ML - unsupervised learning (practical)

IMohorianu

02/03/2021

Required libraries

```r
setwd("~/Dropbox/CSCI/training/BBS/MachineLearning/")

library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.2
```

```r
library(ggfortify)
library(readr)
library(car)
```

```
## Loading required package: carData
```

```r
library(gridExtra)
library(grid)
library(plyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following object is masked from 'package:gridExtra':
##
##     combine

## The following object is masked from 'package:car':
##
##     recode

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(reshape2)
library(caTools)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
##
## Attaching package: 'GGally'
```

```
## The following object is masked from 'package:dplyr':
##
##     nasa
```

```r
library(e1071)
library(C50)
library(tree)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(class)
library(gmodels)
library(dendextend)
```

```
## Warning: package 'dendextend' was built under R version 3.6.2
```

```
##
## ---------------------
## Welcome to dendextend version 1.14.0
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## Or contact: <tal.galili@gmail.com>
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
```

```
##
## Attaching package: 'dendextend'
```

```
## The following object is masked from 'package:stats':
##
```

```
##     cutree
```

The IRIS dataset

From the iris manual page:

The famous (Fisher's or Anderson's) Iris data set, first presented by Fisher in 1936 (http://archive.ics.uci.edu/ml/datasets/Iris), gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica. One class is linearly separable from the other two; the latter are not linearly separable from each other. The data base contains the following attributes: 1). sepal length in cm 2). sepal width in cm 3). petal length in cm 4). petal width in cm 5). classes: - Iris Setosa - Iris Versicolour - Iris Virginica

```
library(datasets)
data(iris)      ##loads the dataset, which can be accessed under the variable name iris
summary(iris)   ##presents the 5 figure summary of the dataset
```

```
##   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  
##                 
##                 
## 
```

```
str(iris)       ##presents the structure of the iris dataframe
```

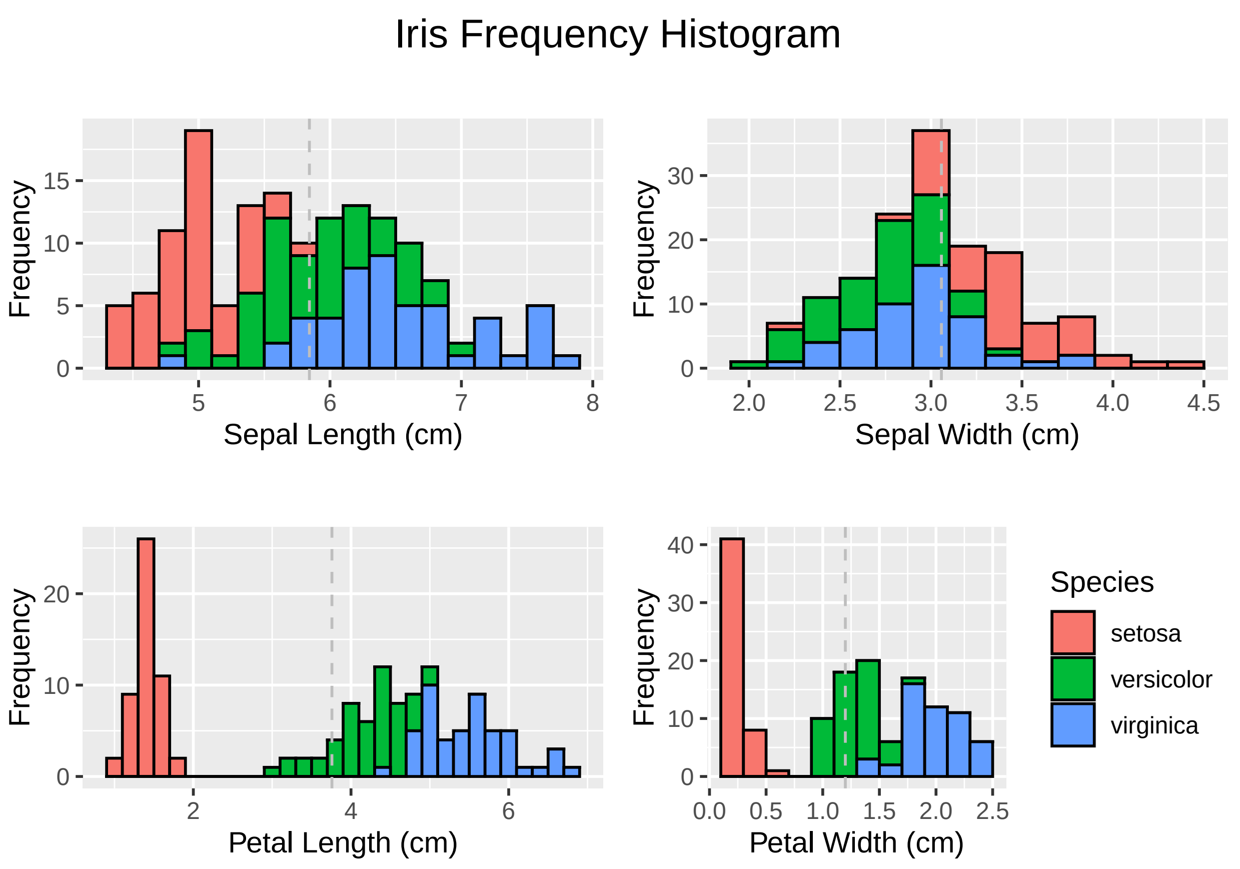
```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Exploring the data

Density & Frequency analyses using Histogram

```
# Sepal length
HistSl <- ggplot(data=iris, aes(x=Sepal.Length))+
  geom_histogram(binwidth=0.2, color="black", aes(fill=Species)) +
  xlab("Sepal Length (cm)") +
  ylab("Frequency") +
  theme(legend.position="none")+
  ggtitle("Histogram of Sepal Length")+
  geom_vline(data=iris, aes(xintercept = mean(Sepal.Length)),linetype="dashed",color="grey")
# Sepal width
HistSw <- ggplot(data=iris, aes(x=Sepal.Width)) +
  geom_histogram(binwidth=0.2, color="black", aes(fill=Species)) +
```

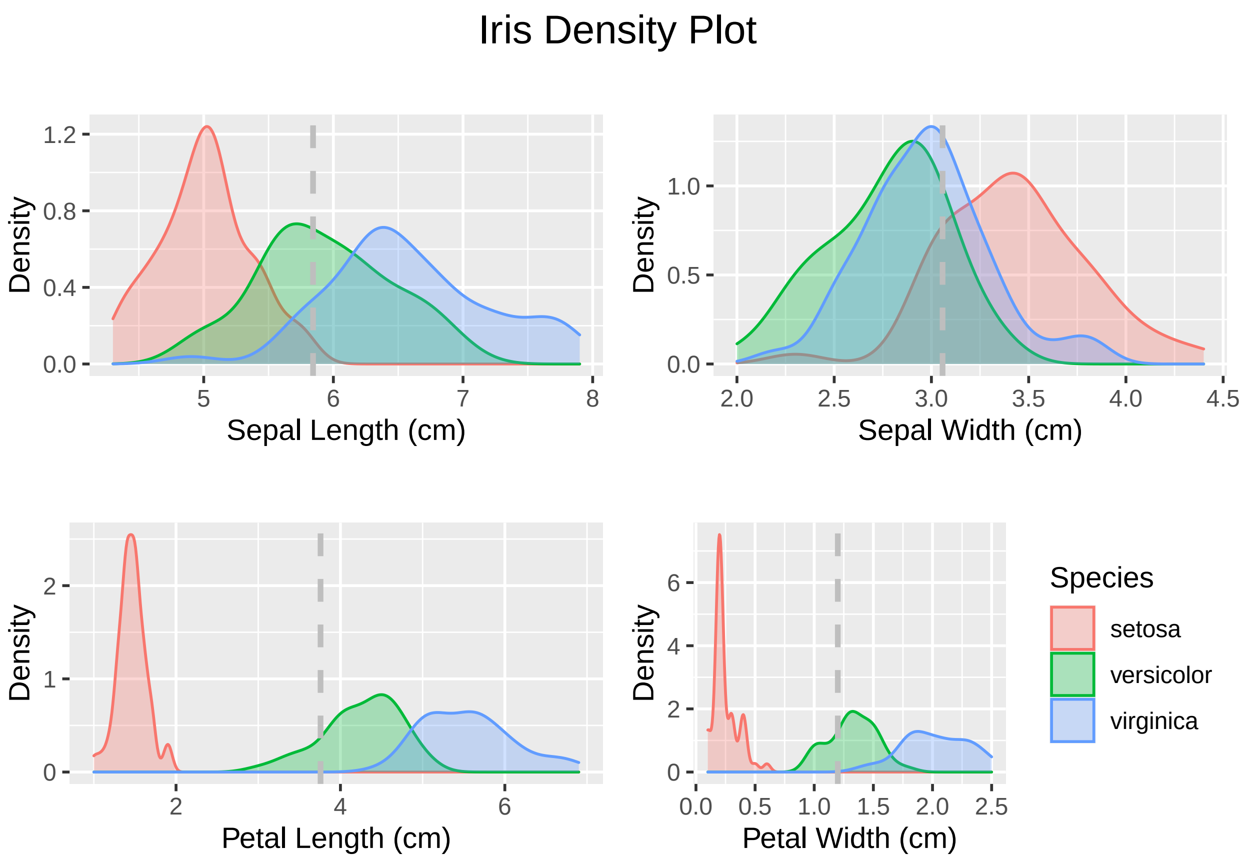
```r
  xlab("Sepal Width (cm)") +
  ylab("Frequency") +
  theme(legend.position="none")+
  ggtitle("Histogram of Sepal Width")+
  geom_vline(data=iris, aes(xintercept = mean(Sepal.Width)),linetype="dashed",color="grey")
# Petal length
HistPl <- ggplot(data=iris, aes(x=Petal.Length))+
  geom_histogram(binwidth=0.2, color="black", aes(fill=Species)) +
  xlab("Petal Length (cm)") +
  ylab("Frequency") +
  theme(legend.position="none")+
  ggtitle("Histogram of Petal Length")+
  geom_vline(data=iris, aes(xintercept = mean(Petal.Length)),
             linetype="dashed",color="grey")
# Petal width
HistPw <- ggplot(data=iris, aes(x=Petal.Width))+
  geom_histogram(binwidth=0.2, color="black", aes(fill=Species)) +
  xlab("Petal Width (cm)") +
  ylab("Frequency") +
  theme(legend.position="right" )+
  ggtitle("Histogram of Petal Width")+
  geom_vline(data=iris, aes(xintercept = mean(Petal.Width)),linetype="dashed",color="grey")
# Plot all visualizations
grid.arrange(HistSl + ggtitle(""),
             HistSw + ggtitle(""),
             HistPl + ggtitle(""),
             HistPw  + ggtitle(""),
             nrow = 2,
             top = textGrob("Iris Frequency Histogram",
                            gp=gpar(fontsize=15))
)
```



Notice the shape of the data, most attributes exhibit a normal distribution. You can see the measurements of very small flowers in the Petal width and length column.

We can review the density distribution of each attribute broken down by class value. Like the scatterplot matrix, the density plot by class can help see the separation of classes. It can also help to understand the overlap in class values for an attribute.

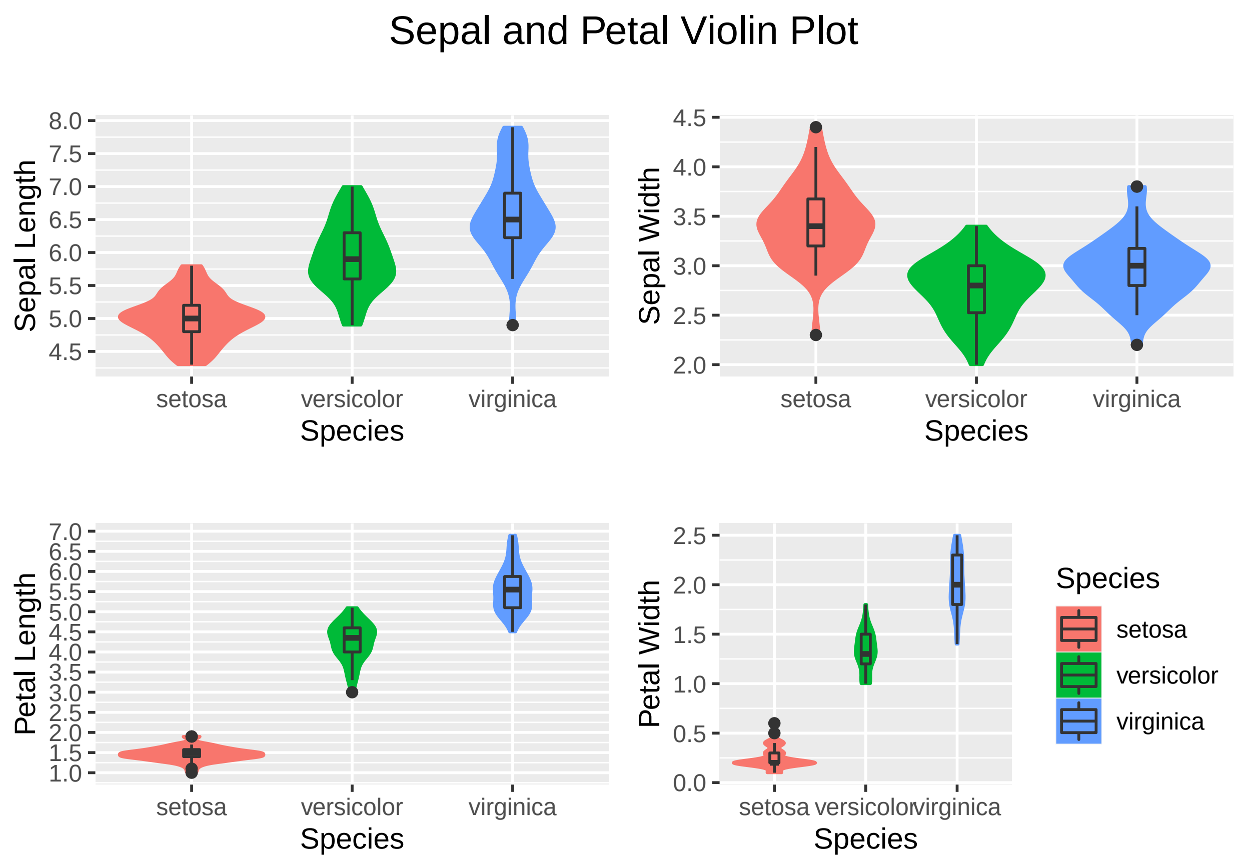
```
DhistPl <- ggplot(iris, aes(x=Petal.Length, colour=Species, fill=Species)) +
  geom_density(alpha=.3) +
  geom_vline(aes(xintercept=mean(Petal.Length),  colour=Species),linetype="dashed",color="grey", size=1
  xlab("Petal Length (cm)") +
  ylab("Density")+
  theme(legend.position="none")
DhistPw <- ggplot(iris, aes(x=Petal.Width, colour=Species, fill=Species)) +
  geom_density(alpha=.3) +
  geom_vline(aes(xintercept=mean(Petal.Width),  colour=Species),linetype="dashed",color="grey", size=1)
  xlab("Petal Width (cm)") +
  ylab("Density")
DhistSw <- ggplot(iris, aes(x=Sepal.Width, colour=Species, fill=Species)) +
  geom_density(alpha=.3) +
  geom_vline(aes(xintercept=mean(Sepal.Width),  colour=Species), linetype="dashed",color="grey", size=1
  xlab("Sepal Width (cm)") +
  ylab("Density")+
  theme(legend.position="none")
DhistSl <- ggplot(iris, aes(x=Sepal.Length, colour=Species, fill=Species)) +
  geom_density(alpha=.3) +
  geom_vline(aes(xintercept=mean(Sepal.Length),
  colour=Species),linetype="dashed", color="grey", size=1)+
  xlab("Sepal Length (cm)") +
  ylab("Density")+
  theme(legend.position="none")
# Plot all density visualizations
grid.arrange(DhistSl + ggtitle(""),
             DhistSw  + ggtitle(""),
             DhistPl + ggtitle(""),
             DhistPw  + ggtitle(""),
             nrow = 2,
             top = textGrob("Iris Density Plot",
                            gp=gpar(fontsize=15))
)
```



We can also visualize the data using the violin plots. They are similar to the Box Plots but they allow the illustration of the number of points at a particular value by the width of the shapes. We can also include the marker for the median and a box for the interquartile range.

```
VpSl <-  ggplot(iris, aes(Species, Sepal.Length, fill=Species)) +
        geom_violin(aes(color = Species), trim = T)+
        scale_y_continuous("Sepal Length", breaks= seq(0,30, by=.5))+
        geom_boxplot(width=0.1)+
        theme(legend.position="none")
VpSw <-  ggplot(iris, aes(Species, Sepal.Width, fill=Species)) +
        geom_violin(aes(color = Species), trim = T)+
        scale_y_continuous("Sepal Width", breaks= seq(0,30, by=.5))+
        geom_boxplot(width=0.1)+
        theme(legend.position="none")
```

```
VpPl <-  ggplot(iris, aes(Species, Petal.Length, fill=Species)) +
        geom_violin(aes(color = Species), trim = T)+
        scale_y_continuous("Petal Length", breaks= seq(0,30, by=.5))+
        geom_boxplot(width=0.1)+
        theme(legend.position="none")
VpPw <-  ggplot(iris, aes(Species, Petal.Width, fill=Species)) +
        geom_violin(aes(color = Species), trim = T)+
        scale_y_continuous("Petal Width", breaks= seq(0,30, by=.5))+
        geom_boxplot(width=0.1)+
        labs(title = "Iris Box Plot", x = "Species")
# Plot all visualizations
grid.arrange(VpSl  + ggtitle(""),
             VpSw  + ggtitle(""),
             VpPl + ggtitle(""),
             VpPw + ggtitle(""),
             nrow = 2,
             top = textGrob("Sepal and Petal Violin Plot",
                            gp=gpar(fontsize=15))
)
```

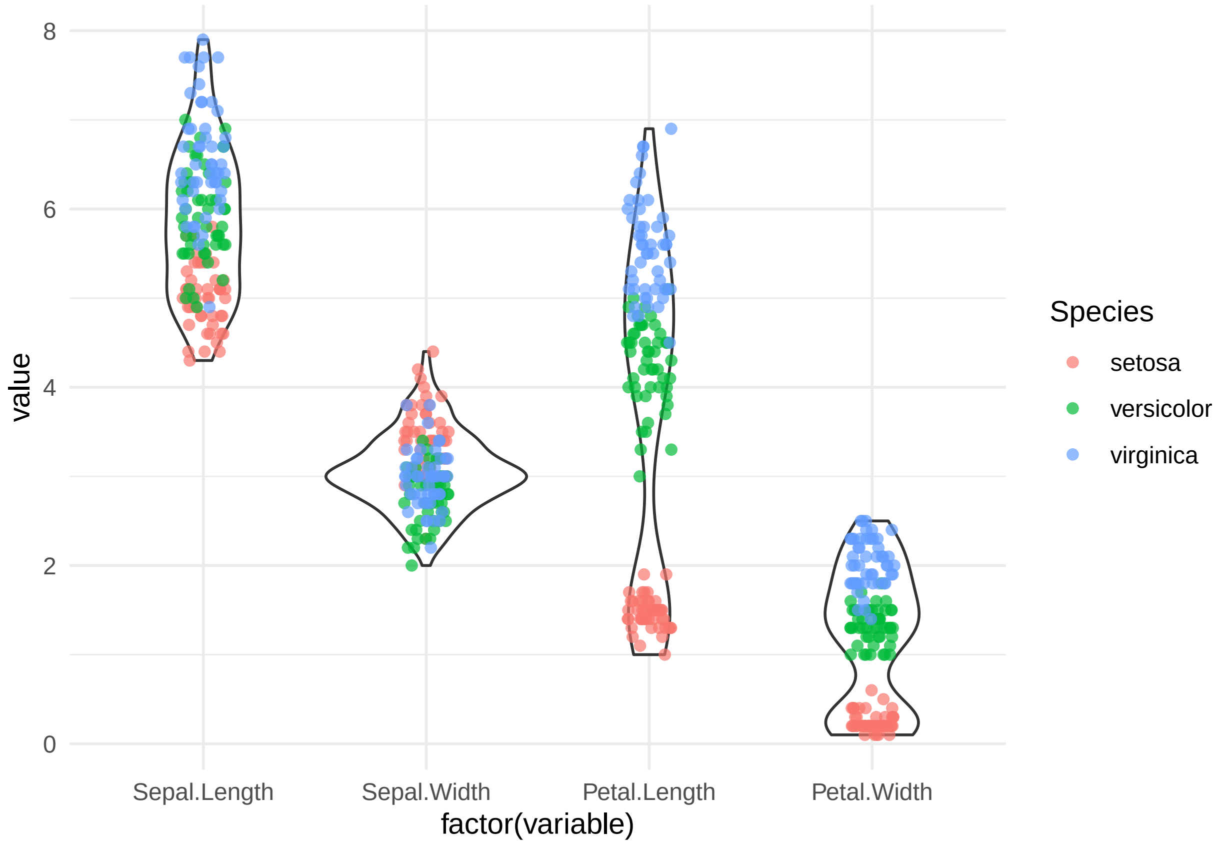


Yet another way to combine violin plots and scatter plots is illustrated below:

```
exploratory_iris <- melt(iris)
```

```
## Using Species as id variables
```

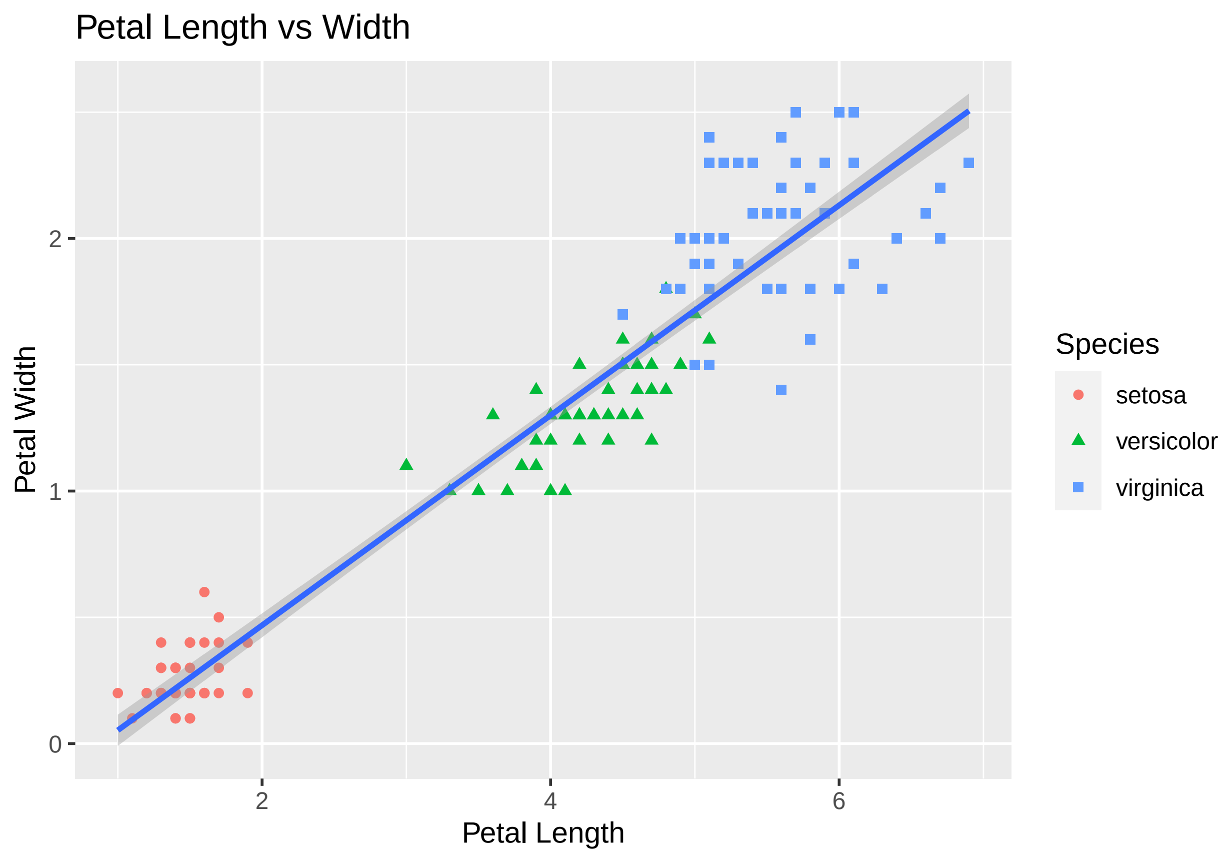
```
exploratory_iris %>%
  ggplot(aes(x = factor(variable), y = value)) +
  geom_violin() +
  geom_jitter(height = 0, width = 0.1, aes(colour = Species), alpha = 0.7) +
  theme_minimal()
```



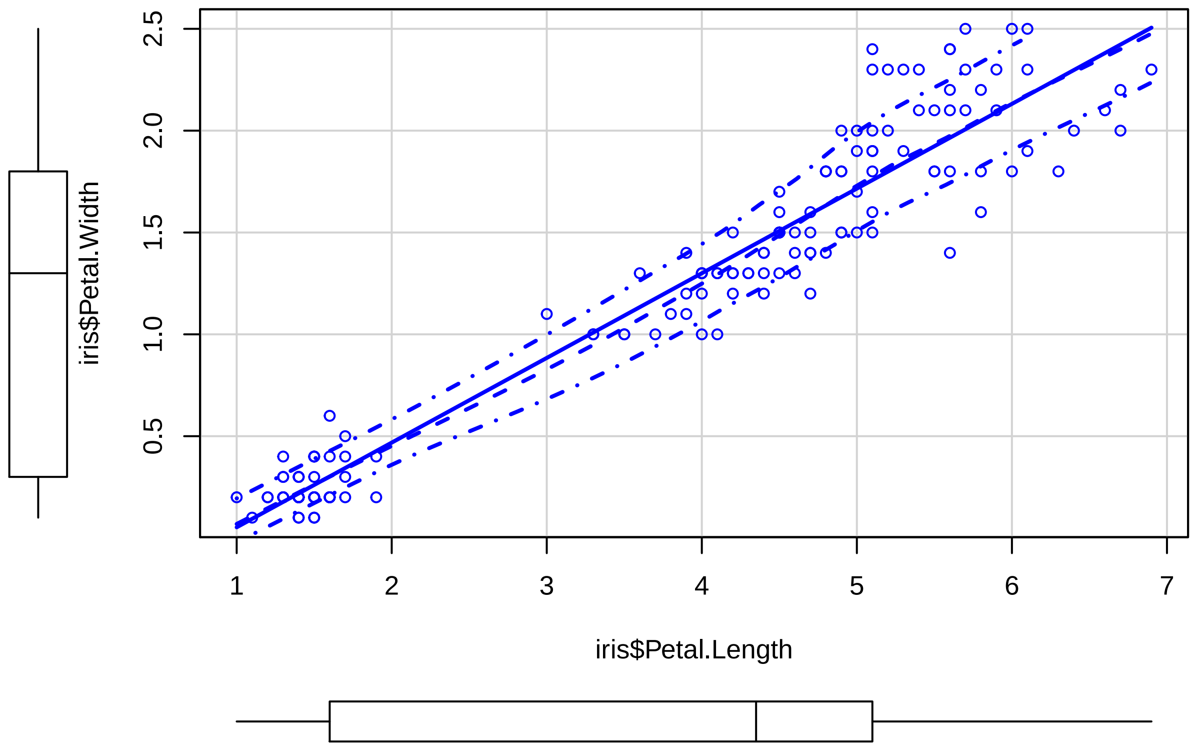
Now let's create a scatterplot of petal lengths versus petal widths with the color & shape by species. There is also a regression line with a 95% confidence band. Notice the petal length of the setosa is clearly a differenciated cluster so it will be a good predictor for ML.

```
ggplot(data = iris, aes(x = Petal.Length, y = Petal.Width))+
  xlab("Petal Length")+
  ylab("Petal Width") +
  geom_point(aes(color = Species,shape=Species))+
  geom_smooth(method='lm')+
  ggtitle("Petal Length vs Width")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



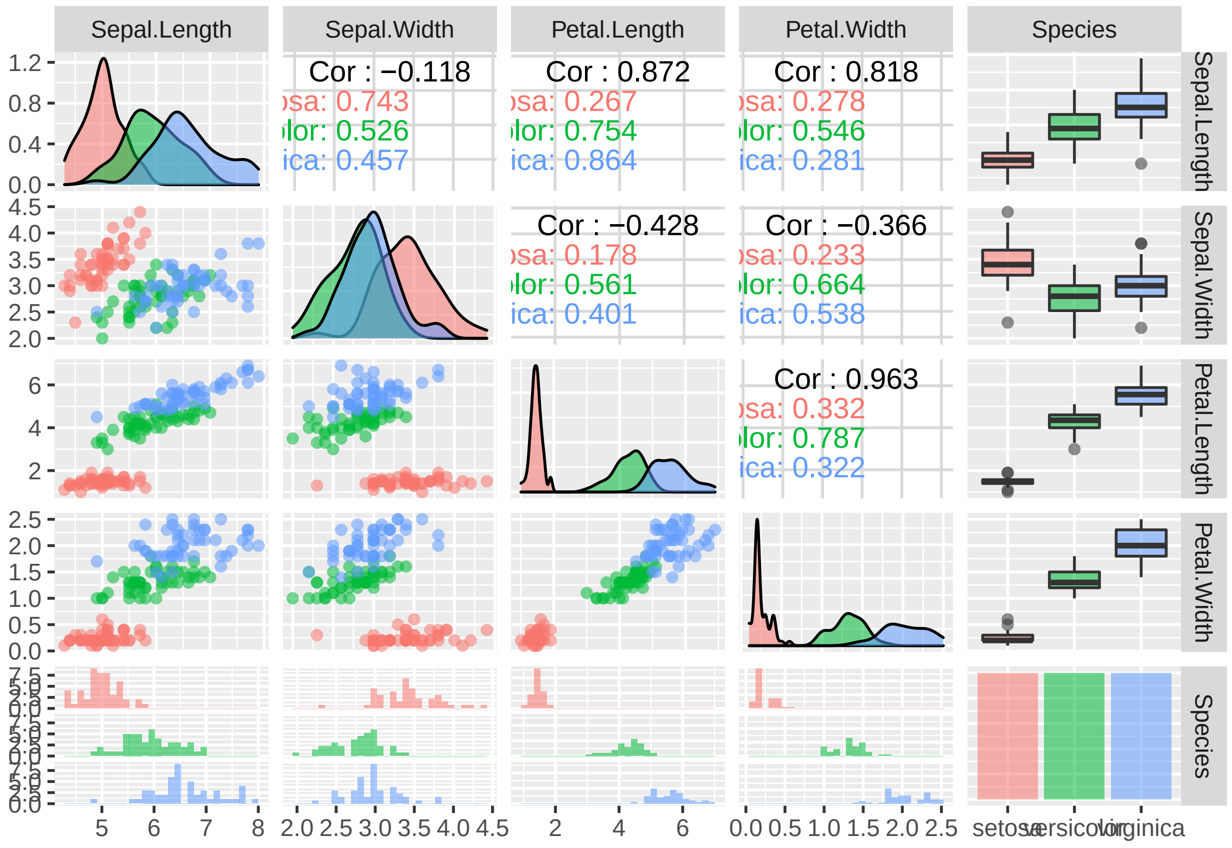
```
# Here is a similar plot with more details on the regression line.

scatterplot(iris$Petal.Length,iris$Petal.Width)
```



And the summary of it all can be obtained using the following function:

```
ggpairs(iris, ggplot2::aes(colour = Species, alpha = 0.4))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
iris.mat <- as.matrix(iris[,1:4])
cov.mat <- cor(iris.mat)
eigen(cov.mat)
```

```
## eigen() decomposition
## $values
## [1] 2.91849782 0.91403047 0.14675688 0.02071484
##
## $vectors
##            [,1]        [,2]       [,3]       [,4]
## [1,]  0.5210659 -0.37741762  0.7195664  0.2612863
## [2,] -0.2693474 -0.92329566 -0.2443818 -0.1235096
## [3,]  0.5804131 -0.02449161 -0.1421264 -0.8014492
## [4,]  0.5648565 -0.06694199 -0.6342727  0.5235971
```

```
iris.pca <- prcomp(iris.mat, center = TRUE, scale = TRUE)
print(iris.pca)
```

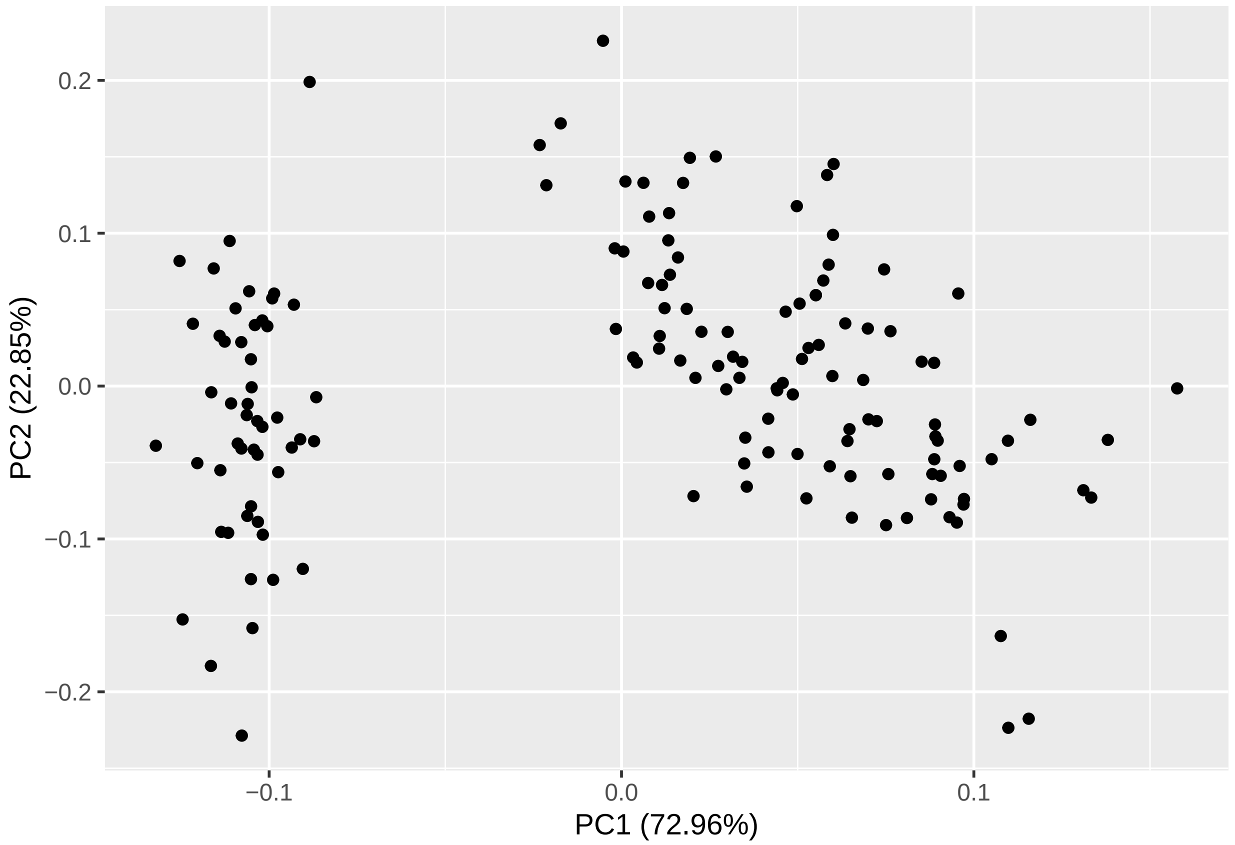
```
## Standard deviations (1, .., p=4):
## [1] 1.7083611 0.9560494 0.3830886 0.1439265
```

```
##
## Rotation (n x k) = (4 x 4):
##                     PC1         PC2        PC3        PC4
## Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
## Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
## Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
## Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
```

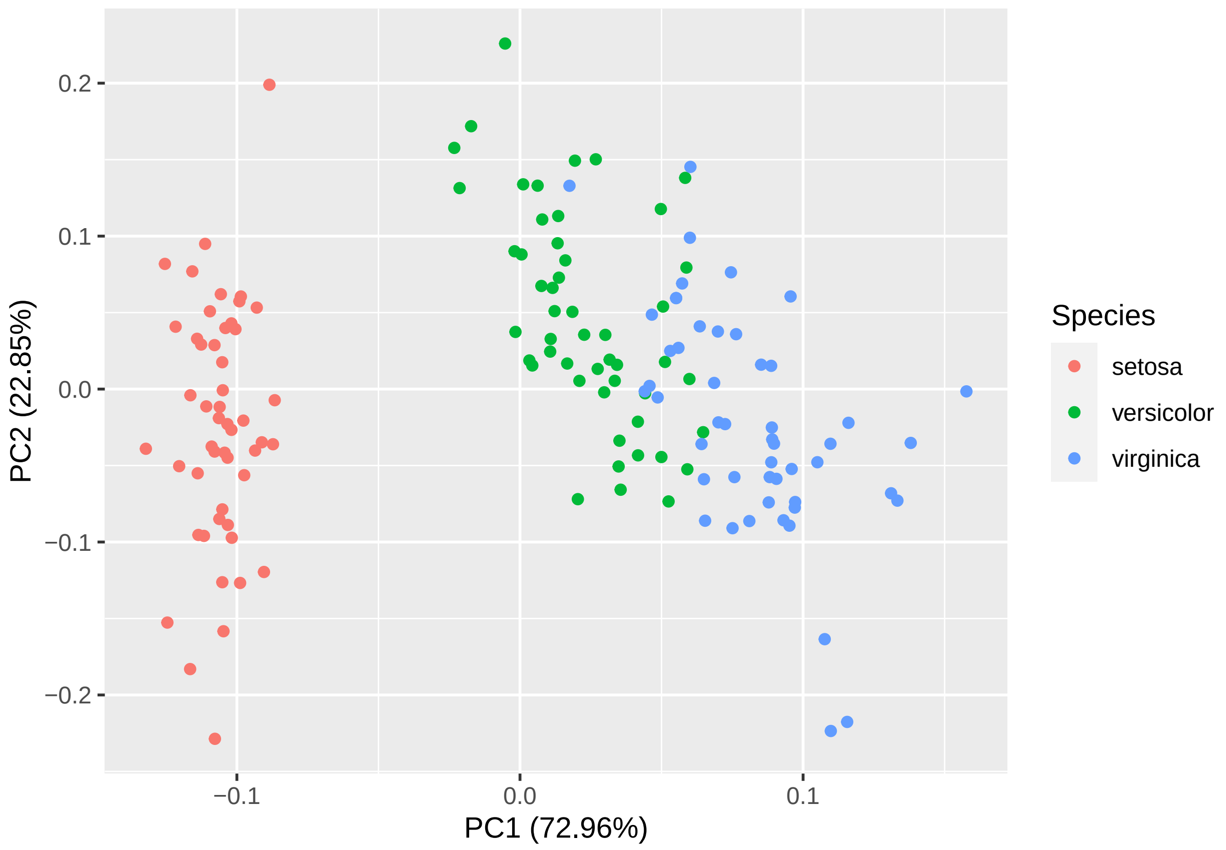
```
summary(iris.pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4
## Standard deviation     1.7084 0.9560 0.38309 0.14393
## Proportion of Variance 0.7296 0.2285 0.03669 0.00518
## Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
```

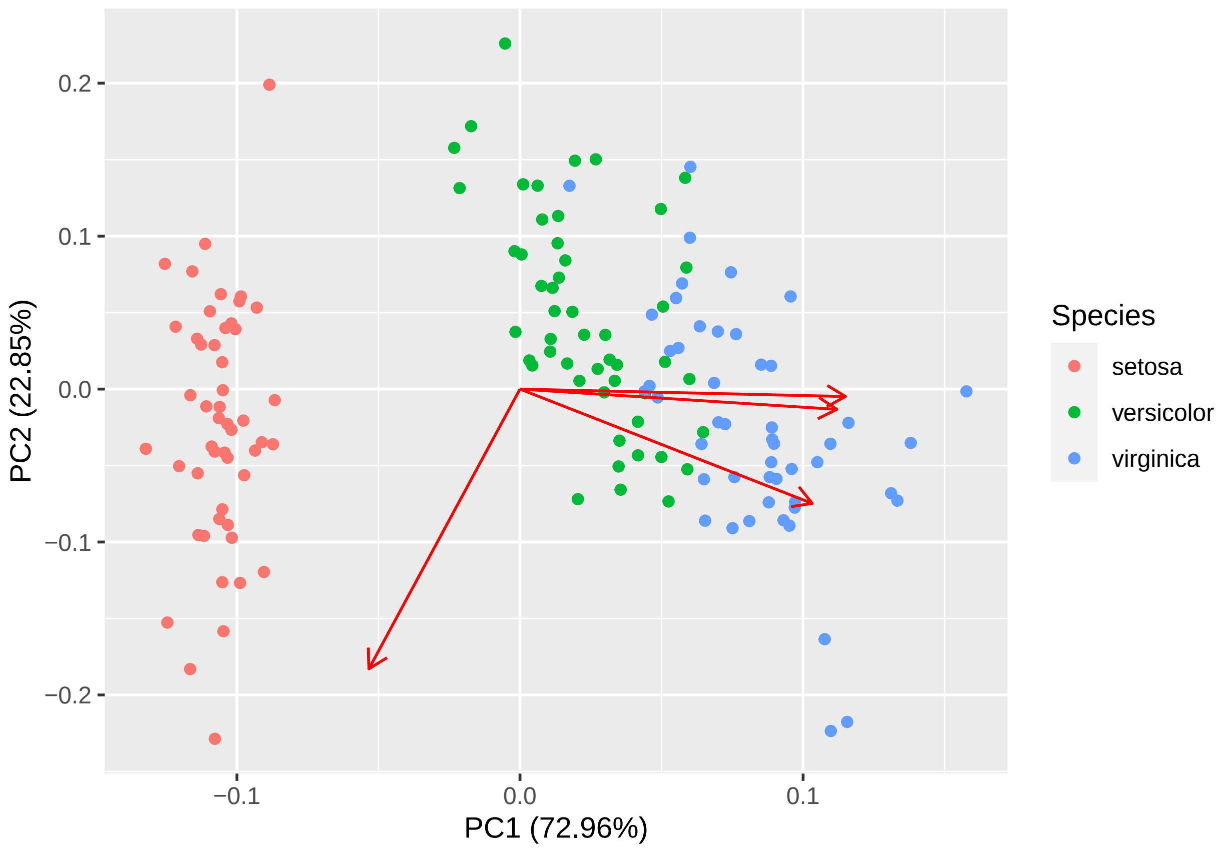
```
autoplot(iris.pca)
```



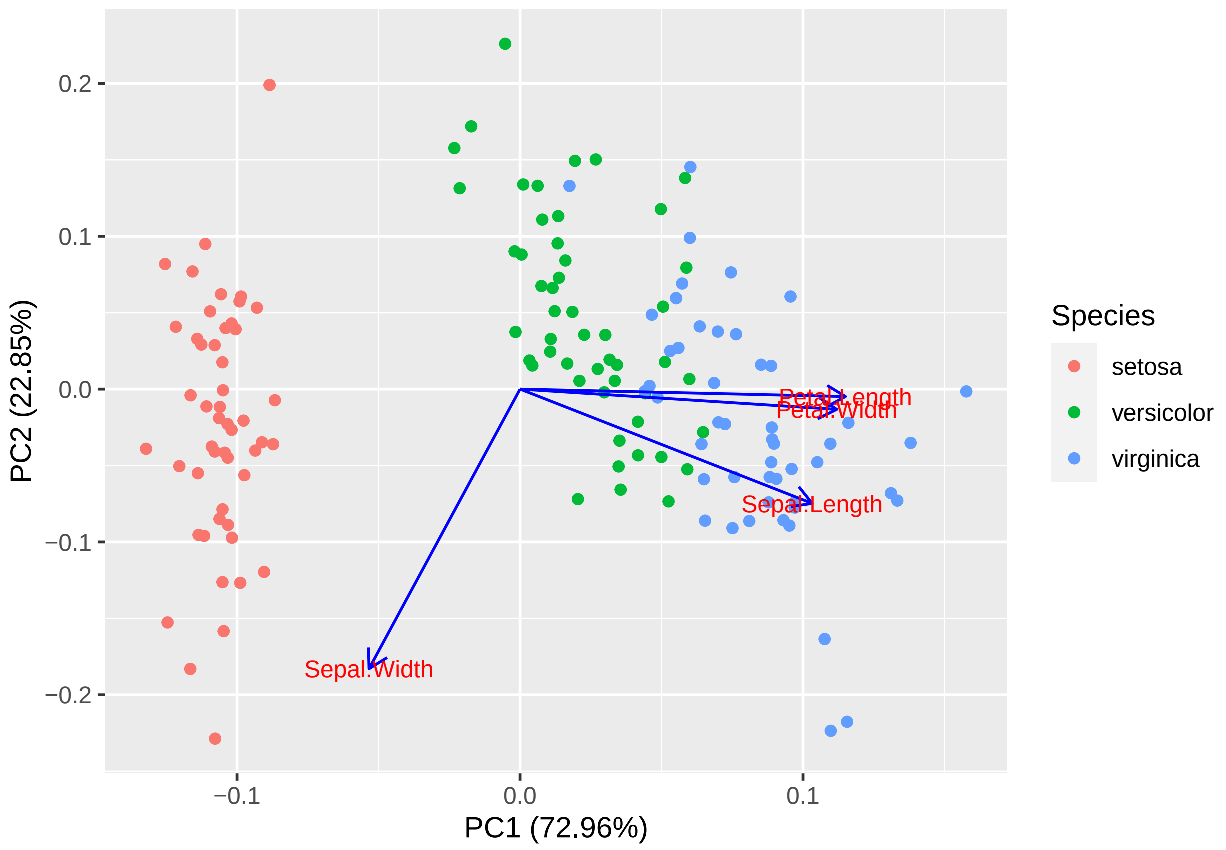
```
autoplot(iris.pca, data = iris, colour = 'Species')
```



```
#Passing loadings = TRUE draws eigenvectors.
autoplot(iris.pca, data = iris, colour = 'Species', loadings = TRUE)
```



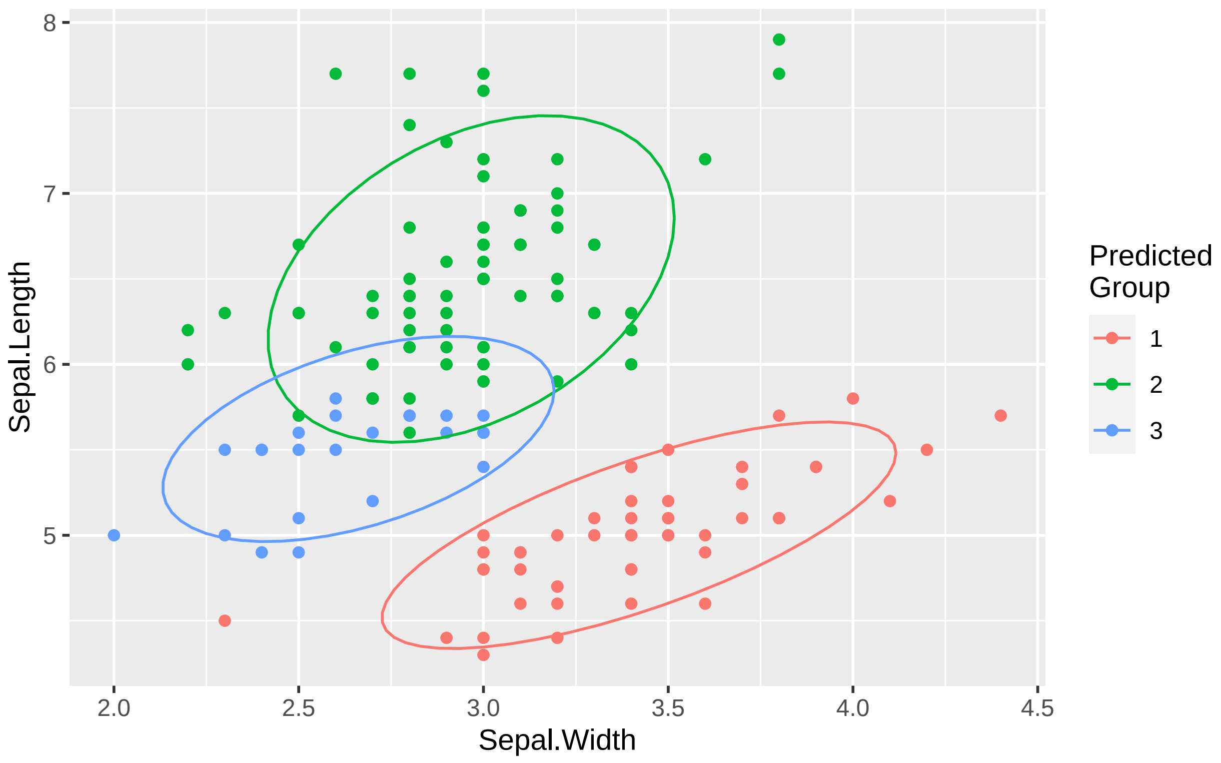
```
#Loadings are interpreted as the coefficients of the linear combination of the initial variables from wh

autoplot(iris.pca, data = iris, colour = 'Species',
         loadings = TRUE, loadings.colour = 'blue',
         loadings.label = TRUE, loadings.label.size = 3)
```



Clustering.

Hierarchical clustering

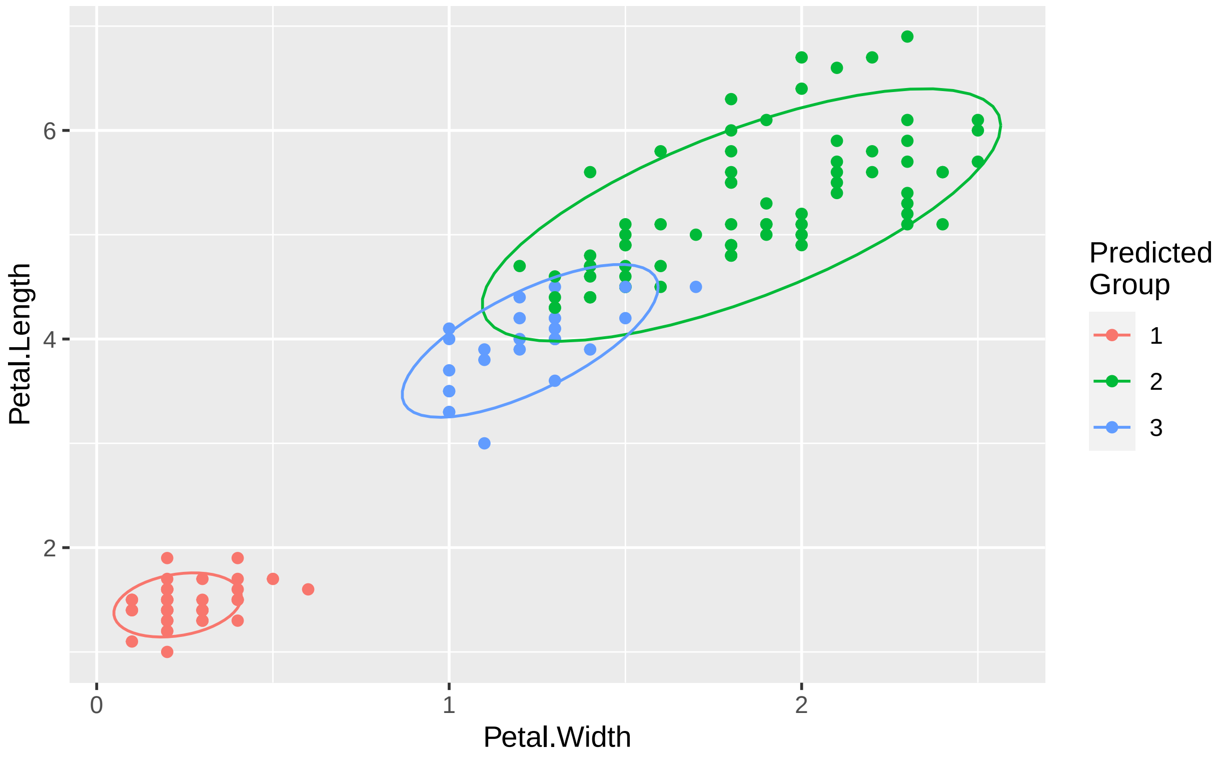
```
dst <- iris.mat %>% dist()
hc.iris.complete <- hclust(dst, method = 'complete')
hc.iris.average  <- hclust(dst, method = 'average')
hc.iris.single   <- hclust(dst, method = 'single')

hcPreds <- cutree(hc.iris.complete, k = 3)

# Make the Data
groupPred <- factor(hcPreds, levels = c(1,2,3), ordered = FALSE)
iris$KMpred <- groupPred

# Plot the Data
ggplot(iris, aes(y = Sepal.Length, x = Sepal.Width, col = KMpred)) +
  geom_point() +
  labs(col = "Predicted\nGroup", caption = "Ellipses represent 90% Normal confidence levels,\n
       predictions made using K-means algorithm with 2 classes") +
  stat_ellipse(level = 0.9)
```



Ellipses represent 90% Normal confidence levels,

predictions made using K−means algorithm with 2 classes

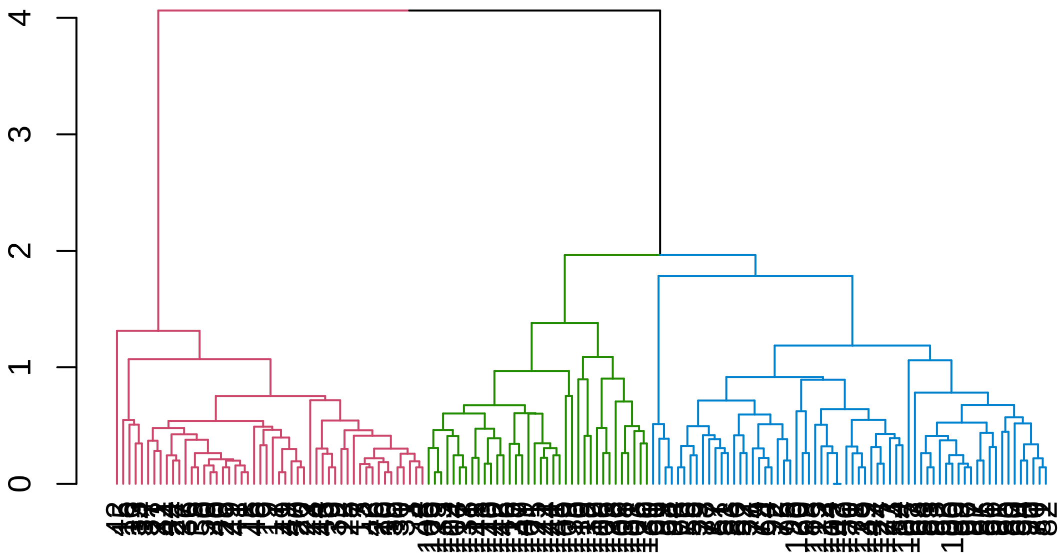
```
ggplot(iris, aes(y = Petal.Length, x = Petal.Width, col = KMpred)) +
  geom_point() +
  labs(col = "Predicted\nGroup", caption = "Ellipses represent 90% Normal confidence levels,\n
       predictions made using K-means algorithm with 2 classes") +
  stat_ellipse(level = 0.9)
```

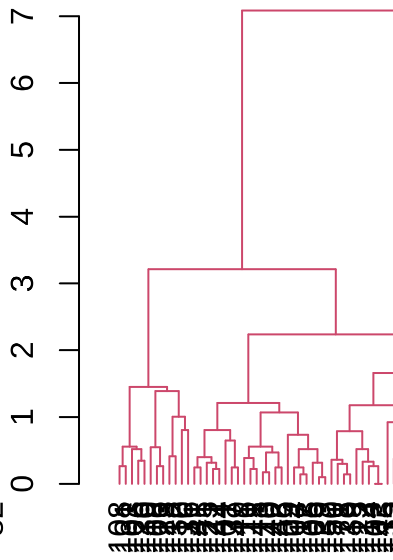


Ellipses represent 90% Normal confidence levels,

predictions made using K−means algorithm with 2 classes

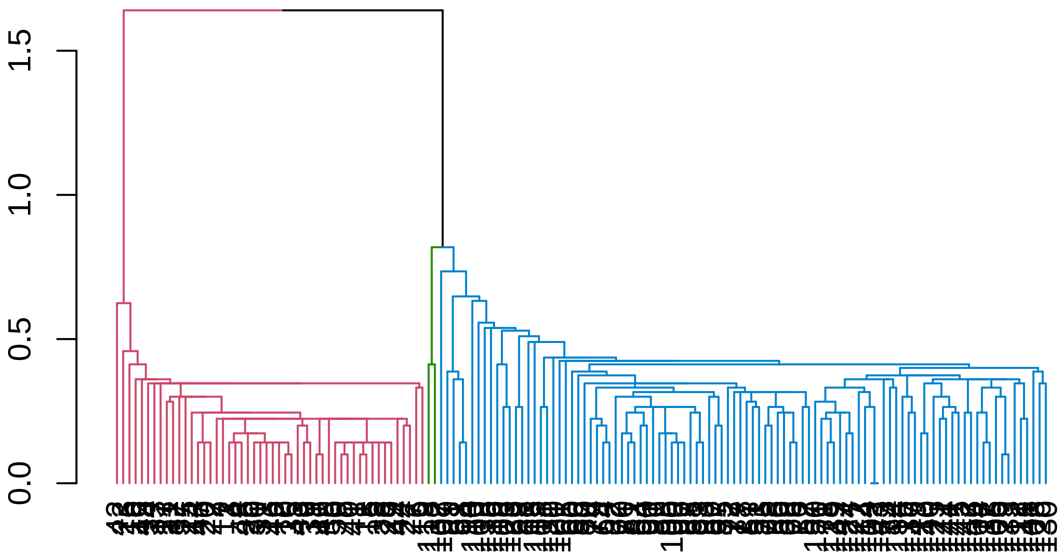
```
mods <- list(hc.iris.average, hc.iris.complete, hc.iris.single)
type.vec <- c("Average", "Complete", "Single")

for (i in 1:3) {
  hc = mods[[i]]
  hc <-  hc %>%    # Comment out for base without dendextend
   as.dendrogram() %>% set("branches_k_color", k = 3) %>%
   set("leaves_cex", 0.2)
 plot(hc, sub = paste("Built Using ", type.vec[i], " Linkage"), xlab = "", cex = 0.3)
}
```





Built Using Average Linkage

Bu



Built Using Single Linkage

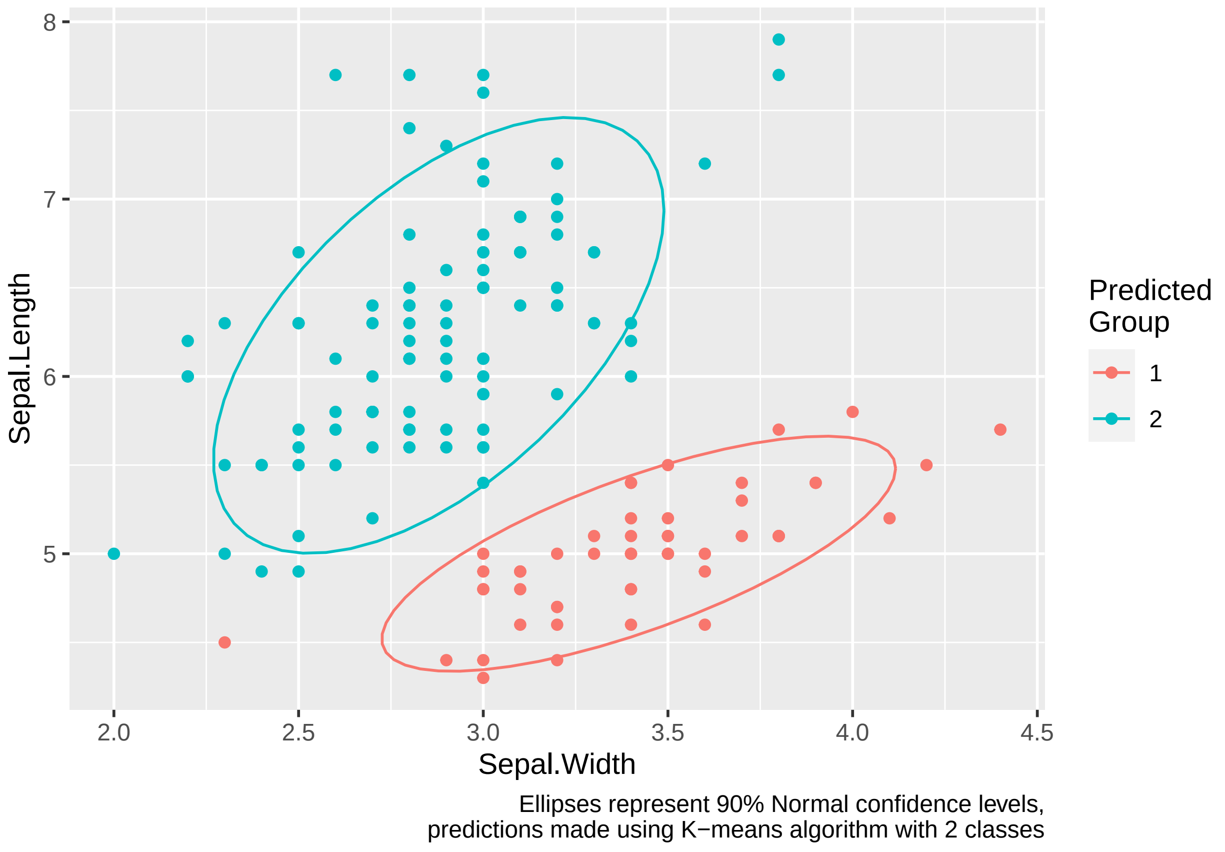
k-means clustering

```
data.frame(
"means"=apply(iris[,1:4], 2, mean), #2 means rows, 1 means cols
"sd"=apply(iris[,1:4], 2, sd)
) %>% print(,digits = 2)
```

```
##              means   sd
## Sepal.Length   5.8 0.83
## Sepal.Width    3.1 0.44
## Petal.Length   3.8 1.77
## Petal.Width    1.2 0.76
```

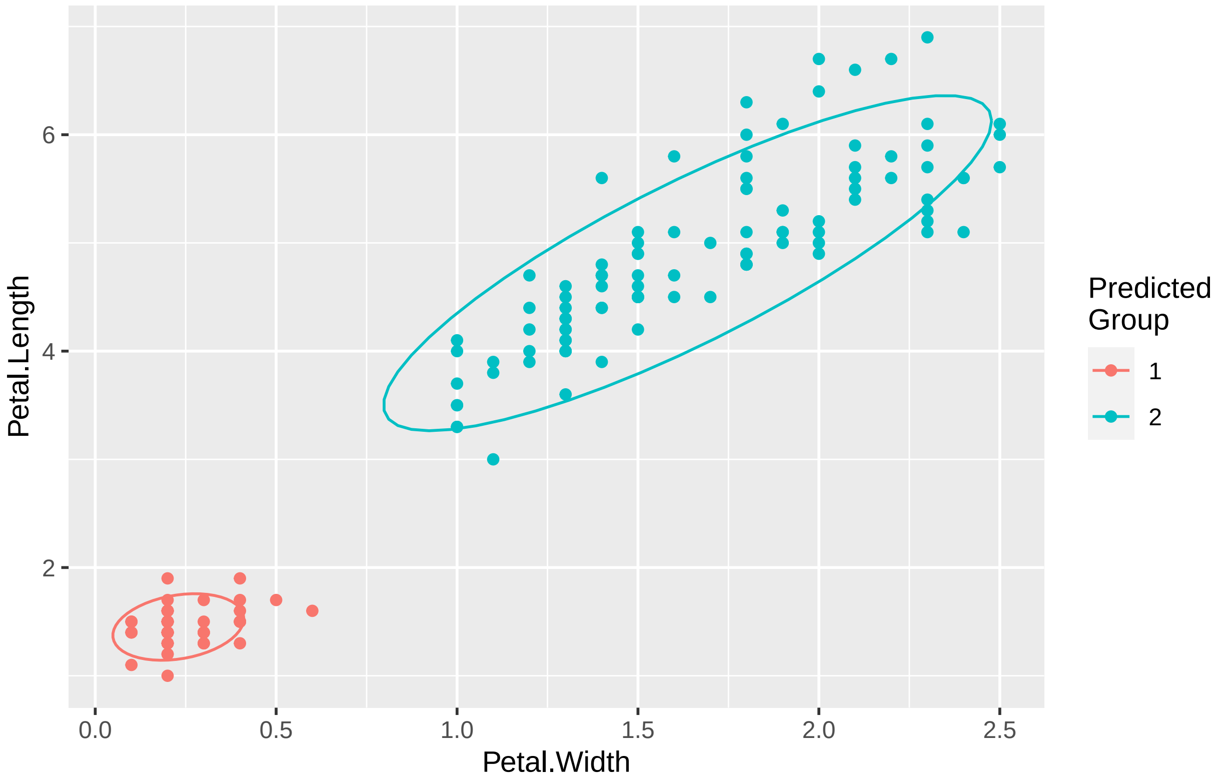
```
## we will scale the data because the petal width is much smaller than all other measurements.

iris.sc <- iris
iris.sc[,1:4] <- scale(iris[,1:4])
iris.sc.input <- iris.sc[,1:4]
```

```
irisKM.mod <- kmeans(iris.sc.input, centers = 2, nstart = 100)

# attached the inferred labels
groupPred <- factor(irisKM.mod$cluster, levels = c(1,2), ordered = FALSE)
iris$KMpred <- groupPred

# Plot the Data
ggplot(iris, aes(y = Sepal.Length, x = Sepal.Width, col = KMpred)) +
  geom_point() +
  labs(col = "Predicted\nGroup",
       caption = "Ellipses represent 90% Normal confidence levels,
       predictions made using K-means algorithm with 2 classes") +
  stat_ellipse(level = 0.9)
```



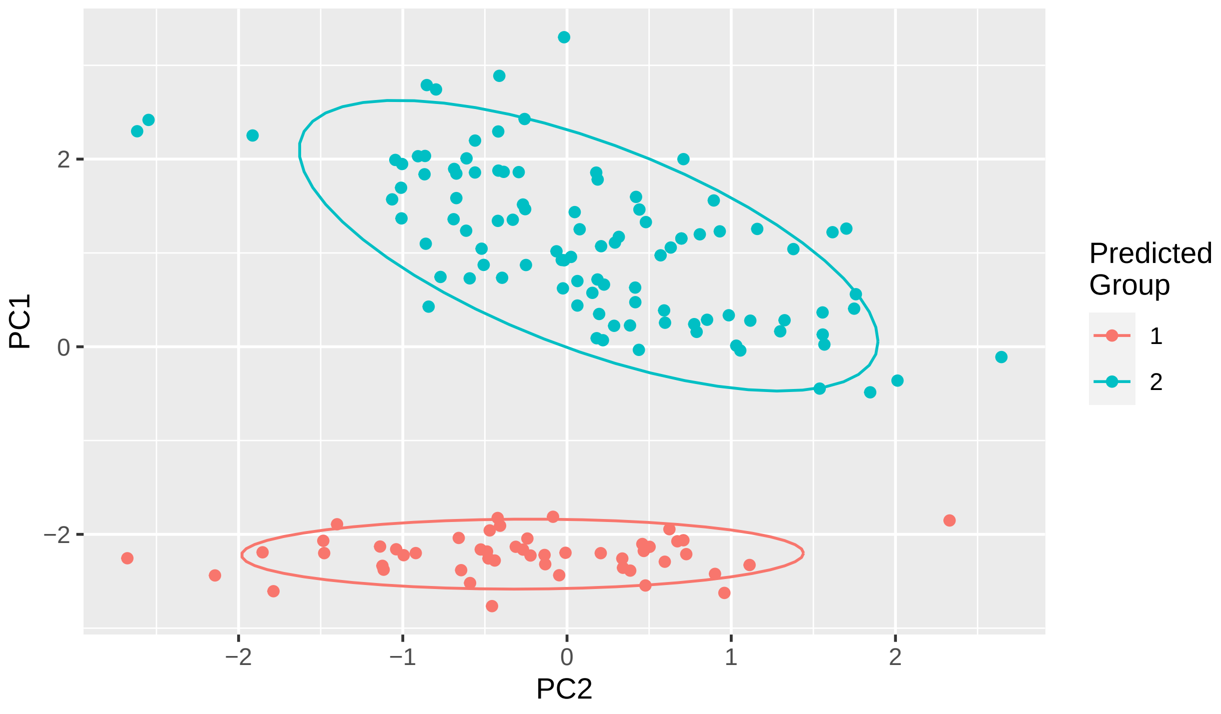
```
## but the sepals are not discriminative features
ggplot(iris, aes(y = Petal.Length, x = Petal.Width, col = KMpred)) +
  geom_point() +
  labs(col = "Predicted\nGroup",
```

```r
      caption = "Ellipses represent 90% Normal confidence levels,
      predictions made using K-means algorithm with 2 classes") +
  stat_ellipse(level = 0.9)
```



Ellipses represent 90% Normal confidence levels,
predictions made using K−means algorithm with 2 classes

```r
## it would be better to represent the "clusters" using the PCA reduction, since it incorporates inform

PCA.mod.iris <- prcomp(x = iris.sc.input)
PCADF <- PCA.mod.iris$x %>% as_tibble()

#Put the predicted groups on the end
PCADF$KM2Pred <- groupPred

#Draw the Plot
ggplot(PCADF, aes(y = PC1, x = PC2, col = KM2Pred)) +
  geom_point() +
  labs(col = "Predicted\nGroup",
      caption = "First two Principle Components of Iris Data,\n
      Ellipses represent 90% Normal confidence levels, \n
      predictions made using K-means algorithm with 2 classes") +
  stat_ellipse(level = 0.9)
```



First two Principle Components of Iris Data,

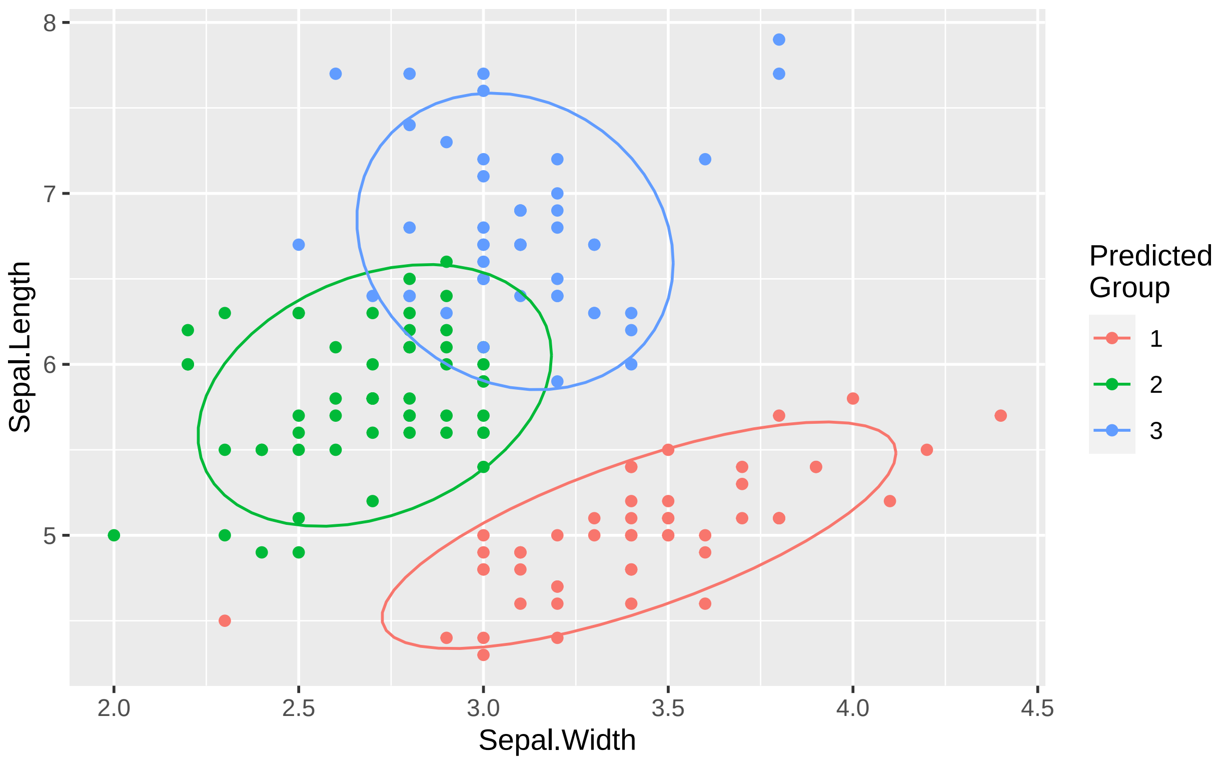
Ellipses represent 90% Normal confidence levels,

predictions made using K−means algorithm with 2 classes

```
irisKM.mod <- kmeans(iris.sc.input, centers = 3, nstart = 100)

# Make the Data
groupPred <- factor(irisKM.mod$cluster, levels = c(1,2,3), ordered = FALSE)
iris$KMpred <- groupPred
groupPred %>% print()
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 2 2 2 3 2 2 2 2 2 2 2 2 3 2 2 2 2 3 2 2 2
##  [75] 2 3 3 3 2 2 2 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3 3
## [112] 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 3 3 3 3 3 3 2 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [149] 3 2
## Levels: 1 2 3
```

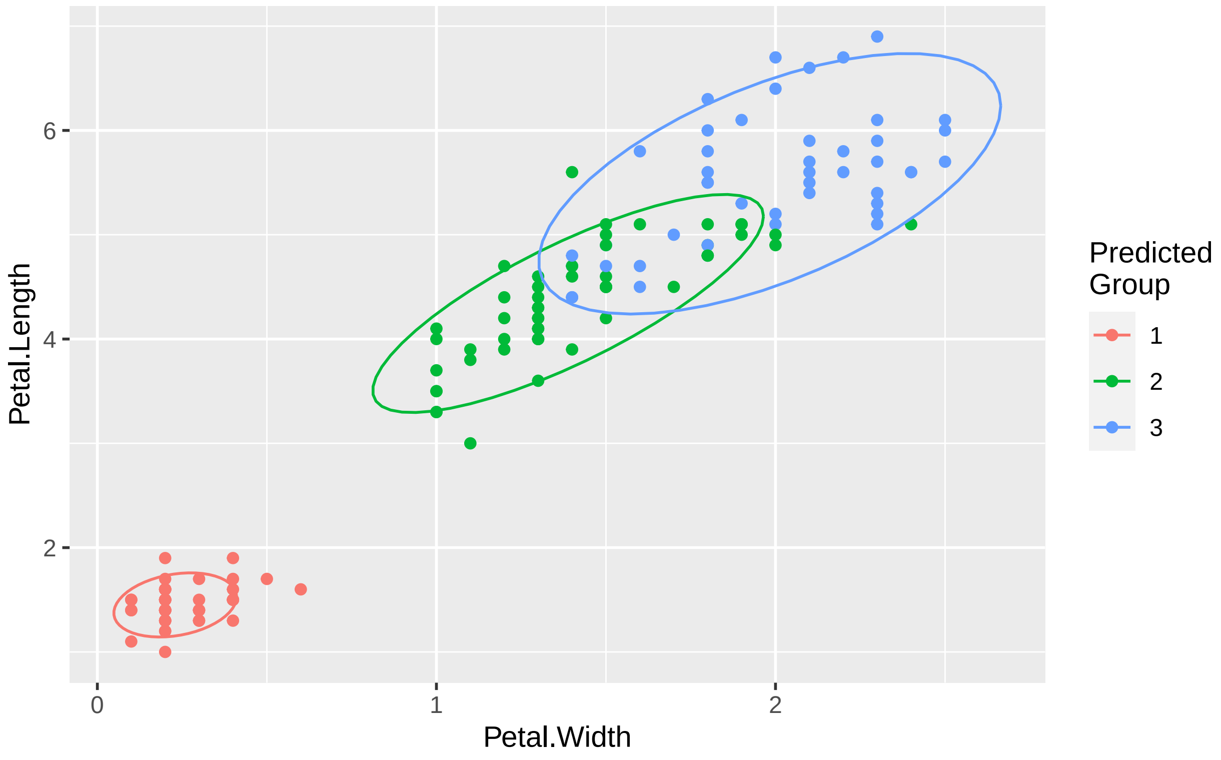
```
# plot the Data
groupPred <- factor(irisKM.mod$cluster, levels = c(1,2,3), ordered = FALSE)
iris$KMpred <- groupPred


ggplot(iris, aes(y = Sepal.Length, x = Sepal.Width, col = KMpred)) +
  geom_point() +
  labs(col = "Predicted\nGroup", caption = "Ellipses represent 90% Normal confidence levels,\n
       predictions made using K-means algorithm with 2 classes") +
  stat_ellipse(level = 0.9)
```



Ellipses represent 90% Normal confidence levels,

predictions made using K−means algorithm with 2 classes

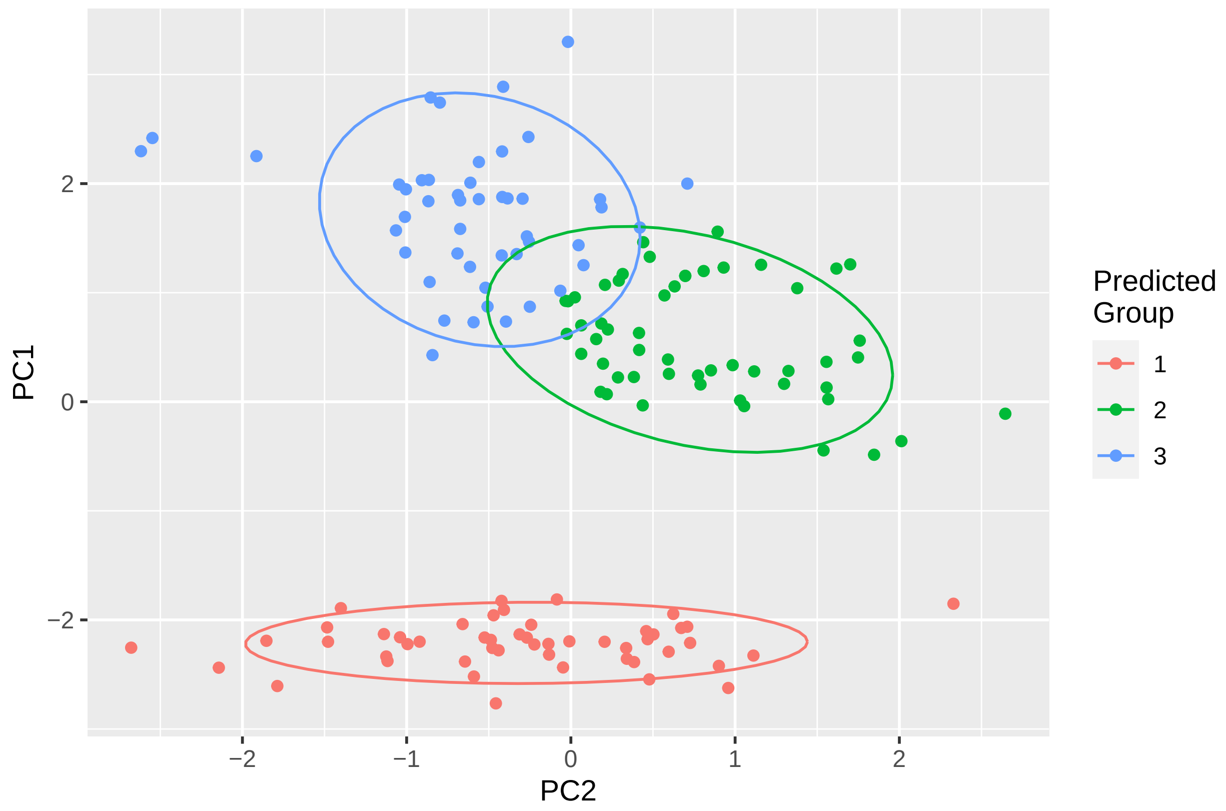
```r
ggplot(iris, aes(y = Petal.Length, x = Petal.Width, col = KMpred)) +
  geom_point() +
  labs(col = "Predicted\nGroup", caption = "Ellipses represent 90% Normal confidence levels,\n
      predictions made using K-means algorithm with 2 classes") +
  stat_ellipse(level = 0.9)
```



Ellipses represent 90% Normal confidence levels,

predictions made using K−means algorithm with 2 classes

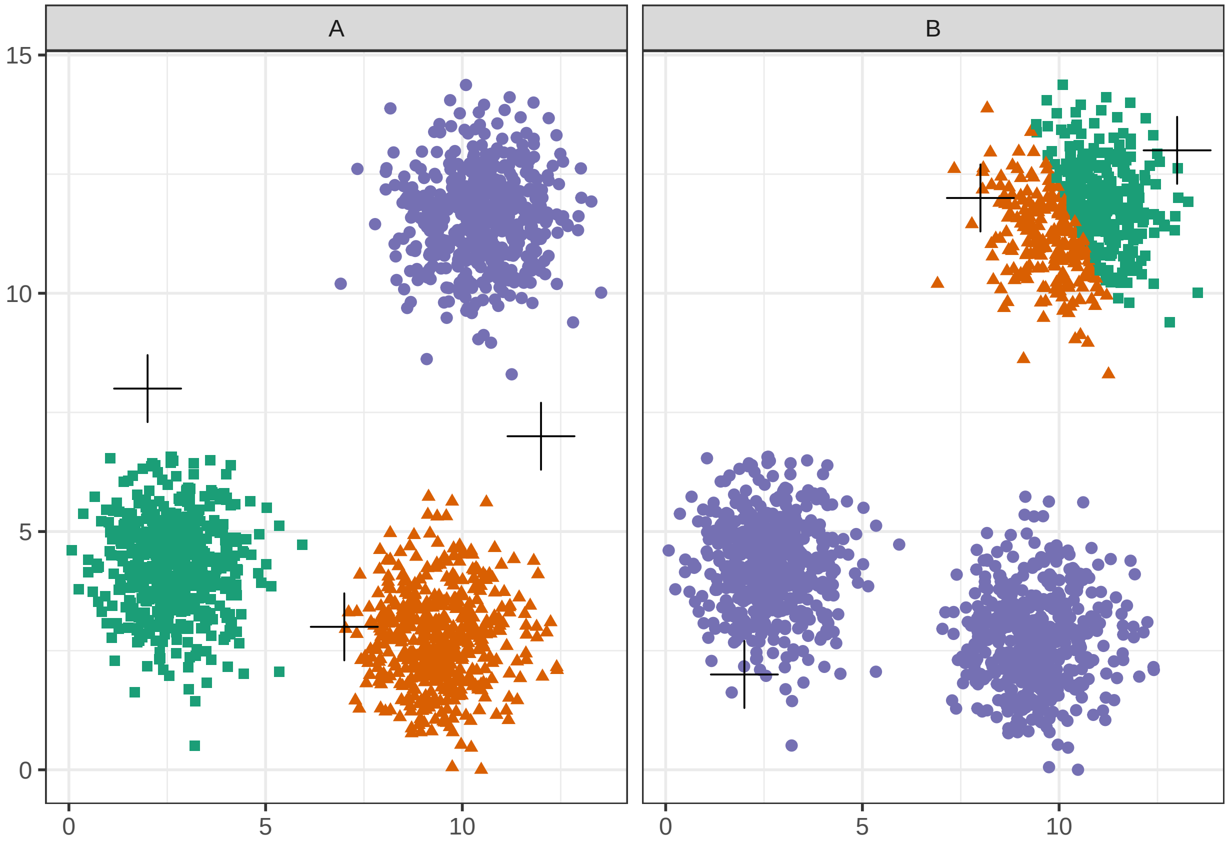
```r
#Put the predicted groups on the end
PCADF$KM2Pred <- groupPred

#Draw the Plot
ggplot(PCADF, aes(y = PC1, x = PC2, col = KM2Pred)) +
  geom_point() +
  labs(col = "Predicted\nGroup", caption = "First two Principle Components of Iris Data, Ellipses repres
  stat_ellipse(level = 0.9)
```



epresent 90% Normal confidence levels, predictions made using K−means algorithm with 2 classes

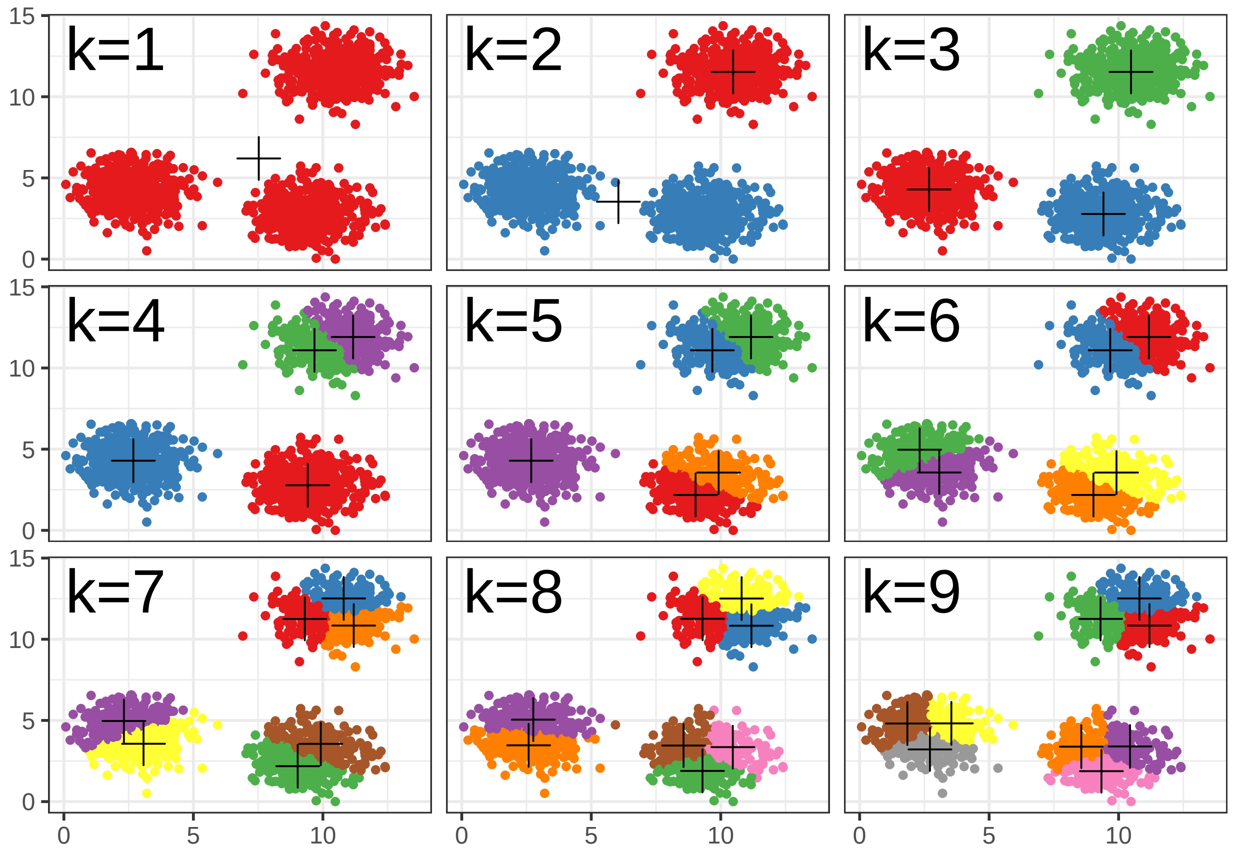
Example on tthe blob data

```r
library(RColorBrewer)
point_shapes <- c(15,17,19)
point_colours <- brewer.pal(3,"Dark2")
point_size = 1.5
center_point_size = 8

blobs <- as.data.frame(read.csv("blobs.csv", header=F))

good_centres <- as.data.frame(matrix(c(2,8,7,3,12,7), ncol=2, byrow=T))
bad_centres <- as.data.frame(matrix(c(13,13,8,12,2,2), ncol=2, byrow=T))

good_result <- kmeans(blobs[,1:2], centers=good_centres)
bad_result <- kmeans(blobs[,1:2], centers=bad_centres)

plotList <- list(
ggplot(blobs, aes(V1,V2)) +
  geom_point(col=point_colours[good_result$cluster], shape=point_shapes[good_result$cluster],
             size=point_size) +
  geom_point(data=good_centres, aes(V1,V2), shape=3, col="black", size=center_point_size) +
  theme_bw(),
ggplot(blobs, aes(V1,V2)) +
  geom_point(col=point_colours[bad_result$cluster], shape=point_shapes[bad_result$cluster],
             size=point_size) +
  geom_point(data=bad_centres, aes(V1,V2), shape=3, col="black", size=center_point_size) +
  theme_bw()
)

pm <- ggmatrix(
  plotList, nrow=1, ncol=2, showXAxisPlotLabels = T, showYAxisPlotLabels = T,
  xAxisLabels=c("A", "B")
) + theme_bw()

pm
```

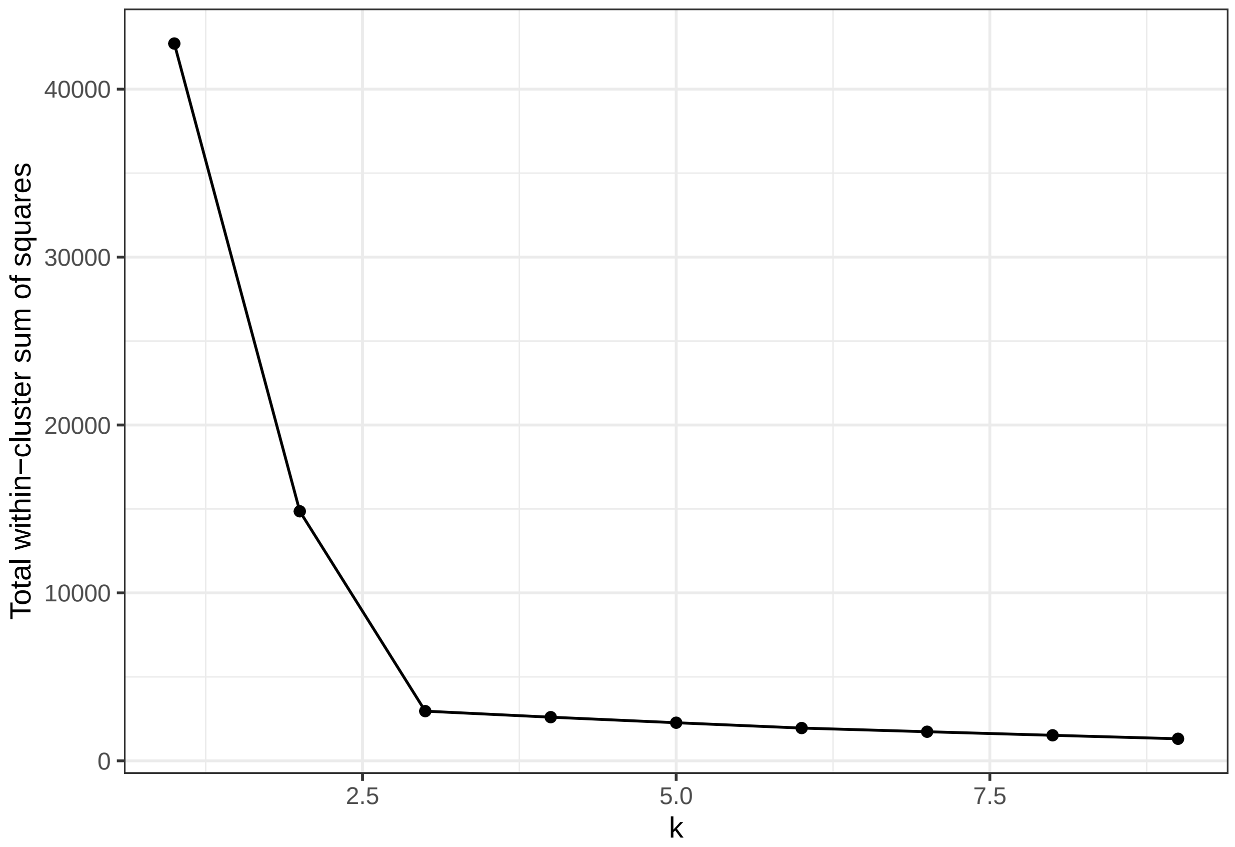


```
point_colours <- brewer.pal(9,"Set1")
k <- 1:9
res <- lapply(k, function(i){kmeans(blobs[,1:2], i, nstart=50)})

plotList <- lapply(k, function(i){
  ggplot(blobs, aes(V1, V2)) +
    geom_point(col=point_colours[res[[i]]$cluster], size=1) +
    geom_point(data=as.data.frame(res[[i]]$centers), aes(V1,V2), shape=3, col="black", size=5) +
    annotate("text", x=2, y=13, label=paste("k=", i, sep=""), size=8, col="black") +
    theme_bw()
}
)

pm <- ggmatrix(
  plotList, nrow=3, ncol=3, showXAxisPlotLabels = T, showYAxisPlotLabels = T
) + theme_bw()
```

```
pm
```



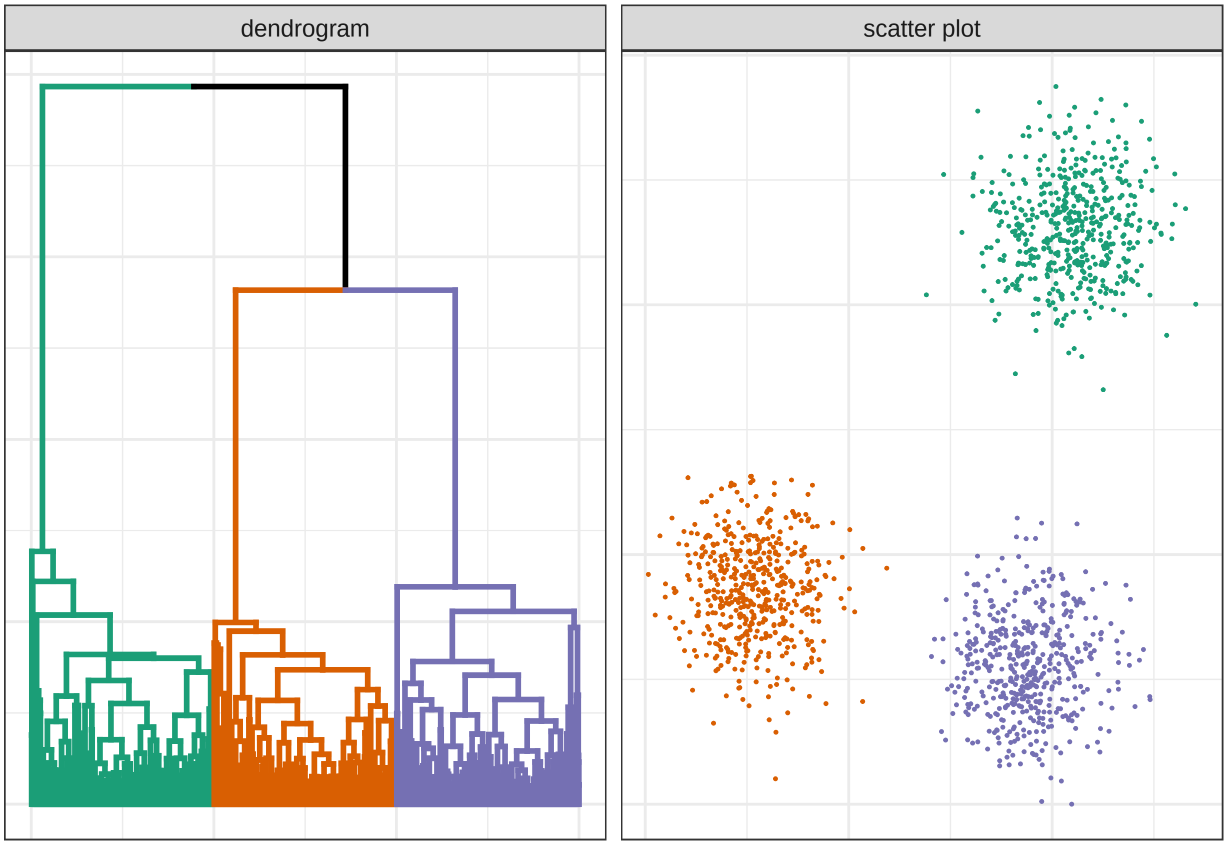
```
## to assess the variability witthin clusters, we can use a sum of squares approach
tot_withinss <- sapply(k, function(i){res[[i]]$tot.withinss})
qplot(k, tot_withinss, geom=c("point", "line"),
      ylab="Total within-cluster sum of squares") + theme_bw()
```



Clustering the blob data using a hierarchical approach:

```r
#Create distance matrix using Euclidean distance metric.
d <- dist(blobs[,1:2])

#Perform hierarchical clustering using the average agglomeration method and convert the result to an ob
dend <- as.dendrogram(hclust(d, method="average"))

#Cut the tree into three clusters
clusters <- cutree(dend,3,order_clusters_as_data=F)

cluster_colours <- brewer.pal(8,"Dark2")
dend <- color_branches(dend, clusters=clusters, col=cluster_colours[1:3])
labels(dend) <- rep("", length(blobs[,1]))

## plot the dendrogram
ggd <- as.ggdend(dend)
ggd$nodes <- ggd$nodes[!(1:length(ggd$nodes[,1])),]
clusters <- clusters[order(as.numeric(names(clusters)))]
plotList <- list(ggplot(ggd),
                 ggplot(blobs, aes(V1,V2)) +
                   geom_point(col=cluster_colours[clusters], size=0.2)
                 )

pm <- ggmatrix(
  plotList, nrow=1, ncol=2, showXAxisPlotLabels = F, showYAxisPlotLabels = F,
  xAxisLabels=c("dendrogram", "scatter plot")
) + theme_bw()
```

```
pm
```


dendrogram
scatter plot