PA 1

Teagan Jenner

3/15/2020

This markdown file was programmed in R and prepared in RStudio.

1: Implement an algorithm to read in the Iris dataset.

Use the built-in datasets library in R to read in the dataset.

```
      # install.packages("datasets")
      library(datasets)

      data(iris)
      summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

```
      names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

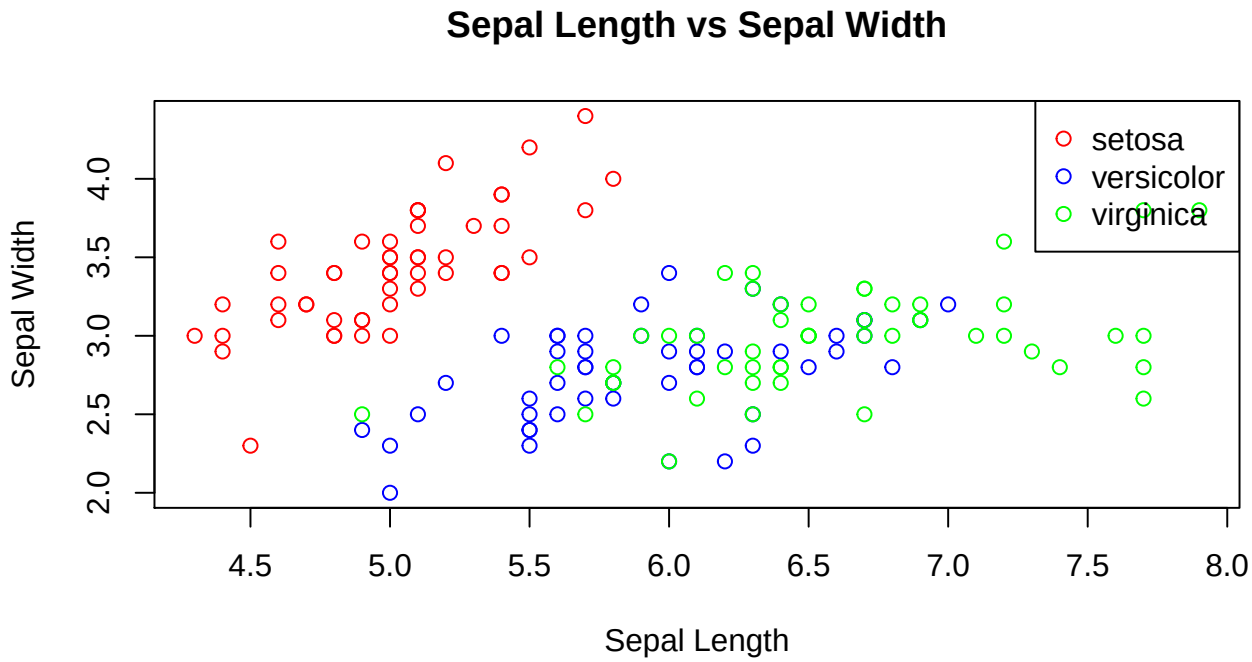
Create 3 new dataframes to contain the data of each class, respectively.

```
      class1 <- iris[1:50,]
      class2 <- iris[51:100,]
      class3 <- iris[101:150,]
```

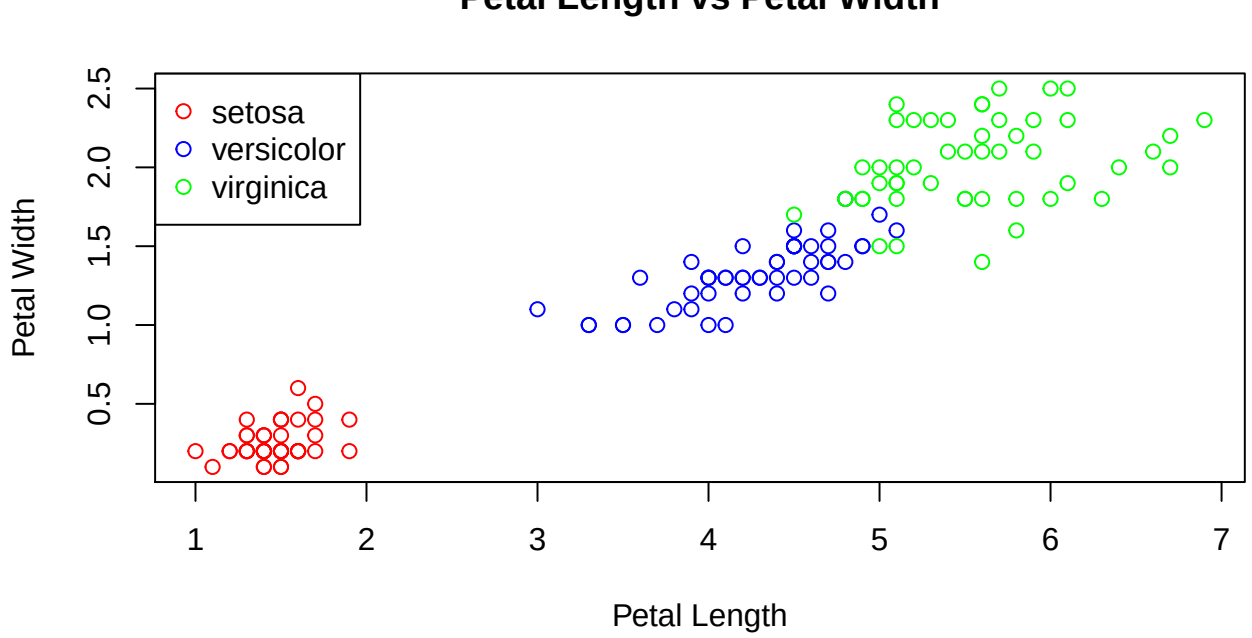
2: Implement an algorithm to visually see two sets of features and the class they belong to.

Use the plot() function in R to see two sets of features at a time. There are six plots in total.

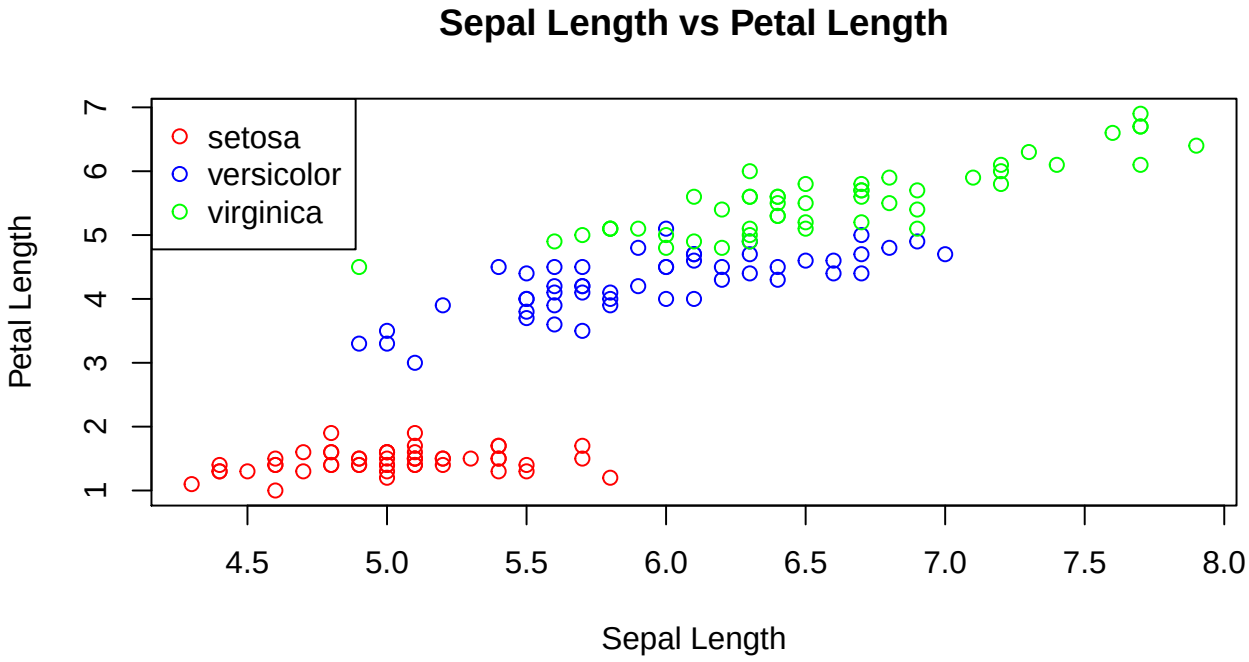
```
plot(x = iris$Sepal.Length, y = iris$Sepal.Width, col = c("red", "blue", "green")[iris$Species]
     main = "Sepal Length vs Sepal Width", xlab = "Sepal Length", ylab = "Sepal Width")
legend(x="topright", legend = levels(iris$Species), col=c("red","blue","green"), pch=1)
```



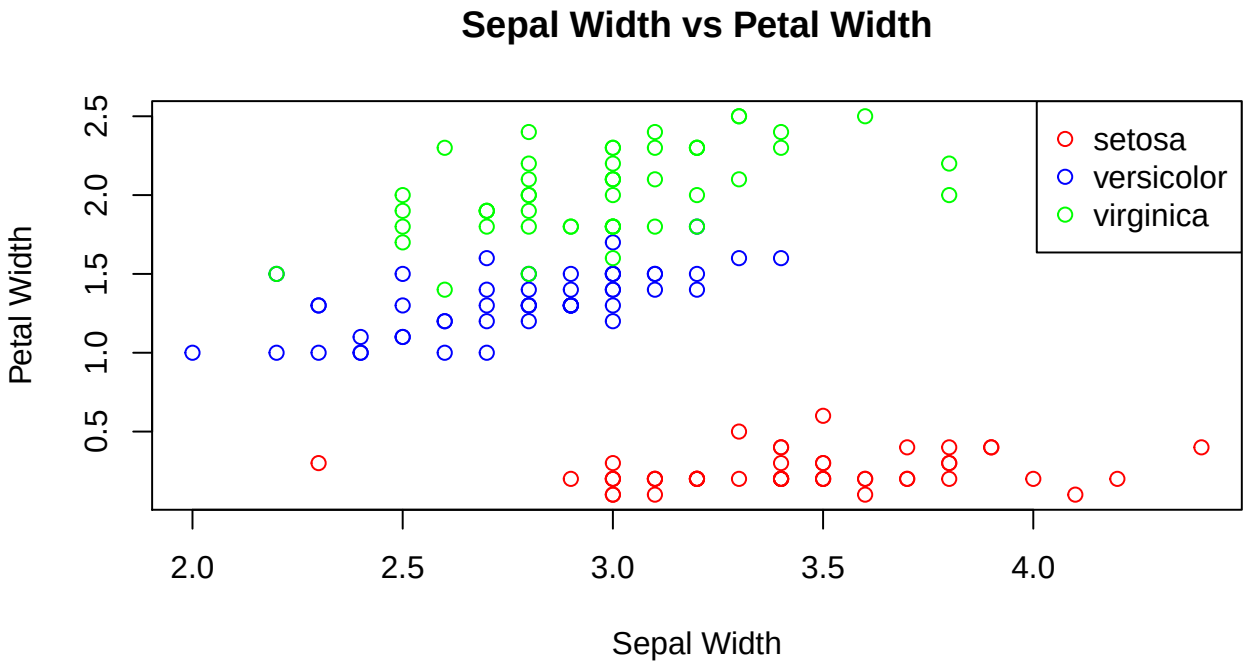

```
plot(x = iris$Petal.Length, y = iris$Petal.Width, col = c("red", "blue", "green")[iris$Species]
     main = "Petal Length vs Petal Width", xlab = "Petal Length", ylab = "Petal Width")
legend(x="topleft", legend = levels(iris$Species), col=c("red","blue","green"), pch=1)
```



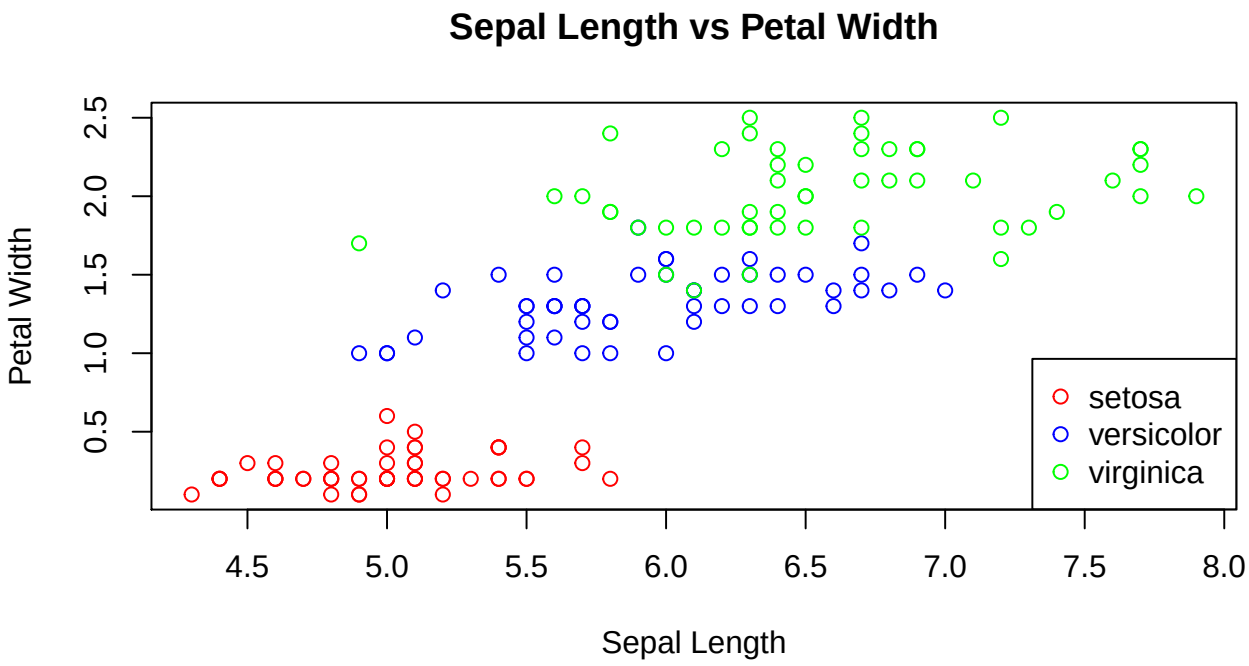

```
plot(x = iris$Sepal.Length, y = iris$Petal.Length, col = c("red", "blue", "green")[iris$Species]
     main = "Sepal Length vs Petal Length", xlab = "Sepal Length", ylab = "Petal Length")
legend(x="topleft", legend = levels(iris$Species), col=c("red","blue","green"), pch=1)
```



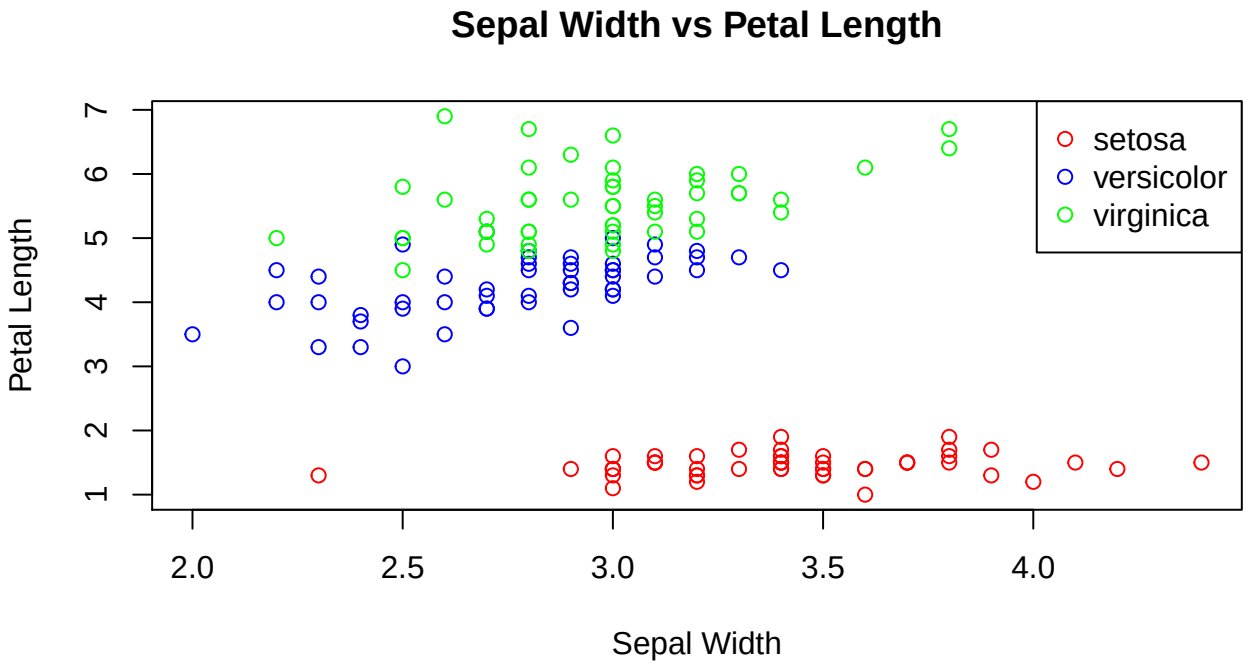

```
plot(x = iris$Sepal.Width, y = iris$Petal.Width, col = c("red", "blue", "green")[iris$Species],
     main = "Sepal Width vs Petal Width", xlab = "Sepal Width", ylab = "Petal Width")
legend(x="topright", legend = levels(iris$Species), col=c("red","blue","green"), pch=1)
```




```
plot(x = iris$Sepal.Length, y = iris$Petal.Width, col = c("red", "blue", "green")[iris$Species],
     main = "Sepal Length vs Petal Width", xlab = "Sepal Length", ylab = "Petal Width")
legend(x="bottomright", legend = levels(iris$Species), col=c("red","blue","green"), pch=1)
```




```
plot(x = iris$Sepal.Width, y = iris$Petal.Length, col = c("red", "blue", "green")[iris$Species]
     main = "Sepal Width vs Petal Length", xlab = "Sepal Width", ylab = "Petal Length")
legend(x="topright", legend = levels(iris$Species), col=c("red","blue","green"), pch=1)
```



3: Sorting

(a): Develop an algorithm (pseudocode) to sort the four features in the dataset.

Use radix sort as a counting sort on vector A for d number of passes. Since we will be sorting an integer vector that has a range less than 100,000, radix sort will use a linear time counting sort as its stable sort.

```
RADIX-SORT(A, d)
```

```
for i = 1 to d
      use linear time counting sort to sort array A on digit i
```

(b): Provide the efficiency (running time) of your sorting algorithm in O-notation.

The radix sort method implemented in the order() function has an asymptotic time complexity of O(n). The sort utilizes hashing to scale time linearly with the input size.

(c): Implement your algorithm in your code of choice.

Sort the four features.

```
s1 <- iris[order(iris$Sepal.Length, decreasing = TRUE, method = "radix") , ]
s2 <- iris[order(iris$Sepal.Width, decreasing = TRUE, method = "radix") , ]
s3 <- iris[order(iris$Petal.Length, decreasing = TRUE, method = "radix") , ]
s4 <- iris[order(iris$Petal.Width, decreasing = TRUE, method = "radix") , ]
```

(d): Determine if any of the four can separate the three plant types.

Create a function to calculate the proportion of observations of each class that fall within the indexes 1:50, 51:100, 101:150. If each class is sorted correctly, each group of 50 would contain the data of only one class.

The 'data' parameter is the already sorted dataset, 'b1' is the lower index boundary and 'b2' is the upper index boundary.

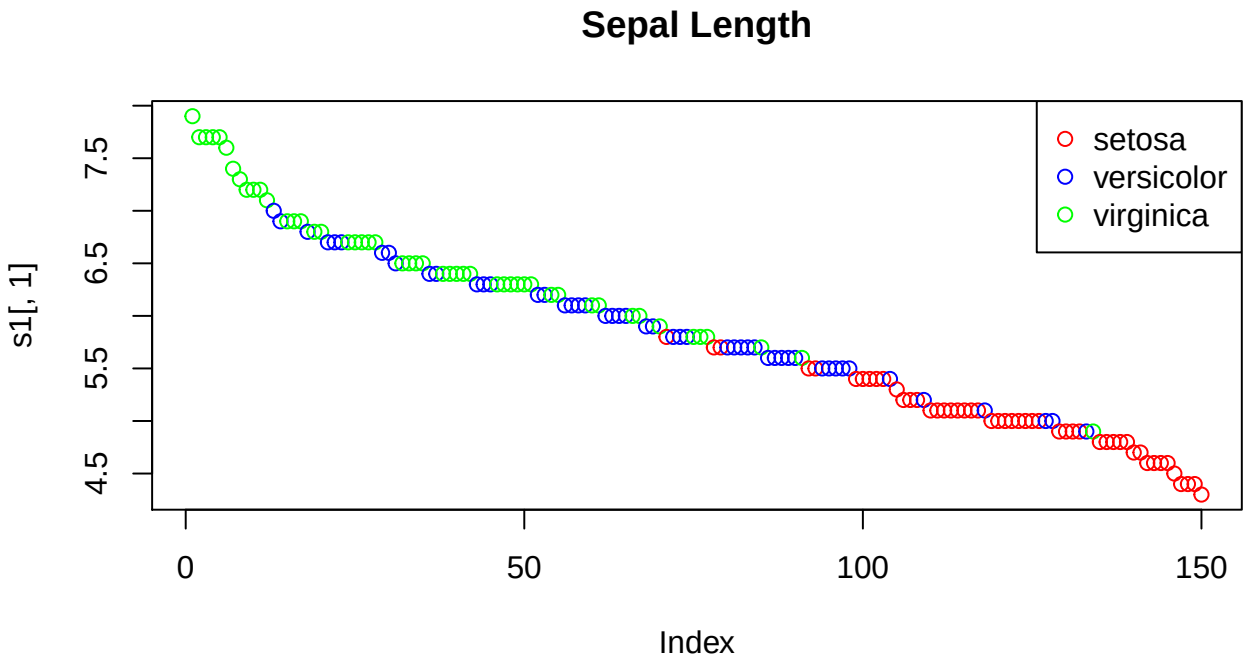
```
propSortCorrect <- function(data, b1, b2){
   p1 <- 0
   p2 <- 0
   p3 <- 0

   for(i in b1:b2){
      if(data[i, 5] == "setosa")
         p1 <- p1 + 1
      if(data[i, 5] == "virginica")
         p2 <- p2 + 1
      if(data[i, 5] == "versicolor")
         p3 <- p3 + 1
   }
   p1 <- p1/50
   p2 <- p2/50
   p3 <- p3/50

   return(c(p1, p2, p3))
}
```

Sepal Length:

```
# plot sorted values by index
plot(s1[,1], col = c("red", "blue", "green")[s1$Species], main = "Sepal Length")
legend(x="topright", legend = levels(s1$Species), col=c("red","blue","green"), pch=1)
```




```
    # FORMAT: % setosa, virginica, versicolor, respectively
    propSortCorrect(s1, 1, 50)
```

```
## [1] 0.00 0.72 0.28
```

```
    propSortCorrect(s1, 51, 100)
```

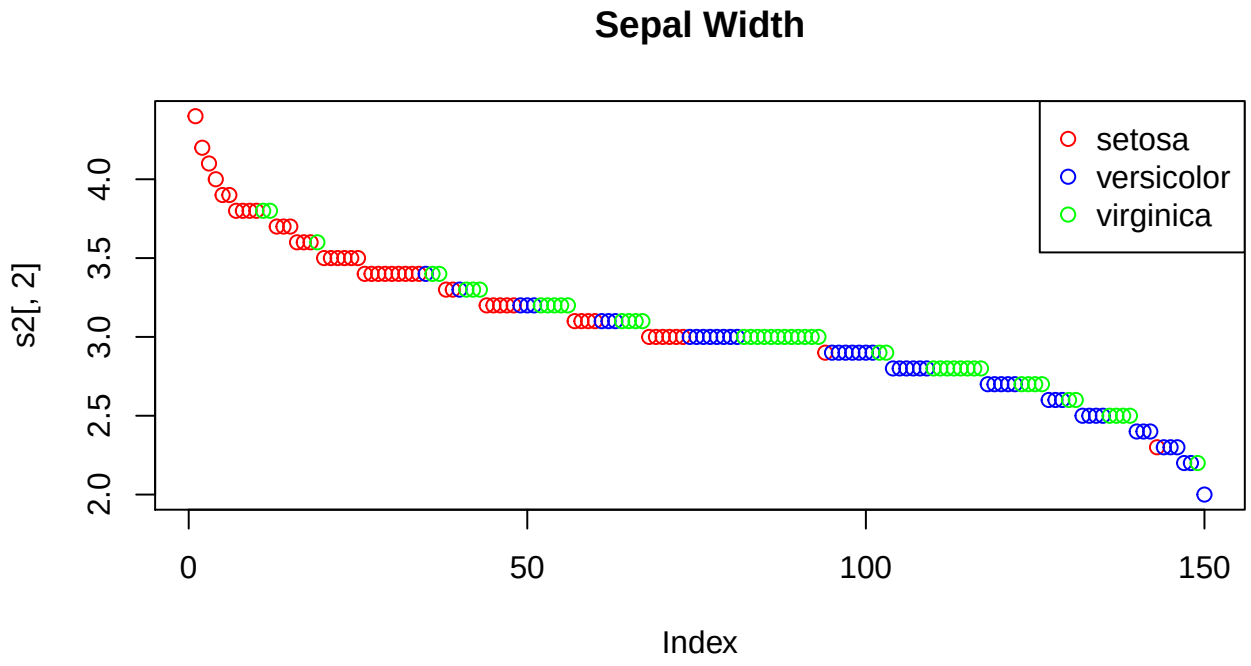
```
## [1] 0.14 0.26 0.60
```

```
    propSortCorrect(s1, 101, 150)
```

```
## [1] 0.86 0.02 0.12
```

Sepal Width:

```
    # plot sorted values by index
    plot(s2[,2], col = c("red", "blue", "green")[s2$Species], main = "Sepal Width")
    legend(x="topright", legend = levels(s2$Species), col=c("red","blue","green"), pch=1)
```




```
    # FORMAT: % setosa, virginica, versicolor, respectively
    propSortCorrect(s2, 1, 50)
```

```
## [1] 0.76 0.16 0.08
```

```
    propSortCorrect(s2, 51, 100)
```

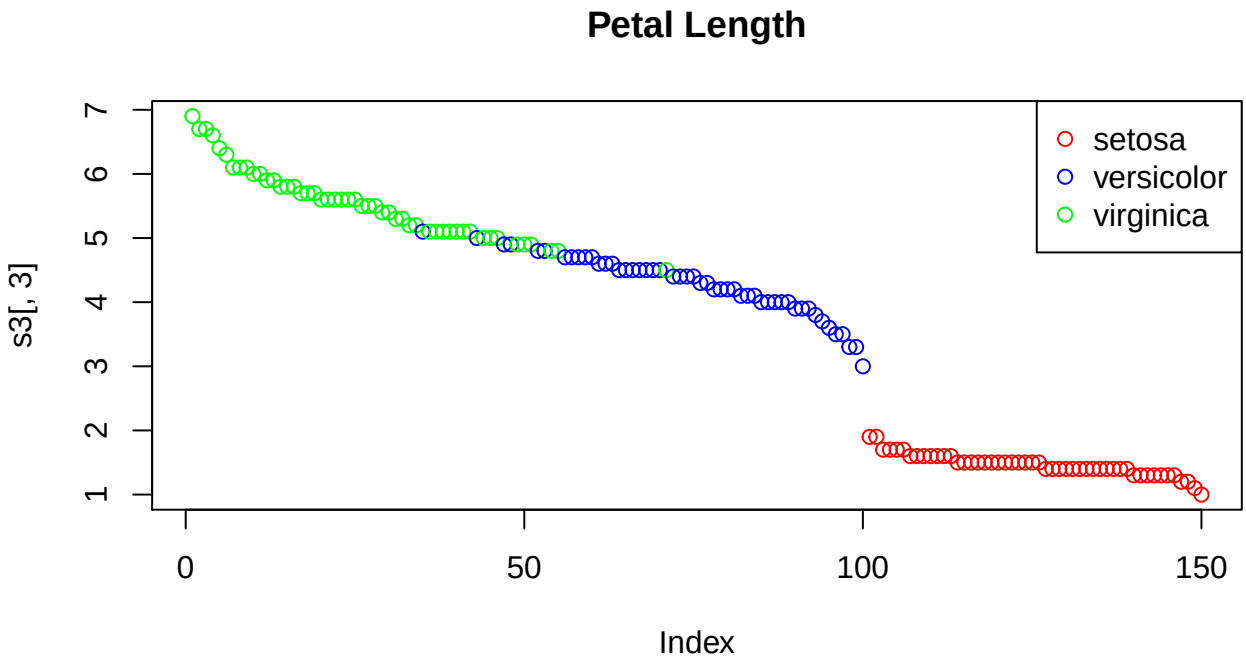
```
## [1] 0.22 0.42 0.36
```

```
    propSortCorrect(s2, 101, 150)
```

```
## [1] 0.02 0.42 0.56
```

Petal Length:

```
    # plot sorted values by index
    plot(s3[,3], col = c("red", "blue", "green")[s3$Species], main = "Petal Length")
    legend(x="topright", legend = levels(s3$Species), col=c("red","blue","green"), pch=1)
```




```
    # FORMAT: % setosa, virginica, versicolor, respectively
    propSortCorrect(s3, 1, 50)
```

```
## [1] 0.00 0.92 0.08
```

```
    propSortCorrect(s3, 51, 100)
```

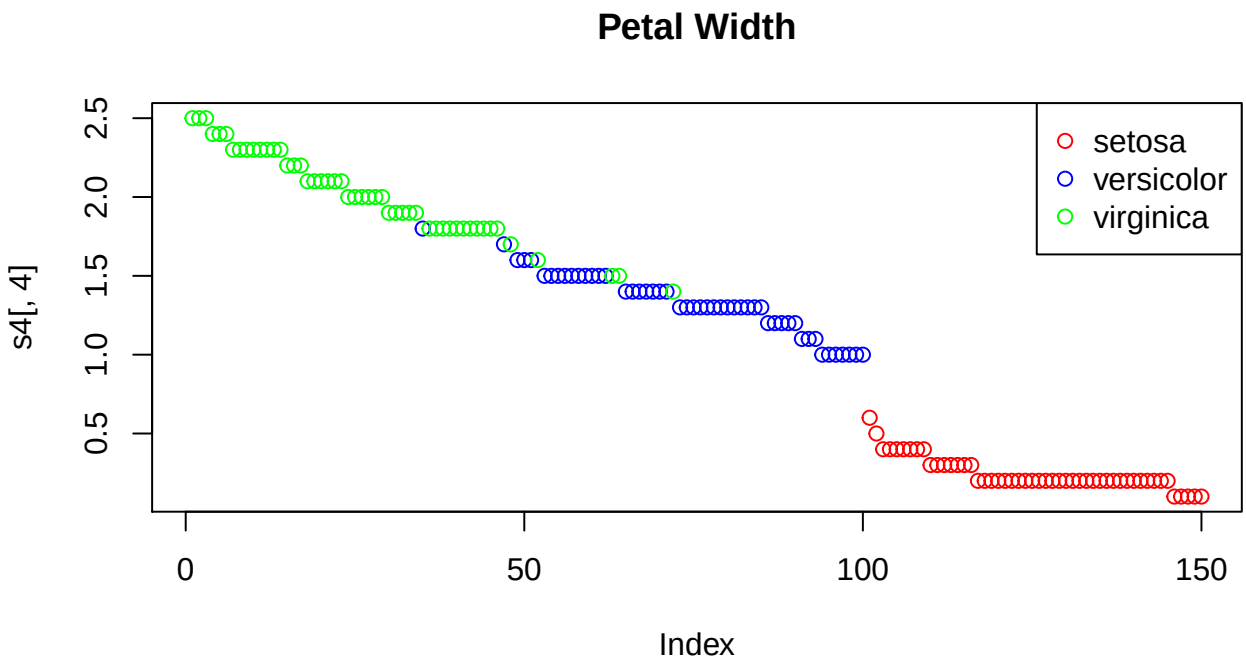
```
## [1] 0.00 0.08 0.92
```

```
    propSortCorrect(s3, 101, 150)
```

```
## [1] 1 0 0
```

Petal Width:

```
    # plot sorted values by index
    plot(s4[,4], col = c("red", "blue", "green")[s4$Species], main = "Petal Width")
    legend(x="topright", legend = levels(s4$Species), col=c("red","blue","green"), pch=1)
```




```
      # FORMAT: % setosa, virginica, versicolor, respectively
      propSortCorrect(s4, 1, 50)
```

```
## [1] 0.00 0.92 0.08
```

```
      propSortCorrect(s4, 51, 100)
```

```
## [1] 0.00 0.08 0.92
```

```
      propSortCorrect(s4, 101, 150)
```

```
## [1] 1 0 0
```

(e) Provide an explanation of the results.

Was there any feature that could separate the plant types?

After examing both the plots of sorted values and the proportion of each plant class (setosa, versicolor, virginica) that were found to be correct based on the overall plot, Sepal Length and Sepal Width do not appear to clearly separate the plant types.

Examining the plots of Petal Length and Petal Width show that these two features may separate the plant types after outliers are removed, but they cannot correctly separate plant types with all of the original data. Each of these features correctly separates Setosa from the other two plant types and has 0.92 (or 46 out of 50) of the other two remaining classes separated correctly.

** Examing the data again after the removal of outliers did not lead to sorting correctly separating the 3 features **

As a broad approach, sorting is not a good way to distinguish between plant types.

What was the metric used to determine separation?

I created a function propSortCorrect() to calculate what proportion of observations were correctly sorted based on the plot of each feature.

My logic for this metric is that if each class is completely separated, the first 50 sorted obs will be one class, the second 50 will be of another class, and the final 50 will be of the final remaining class. By looking at the

plot, we can see which class (Setosa, Virginica, Versicolor) should be first, second, and third, respectively. Looking at this calculated proportion can tell us how well the sorted feature can separate each plant type.

Ex.) Petal Length

Looking at the plot, we expect the first group to be Virginica, the second group to be Versicolor, and the final group to be Setosa. Below is the calculated successful separation of each plant type.

```
# FORMAT: setosa, virginica, versicolor
[1] 0.00 0.92 0.08
[1] 0.00 0.08 0.92
[1] 1 0 0
```

We see that 92% (46 out of 50) of the sorted data are Virginica and 8% (4 out of 50) are Versicolor. Only 4 values in the first 50 indexes were not correctly separated.

We see that 92% (46 out of 50) of the sorted data are Versicolor and 8% (4 out of 50) are Virginica. Only 4 values in the second 50 indexes were not correctly separated.

We see that 100% (50 out of 50) of the sorted data are Setosa and completely separated in the final 50 indexes.

4 Outlier Removal

(a): Develop an algorithm (pseudocode) to remove in sequential order observations that are furthest from the data class mean.

```
WILKS-OUTLIER-REMOVAL(m)
      Calculate n, the length of m
      Calculate p, the number of columns of m
      Calculate mahalanobis distance (which follows Beta distribution)
      Calculate F statistic
      Calculate p value for each row
      Bind results together
```

(b): Provide the running time of your algorithm in O-notation.

Since the malahanobis() function is used within my propose algorithm for Wilks' Outlier removal, to check that the runtime is O(n), I have run the algorithm a number of times with different size of inputs.

```r
Wilks <- function(d){
   n <- nrow(d) # number of rows
   p <- ncol(d) # number of columns
   # beta distribution, since mahalanobis follows beta for mult norm distribution
   u <- n * mahalanobis(d, center = colMeans(d), cov = cov(d)) / (n - 1) ^ 2
   w <- 1 - u
   Fstat <- ((n - p - 1) / p) * (1 / w - 1) # compute F statistic
   p <- 1 - round(pf(Fstat, p, n - p - 1), 3) # compute p value for each row
   cbind(w, Fstat, p) # bind results together
}
```

The function is provided above, since the test of the run time follows:

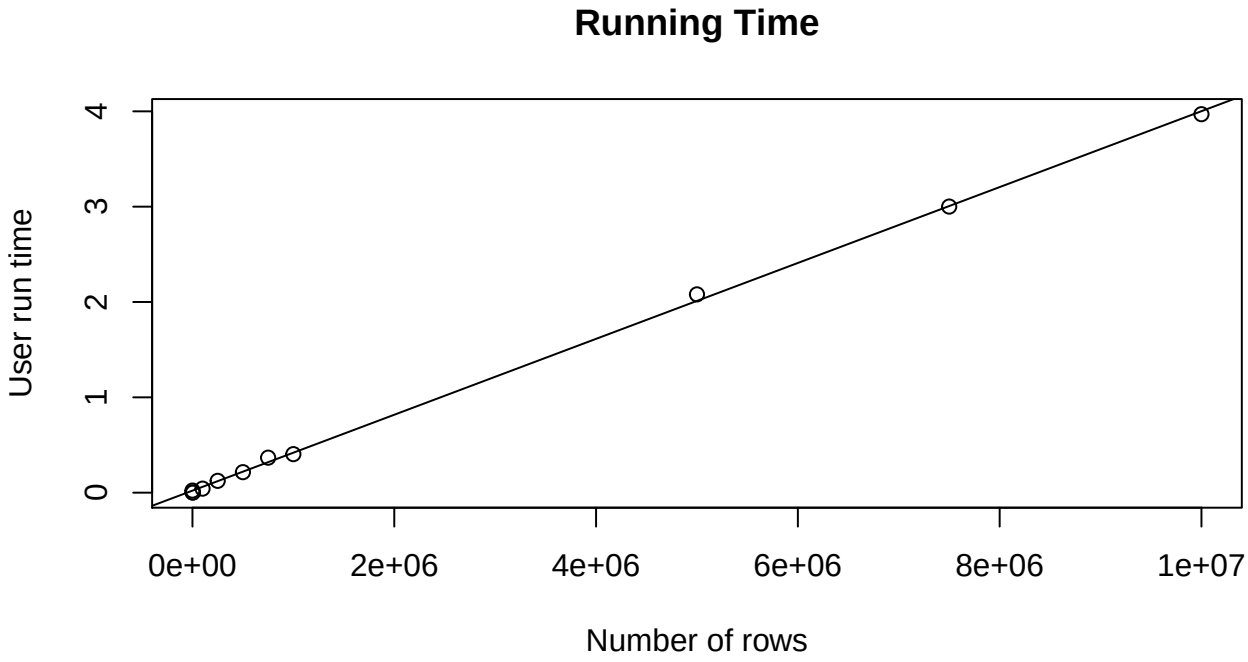
```r
dat <- class1[,-c(1,2,5)] # 150 rows
run0 <- system.time(Wilks(dat))
dat <- cbind(runif(1000), runif(1000)) # 1000 rows
run1 <- system.time(Wilks(dat))
dat <- cbind(runif(5000), runif(5000)) # 5000 rows
run2 <- system.time(Wilks(dat))
```

```r
dat <- cbind(runif(100000), runif(100000)) # 100000 rows
run3 <- system.time(Wilks(dat))
dat <- cbind(runif(250000), runif(250000)) # 250000 rows
run4 <- system.time(Wilks(dat))
dat <- cbind(runif(500000), runif(500000)) # 500000 rows
run5 <- system.time(Wilks(dat))
dat <- cbind(runif(750000), runif(750000)) # 750000 rows
run6 <- system.time(Wilks(dat))
dat <- cbind(runif(1000000), runif(1000000)) # 1000000 rows
run7 <- system.time(Wilks(dat))
dat <- cbind(runif(2500000), runif(2500000)) # 2500000 rows
run8 <- system.time(Wilks(dat))
dat <- cbind(runif(5000000), runif(5000000)) # 5000000 rows
run9 <- system.time(Wilks(dat))
dat <- cbind(runif(7500000), runif(7500000)) # 7500000 rows
run10 <- system.time(Wilks(dat))
dat <- cbind(runif(10000000), runif(10000000)) # 10000000 rows
run11 <- system.time(Wilks(dat))

x = c(150, 1000, 5000, 100000, 250000, 500000, 750000, 1000000, 2500000, 5000000, 7500000, 10000000)
y = c(run0[1], run1[1], run2[1], run3[1], run4[1], run5[1], run6[1], run7[1], run8[8], run9[1], run10[1]
fit <- lm(y ~ x)
summary(fit)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.031656 -0.019966 -0.005583  0.001851  0.068489
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.137e-02  1.180e-02    1.81    0.104
## x           3.980e-07  2.892e-09  137.61 2.87e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03245 on 9 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.9995, Adjusted R-squared:  0.9995
## F-statistic: 1.894e+04 on 1 and 9 DF,  p-value: 2.871e-16
```

```r
plot(x, y, xlab = "Number of rows", ylab = "User run time", main = "Running Time") + abline(fit)
```




```
## integer(0)
```

After graphing the plot above and taking a look at the linear model, it appears that a linear model fits the data, supporting that the runtime is O(n).

(c): Implement your algorithm in your code of choice.

Use the existing mahalanobis() function to write a function to perform Wilks' Outlier Removal.

The F statistic and p-value for each value will also be calculated and used to determine if the value is extreme enough to be considered an outlier.

```
Wilks <- function(d){
   n <- nrow(d) # number of rows
   p <- ncol(d) # number of columns
   # beta distribution, since mahalanobis follows beta for mult norm distribution
   u <- n * mahalanobis(d, center = colMeans(d), cov = cov(d)) / (n - 1) ^ 2
   w <- 1 - u
   Fstat <- ((n - p - 1) / p) * (1 / w - 1) # compute F statistic
   p <- 1 - round(pf(Fstat, p, n - p - 1), 3) # p value for each row
   cbind(w, Fstat, p) # bind results together
}
```

(d): Determine if the data contains an outlier by plotting each class individually.

Remove outliers from Class 1, Petal Width and Petal Length

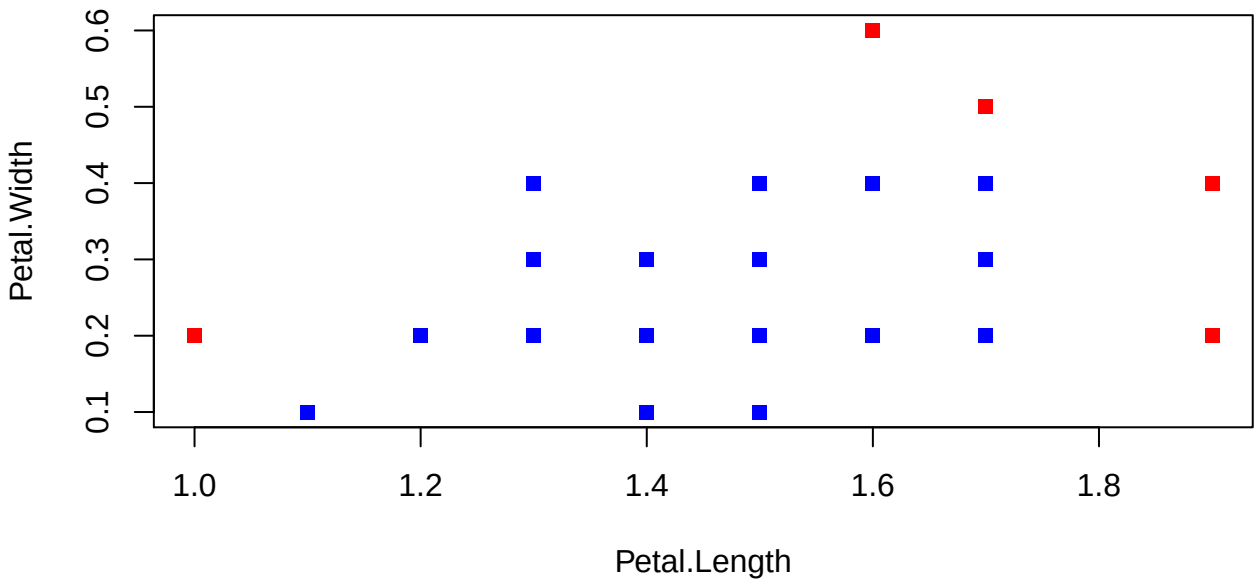
After removing an outlier from the original Class 1 data, create a new df to reference for the remaining set of features, so that outliers are not removed multiple times.

```
      # remove the two desired features
      dat <- class1[,-c(1,2,5)]
      # plot the two features
      plot(dat, col = "blue", pch = c(15,16,17)[as.numeric(iris$Species)])
      # use Wilks' Outlier Removal
```

```
      dat.rows <- Wilks(dat)
      head(dat.rows)
```

```
##           w     Fstat     p
## 1 0.9949782 0.1186074 0.888
## 2 0.9949782 0.1186074 0.888
## 3 0.9815006 0.4429310 0.645
## 4 0.9929396 0.1670991 0.847
## 5 0.9949782 0.1186074 0.888
## 6 0.9371699 1.5754949 0.218
```

```
      # return indexes which contain outliers
      outliers <- which(dat.rows[,"p"] < 0.05)
      # plot the outliers in red
      points(dat[outliers,], col = "red", pch = c(15,16,17)[as.numeric(iris$Species[outliers])])
```



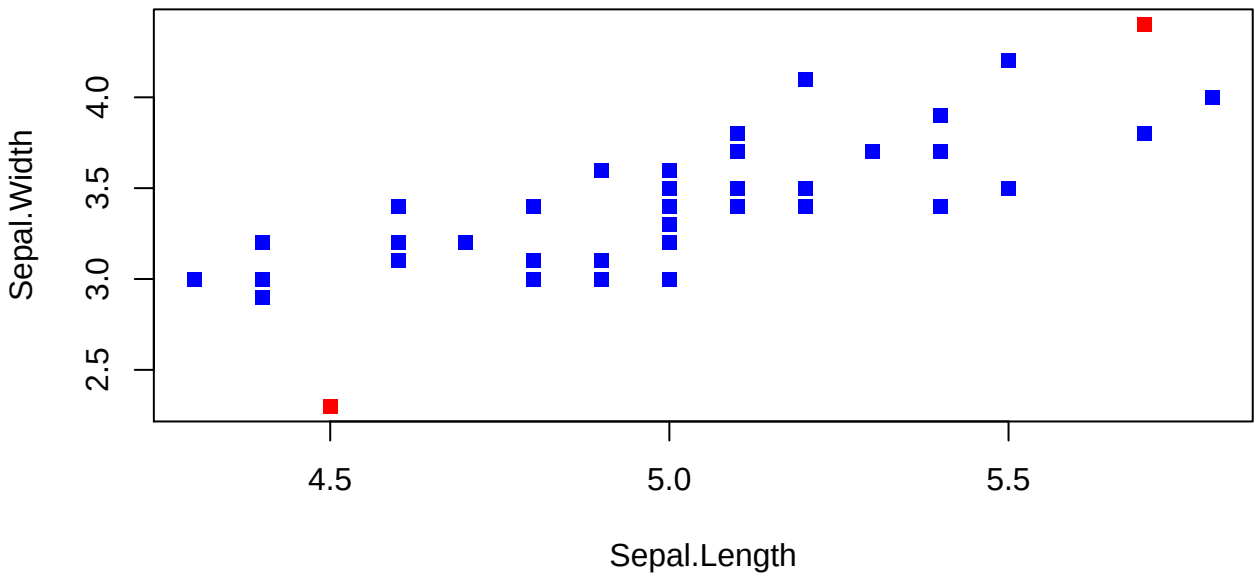

```
      # remove the outliers from the class
      class1OR <- class1[-outliers,]
```

Remove outliers from Class 1, Sepal Width and Sepal Length

```
      # remove the two desired features
      dat <- class1OR[,-c(3,4,5)]
      # plot the two features
      plot(dat, col = "blue", pch = c(15,16,17)[as.numeric(iris$Species)])
      # use Wilks' Outlier Removal
      dat.rows <- Wilks(dat)
      head(dat.rows)
```

```
##           w      Fstat     p
## 1 0.9987059 0.02721125 0.973
## 2 0.9591439 0.89452384 0.416
## 3 0.9818093 0.38908266 0.680
## 4 0.9695140 0.66033746 0.522
## 5 0.9857032 0.30458686 0.739
## 6 0.9646038 0.77059639 0.469
```

```
      # return indexes which contain outliers
      outliers <- which(dat.rows[,"p"] < 0.05)
      # plot the outliers in red
      points(dat[outliers,], col = "red", pch = c(15,16,17)[as.numeric(iris$Species[outliers])])
```



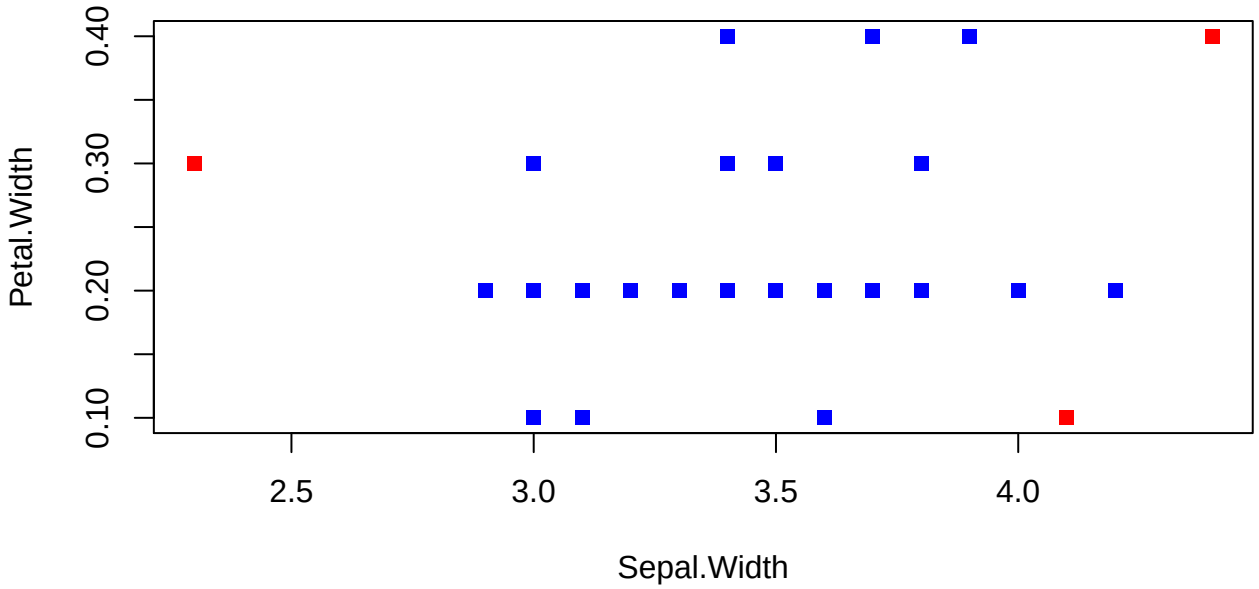

```
      # remove the outliers from the class
      classOR <- class1OR[-outliers,]
```

Remove outliers from Class 1, Petal Width and Sepal Width

```
      # remove the two desired features
      dat <- class1OR[,-c(1,3,5)]
      # plot the two features
      plot(dat, col = "blue", pch = c(15,16,17)[as.numeric(iris$Species)])
      # use Wilks' Outlier Removal
      dat.rows <- Wilks(dat)
      head(dat.rows)
```

```
##           w     Fstat     p
## 1 0.9944639 0.1169042 0.890
## 2 0.9737821 0.5653998 0.572
## 3 0.9917533 0.1746210 0.840
## 4 0.9843841 0.3331362 0.719
## 5 0.9889019 0.2356763 0.791
## 6 0.8963761 2.4276670 0.101
```

```
      # return indexes which contain outliers
      outliers <- which(dat.rows[,"p"] < 0.05)
      # plot the outliers in red
      points(dat[outliers,], col = "red", pch = c(15,16,17)[as.numeric(iris$Species[outliers])])
```



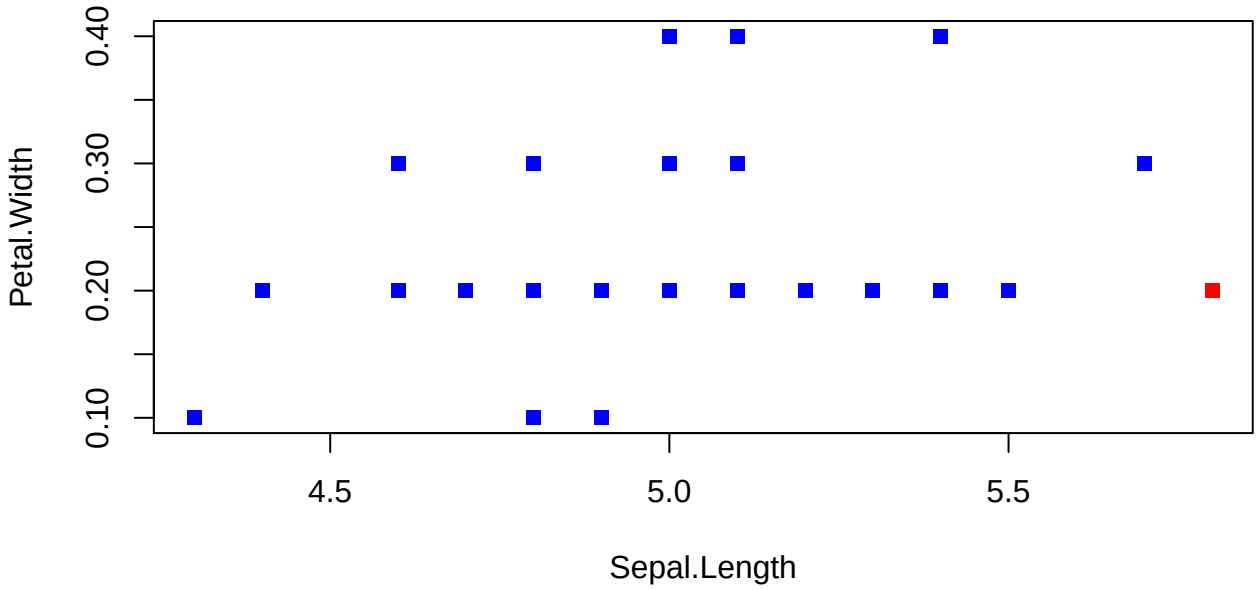

```
      # remove the outliers from the class
      class1OR <- class1OR[-outliers,]
```

Remove outliers from Class 1, Petal Width and Sepal Length

```
      # remove the two desired features
      dat <- class1OR[,-c(2,3,5)]
      # plot the two features
      plot(dat, col = "blue", pch = c(15,16,17)[as.numeric(iris$Species)])
      # use Wilks' Outlier Removal
      dat.rows <- Wilks(dat)
      head(dat.rows)
```

```
##           w      Fstat     p
## 1 0.9927177 0.14304615 0.867
## 2 0.9958919 0.08043858 0.923
## 3 0.9809000 0.37970150 0.687
## 4 0.9665919 0.67397505 0.516
## 5 0.9965756 0.06700607 0.935
## 6 0.8822322 2.60302522 0.087
```

```
      # return indexes which contain outliers
      outliers <- which(dat.rows[,"p"] < 0.05)
      # plot the outliers in red
      points(dat[outliers,], col = "red", pch = c(15,16,17)[as.numeric(iris$Species[outliers])])
```



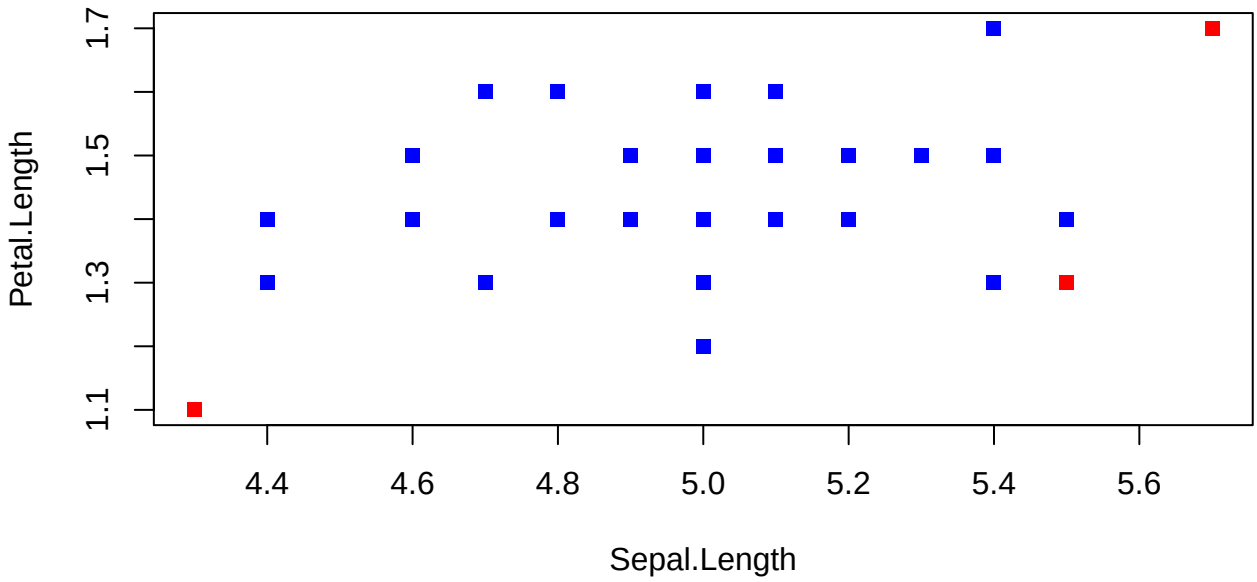

```
      # remove the outliers from the class
      class1OR <- class1OR[-outliers,]
```

Remove outliers from Class 1, Sepal Length and Petal Length

```
      # remove the two desired features
      dat <- class1OR[,-c(2,4,5)]
      # plot the two features
      plot(dat, col = "blue", pch = c(15,16,17)[as.numeric(iris$Species)])
      # use Wilks' Outlier Removal
      dat.rows <- Wilks(dat)
      head(dat.rows)
```

```
##           w      Fstat     p
## 1 0.9886088 0.21892751 0.804
## 2 0.9958340 0.07948478 0.924
## 3 0.9618271 0.75406993 0.477
## 4 0.9432534 1.14305031 0.330
## 5 0.9949959 0.09555630 0.909
## 6 0.9030038 2.04088529 0.144
```

```
      # return indexes which contain outliers
      outliers <- which(dat.rows[,"p"] < 0.05)
      # plot the outliers in red
      points(dat[outliers,], col = "red", pch = c(15,16,17)[as.numeric(iris$Species[outliers])])
```



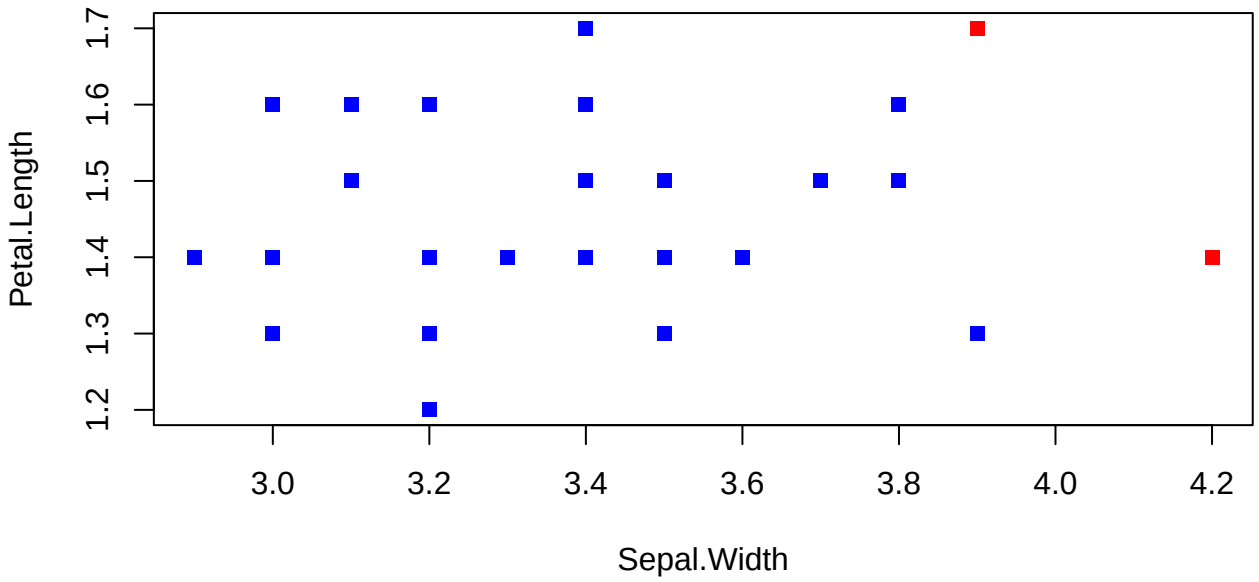

```
      # remove the outliers from the class
      class1OR <- class1OR[-outliers,]
```

Remove outliers from Class 1, Sepal Width and Petal Length

```
      # remove the two desired features
      dat <- class1OR[,-c(1,4,5)]
      # plot the two features
      plot(dat, col = "blue", pch = c(15,16,17)[as.numeric(iris$Species)])
      # use Wilks' Outlier Removal
      dat.rows <- Wilks(dat)
      head(dat.rows)
```

```
##           w     Fstat     p
## 1 0.9874626 0.2221895 0.802
## 2 0.9527423 0.8680313 0.429
## 3 0.9431950 1.0539578 0.359
## 4 0.9684991 0.5691958 0.571
## 5 0.9762343 0.4260238 0.656
## 6 0.8242980 3.7301870 0.034
```

```
      # return indexes which contain outliers
      outliers <- which(dat.rows[,"p"] < 0.05)
      # plot the outliers in red
      points(dat[outliers,], col = "red", pch = c(15,16,17)[as.numeric(iris$Species[outliers])])
```



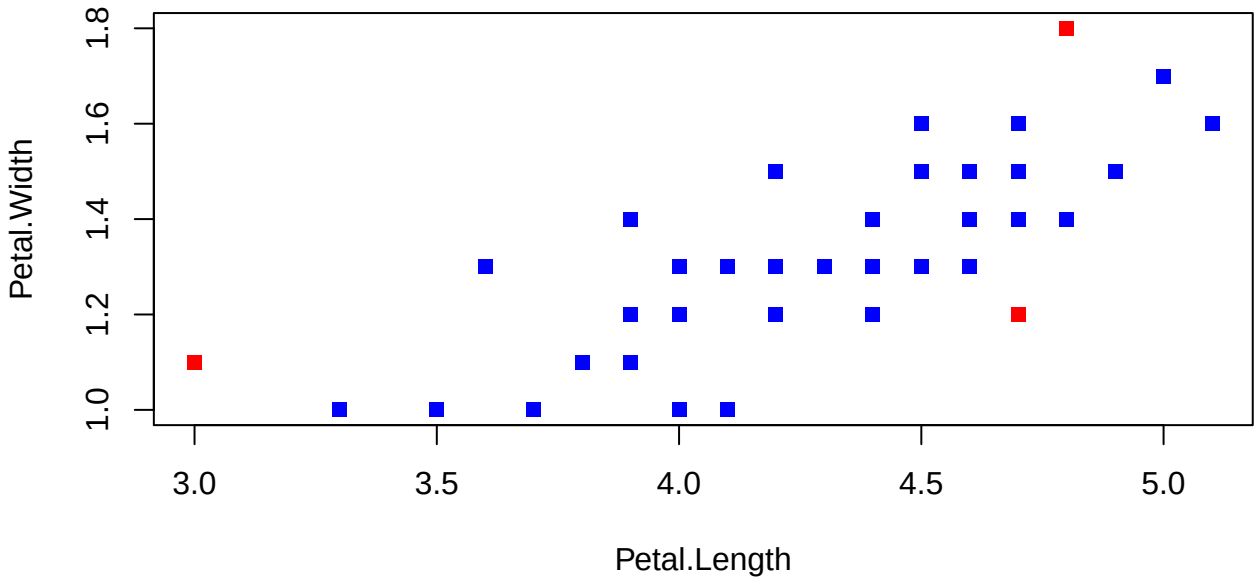

```
      # remove the outliers from the class
      class1OR <- class1OR[-outliers,]
```

Remove outliers from Class 2, Petal Width and Petal Length

```
      # remove the two desired features
      dat <- class2[,-c(1,2,5)]
      # plot the two features
      plot(dat, col = "blue", pch = c(15,16,17)[as.numeric(iris$Species)])
      # use Wilks' Outlier Removal
      dat.rows <- Wilks(dat)
      head(dat.rows)
```

```
##            w     Fstat     p
## 51 0.9745669 0.6132751 0.546
## 52 0.9820788 0.4288331 0.654
## 53 0.9593675 0.9953057 0.377
## 54 0.9885828 0.2714035 0.763
## 55 0.9838238 0.3863913 0.682
## 56 0.9790315 0.5033137 0.608
```

```
      # return indexes which contain outliers
      outliers <- which(dat.rows[,"p"] < 0.05)
      # plot the outliers in red
      points(dat[outliers,], col = "red", pch = c(15,16,17)[as.numeric(iris$Species[outliers])])
```



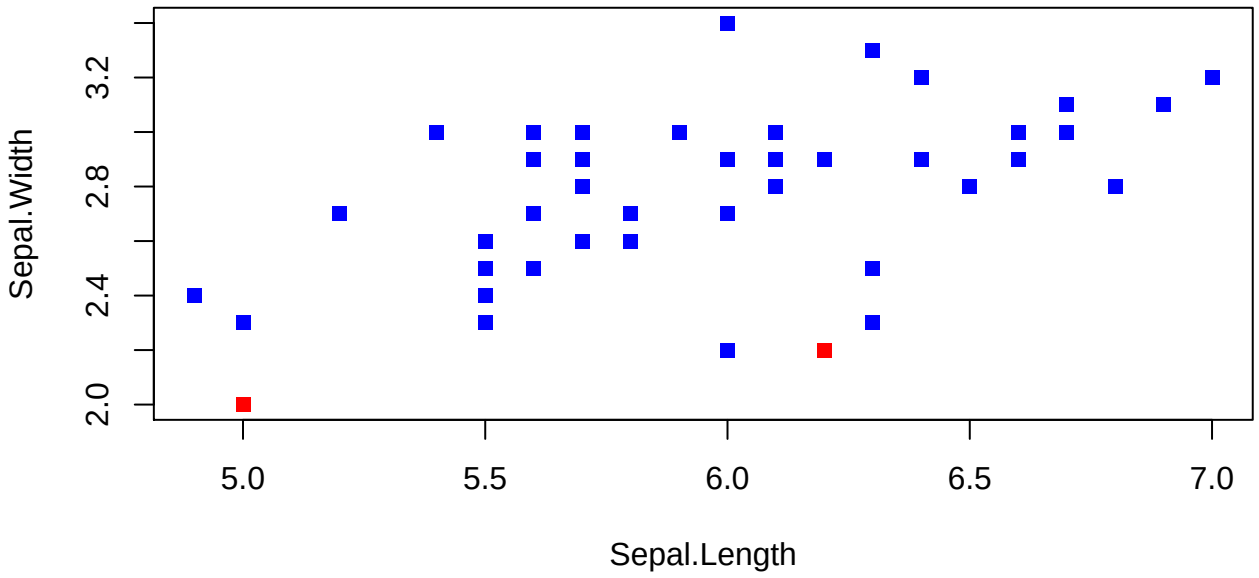

```
      # remove the outliers from the class
      class2OR <- class2[-outliers,]
```

Remove outliers from Class 2, Sepal Width and Sepal Length

```
      # remove the two desired features
      dat <- class2OR[,-c(3,4,5)]
      # plot the two features
      plot(dat, col = "blue", pch = c(15,16,17)[as.numeric(iris$Species)])
      # use Wilks' Outlier Removal
      dat.rows <- Wilks(dat)
      head(dat.rows)
```

```
##            w     Fstat     p
## 51 0.9058312 2.2870855 0.114
## 52 0.9572373 0.9828067 0.382
## 53 0.9250466 1.7825853 0.180
## 54 0.9511593 1.1296682 0.332
## 55 0.9686995 0.7108623 0.497
## 56 0.9906806 0.2069547 0.814
```

```
      # return indexes which contain outliers
      outliers <- which(dat.rows[,"p"] < 0.05)
      # plot the outliers in red
      points(dat[outliers,], col = "red", pch = c(15,16,17)[as.numeric(iris$Species[outliers])])
```



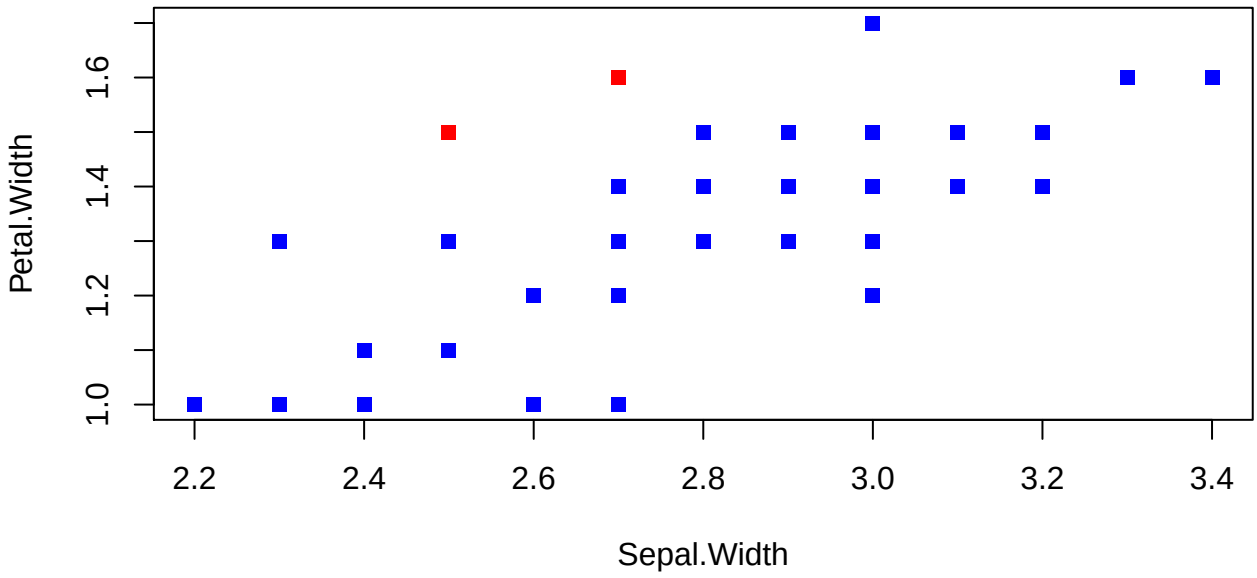

```
      # remove the outliers from the class
      class2OR <- class2OR[-outliers,]
```

Remove outliers from Class 2, Petal Width and Sepal Width

```
      # remove the two desired features
      dat <- class2OR[,-c(1,3,5)]
      # plot the two features
      plot(dat, col = "blue", pch = c(15,16,17)[as.numeric(iris$Species)])
      # use Wilks' Outlier Removal
      dat.rows <- Wilks(dat)
      head(dat.rows)
```

```
##            w      Fstat     p
## 51 0.9389466 1.36549032 0.266
## 52 0.9537266 1.01888831 0.370
## 53 0.9718751 0.60771376 0.549
## 54 0.8814304 2.82491106 0.071
## 55 0.9612890 0.84566655 0.436
## 56 0.9989134 0.02284332 0.977
```

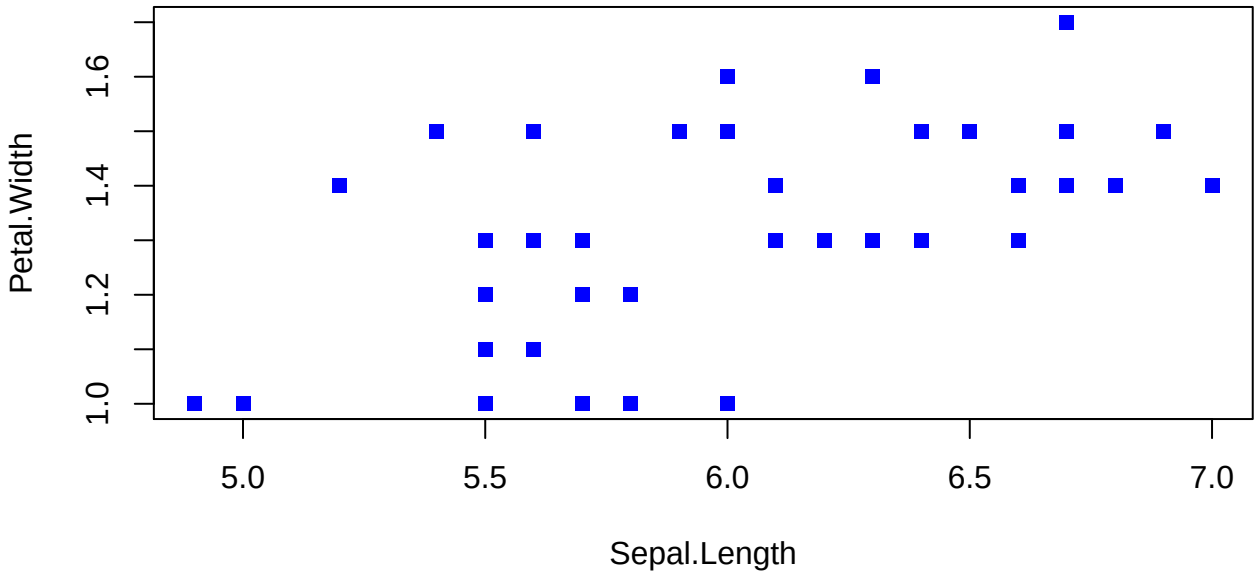
```
      # return indexes which contain outliers
      outliers <- which(dat.rows[,"p"] < 0.05)
      # plot the outliers in red
      points(dat[outliers,], col = "red", pch = c(15,16,17)[as.numeric(iris$Species[outliers])])
```




```
# remove the outliers from the class
class2OR <- class2OR[-outliers,]
```

Remove outliers from Class 2, Petal Width and Sepal Length

```
# remove the two desired features
dat <- class2OR[,-c(2,3,5)]
# plot the two features
plot(dat, col = "blue", pch = c(15,16,17)[as.numeric(iris$Species)])
```




```
# use Wilks' Outlier Removal
dat.rows <- Wilks(dat)
head(dat.rows)
```

```
##            w     Fstat     p
## 51 0.8865167 2.5602071 0.090
## 52 0.9718333 0.5796607 0.565
## 53 0.9190800 1.7608922 0.185
```

```
## 54 0.9753678 0.5050848 0.607
## 55 0.9665045 0.6931273 0.506
## 56 0.9927378 0.1463068 0.864
```

```
      # return indexes which contain outliers
      outliers <- which(dat.rows[,"p"] < 0.05)
      length(outliers) # no outliers
```

```
## [1] 0
```

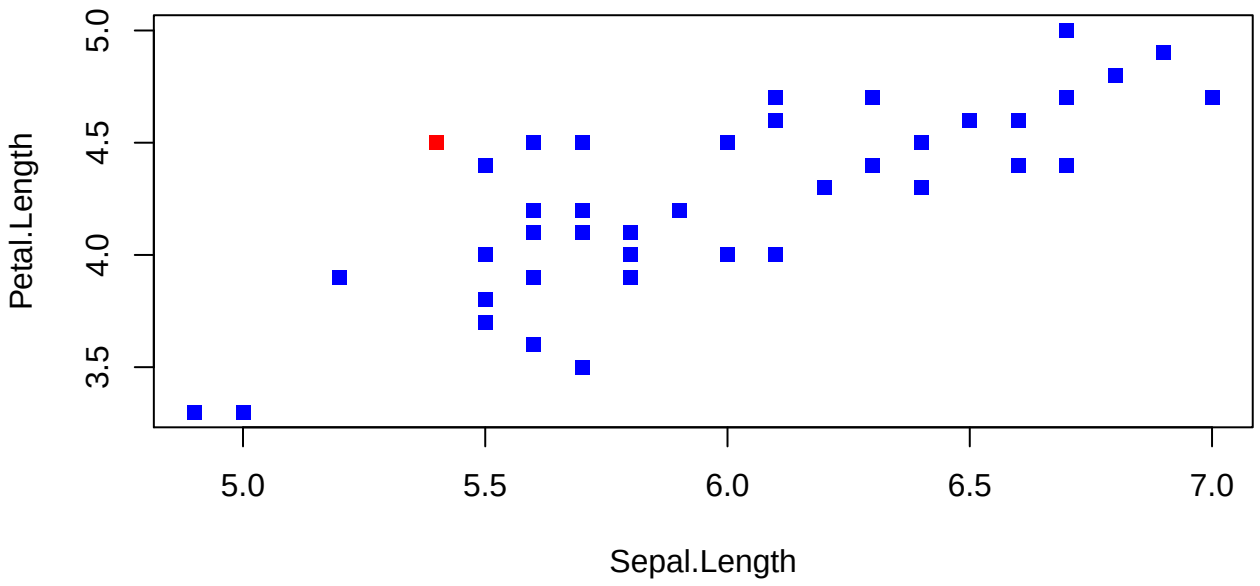
This feature combination (Petal Width and Sepal Length) in Class 2 has no outliers.

Remove outliers from Class 2, Sepal Length and Petal Length

```
      # remove the two desired features
      dat <- class2OR[,-c(2,4,5)]
      # plot the two features
      plot(dat, col = "blue", pch = c(15,16,17)[as.numeric(iris$Species)])
      # use Wilks' Outlier Removal
      dat.rows <- Wilks(dat)
      head(dat.rows)
```

```
##           w     Fstat     p
## 51 0.8898547 2.4755806 0.097
## 52 0.9821382 0.3637335 0.697
## 53 0.9165307 1.8214173 0.175
## 54 0.9804472 0.3988555 0.674
## 55 0.9729562 0.5559103 0.578
## 56 0.9329279 1.4378833 0.249
```

```
      # return indexes which contain outliers
      outliers <- which(dat.rows[,"p"] < 0.05)
      # plot the outliers in red
      points(dat[outliers,], col = "red", pch = c(15,16,17)[as.numeric(iris$Species[outliers])])
```



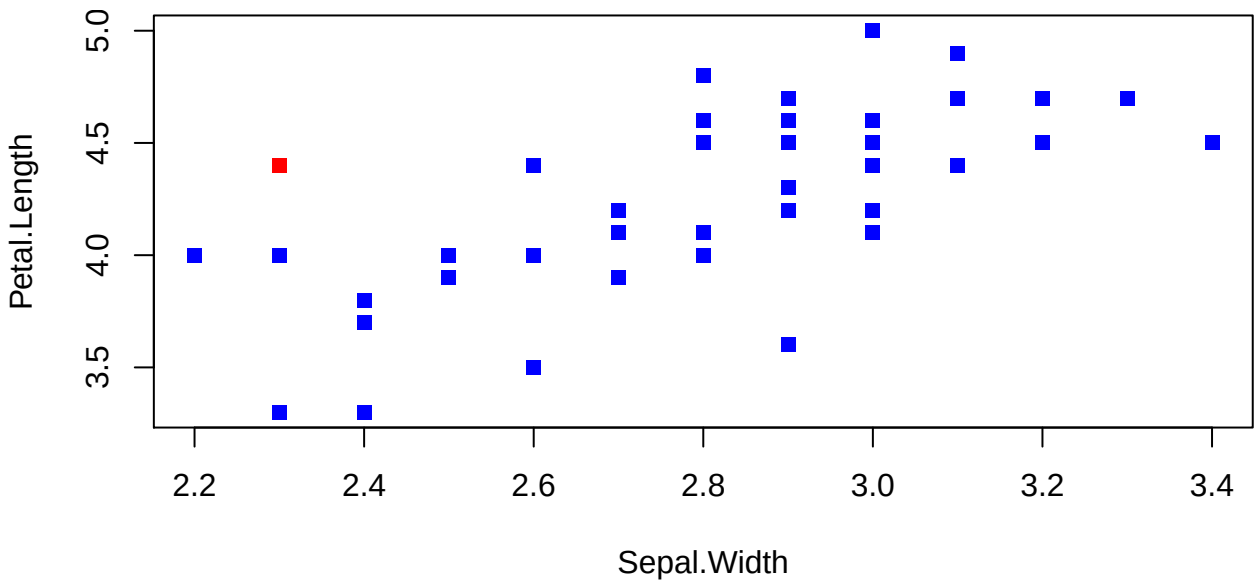

```
      # remove the outliers from the class
      class2OR <- class2OR[-outliers,]
```

Remove outliers from Class 2, Sepal Width and Petal Length

```
      # remove the two desired features
      dat <- class2OR[,-c(1,4,5)]
      # plot the two features
      plot(dat, col = "blue", pch = c(15,16,17)[as.numeric(iris$Species)])
      # use Wilks' Outlier Removal
      dat.rows <- Wilks(dat)
      head(dat.rows)
```

```
##           w     Fstat     p
## 51 0.9507108 1.0109690 0.373
## 52 0.9498569 1.0294086 0.367
## 53 0.9351016 1.3533481 0.270
## 54 0.9132979 1.8511934 0.171
## 55 0.9659435 0.6875152 0.509
## 56 0.9821640 0.3541182 0.704
```

```
      # return indexes which contain outliers
      outliers <- which(dat.rows[,"p"] < 0.05)
      # plot the outliers in red
      points(dat[outliers,], col = "red", pch = c(15,16,17)[as.numeric(iris$Species[outliers])])
```




```
      # remove the outliers from the class
      class2OR <- class2OR[-outliers,]
```

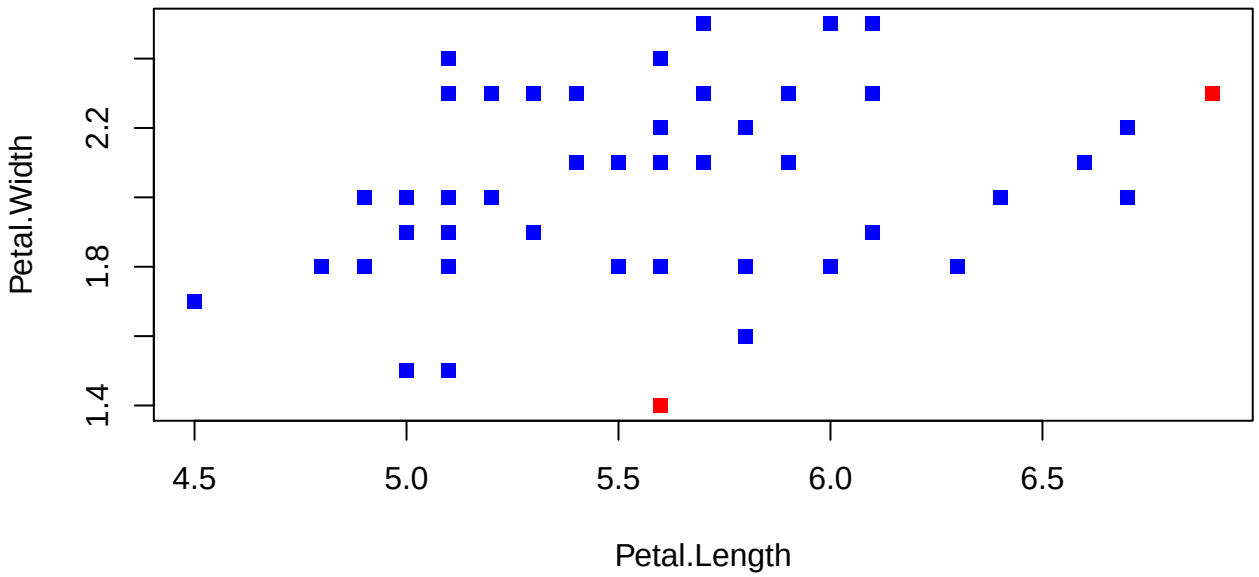
Remove outliers from Class 3, Petal Width and Petal Length

```
      # remove the two desired features
      dat <- class3[,-c(1,2,5)]
      # plot the two features
      plot(dat, col = "blue", pch = c(15,16,17)[as.numeric(iris$Species)])
      # use Wilks' Outlier Removal
      dat.rows <- Wilks(dat)
      head(dat.rows)
```

```
##           w     Fstat     p
```

```
## 101 0.9364529 1.5946951 0.214
## 102 0.9851486 0.3542699 0.704
## 103 0.9916179 0.1986443 0.821
## 104 0.9830201 0.4059200 0.669
## 105 0.9902437 0.2315330 0.794
## 106 0.9221878 1.9828801 0.149
```

```
      # return indexes which contain outliers
      outliers <- which(dat.rows[,"p"] < 0.05)
      # plot the outliers in red
      points(dat[outliers,], col = "red", pch = c(15,16,17)[as.numeric(iris$Species[outliers])])
```



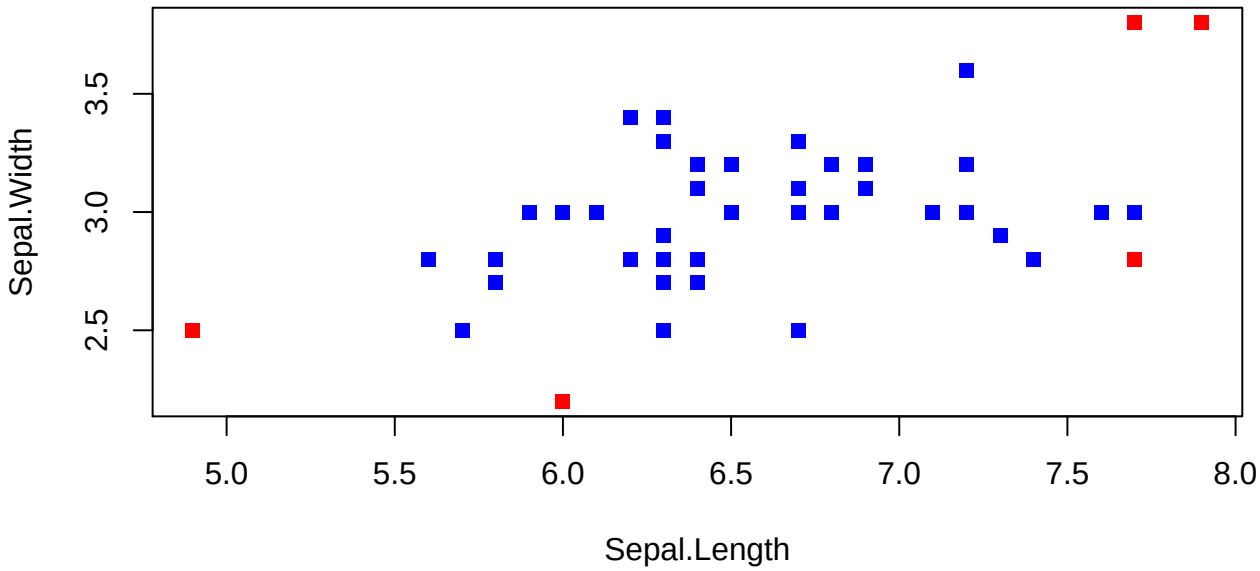

```
      # remove the outliers from the class
      class3OR <- class3[-outliers,]
```

Remove outliers from Class 3, Sepal Width and Sepal Length

```
      # remove the two desired features
      dat <- class3OR[,-c(3,4,5)]
      # plot the two features
      plot(dat, col = "blue", pch = c(15,16,17)[as.numeric(iris$Species)])
      # use Wilks' Outlier Removal
      dat.rows <- Wilks(dat)
      head(dat.rows)
```

```
##             w      Fstat     p
## 101 0.9533264 1.10156945 0.341
## 102 0.9644415 0.82956319 0.443
## 103 0.9798905 0.46175045 0.633
## 104 0.9957027 0.09710572 0.908
## 105 0.9994224 0.01300317 0.987
## 106 0.9218813 1.90661380 0.160
```

```
      # return indexes which contain outliers
      outliers <- which(dat.rows[,"p"] < 0.05)
      # plot the outliers in red
      points(dat[outliers,], col = "red", pch = c(15,16,17)[as.numeric(iris$Species[outliers])])
```



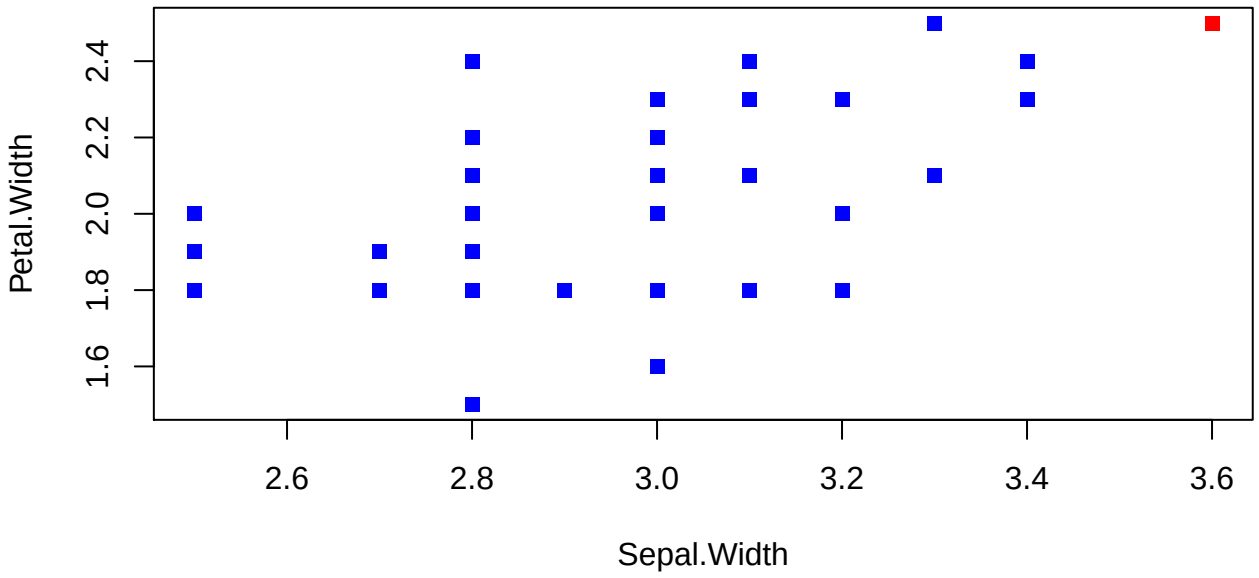

```
    # remove the outliers from the class
    class3OR <- class3OR[-outliers,]
```

Remove outliers from Class 3, Petal Width and Sepal Width

```
    # remove the two desired features
    dat <- class3OR[,-c(1,3,5)]
    # plot the two features
    plot(dat, col = "blue", pch = c(15,16,17)[as.numeric(iris$Species)])
    # use Wilks' Outlier Removal
    dat.rows <- Wilks(dat)
    head(dat.rows)
```

```
##             w      Fstat     p
## 101 0.9230058 1.66833709 0.201
## 102 0.9675646 0.67045469 0.517
## 103 0.9990366 0.01928693 0.981
## 104 0.9754422 0.50352211 0.608
## 105 0.9893498 0.21529700 0.807
## 106 0.9990366 0.01928693 0.981
```

```
    # return indexes which contain outliers
    outliers <- which(dat.rows[,"p"] < 0.05)
    # plot the outliers in red
    points(dat[outliers,], col = "red", pch = c(15,16,17)[as.numeric(iris$Species[outliers])])
```



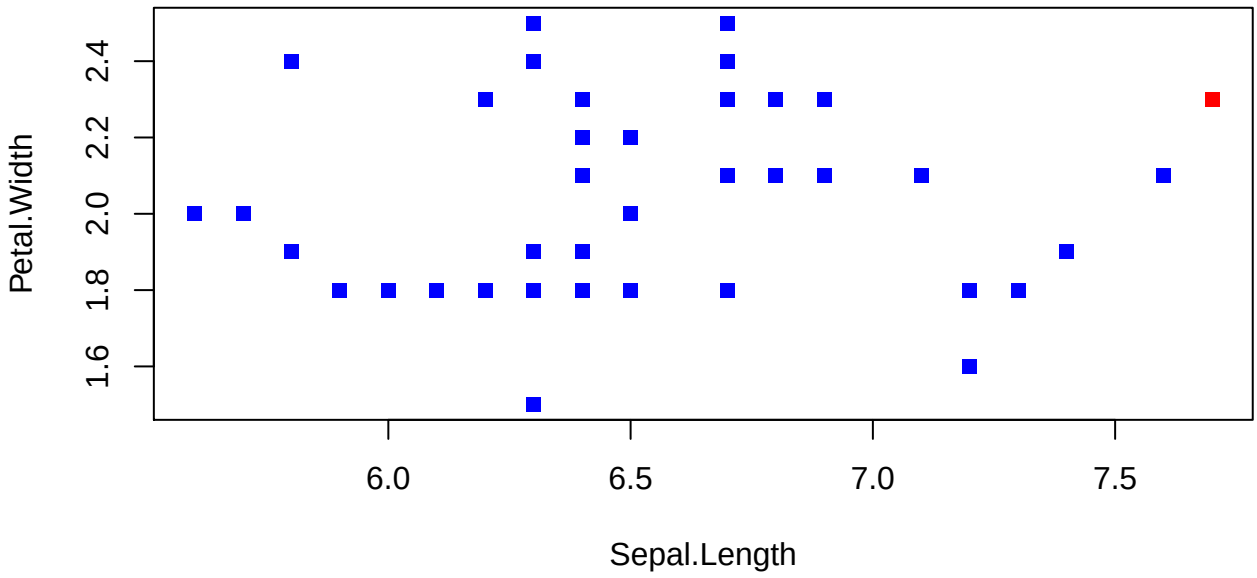

```
      # remove the outliers from the class
      class3OR <- class3OR[-outliers,]
```

Remove outliers from Class 3, Petal Width and Sepal Length

```
      # remove the two desired features
      dat <- class3OR[,-c(2,3,5)]
      # plot the two features
      plot(dat, col = "blue", pch = c(15,16,17)[as.numeric(iris$Species)])
      # use Wilks' Outlier Removal
      dat.rows <- Wilks(dat)
      head(dat.rows)
```

```
##             w     Fstat     p
## 101 0.9055870 2.0329955 0.145
## 102 0.9436727 1.1639449 0.323
## 103 0.9675718 0.6535439 0.526
## 104 0.9739292 0.5219886 0.597
## 105 0.9895105 0.2067135 0.814
## 106 0.8872027 2.4791934 0.097
```

```
      # return indexes which contain outliers
      outliers <- which(dat.rows[,"p"] < 0.05)
      # plot the outliers in red
      points(dat[outliers,], col = "red", pch = c(15,16,17)[as.numeric(iris$Species[outliers])])
```



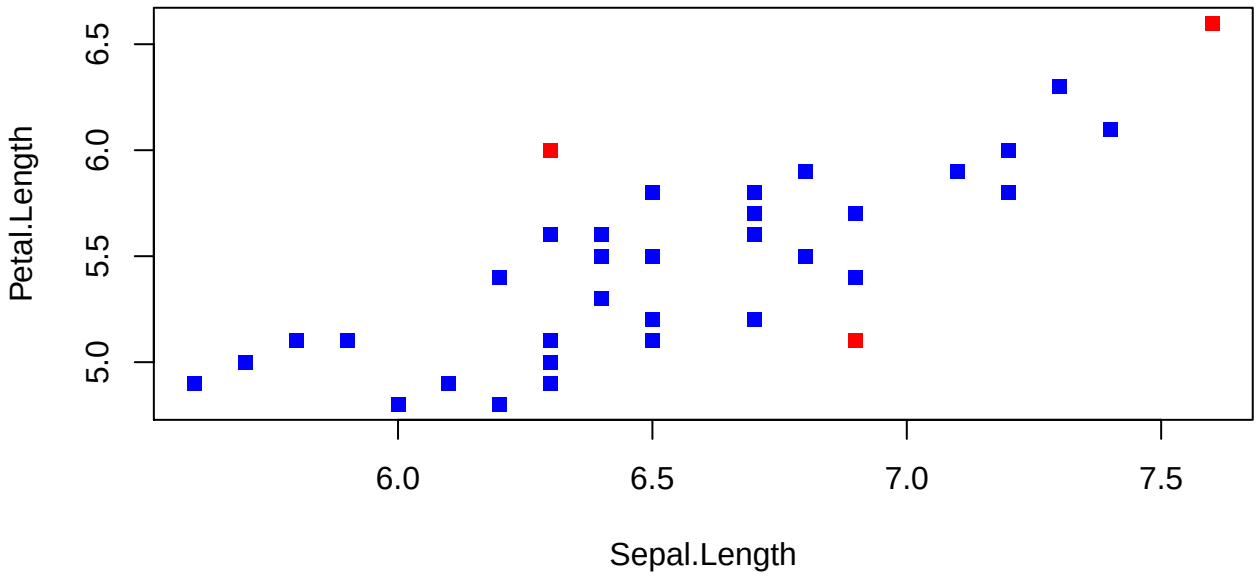

```
      # remove the outliers from the class
      class3OR <- class3OR[-outliers,]
```

Remove outliers from Class 3, Sepal Length and Petal Length

```
      # remove the two desired features
      dat <- class3OR[,-c(2,4,5)]
      # plot the two features
      plot(dat, col = "blue", pch = c(15,16,17)[as.numeric(iris$Species)])
      # use Wilks' Outlier Removal
      dat.rows <- Wilks(dat)
      head(dat.rows)
```

```
##             w     Fstat     p
## 101 0.8296789 3.9004266 0.029
## 102 0.9377342 1.2616052 0.295
## 103 0.9588542 0.8153179 0.450
## 104 0.9665847 0.6568385 0.524
## 105 0.9578031 0.8370634 0.441
## 106 0.8128698 4.3739760 0.020
```

```
      # return indexes which contain outliers
      outliers <- which(dat.rows[,"p"] < 0.05)
      # plot the outliers in red
      points(dat[outliers,], col = "red", pch = c(15,16,17)[as.numeric(iris$Species[outliers])])
```



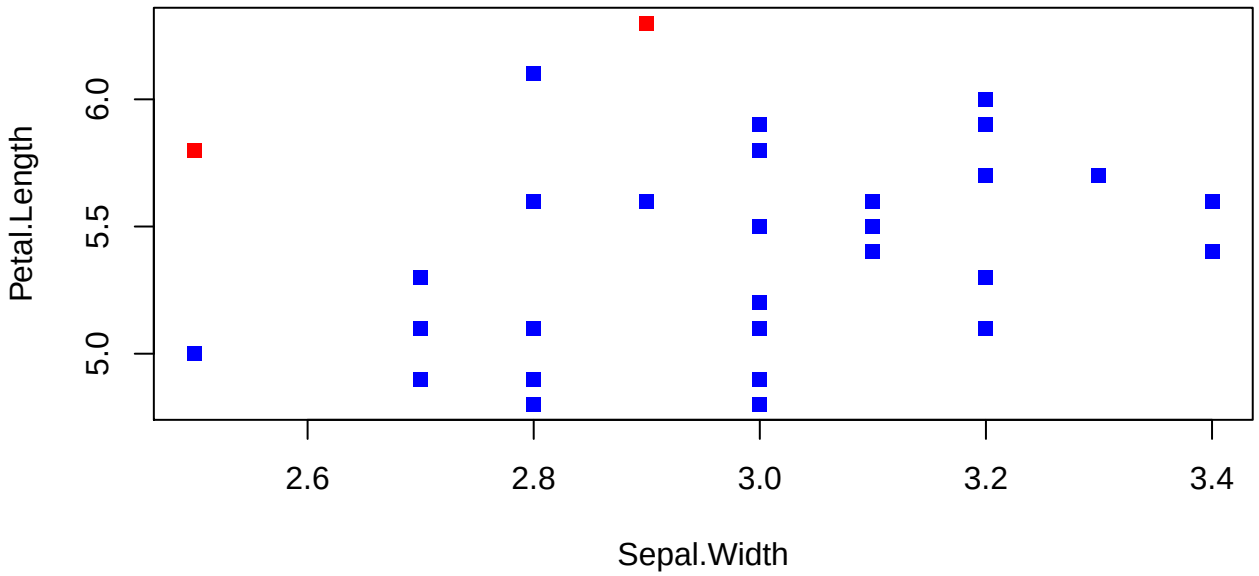

```
      # remove the outliers from the class
      class3OR <- class3OR[-outliers,]
```

Remove outliers from Class 3, Sepal Width and Petal Length

```
      # remove the two desired features
      dat <- class3OR[,-c(1,4,5)]
      # plot the two features
      plot(dat, col = "blue", pch = c(15,16,17)[as.numeric(iris$Species)])
      # use Wilks' Outlier Removal
      dat.rows <- Wilks(dat)
      head(dat.rows)
```

```
##             w     Fstat     p
## 102 0.9601535 0.7262521 0.491
## 103 0.9556268 0.8125880 0.452
## 104 0.9890776 0.1932535 0.825
## 105 0.9724999 0.4948611 0.614
## 108 0.8269564 3.6619379 0.036
## 109 0.8163202 3.9376656 0.029
```

```
      # return indexes which contain outliers
      outliers <- which(dat.rows[,"p"] < 0.05)
      # plot the outliers in red
      points(dat[outliers,], col = "red", pch = c(15,16,17)[as.numeric(iris$Species[outliers])])
```




```
# remove the outliers from the class
class3OR <- class3OR[-outliers,]
```

Create another dataframe of the original dataset with the outliers identified above removed.

```
irisRemoveOutlier <- rbind(class1OR, class2OR, class3OR)
```

(e): Provide an explanation of the results:

Was there any class that had obvious outliers? If so, how did you determine the outlier, if not, why not?

Yes.

Example) When viewing the plot for Class 1 Sepal Length vs Sepal Width, there was one point located at approximately (4.5, 2.3) which appeared to be a significant outlier. This is highlighted in red as an outlier that was picked up and removed by the Wilks' Outlier Removal method that was implemented.

In the Class 3 comparison of Petal Width and Sepal Length, one outlier was determined to exist. When viewing the plot, we can see why this point was identified as an outlier. Since the distribution amongst these two features is not as clear, determining an outlier in this case would be difficult. Therefore, I am content that this method only identified one outlier and left most of the data intact.

After comparing all pairing of features per class, Class 1 and Class 3 had the most outliers removed (14). Class 2 had the least number of outliers (9).

Overall, it appears that this method removed the most apparent outliers from each class. In fact, this method may have removed more outliers than are necessary or ideal, since our original sample size is only 50 observations to begin with. Performing the outlier removal on all feature pairings contributed to the larger number of outliers detected. The implementation of this outlier removal method is a good learning practice, but I am skeptical of the usefullness if this many outliers are removed from such a small dataset. It would be useful to track how the removal of an outlier affects the mean and sd of the data, to help determine when no significant difference is made and therefore no more outliers should be removed.

What was the metric used?

After using Wilks' Outlier Removal and calculating the distance using the Mahalanobis Distance, the corresponding F-statistic and p-value was calculated for each data point. If the p-value was less than alpha = 0.05 and therefore justifiably statistically different, the outlier was removed from the dataset.

5 Feature Ranking

(a): Develop an algorithm (pseudocode) to rank the four features of the dataset.

Calculate Fisher's Discriminant Ratio to rank the four features of the dataset.

```
Calculate mean vectors for each class
Calculate sd vectors for each class

Calculate FDR using mean vectors and sd vectors (-, ^, /)
```

(b): Provide the running time of your feature ranking algorithm in O-notation.

The running time of this feature ranking algorithm is O(n).

(c): Implement your algorithm in your code of choice.

Calculate the mean vectors and sd vectors for each class.

```
# mean vectors
   mu1 <- apply(class1[, 1:4], 2, mean)
   mu2 <- apply(class2[, 1:4], 2, mean)
   mu3 <- apply(class3[, 1:4], 2, mean)

# sd vectors
   sd1 <- apply(class1[, 1:4], 2, sd)
   sd2 <- apply(class2[, 1:4], 2, sd)
   sd3 <- apply(class3[, 1:4], 2, sd)
```

Calculate the FDR for Class 1 compared to Class 2:

```
# Class 1 vs Class 2
   # Sepal Length
      FDR12.SL <- c((mu1[1] - mu2[1]) ^ 2 / (sd1[1]^2 + sd2[1]^2), "Sepal Length", "Class 1", "Class 2"
   # Sepal Width
      FDR12.SW <- c((mu1[2] - mu2[2]) ^ 2 / (sd1[2]^2 + sd2[2]^2), "Sepal Width", "Class 1", "Class 2")
   # Petal Length
      FDR12.PL <- c((mu1[3] - mu2[3]) ^ 2 / (sd1[3]^2 + sd2[3]^2), "Petal Length", "Class 1", "Class 2"
   # Petal Width
      FDR12.PW <- c((mu1[4] - mu2[4]) ^ 2 / (sd1[4]^2 + sd2[4]^2), "Petal Width", "Class 1", "Class 2")
```

Calculate the FDR for Class 1 compared to Class 3:

```
# Class 1 vs Class 3
   # Sepal Length
      FDR13.SL <- c((mu1[1] - mu3[1]) ^ 2 / (sd1[1]^2 + sd3[1]^2), "Sepal Length", "Class 1", "Class 3"
   # Sepal Width
      FDR13.SW <- c((mu1[2] - mu3[2]) ^ 2 / (sd1[2]^2 + sd3[2]^2), "Sepal Width", "Class 1", "Class 3")
   # Petal Length
      FDR13.PL <- c((mu1[3] - mu3[3]) ^ 2 / (sd1[3]^2 + sd3[3]^2), "Petal Length", "Class 1", "Class 3"
   # Petal Width
      FDR13.PW <- c((mu1[4] - mu3[4]) ^ 2 / (sd1[4]^2 + sd3[4]^2), "Petal Width", "Class 1", "Class 3")
```

Calculate the FDR for Class 2 compared to Class 3:

```
# Class 2 vs Class 3
   # Sepal Length
      FDR23.SL <- c((mu2[1] - mu3[1]) ^ 2 / (sd2[1]^2 + sd3[1]^2), "Sepal Length", "Class 2", "Class 3"
```

```
   # Sepal Width
      FDR23.SW <- c((mu2[2] - mu3[2]) ^ 2 / (sd2[2]^2 + sd3[2]^2), "Sepal Width", "Class 2", "Class 3")
   # Petal Length
      FDR23.PL <- c((mu2[3] - mu3[3]) ^ 2 / (sd2[3]^2 + sd3[3]^2), "Petal Length", "Class 2", "Class 3"
   # Petal Width
      FDR23.PW <- c((mu2[4] - mu3[4]) ^ 2 / (sd2[4]^2 + sd3[4]^2) , "Petal Width", "Class 2", "Class 3"
```

(d): Determine if any of the four features can separate the three plant types.

Create a single data frame containing the Fishers' Discriminant Ratio for each of the 3 class comparisons.

```
   FDR <- rbind(FDR12.SL, FDR12.SW, FDR12.PL, FDR12.PW, FDR13.SL, FDR13.SW, FDR13.PL, FDR13.PW, FDR23.SL
   colnames(FDR) <- c("Fisher's Discriminant Ratio", "Feature", "Class X", "Class Y")
   FDR <- as.data.frame(FDR)
   FDR
```

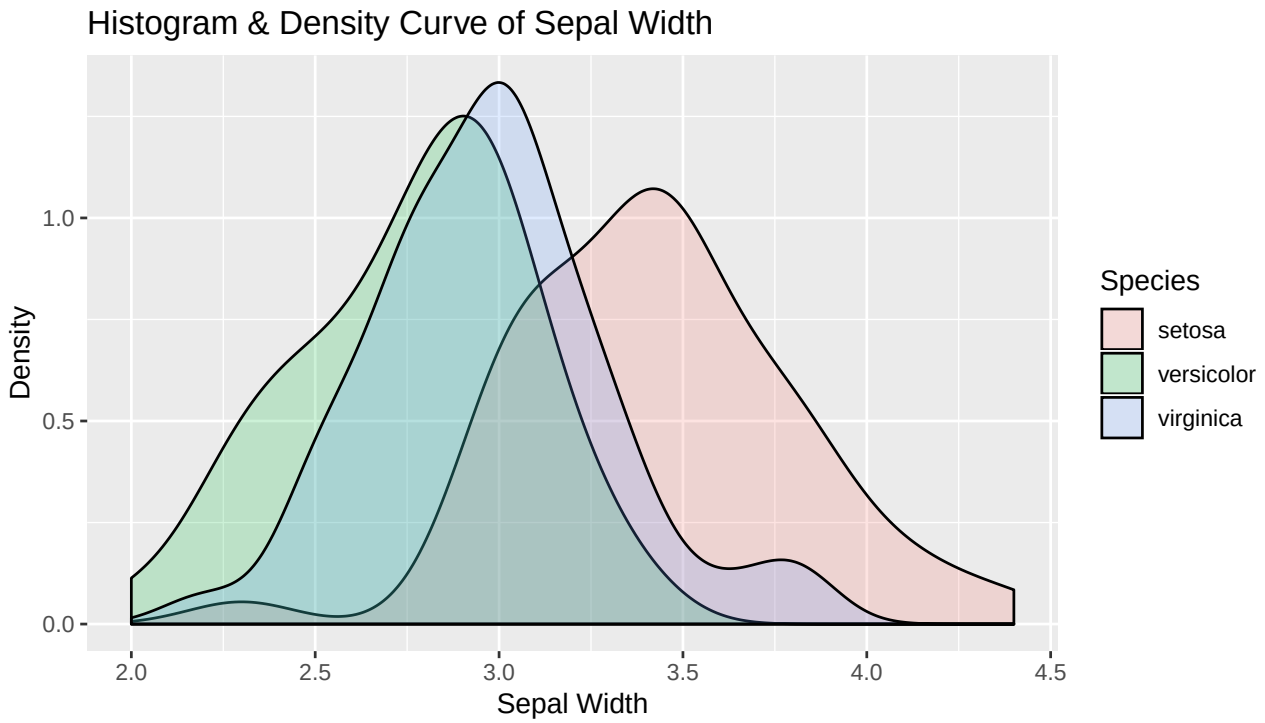
```
##          Fisher's Discriminant Ratio      Feature Class X Class Y
## FDR12.SL            2.21382304083914 Sepal Length Class 1 Class 2
## FDR12.SW             1.7879313657739  Sepal Width Class 1 Class 2
## FDR12.PL            31.1934976987754 Petal Length Class 1 Class 2
## FDR12.PW            23.2293935945375  Petal Width Class 1 Class 2
## FDR13.SL            4.73470043627659 Sepal Length Class 1 Class 3
## FDR13.SW              0.832140067562  Sepal Width Class 1 Class 3
## FDR13.PL            49.9723763305817 Petal Length Class 1 Class 3
## FDR13.PW            36.6124893877936  Petal Width Class 1 Class 3
## FDR23.SL           0.633750030424729 Sepal Length Class 2 Class 3
## FDR23.SW           0.205538039753256  Sepal Width Class 2 Class 3
## FDR23.PL            3.17710512414157 Petal Length Class 2 Class 3
## FDR23.PW             4.2780272254294  Petal Width Class 2 Class 3
```

Looking at the single data frame above that contains the Fisher's Discriminant Ratio, the features that best separate the class data will be at either extreme, either the largest values or smallest values. It makes sense that the largest values would best separate the classes, but by looking at the density plots of each feature, we can test this hypothesis:

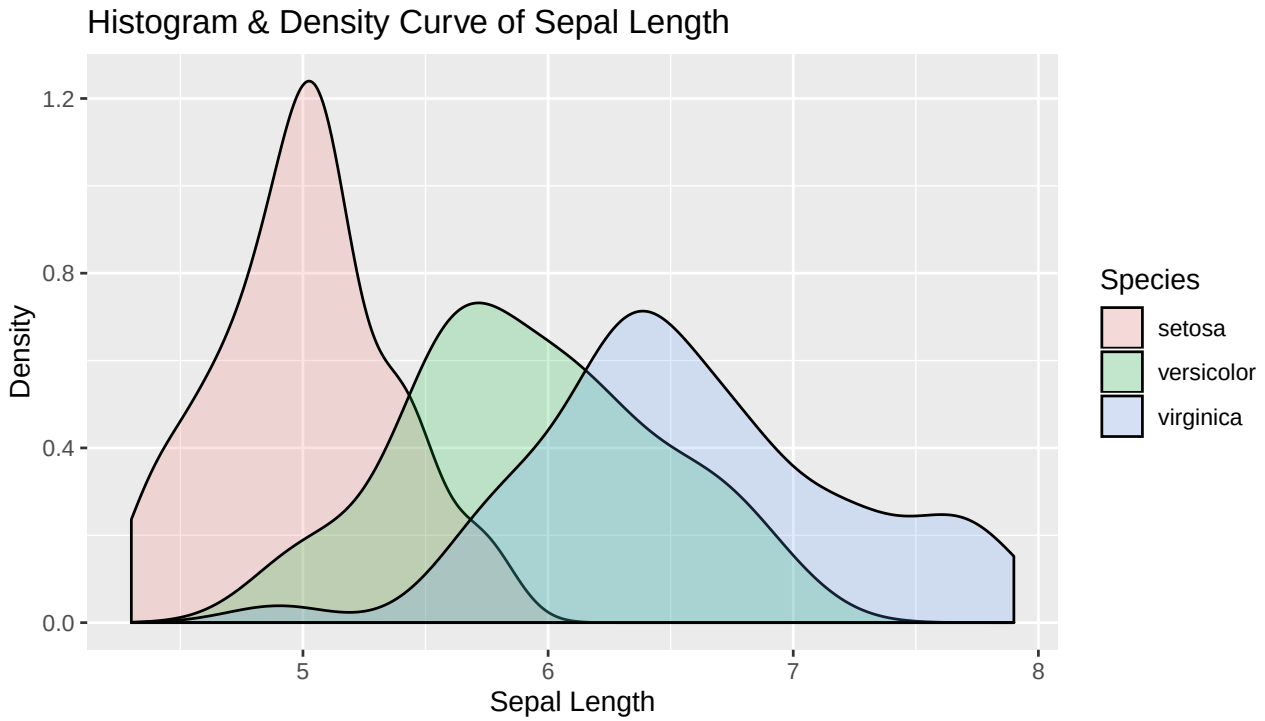
```
require(ggplot2)
```

```
## Loading required package: ggplot2
```

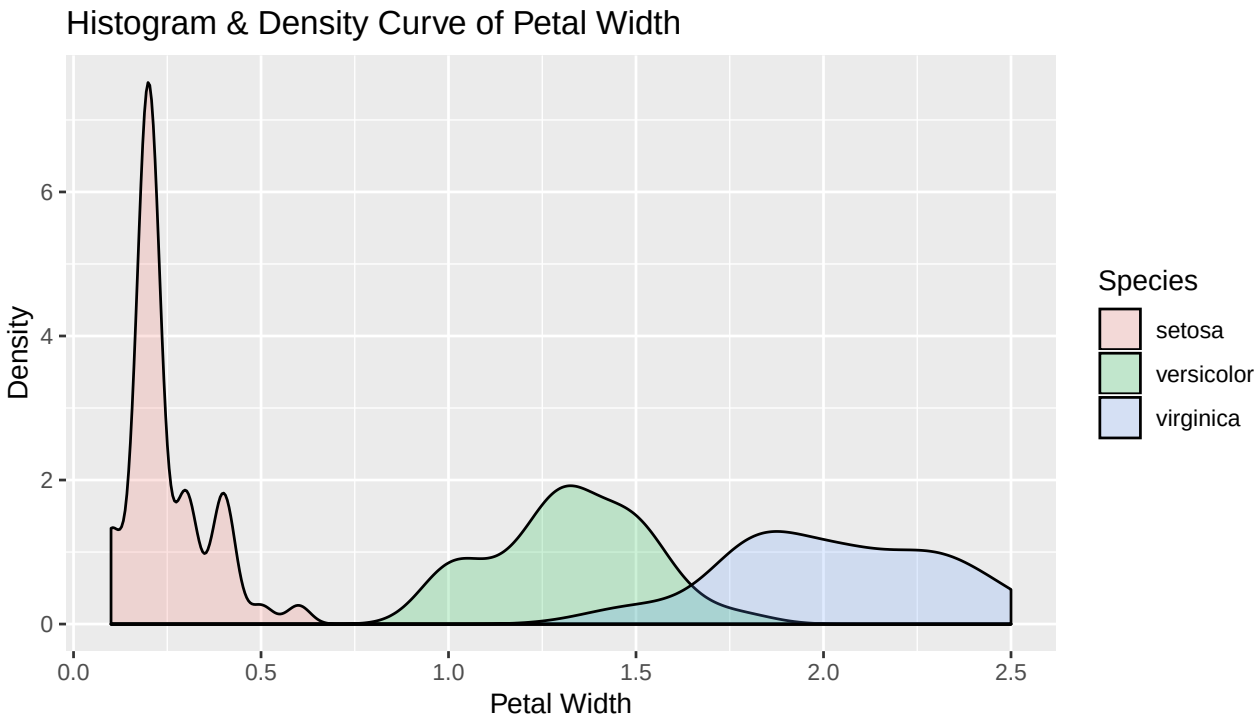
```
# Sepal Width
   density1 <- ggplot(data=iris, aes(x=Sepal.Width, fill=Species))
   density1 + geom_density(stat="density", alpha=I(0.2)) + xlab("Sepal Width") +  ylab("Density") + ggt
```



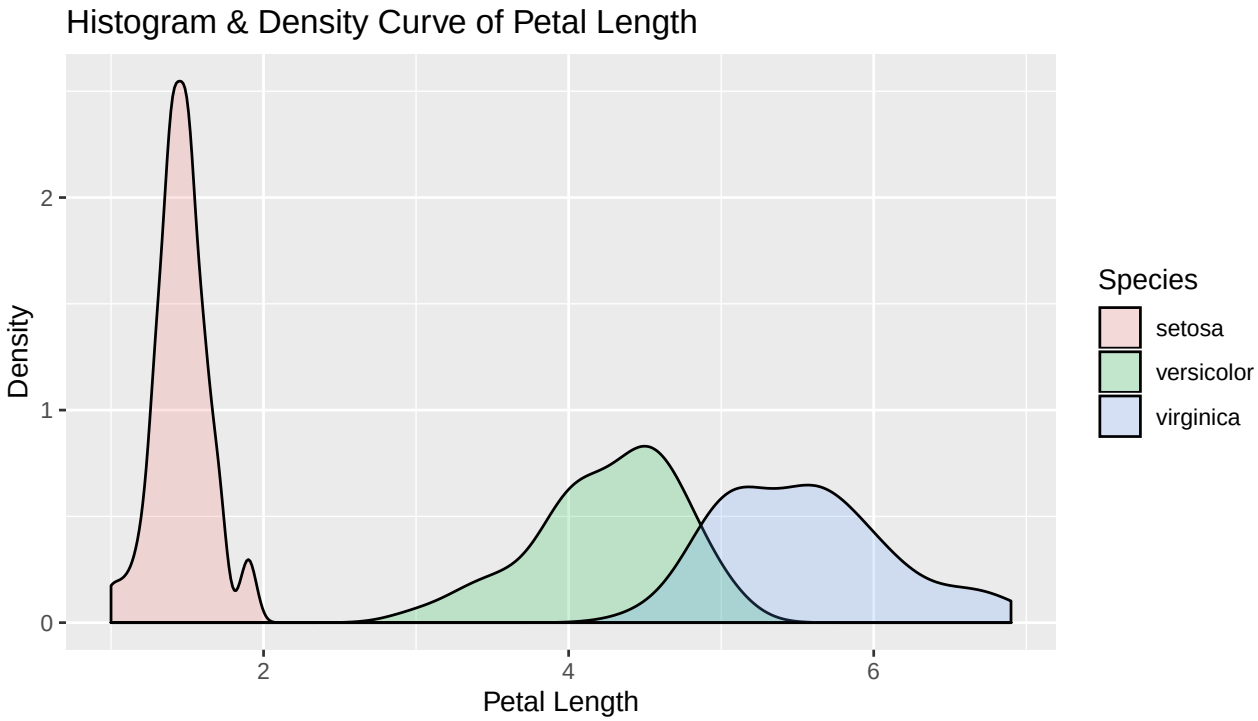

```
# Sepal Length
   density2 <- ggplot(data=iris, aes(x=Sepal.Length, fill=Species))
   density2 + geom_density(stat="density", alpha=I(0.2)) + xlab("Sepal Length") +  ylab("Density") + gg
```




```
# Petal Width
   density3 <- ggplot(data=iris, aes(x=Petal.Width, fill=Species))
   density3 + geom_density(stat="density", alpha=I(0.2)) + xlab("Petal Width") +  ylab("Density") + ggt
```




```
# Petal Length
   density4 <- ggplot(data=iris, aes(x=Petal.Length, fill=Species))
   density4 + geom_density(stat="density", alpha=I(0.2)) + xlab("Petal Length") +  ylab("Density") + gg
```



After looking at the density plots, we can see that Petal Width and Petal Length are better separators of the three classes than either Sepal Width or Sepal Length. This corresponds to the significantly larger values in the table of FDR. Therefore, the largest FDR values provide the best class separability.

After breaking down the calculation for Fisher's Discriminant Ratio, this also makes sense. The greater the

square of the distance between the two means and the smaller the within-class scatter is (class variance), the better the expected line.

(e): Provide an explanation of the results:

Was there any feature that could separate the plant types?

The best separators of plant types were Petal Width and Petal Length. Especially for separation between Class 1 and Class 3, the ratios were PW: 36.61 and PL: 49.97. The separation between Class 1 and Class 2 had ratios of PW: 23.23 and PL: 31.19. The smallest separation with these two features is found between Class 2 and Class 3 with FDR of PW: 4.28 and PL: 3.18.

Overall, the best single feature to separate Class 1/Class 2 and Class 1/Class 3 is Petal Length, while Petal Width is identified as the best separator for Class 2/Class 3.

If a feature could not separate the plant types, what conclusion can be drawn from the feature?

Neither Sepal Width or Sepal Length are a good separator of plant types. These two features have a smaller distance between the means of each class and have larger variances, which makes them non-distinguishable. The combination of both of these things is what makes these features poor separators of plant types.

Can a metric be developed to complement the ranking method?

Another metric could be developed to determine if this ranking method yields results that will separate the three classes within a reasonable amount of accuracy. Right now, the FDR give support in which order to rank each of the four features, but does not determine which values are significant, in implementation.

This analysis also did not consider combining multiple features in order to separate plant types. So, developing a metric that takes into account the feature ranking discovered in this section while also considering the combination of multiple features separating classes would be more useful.

REFERENCES

[1] JonJon. "Removing Multivariate Outliers With Mvoutlier." Stack Overflow, 1 June 1967, stackoverflow.com/questions/45289225/removing-multivariate-outliers-with-mvoutlier.

[2] Lee, Jihui. "Data Visualization with ggplot2." Columbia.edu, Columbia University, 24 Feb. 2015, www.mailman.columbia.edu/sites/default/files/media/fdawg_ggplot2.html.

[3] R Documentation, API Documentation, www.rdocumentation.org/.

[4] "Sort: Sorting or Ordering Vectors." R Package Documentation, rdrr.io/r/base/sort.html.