Question 4.2

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 data is available from the R library datasets and can be accessed with iris once the library is loaded. It is also available at the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Iris). *The response values are only given to see how well a specific method performed and should not be used to build the model.*

Use the R function `kmeans` to cluster the points as well as possible. Report the best combination of predictors, your suggested value of k, and how well your best clustering predicts flower type.

(1) Best k

```
K-means clustering with 3 clusters of sizes 39, 50, 61

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1    0.7072650   0.4508547   0.79704476  0.82478632
2    0.1961111   0.5950000   0.07830508  0.06083333
3    0.4412568   0.3073770   0.57571548  0.54918033

Clustering vector:
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 
  2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2 
 41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80 
  2   2   2   2   2   2   2   2   2   2   1   3   1   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   1   3   3 
 81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 
  3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   1   3   1   1   1   1   3   1   1   1   1   1   1   3   1   1   1   1   1   3 
121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 
  1   3   1   3   1   1   3   3   1   1   1   1   1   3   3   1   1   1   3   1   1   1   3   1   1   1   3   1   1   3 

Within cluster sum of squares by cluster:
[1] 2.073324 1.829062 3.079830
 (between_SS / total_SS =  83.0 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
```

After the train the model, k =2 is the best cluster possible. Because the (between_SS / total_SS = 70.5 %).

Besides, I also train k = 3, but with higher (between_SS / total_SS = 83 %).

```
K-means clustering with 2 clusters of sizes 50, 100

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1    0.1961111   0.5950000   0.07830508  0.06083333
2    0.5450000   0.3633333   0.66203390  0.65666667

Clustering vector:
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 
  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   1   1 
 41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80 
  1   1   1   1   1   1   1   1   1   1   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2 
 81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 
  2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2 
121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 
  2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2 

Within cluster sum of squares by cluster:
[1]  1.829062 10.298729
 (between_SS / total_SS =  70.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
```

(2) Accuracy.

When k = 3, the accuracy rate is 133/150 = 88.67%.

When k = 2, the accuracy rate is 150/150= 100%.

```
             1  2  3
  setosa     0 50  0
  versicolor  3  0 47
  virginica  36  0 14
> # plot
> autoplot(kmeans_cluster3, df)
> #
> #
> autoplot(kmeans_cluster2, df)
> table(df$Species, kmeans_cluster2$cluster)

             1  2
  setosa    50  0
  versicolor  0 50
  virginica   0 50
>
```

Reasoning & detailed code:

```
# read data from local path
df <- read.csv("D:/GEORGIA INSTITUTE OF TECHNOLOGY/ISYE_6501/WEEK2/hw2-SP22/iris.txt", sep="")
#check the data head
head(df)
#
# separate the response varible from the dataset
df_split <- df[,-5]
head(df_split)
#
```

Important thing here is to separate response value from the dataset to prevent using response value build the model.

```
# normalized the data
df_preprocessed <- preProcess(df_split, method=c("range"))
df_normalized <- predict(df_preprocessed, df_split)
head(df_normalized)
```

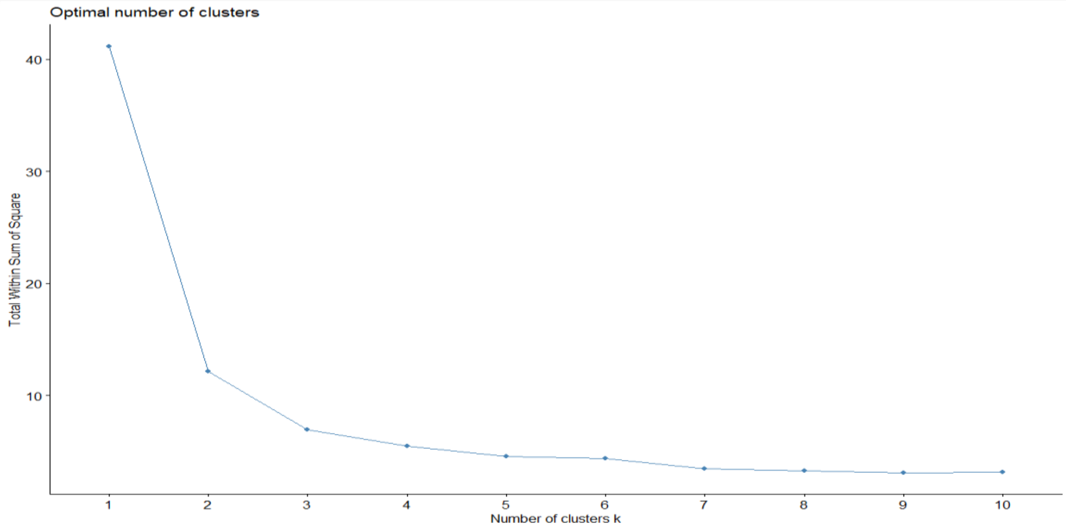
Secondly, scaled or normalized the data. Normalization is the way I used here because it is a more radical transformation, can change the observation so that observations can be described as a normal distribution, which is one of underlying assumption of machine learning. However, scaled is better way under some scenario, vice versa. But here I chose to normalize the predicators.

Thirdly, it is crucial to figure out the k clusters before build up a model. Hence, it is reasonable to use the optimal number of clusters. I used the combination of Elbow method and silhouette method to find the best K. Occasionally, Elbow method is enough, but the combination of two methods can provide much more confidence.

```
# to reproduce the results you have to use set.seed()
set.seed(9876)
# set up elbow method
fviz_nbclust(df_normalized, kmeans, method = "wss")

# use silhouette_score
set.seed(9876)
fviz_nbclust(df_normalized, kmeans, method='silhouette')
```

I set the seed before I apply two methods in order to make my results more reproducible.

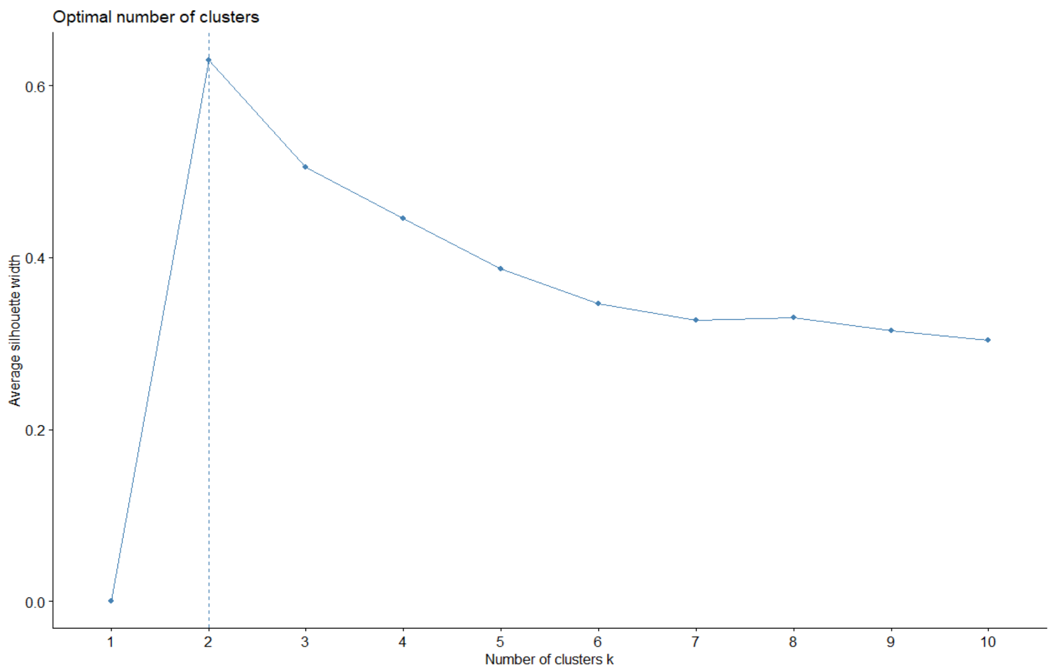
ELBOW:



The idea of this method is that it plots the various values of cost with changing k. As the value of K increases, there will be fewer elements in the cluster. So average distortion will decrease. The lesser number of elements means closer to the centroid. So, the point where this distortion declines the most is the elbow point. From this plot, 2 or 3 are the possible k we can chose.

SILHOUETTE:

This approach measures the quality of a clustering. It determines how well each object lies within its cluster. A high average silhouette width indicates a good clustering.



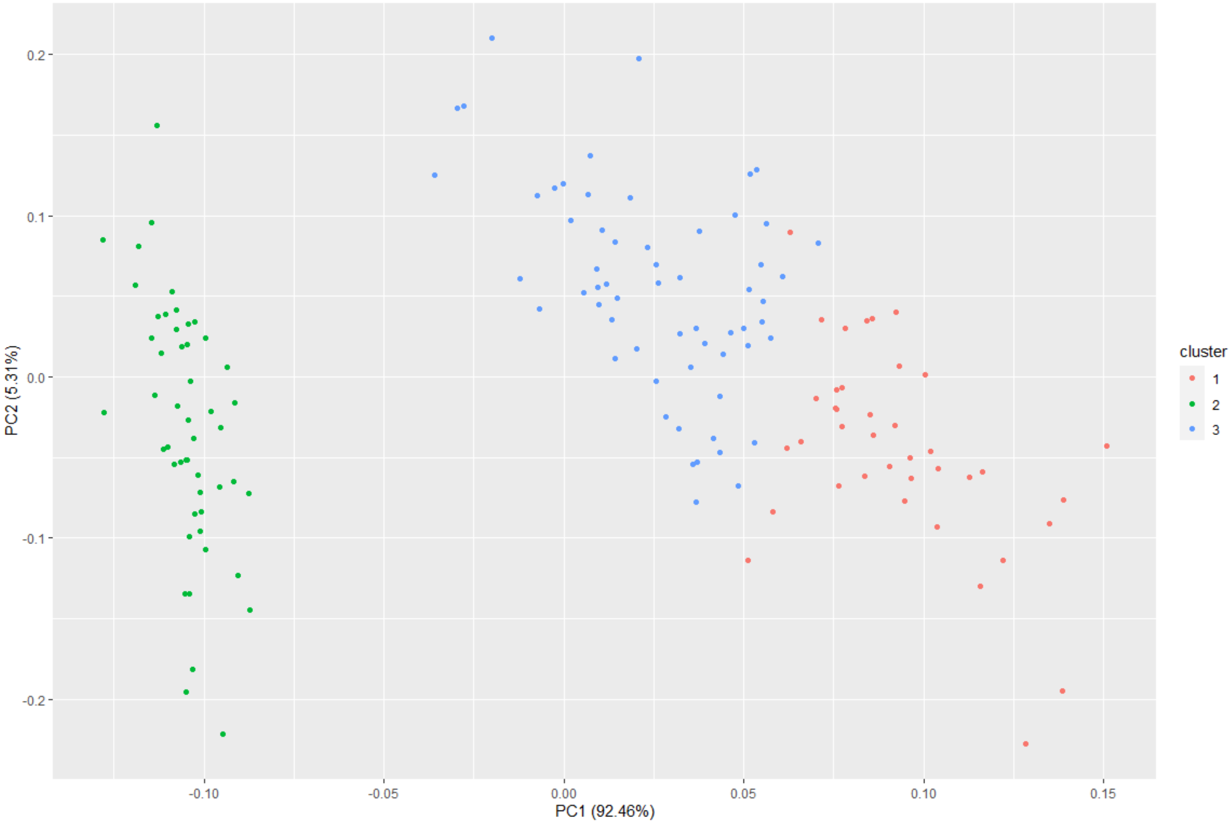
It can be observed that both k=2 and k=3 have high average silhouette score, but k=2 have highest score. So these two can be the our potential k for building up a model. Hence, I used k=2, k=3 buildup model separately.

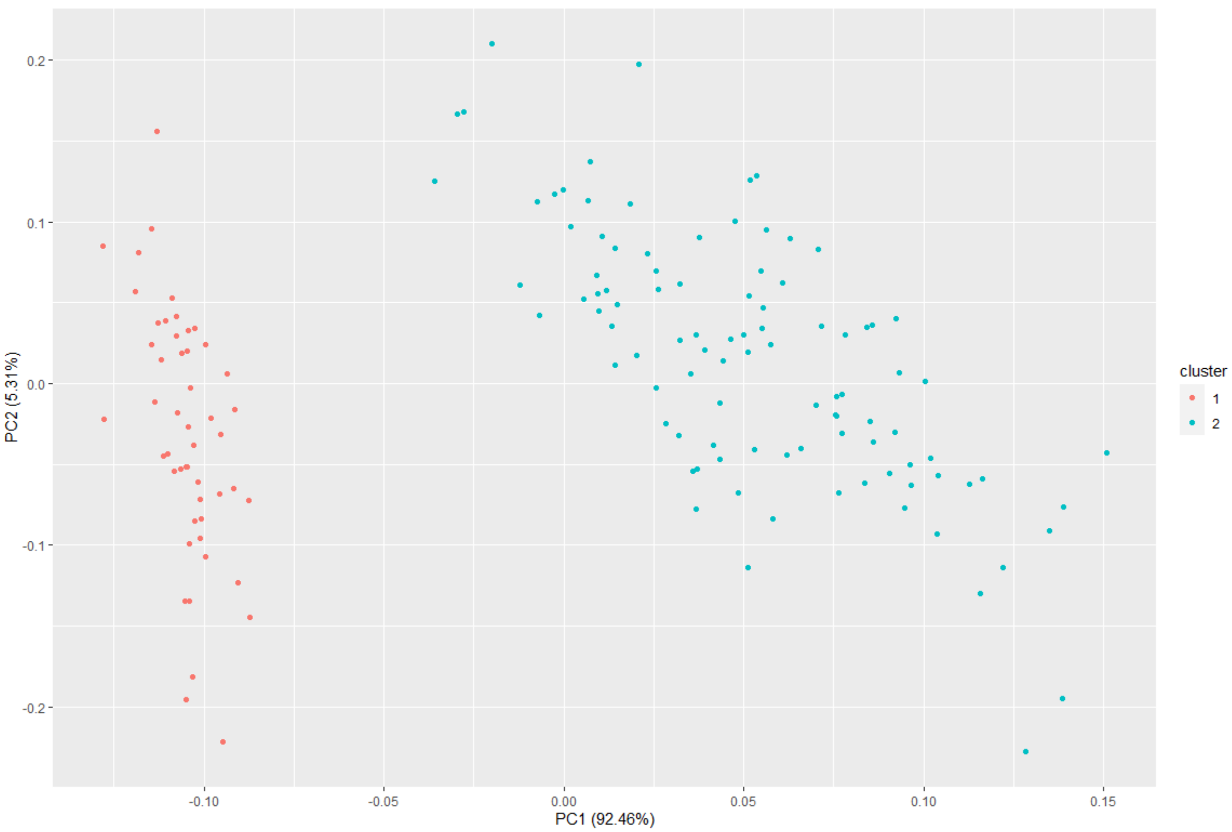
```
K-means clustering with 2 clusters of sizes 50, 100

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1    0.1961111   0.5950000   0.07830508  0.06083333
2    0.5450000   0.3633333   0.66203390  0.65666667

Clustering vector:
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 
  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   1   1 
 41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80 
  1   1   1   1   1   1   1   1   1   1   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2 
 81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 
  2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2 
121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 
  2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2 

Within cluster sum of squares by cluster:
[1]  1.829062 10.298729
 (between_SS / total_SS =  70.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
K-means clustering with 3 clusters of sizes 39, 50, 61

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1    0.7072650   0.4508547   0.79704476  0.82478632
2    0.1961111   0.5950000   0.07830508  0.06083333
3    0.4412568   0.3073770   0.57571548  0.54918033

Clustering vector:
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 
  2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2 
 41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80 
  2   2   2   2   2   2   2   2   2   2   1   3   1   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   1   3   3 
 81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 
  3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   1   3   1   1   1   1   3   1   1   1   1   1   1   3   1   1   1   1   1   3 
121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 
  1   3   1   3   1   1   3   3   1   1   1   1   1   3   3   1   1   1   3   1   1   1   3   1   1   1   3   1   1   3 

Within cluster sum of squares by cluster:
[1] 2.073324 1.829062 3.079830
 (between_SS / total_SS =  83.0 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
```

Take K=3 as an example, as you can observed that we build up clusters with 3 clusters of sizes 39, 50, 61. Each cluster means listed below, and cluster vector. An important indicator is Within cluster sum of squares by cluster, which is a measure of the variability of the observations within each cluster. Generally, a cluster that has a small sum of squares is more compact than a cluster that has a large sum of squares. Clusters that have higher values exhibit greater variability of the observations within the cluster.

In conclusion, K =2 is the best k I chose after validate by combining elbow method, silhouette method and accuracy test. Visualizations have been provided below.

(notice that cluster = 3 there are misclassified points above)


PC2 (5.31%)
PC1 (92.46%)
cluster
1
2
3


PC2 (5.31%)
PC1 (92.46%)
cluster
1
2