```r
library("e1071")
library("class")
library("caret")
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
library(ggplot2)
library(stats)
```

```r
abalone <- read.csv(url("https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data

# rename columns
colnames(abalone) <- c("sex", "length", 'diameter', 'height', 'whole_weight', 'shucked_wieght', 'viscera

# 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'))

abalone$sex <- as.factor(abalone$sex)

# drop the sex column (categorical variable)
abalone.norm <- abalone[,-1]

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

```r
# Train classifiers using three different subsets of features
# 1st model: using length, diameter, and height
classifier1 <- naiveBayes(abalone[, c("length", "diameter", "height")], abalone$age.group)

# 2nd model: using whole weight, shucked weight, and shell weight
classifier2 <- naiveBayes(abalone[, c("whole_weight", "shucked_wieght", "shell_weight")], abalone$age.g

# 3rd model: using length, diameter, and whole weight
classifier3 <- naiveBayes(abalone[, c("length", "diameter", "whole_weight")], abalone$age.group)

# Evaluate classification with contingency tables for all models
contingency1 <- table(predict(classifier1, abalone[, c("length", "diameter", "height")]), abalone$age.g
contingency2 <- table(predict(classifier2, abalone[, c("whole_weight", "shucked_wieght", "shell_weight"
contingency3 <- table(predict(classifier3, abalone[, c("length", "diameter", "whole_weight")]), abalone

# Print contingency tables
print("Contingency Table for Model 1:")
```

```
## [1] "Contingency Table for Model 1:"
```

```r
print(contingency1)
```

```
##          Actual
## Predicted young adult  old
##     young   988   291   79
##     adult   406  1172  564
##     old      13   347  317
```

```r
print("Contingency Table for Model 2:")
```

```
## [1] "Contingency Table for Model 2:"
```

```r
print(contingency2)
```

```
##          Actual
## Predicted young adult  old
##     young  1158   497  192
##     adult   244  1090  515
##     old       5   223  253
```

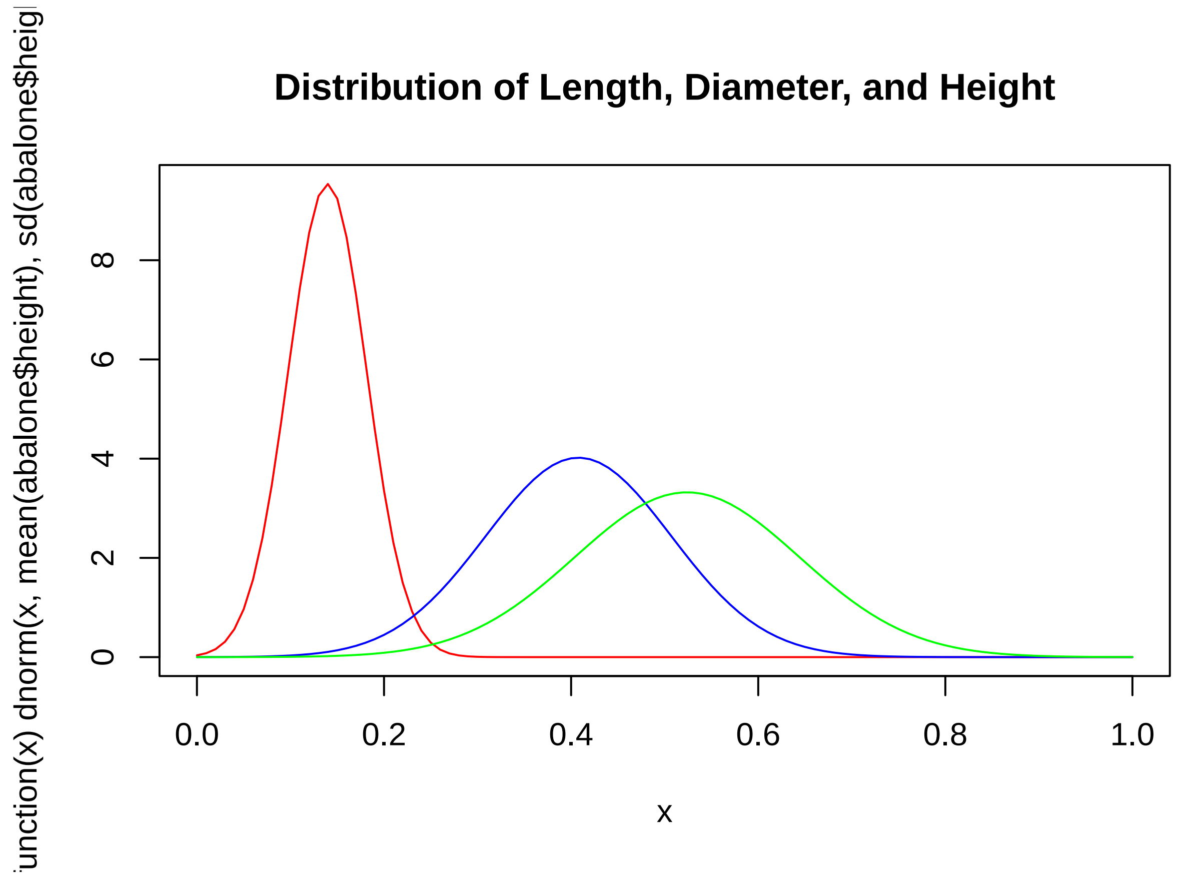
```r
print("Contingency Table for Model 3:")
```

```
## [1] "Contingency Table for Model 3:"
```

```r
print(contingency3)
```

```
##          Actual
## Predicted young adult  old
##     young  1053   345  125
##     adult   349  1204  608
##     old       5   261  227
```

```r
# Plot the distribution of classes along three different features (normalized data)
plot(function(x) dnorm(x, mean(abalone$height), sd(abalone$height)), col = "red", main = "Distribution 
curve(dnorm(x, mean(abalone$diameter), sd(abalone$diameter)), add = TRUE, col = "blue")
curve(dnorm(x, mean(abalone$length), sd(abalone$length)), add = TRUE, col = "green")
```

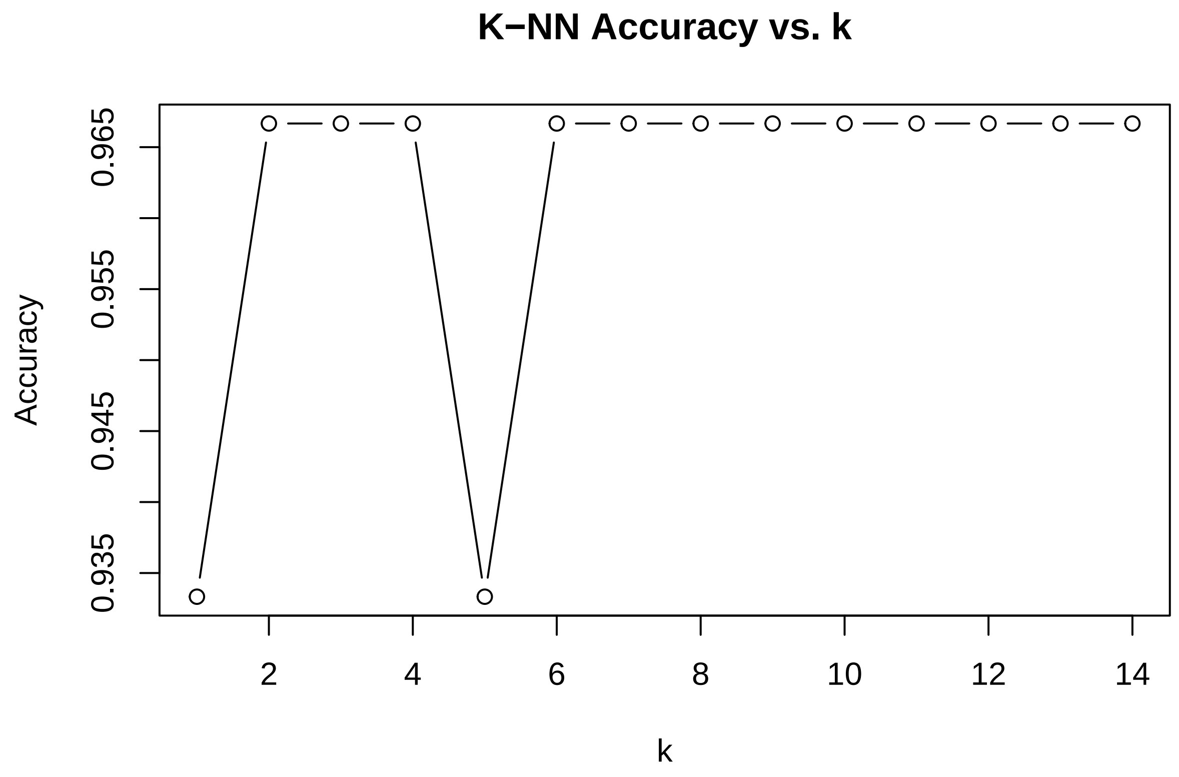


```r
# Normalize the first 4 columns and keep the species column
iris.norm <- as.data.frame(scale(iris[1:4]))
iris.norm$species <- iris$Species

# Create a random sample of 80% of the data
set.seed(123)  # Set seed for reproducibility
sample_index <- sample(seq_len(nrow(iris.norm)), size = 0.8 * nrow(iris.norm))

# Split the data into training and testing sets
train_data <- iris.norm[sample_index, ]
test_data <- iris.norm[-sample_index, ]
```

```r
k <- 3  # Set the number of neighbors (k)
KNNpred <- knn(train = train_data[1:4], test = test_data[1:4], cl = train_data$species, k = k)

# Create a contingency table / confusion matrix
contingency.table <- table(KNNpred, test_data$species)  # Adjust to test_data if needed

# Print the confusion matrix
print(contingency.table)
```

```
##
## KNNpred      setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         14         0
##   virginica       0          1         5
```

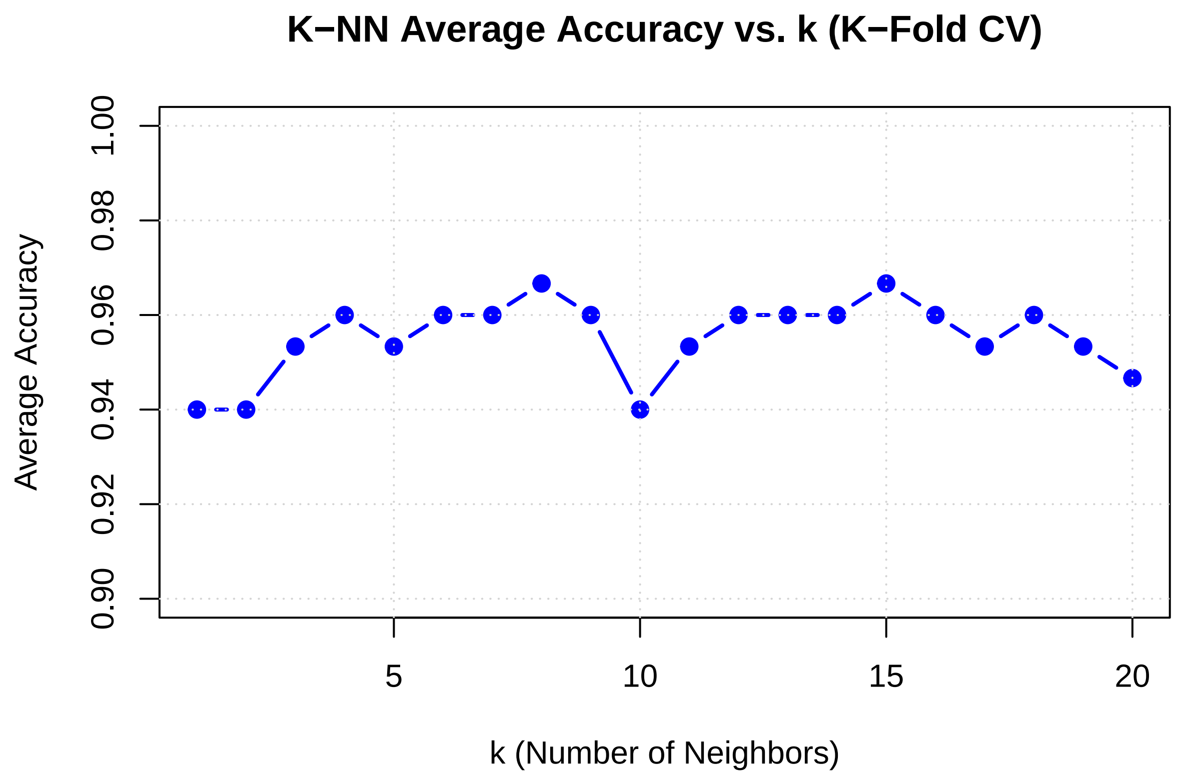
```r
ks <- c(1,2,3,4,5,6,7,8,9,10,11,12,13,14)

# Initialize an accuracy vector
accuracy <- numeric(length(ks))

# Loop through different values of k
for (i in seq_along(ks)) {
  k <- ks[i]  # Get the current value of k
  KNNpred <- knn(train = train_data[1:4], test = test_data[1:4], cl = train_data$species, k = k)

  # Create a confusion matrix
  cm <- as.matrix(table(KNNpred, test_data$species))

  # Calculate accuracy
  accuracy[i] <- sum(diag(cm)) / sum(cm)  # or use length(test_data$species)
}

# Plot the accuracy
plot(ks, accuracy, type = "b", xlab = "k", ylab = "Accuracy", main = "K-NN Accuracy vs. k")
```



The above is a step function, so here i use k-fold cross validation for smoother results.

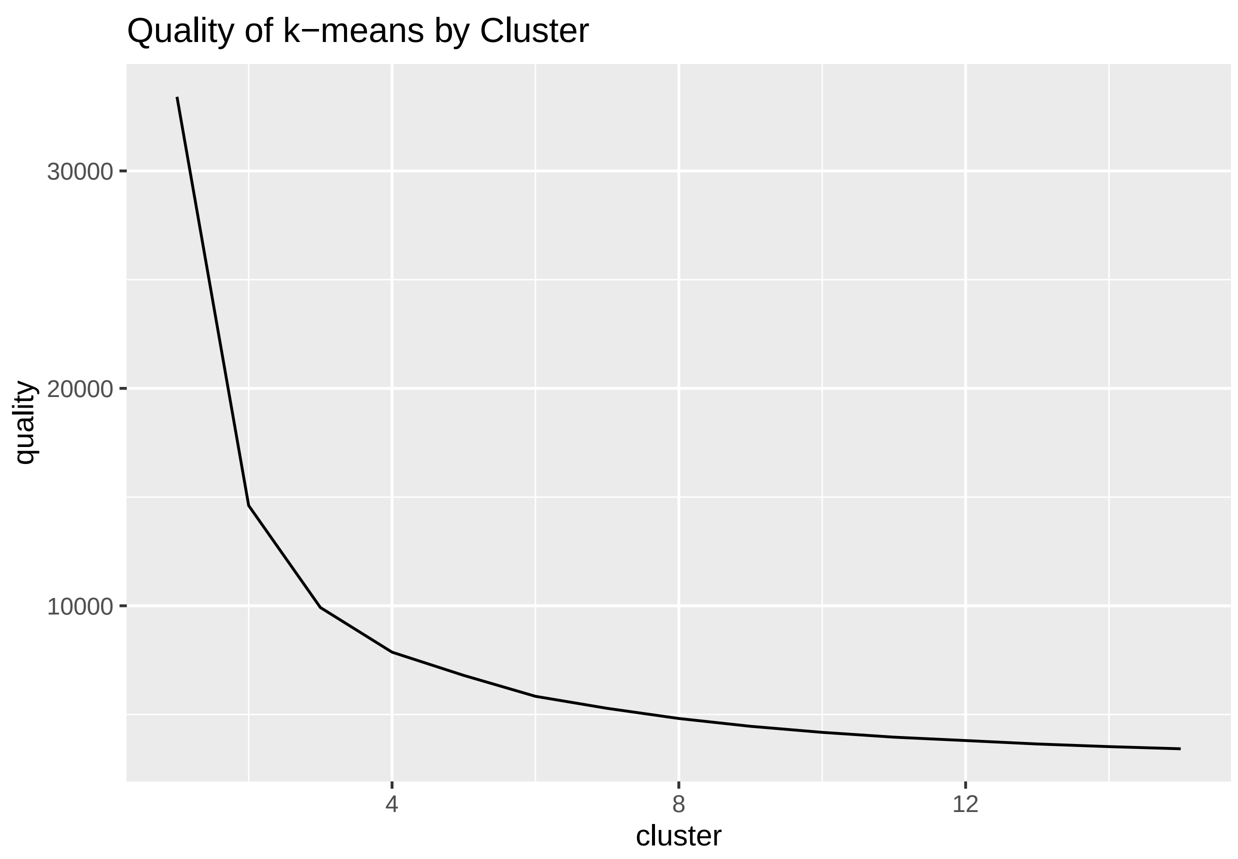
```r
# Load the iris dataset
data(iris)
```

```r
# Normalize the first 4 columns and keep the species column
iris.norm <- as.data.frame(scale(iris[1:4]))
iris.norm$species <- iris$Species

# Set the number of folds for cross-validation
k_folds <- 10

# Create a k-fold cross-validation partition
set.seed(123)  # Set seed for reproducibility
folds <- createFolds(iris.norm$species, k = k_folds, list = FALSE)

# Initialize a matrix to store accuracy for different k values across folds
k_values <- seq(1, 20)
fold_accuracy <- matrix(0, nrow = k_folds, ncol = length(k_values))

# Loop through each fold
for (fold in seq_len(k_folds)) {
  # Split the data into training and testing sets based on the fold
  test_data <- iris.norm[folds == fold, ]
  train_data <- iris.norm[folds != fold, ]

  # Loop through different values of k
  for (i in seq_along(k_values)) {
    k <- k_values[i]  # Get the current value of k
    KNNpred <- knn(train = train_data[1:4], test = test_data[1:4], cl = train_data$species, k = k)

    # Create a confusion matrix
    cm <- as.matrix(table(KNNpred, test_data$species))

    # Calculate accuracy for this k
    fold_accuracy[fold, i] <- sum(diag(cm)) / sum(cm)  # Store accuracy for the current fold and k
  }
}

# Average accuracy across all folds for each k value
average_accuracy <- colMeans(fold_accuracy)

# Plot the average accuracy vs. k values
plot(k_values, average_accuracy, type = "b", pch = 19, col = "blue", lwd = 2,
     xlab = "k (Number of Neighbors)", ylab = "Average Accuracy",
     main = "K-NN Average Accuracy vs. k (K-Fold CV)", ylim = c(0.9, 1))
grid()  # Add grid lines
```

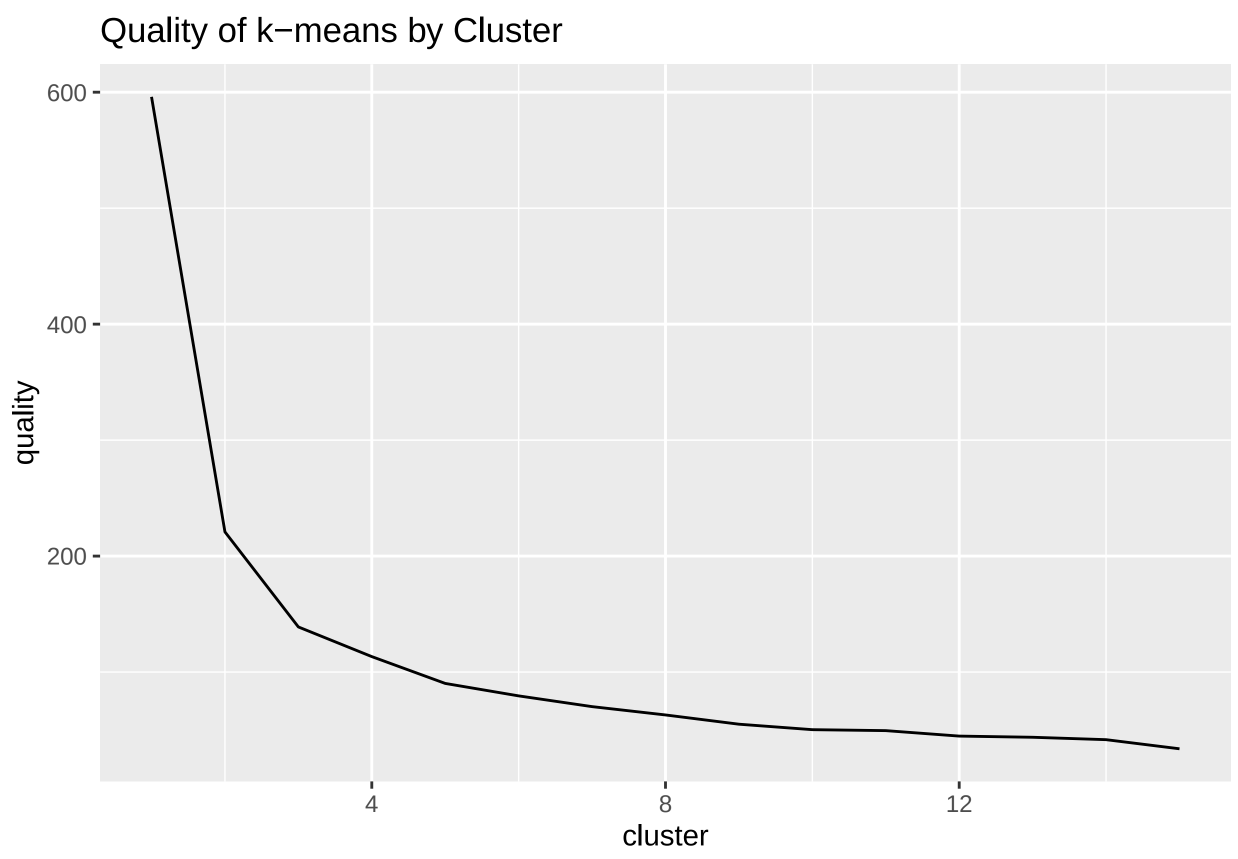


```r
# load iris and abalone datasets for K-Means

abalone <- read.csv(url("https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data

# rename columns
colnames(abalone) <- c("sex", "length", 'diameter', 'height', 'whole_weight', 'shucked_wieght', 'viscera

abalone.norm <- as.data.frame(lapply(abalone[2:9], scale))

iris <- datasets::iris
iris.norm <- as.data.frame(lapply(iris[1:4], scale))
```

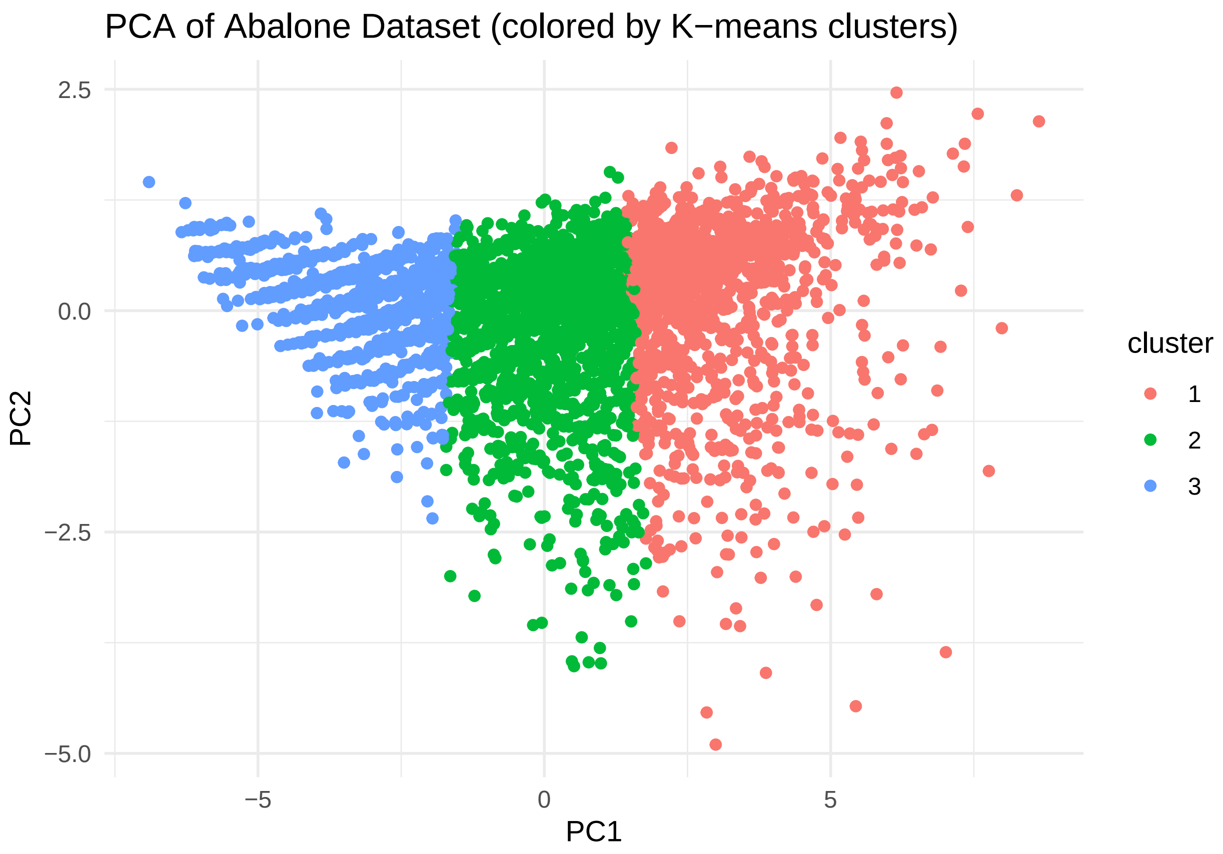
```r
# A user-defined function to examine clusters and plot the results
wssplot <- function(data, nc=15, seed=10){
  wss <- data.frame(cluster=1:nc, quality=c(0))
  for (i in 1:nc){
    set.seed(seed)
    wss[i,2] <- kmeans(data, centers=i)$tot.withinss}
  ggplot(data=wss,aes(x=cluster,y=quality)) +
    geom_line() +
    ggtitle("Quality of k-means by Cluster")
}

wssplot(abalone.norm, nc = 15, seed=1)
```



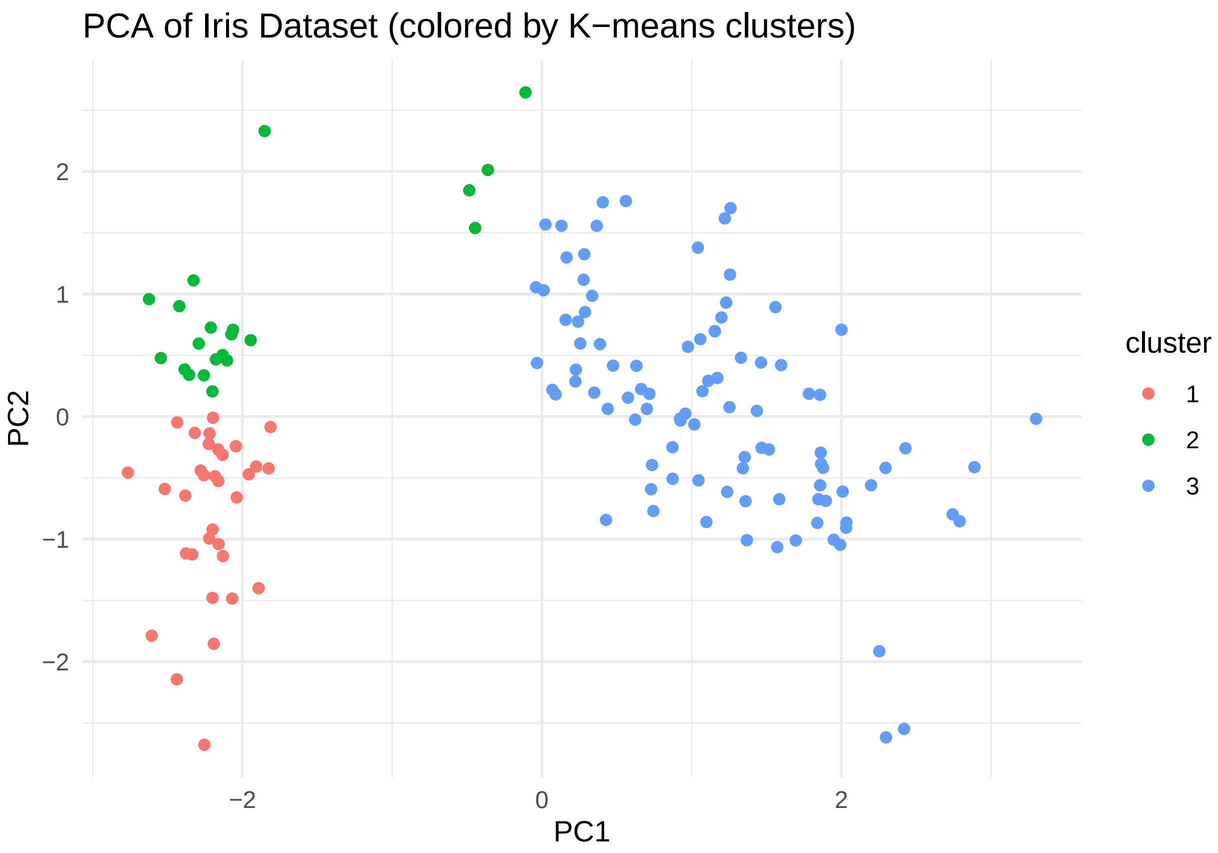
```
wssplot(iris.norm, nc = 15, seed=1)
```



```r
# Perform K-means clustering with k = 3 for abalone and iris datasets
set.seed(1)
kmeans_abalone <- kmeans(abalone.norm, centers = 3)
kmeans_iris <- kmeans(iris.norm, centers = 3)

# Add the K-means cluster result as a new column in the normalized data
abalone.norm$cluster <- as.factor(kmeans_abalone$cluster)
iris.norm$cluster <- as.factor(kmeans_iris$cluster)

# Perform PCA on the normalized data
abalone_pca <- prcomp(abalone.norm[1:8], center = TRUE, scale. = TRUE)
iris_pca <- prcomp(iris.norm[1:4], center = TRUE, scale. = TRUE)

# Create a data frame for plotting the PCA results, including the clusters
abalone_pca_data <- data.frame(abalone_pca$x[, 1:2], cluster = abalone.norm$cluster)
iris_pca_data <- data.frame(iris_pca$x[, 1:2], cluster = iris.norm$cluster)

# Plot PCA results for abalone dataset, colored by K-means cluster
abalone_plot <- ggplot(abalone_pca_data, aes(x = PC1, y = PC2, color = cluster)) +
  geom_point() +
  ggtitle("PCA of Abalone Dataset (colored by K-means clusters)") +
  theme_minimal()

# Plot PCA results for iris dataset, colored by K-means cluster
iris_plot <- ggplot(iris_pca_data, aes(x = PC1, y = PC2, color = cluster)) +
  geom_point() +
  ggtitle("PCA of Iris Dataset (colored by K-means clusters)") +
  theme_minimal()

# Print the plots
abalone_plot
```



```
iris_plot
```


PCA of Iris Dataset (colored by K−means clusters)
PC2
PC1
2
1
0
−1
−2
−2
0
2
cluster
1
2
3