K-Means Clustering on the Iris Data Set

Eric Cai

2023-09-07

The iris data set iris.txt contains 150 data points, each with four predictor variables and one categorical response. The predictors are the width and length of the sepal and petal of flowers and the response is the type of flower. The response values are only given to see how well a specific method performed and should not be used to build the model.

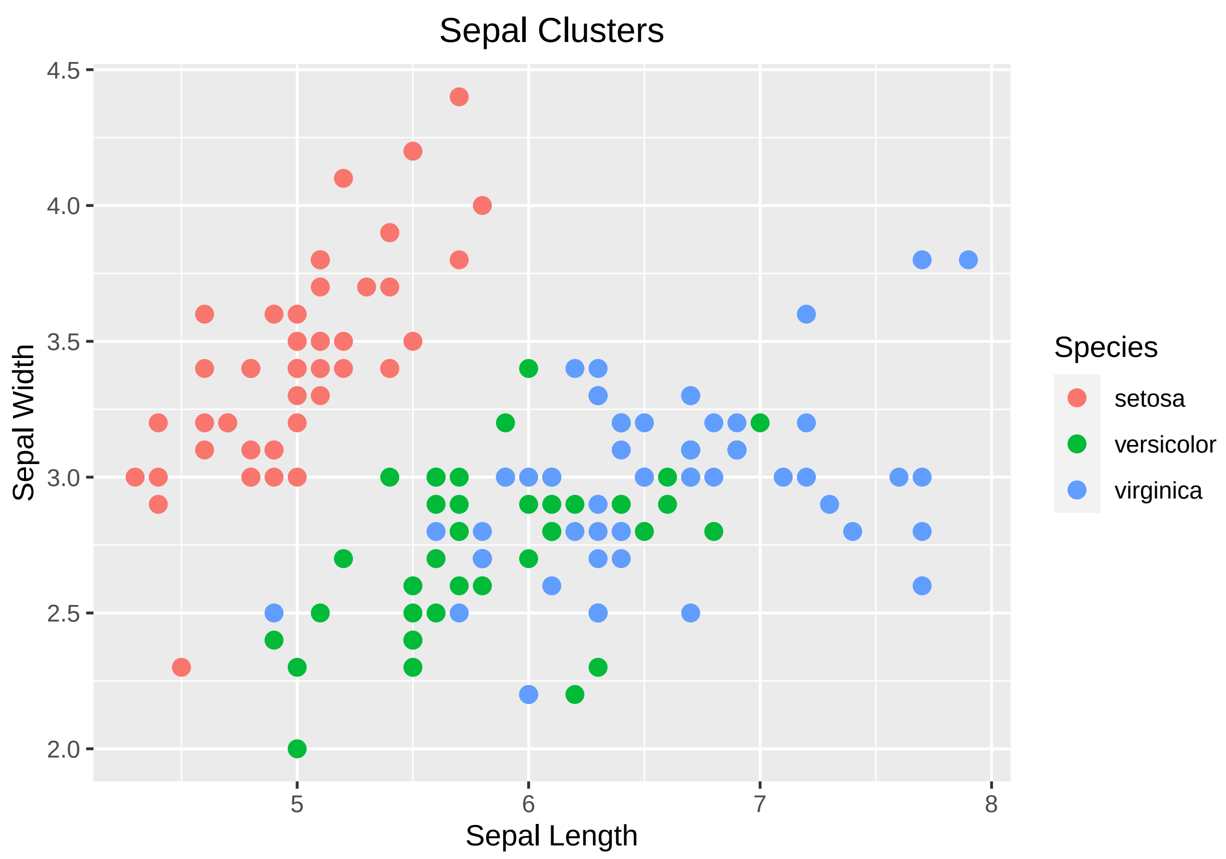
Using the R function kmeans, I will cluster the observations found in iris.txt, report the best combination of predictors, suggest value of k-clusters, and analyze how well it predicts flower type.

```
iris_file <- 'data 4.1/iris.txt'
iris <- read.table(iris_file)
# let's inspect it to see what the table looks like
head(iris)
```

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

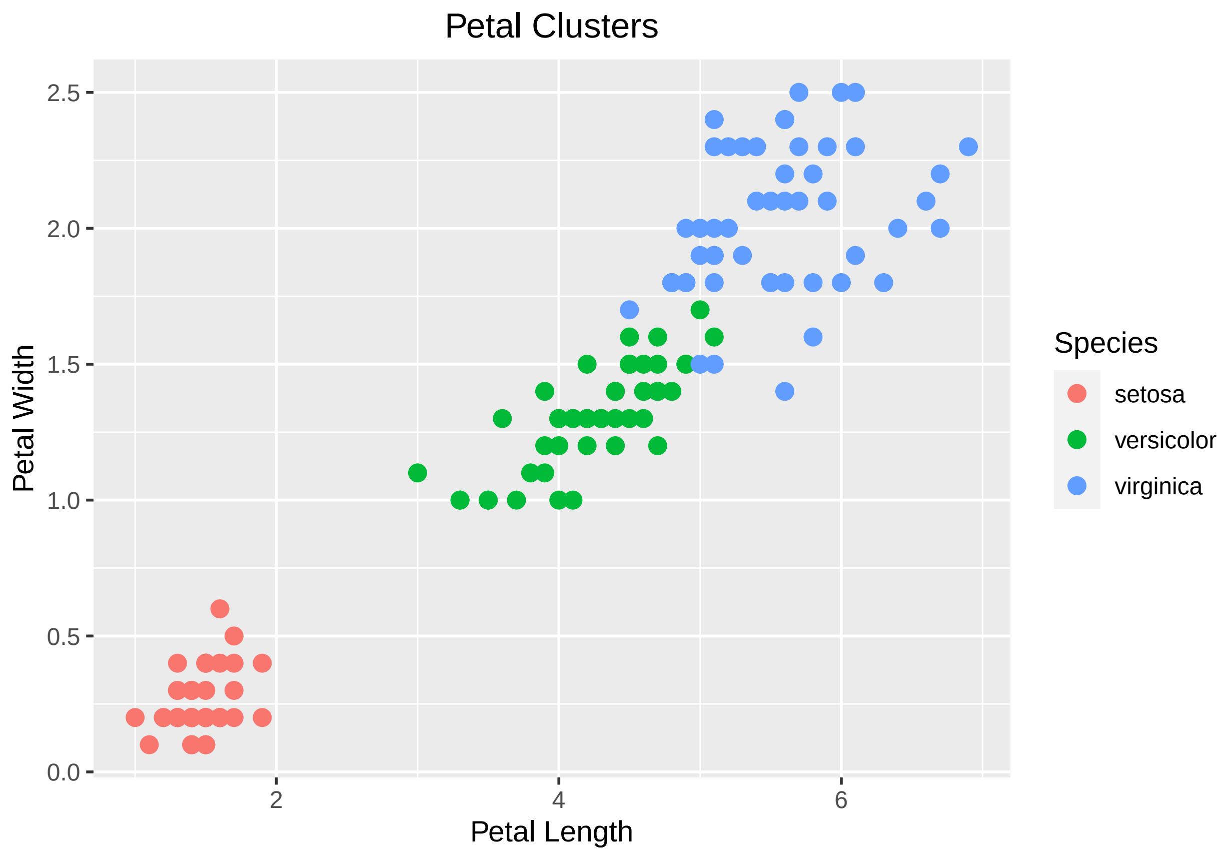
Plotting Iris Let's also see how the data clusters on a scatter plot using the length and width of the sepal only.

```
ggplot(iris,
       aes(x = Sepal.Length,
           y = Sepal.Width,
           color = Species)) +
  geom_point(shape = 16,
             size = 3) +
  labs(
    title = 'Sepal Clusters',
    x = 'Sepal Length',
    y = 'Sepal Width'
  ) +
theme(plot.title = element_text(hjust = 0.5))
```



Normally, we don't have the luxury of knowing what the number of clusters or even what the clusters are. Since we do, we know that the data reveals three cluster-species of irises: setosa, versicolor, and virginica, with versicolor and verginica overlapping in similar features, which is interesting. Let's see how it plots when we compare petal features.

```
ggplot(iris,
       aes(x = Petal.Length,
           y = Petal.Width,
           color = Species)) +
  geom_point(shape = 16,
             size = 3) +
  labs(
    title = 'Petal Clusters',
    x = 'Petal Length',
    y = 'Petal Width'
  ) +
theme(plot.title = element_text(hjust = 0.5))
```



So this is interesting! The petal clusters seem to separate better when we're using the petal predictors, as opposed to the sepal predictors. We'll keep that in mind when we build our model.

Even though we won't be using our response values to build the model, we should still see what the breakdown of our response vector.

```
table(iris$Species)
```

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

```
# since we won't be using the response values, we'll exclude it.
iris_data <- iris[, -5]
```

Hopefully, knowing the breakdown doesn't bias our experiment on k-means clustering!

Scaling Our Data Now, what we find in our data frame is that the sepal features and the petal features are on different scales. So before we start, we need to scale our data. Let's scale our data down to the same interval such that our data is between 0 to 1 (min-max normalization).

```
normalize <- function(x){
        return((x-min(x))/(max(x)-min(x)))
}
iris_normalized <- as.data.frame(lapply(iris_data, normalize))
head(iris_normalized)
```

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

We know that a 3 cluster solution best matches the data, but just for the sake of this assignment, let's choose from 1 to 7 clusters and "see" which model accurately predicts the correct amount of clusters. Per Dr. Sokol's recommendation, I'm going to be making elbow diagrams to determine where the kink in the curve is, thereby getting a better sense of how many clusters we should use.

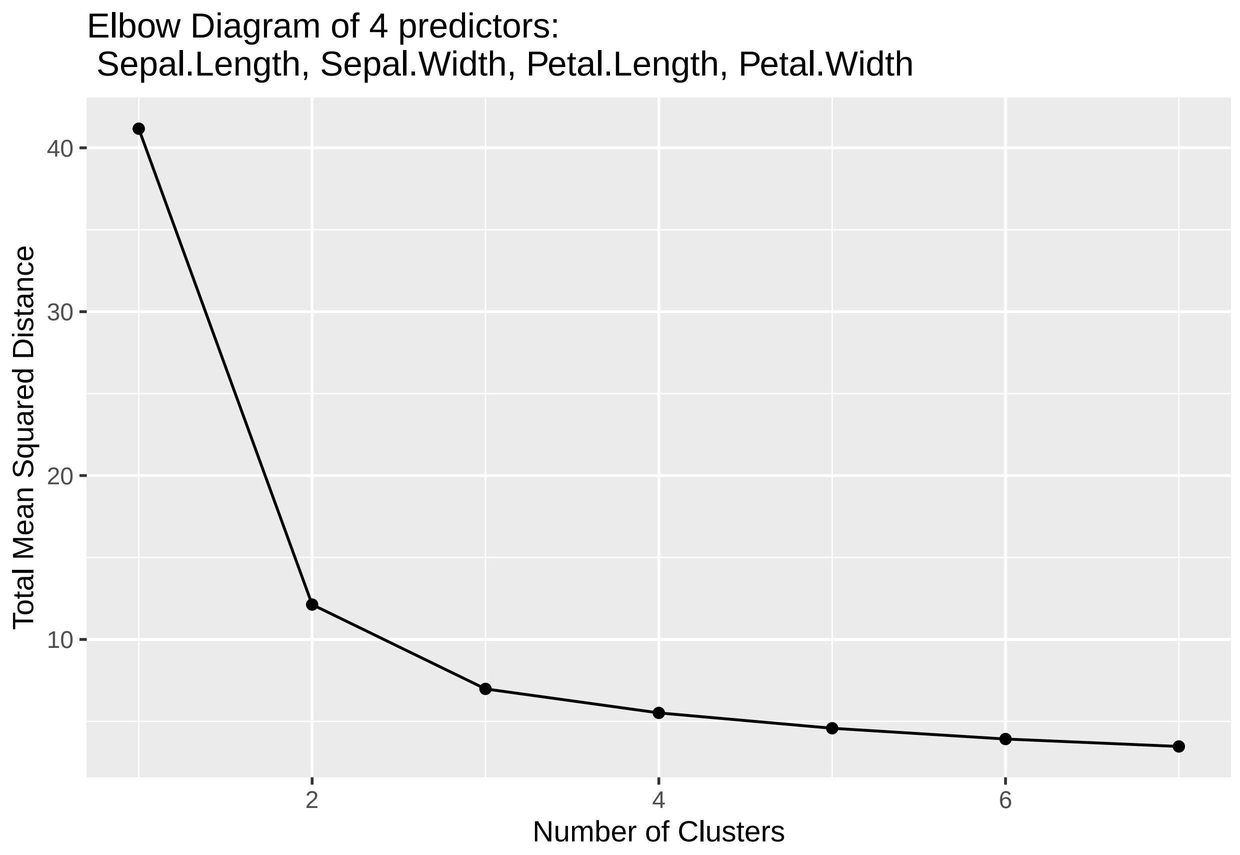
Building Our Model To construct our algorithm, we'll be using kmeans(). One argument of note is the nstart argument, where it says, "if centers is a number, how many random sets should be chosen?"

ISL (*Introduction to Statistical Learning*) recommends always running K-means clustering with a large value of nstart (20 <= nstart <= 50), because it helps the algorithm explore a broader range of starting points for cluster centroids, reducing the risk of converging to a suboptimal clustering result. The only downside is that, if our data is complex, which thankfully it's not, it could be computationally hard.

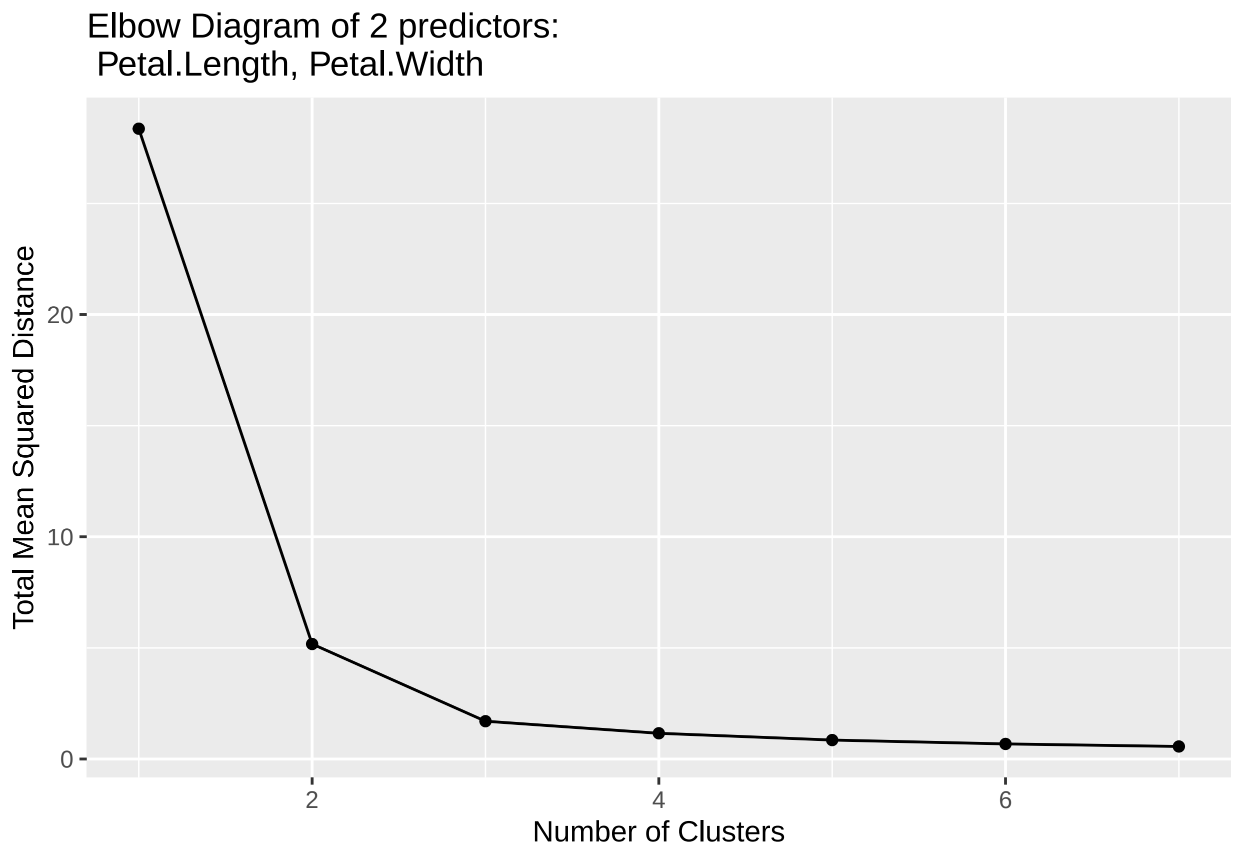
```r
# we'll store this as a function for later use
elbow_diagram <- function(dataset){
        tot_dist <- data.frame(K = integer(), totwithinss = double())
        k_vals <- 1:7
        for(k in k_vals){
                kmeans_result <- kmeans(dataset,
                                        centers=k,
                                        nstart=25)
                tot_dist <- rbind(tot_dist,
                                  data.frame(K = k,
                                             totwithinss = kmeans_result$tot.withinss))
        }
        return(ggplot(tot_dist, aes(x = K, y = totwithinss)) +
          geom_line() +
          geom_point() +
          labs(x = "Number of Clusters", y = "Total Mean Squared Distance") +
          ggtitle(paste("Elbow Diagram of",
                        ncol(dataset),
                        "predictors:\n",
                        paste(colnames(dataset), collapse= ', '))))
}
```

Elbow Diagrams The elbow diagram used all four predictors and as expected, the total mean squared distance decreases as our number of clusters increase. This means that each observation gets closer and closer to their respective cluster centroid as the number of centroids increase, which is good because the total mean squared distance is the function we're trying to minimize.

If you'll remember, our petal clusters graph above separated nicely. How would the elbow diagram look if we just included the petal clusters as our dataset?

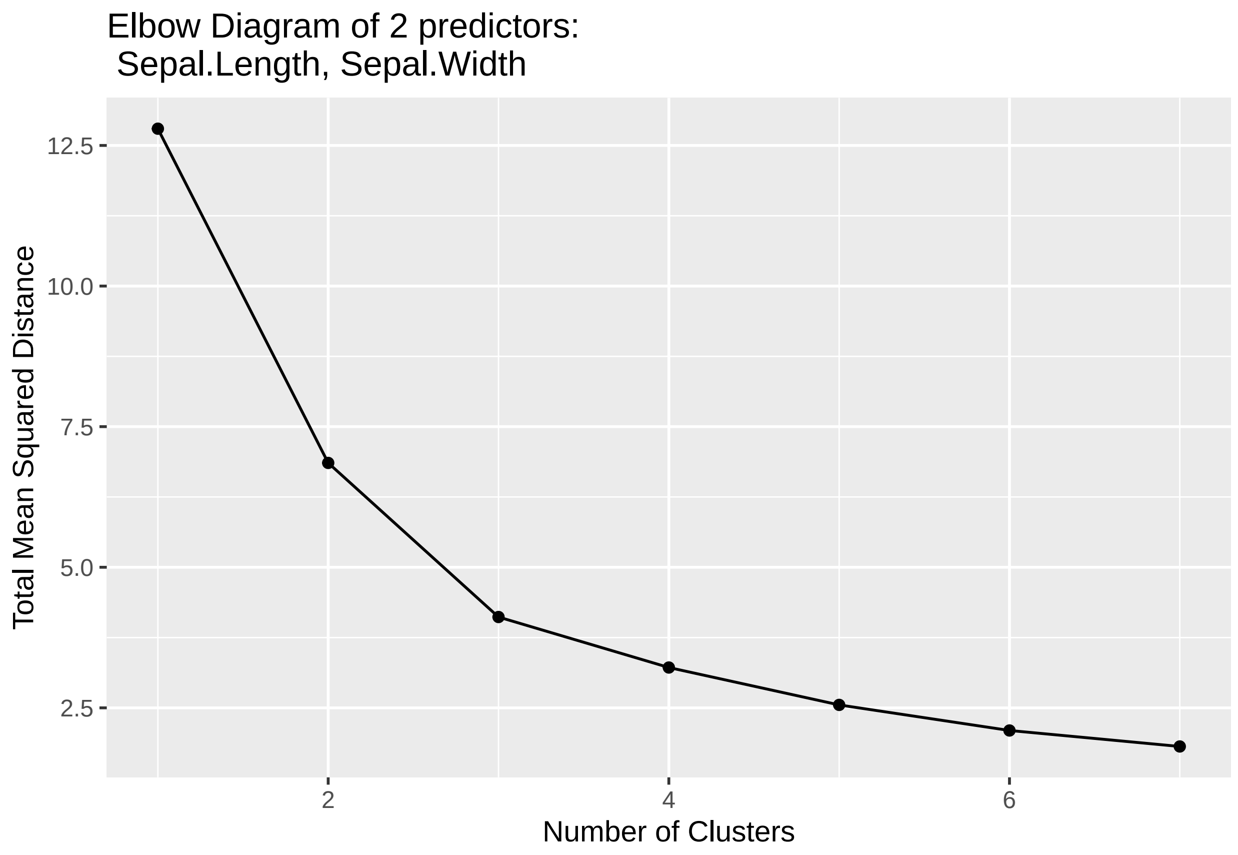


```
elbow_diagram(dataset=iris_normalized[, 3:4])
```



And for our sepal clusters?

```r
elbow_diagram(dataset=iris_normalized[, 1:2])
```



Determining Best Predictors It looks like the points all kink at cluster 3, which is what we expected, but it doesn't really seem to help us identify the best predictors. But since we saw that our petal predictors seemed to cluster the best and looked to be highly correlated with each other, I'm inclined to choose those as our best predictors. However, I want a more robust, statistical way of proving that they are, rather than by graphical representation only.

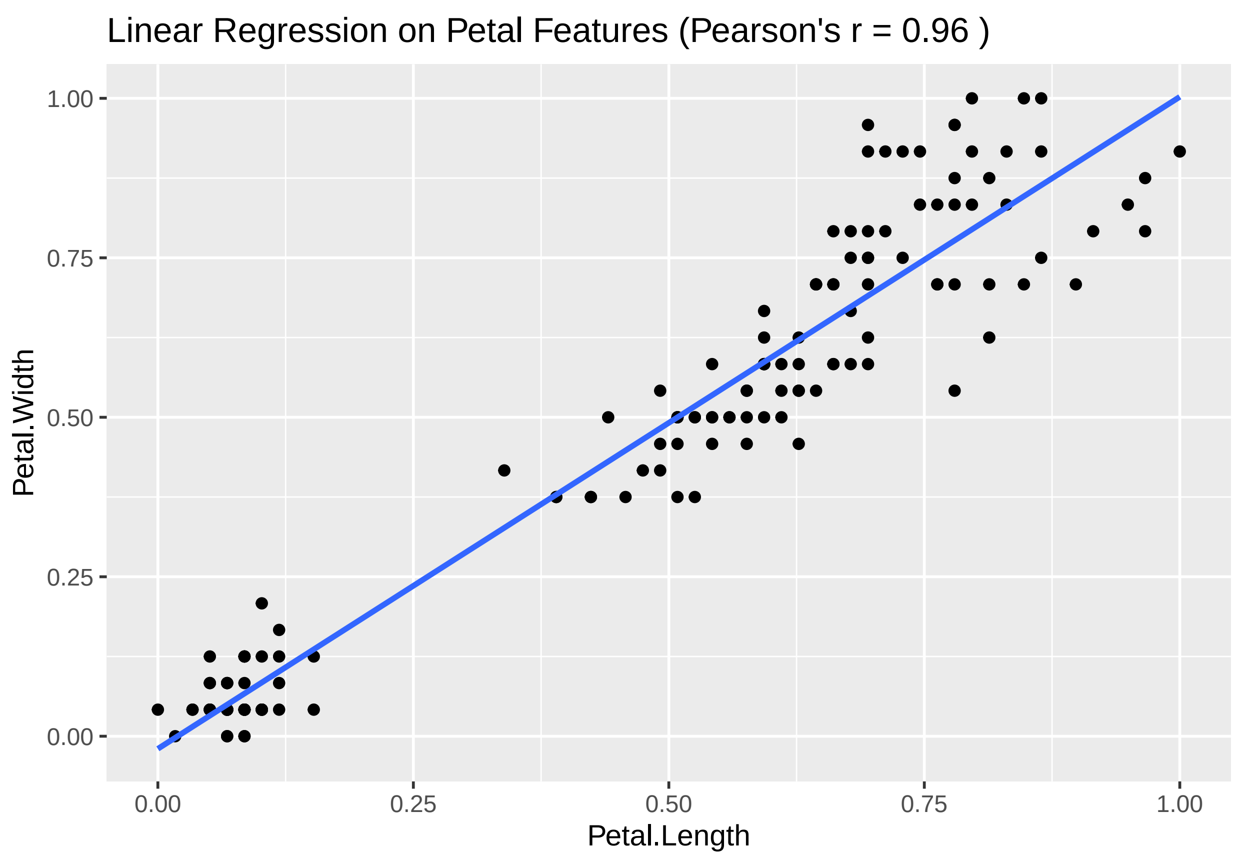
Using Pearson's Correlation Coefficient (r), I'm going to see if there's any relationship between petal length and petal width (there obviously is, we can draw a line right through it haha).

```r
petal_corr <- cor(iris_normalized[, 3], iris_normalized[, 4])
cat("The Pearson Correlation Coefficient calculates", petal_corr, 'for our petal features')
```

```
## The Pearson Correlation Coefficient calculates 0.9628654 for our petal features
```

As r approaches 1, the linear relationship between two variables strengthen such that as one variable increases, the other tends to increase as well. A coefficient of 0.9628654 suggests very high correlation. Since it suggests high correlation (we can also visually see it), let's use linear regression to fit a line through our petal features.

```r
ggplot(iris_normalized[,3:4], aes(Petal.Length, Petal.Width)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE) +
  labs(title = paste("Linear Regression on Petal Features (Pearson's r =",
                     round(petal_corr, 2), ")"))
```



Pretty neat! Where correlation quantifies the strength of the linear relationship between petal width and length, regression expresses this relationship in the form of a linear equation. From our results, it's pretty clear that petal length and width are our best predictors.

K Means Clustering We're almost done (to my relief and yours as well). We just need to determine the best k (we technically found it above in the elbow diagram, but I would prefer not to eyeball it), and see how well it predicts the species type across 1 through 7 clusters.

```r
## let's use a for loop to get our best k
clusters <- 1:7
# tot_dist <- data.frame(K = integer(), clusters = )
clustering_results <- data.frame(Cluster=integer(0),
                                 Species=integer(0),
                                 Clusters_Used = character(0))
for(k in clusters){
        kmeans_result <- kmeans(iris_normalized[, 3:4],
                        centers=k,
                        nstart=25)
        cluster_size <- table(kmeans_result$cluster)
        result_df <- data.frame(Clusters_Used = NA,
                                Species = as.integer(cluster_size)
                                )
        result_df$Clusters_Used[1] <- paste('K =', k)
        clustering_results <- rbind(clustering_results, result_df)
}
clustering_results$Clusters_Used <- ifelse(is.na(clustering_results$Clusters_Used), "",
                                            clustering_results$Clusters_Used)
kable(clustering_results,
      format='markdown',
      col.names=c('Clusters','Species Predicted'),
      align='c',
      caption='Clustering Results'
)
```

Table 1: Clustering Results

Clusters	Species Predicted
K = 1	150
K = 2	100
	50
K = 3	52
	50
	48
K = 4	42
	27
	50
	31
K = 5	29
	23
	50
	25
	23
K = 6	28
	50
	23
	8
	20
	21
K = 7	8
	20
	50
	22
	20
	20
	10

At K = 3, we get 3 clusters of sizes 50, 52, 48, which is pretty close to our original response vector.

```
best_cluster <- 3
kmeans_result <- function(k, data_set){
        kmeans(data_set[, 3:4],
               centers=k,
               nstart=25)
}
comparison_clusters <- table(kmeans_result(k=best_cluster,
                                            data_set=iris_normalized)$cluster,
                             iris$Species)
kable(comparison_clusters,
      format='markdown',
      row.names = TRUE,
      align='c',
      caption='Contingency Table Between Predicted Clusters and Actual Species'
)
```

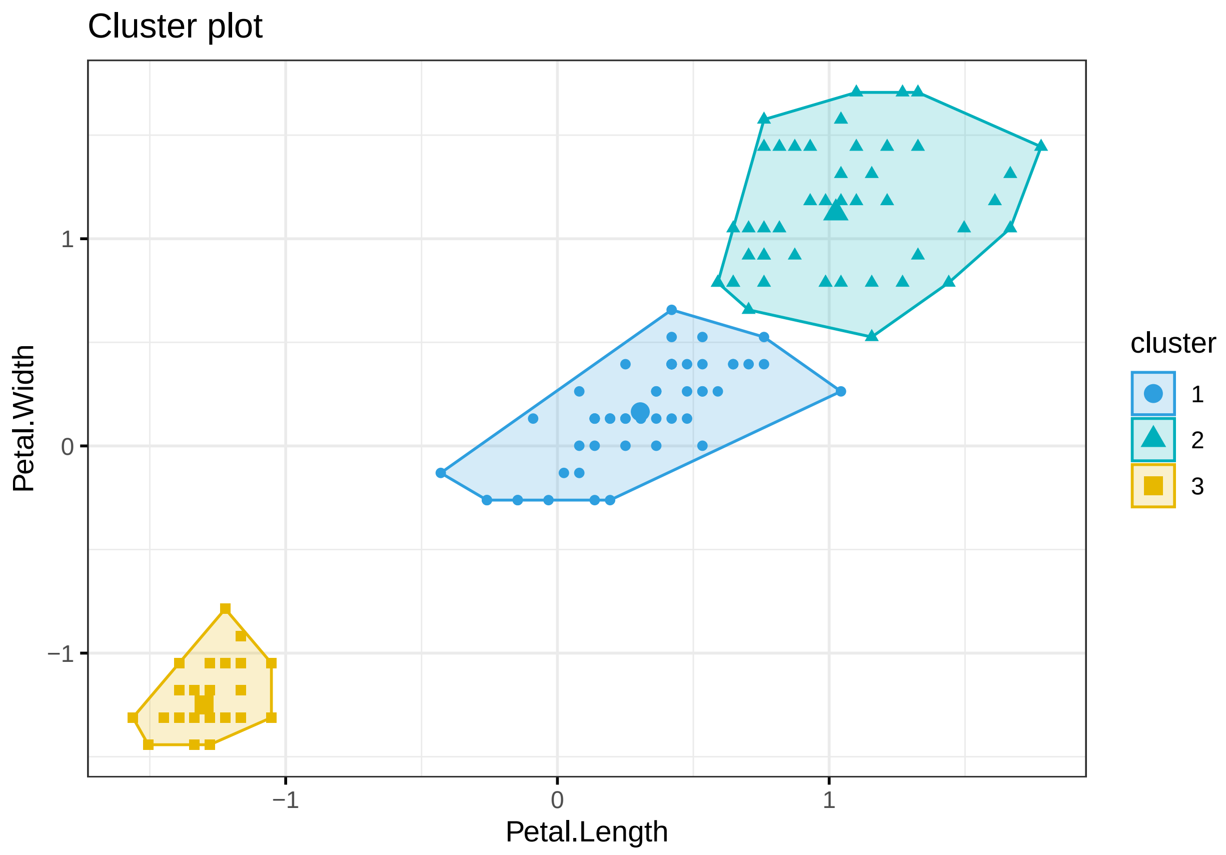
Table 2: Contingency Table Between Predicted Clusters and Actual Species

	setosa	versicolor	virginica
1	0	2	46
2	0	48	4
3	50	0	0

In our contingency table, we got setosa labeled correctly, but missed a few for versicolor and virginica. We predicted 52 for label 1, 48 for label 2, and 50 for label 3. All in all, our clustering algorithm performed pretty well.

Here's how it plots using the factoextra package.

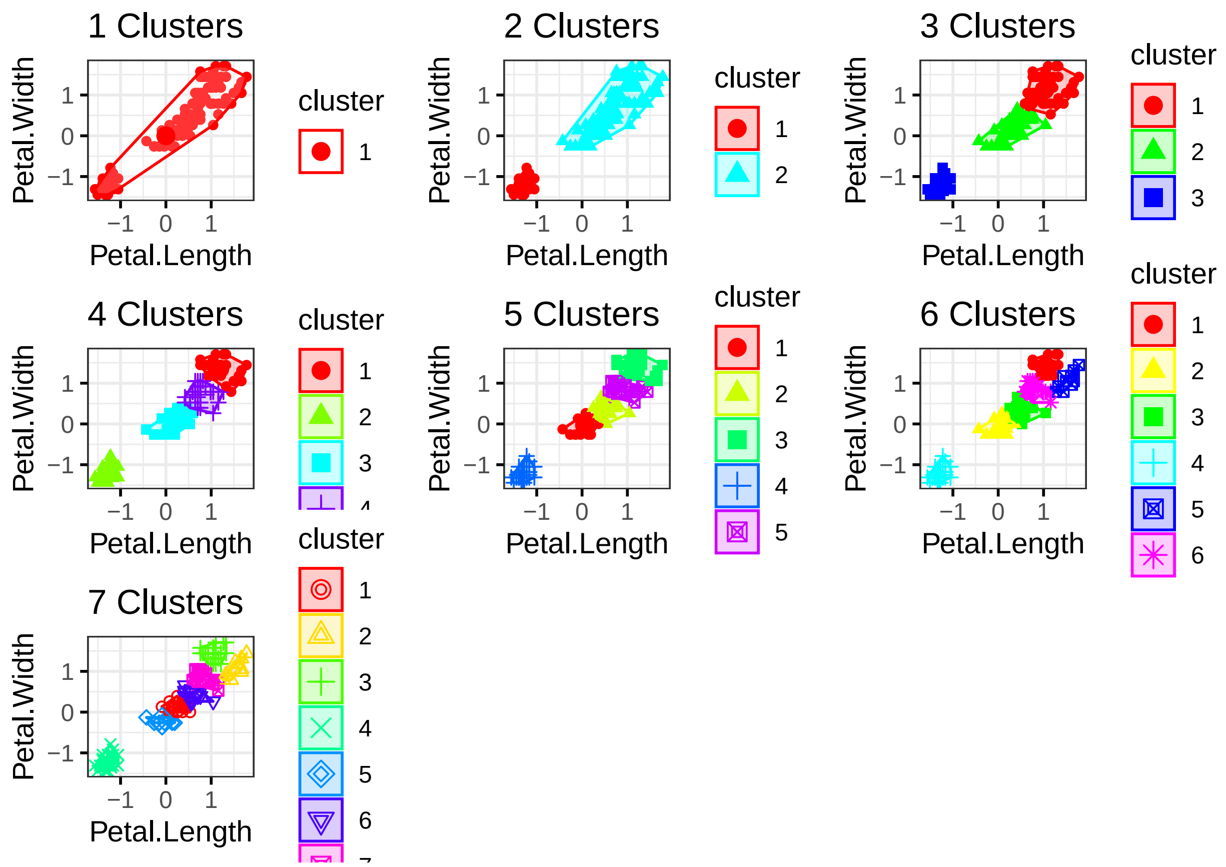
```
fviz_cluster(kmeans_result(k=best_cluster,
                           data_set=iris_normalized),
             data = iris_normalized[, 3:4],
             palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)
```



The correct clustering of cluster 3 makes sense because it had the most separation from the other two species as we saw in our first scatter plot. Understandably, our algorithm mixed up a few points in other two clusters because they were so close together. But this is how our clustering algorithm grouped our "unsupervised" data.

Let's see how fviz_cluster clusters our data when we augment our cluster amount.

```r
fviz_graphs <- function(x, index){
        num_colors <- index + 1
        palette <- hsv(seq(0, 1, length.out = num_colors), 1, 1)
        fviz_cluster(x,
                     data = iris_normalized[, 3:4],
                     palette = palette,
                     geom = "point",
                     ellipse.type = "convex",
                     ggtheme = theme_bw()) +
        ggtitle(paste(index ,'Clusters'))
}
cluster_results <- list()
cluster_graphs <- list()
for(i in clusters){
        cluster_results[[i]] <- kmeans_result(k=i,
                                              data_set=iris_normalized)
        cluster_graphs[[i]] <- fviz_graphs(cluster_results[[i]], i)
}
do.call(gridExtra::grid.arrange, cluster_graphs)
```



Conclusion

Though we were fortunate enough to be given actual values of the data set, in reality we will not have this luxury. If we didn't know the number of categorical labels, 2, 3, 4, and 5-clusters could have easily passed as a valid clustering model. As such, we must be careful about how the results of a clustering analysis are reported. That is, the results should not be taken as the absolute truth about a data set (as seen above). Performing unsupervised learning should cause us to take a step back and see our results as a starting point for the development of a scientific hypothesis and further study, preferably done on an independent data set.