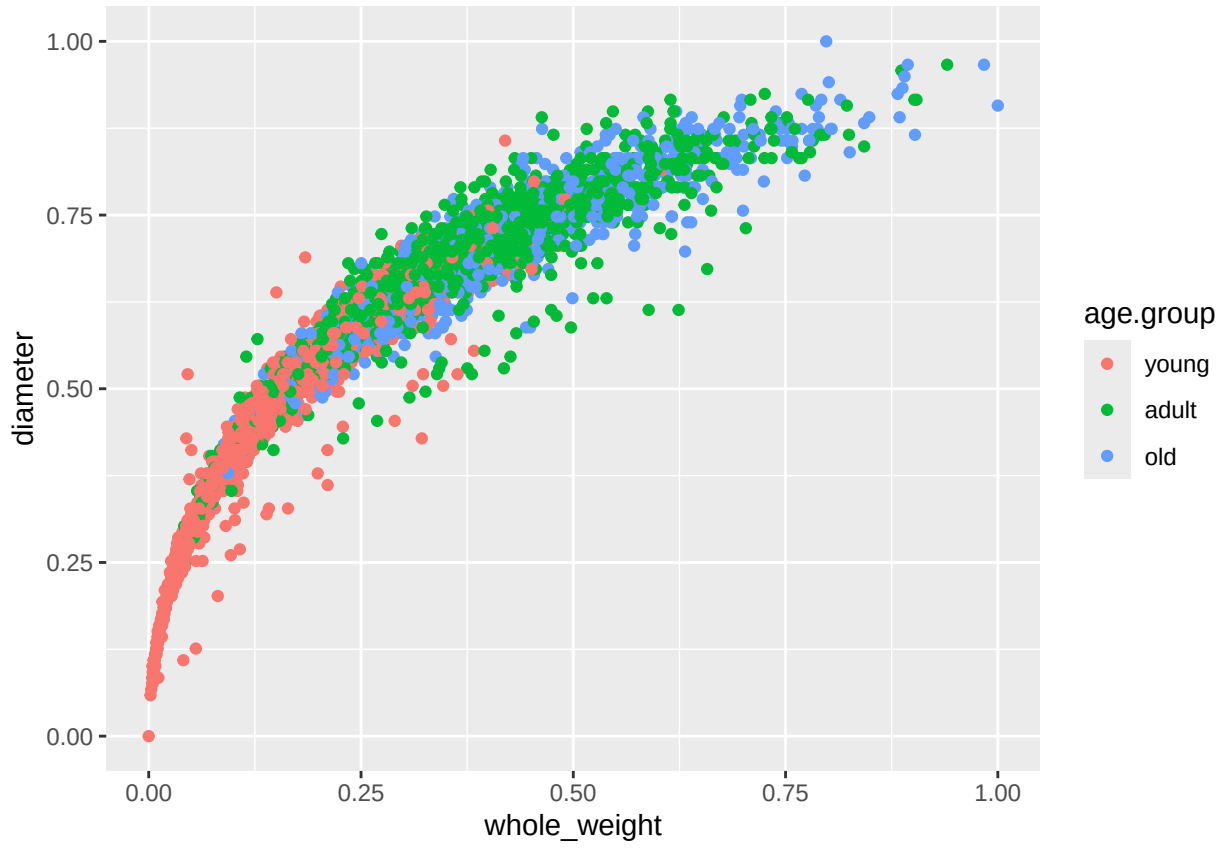

```
wss <- c()
ks <- c(2,3,4,5)
for (k in ks) {
  iris.km <- kmeans(iris[,-5], centers = k)
  wss <- c(wss,iris.km$tot.withinss)
}
plot(ks,wss,type = "b")
```




```
labeled.clusters.iris <- as.character(assigned.clusters.iris)
labeled.clusters.iris[labeled.clusters.iris==1] <- "setosa"
labeled.clusters.iris[labeled.clusters.iris==2] <- "versivolor"
labeled.clusters.iris[labeled.clusters.iris==3] <- "virginica"
table(labeled.clusters.iris, iris[,5])
```

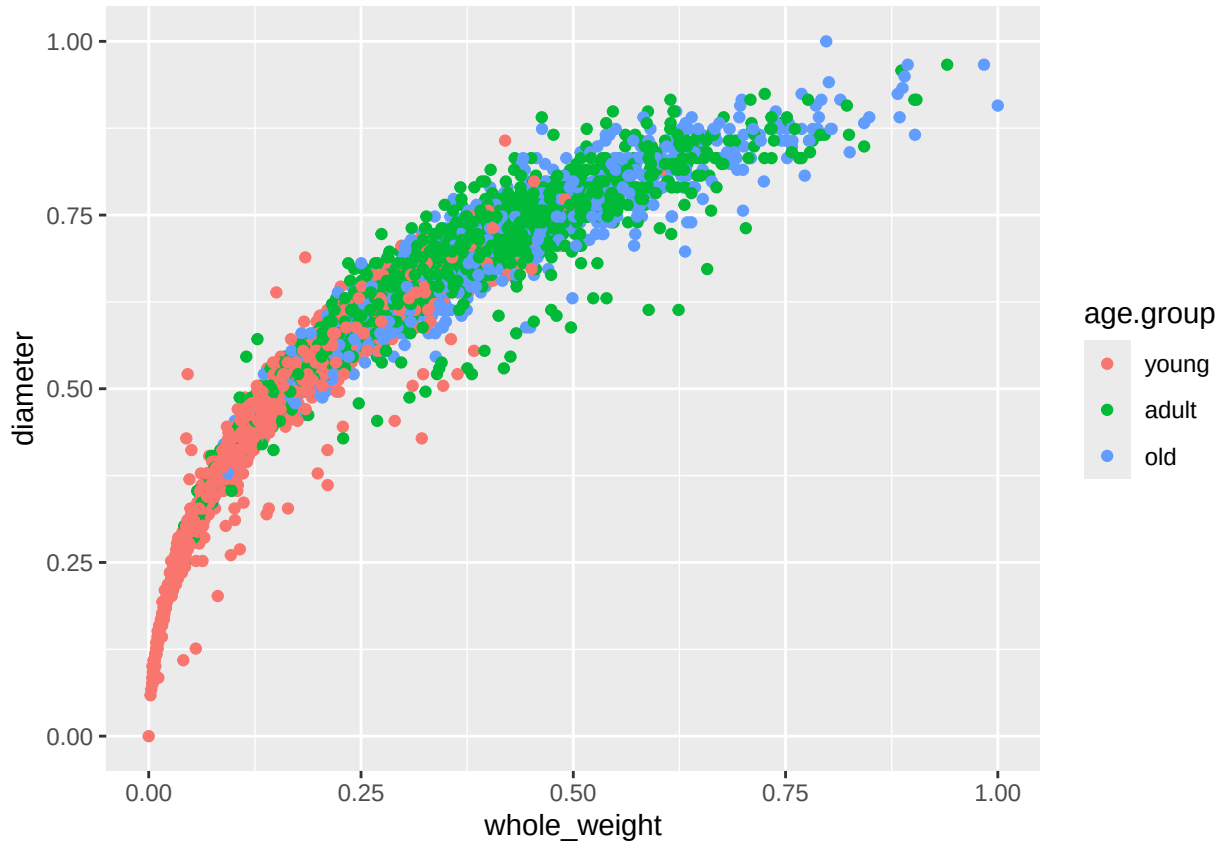
```
##
## labeled.clusters.iris setosa versicolor virginica
##            setosa         50          0         0
##            versivolor      0         48        14
##            virginica       0          2        36
```

```
# Abalone dataset
# Plot abalone length vs. width, color by age
abalone.norm <- abalone.norm[,-8]
ggplot(abalone.norm, aes(x = whole_weight, y = diameter, colour = age.group)) +
  geom_point()
```

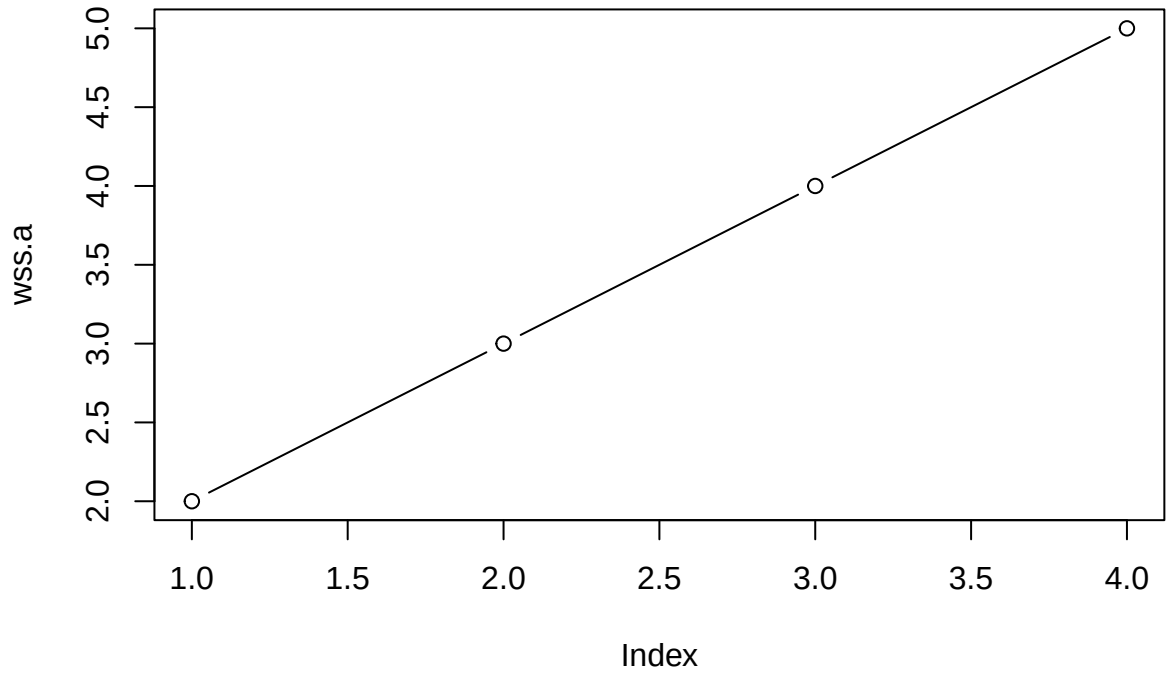



```
# set seed for random number generator
set.seed(123)

# run k-means
abalone.km <- kmeans(abalone.norm[,-8], centers = 3)
assigned.clusters <- as.factor(abalone.km$cluster)
ggplot(abalone.norm, aes(x = whole_weight, y = diameter, colour = age.group)) +
  geom_point()
```




```
wss.a <- c()
ks <- c(2,3,4,5)
for (k in ks) {
  abalone.km <- kmeans(abalone.norm[,-8], centers = k)
  wss <- c(wss.a,abalone.km$tot.withinss)
}
plot(ks,wss.a,type = "b")
```




```
labeled.clusters <- as.character(assigned.clusters)
labeled.clusters[labeled.clusters==1] <- "adult"
labeled.clusters[labeled.clusters==2] <- "young"
labeled.clusters[labeled.clusters==3] <- "old"
table(labeled.clusters, abalone.norm[,8])
```

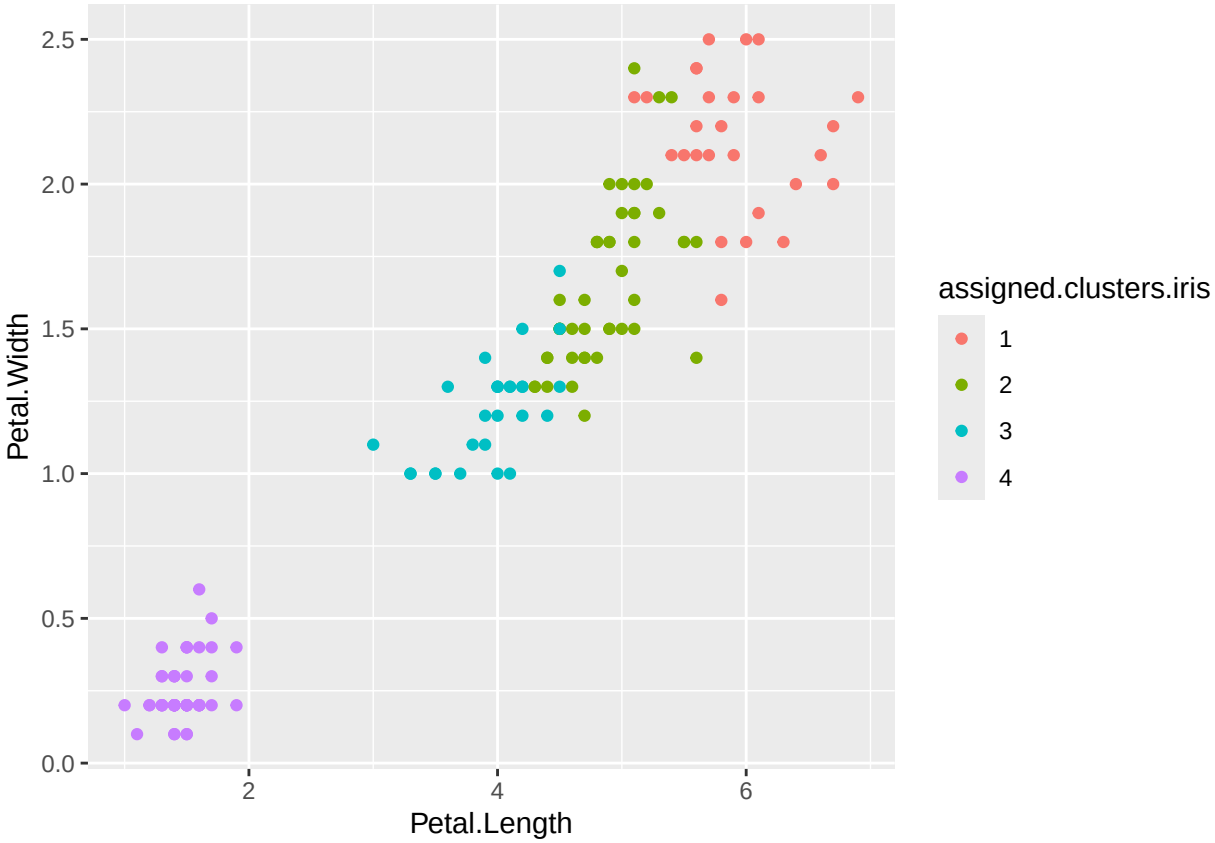
```
##
## labeled.clusters young adult old
##            adult    53   724 440
##            old     918   252  67
##            young   436   834 453
```

Try different values of k for both

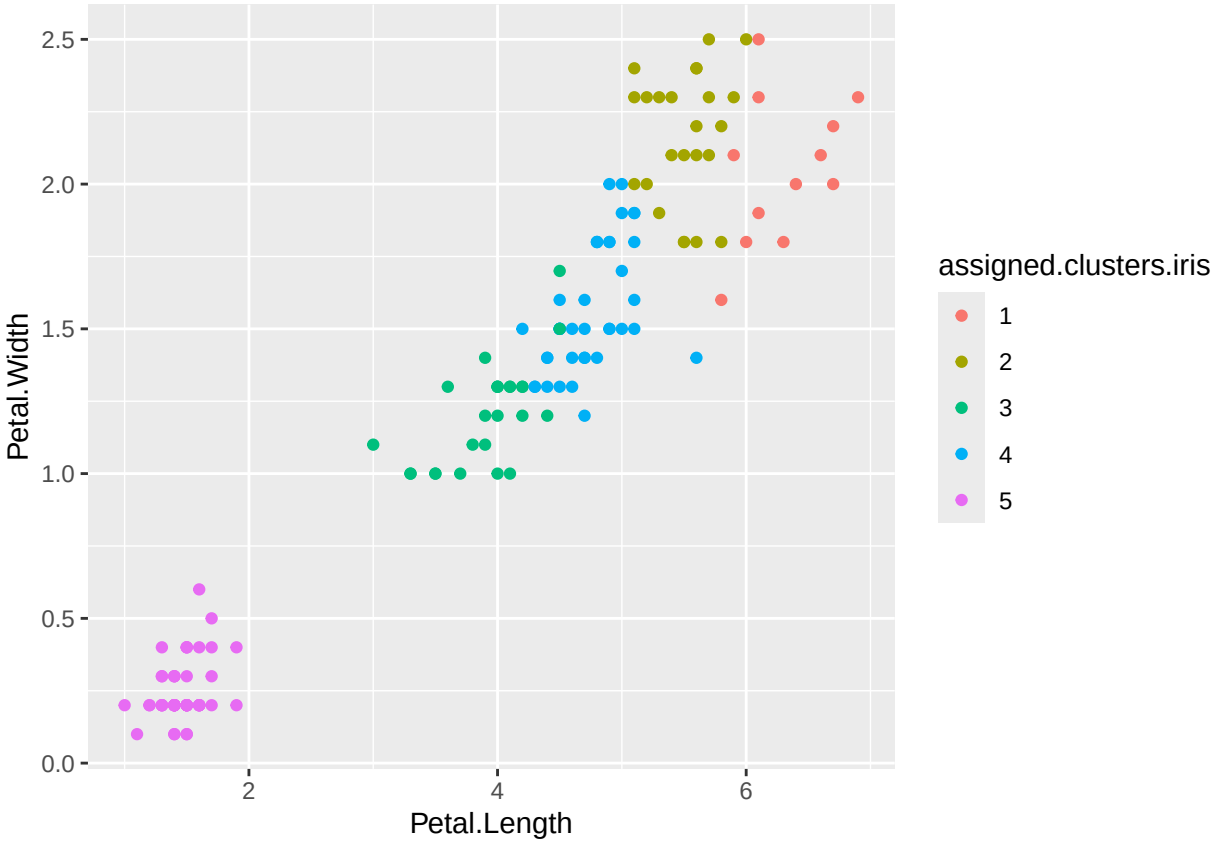
&

Evaluate clustering using Plot the best clustering output for both

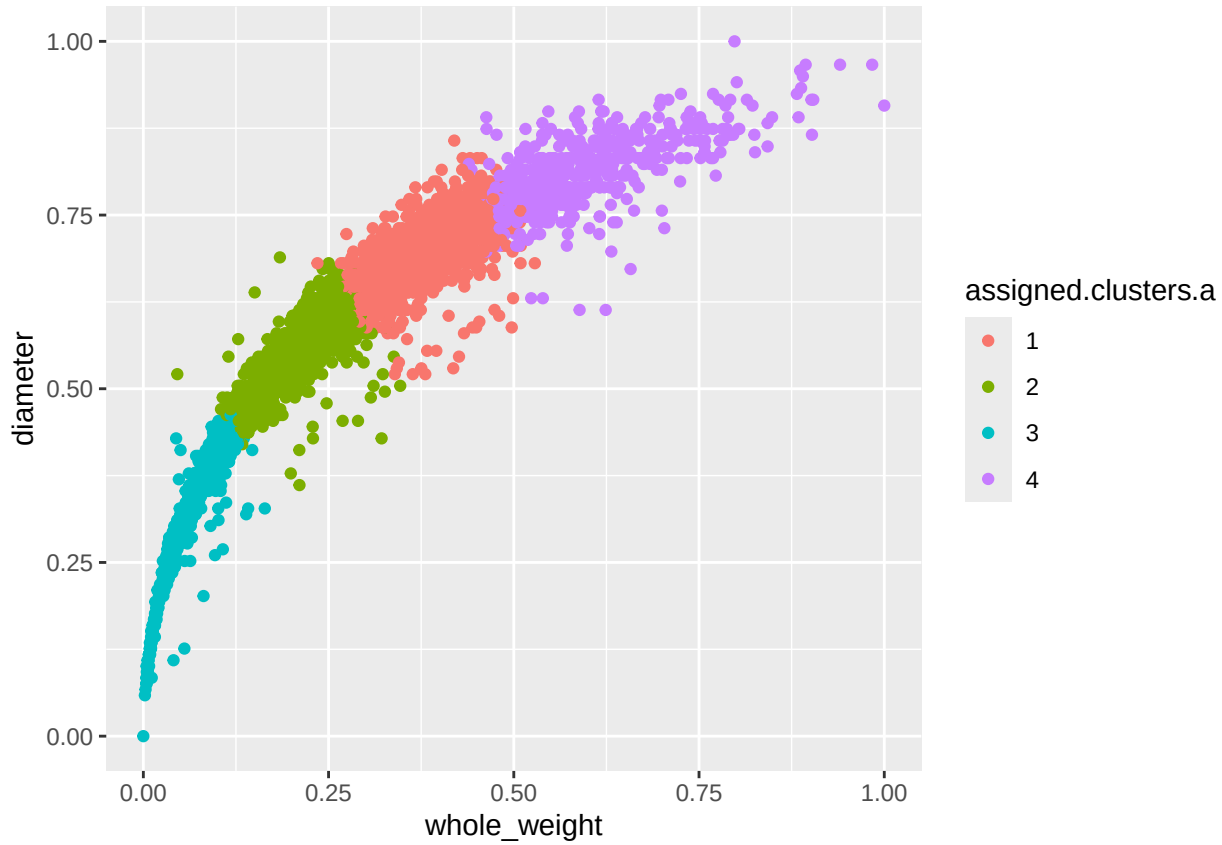
```
# Iris dataset
# run k-means
iris.km <- kmeans(iris[,-5], centers = 4)
assigned.clusters.iris <- as.factor(iris.km$cluster)
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = assigned.clusters.iris)) +
  geom_point()
```



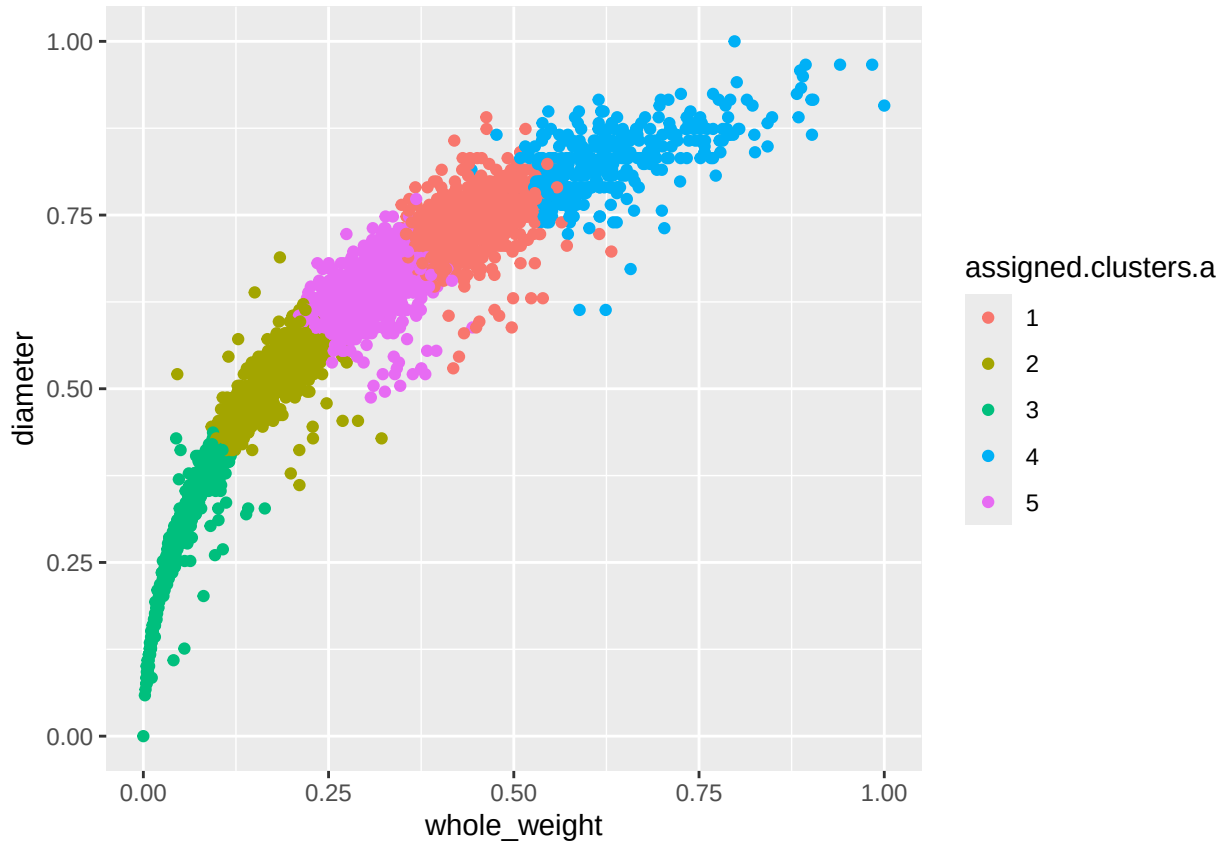

```
# run k-means
iris.km <- kmeans(iris[,-5], centers = 5)
assigned.clusters.iris <- as.factor(iris.km$cluster)
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = assigned.clusters.iris)) +
  geom_point()
```




```
# Abalone dataset
abalone.km <- kmeans(abalone.norm[,-8], centers = 4)
assigned.clusters.a <- as.factor(abalone.km$cluster)
ggplot(abalone.norm, aes(x = whole_weight, y = diameter, colour = assigned.clusters.a)) +
  geom_point()
```




```
abalone.km <- kmeans(abalone.norm[,-8], centers = 5)
assigned.clusters.a <- as.factor(abalone.km$cluster)
ggplot(abalone.norm, aes(x = whole_weight, y = diameter, colour = assigned.clusters.a)) +
  geom_point()
```


1.00
0.75
0.50
0.25
0.00
diameter
0.00
0.25
0.50
0.75
1.00
whole_weight
assigned.clusters.a
1
2
3
4
5