K - Means Clustering

K-means clustering is one of the simplest and popular **UNSUPERVISED** machine learning algorithms.

Introduction

****

- **K - Means Clustering make inferences from data using only input vectors without labelled outcomes.**

- **K - means clustering makes partition of "n" observations into "k" clusters in which each observation belongs to the cluster with the nearest centroid**

- **A cluster refers to a collection of data points aggregated together because of certain similarities**
  **
**

Applications Of K - Means Clustering

- **Academic performance - Based on the scores, students are categorized into grades like A, B, or C.**

- **Search engines - Clustering forms a backbone of search engines. When a search is performed, the search results need to be grouped, and the search engines use clustering to do this.**

- **Customer segmentation - It helps marketers improve their customer base, work on target areas, and segment customers based on purchase history, interests, or activity monitoring.**
  ****

Example For clustering

****

```
A bank wants to give credit card offers to its customers. Currently, they
look at the details of each customer and based on this information they
decide which offer should be given to which customer.
```

The bank will have millions of customers. So here clustering plays a vital role in segregating customers into different groups. For instance, the bank can group the customers based on their income:

Steps to perform K - Means Clustering

1. **Specify the number of clusters (K) to be created**

2. **Select randomly (K) objects from the dataset as the initial cluster centroids**

3. **Assign each observation to their closest centroid, based on the Euclidean distance between the object and the centroid**

4. **For each of the k clusters update the cluster centroid by calculating the new mean values of all the data points in the cluster.**

5. **Iteratively minimize the total within sum of square. That is, iterating steps 3 and 4 until the cluster assignments stop changing**

Importing Packages

In [39]:
```
library(factoextra)
library(ggplot)
library(cluster)
```

Loading the Dataset

USArrests - This data set contains statistics, in arrests per 100,000 residents for assault, murder, and rape in each of the 50 US states in 1973. Also given is the percent of the population living in urban areas.

In [26]:
```
#Inbuilt Dataset
df = USArrests
df
```

	Murder	Assault	UrbanPop	Rape

	Murder	Assault	UrbanPop	Rape
Alabama	13.2	236	58	21.2
Alaska	10.0	263	48	44.5
Arizona	8.1	294	80	31.0
Arkansas	8.8	190	50	19.5
California	9.0	276	91	40.6
Colorado	7.9	204	78	38.7
Connecticut	3.3	110	77	11.1
Delaware	5.9	238	72	15.8
Florida	15.4	335	80	31.9
Georgia	17.4	211	60	25.8
Hawaii	5.3	46	83	20.2
Idaho	2.6	120	54	14.2
Illinois	10.4	249	83	24.0
Indiana	7.2	113	65	21.0
Iowa	2.2	56	57	11.3
Kansas	6.0	115	66	18.0
Kentucky	9.7	109	52	16.3
Louisiana	15.4	249	66	22.2
Maine	2.1	83	51	7.8
Maryland	11.3	300	67	27.8
Massachusetts	4.4	149	85	16.3
Michigan	12.1	255	74	35.1
Minnesota	2.7	72	66	14.9
Mississippi	16.1	259	44	17.1
Missouri	9.0	178	70	28.2
Montana	6.0	109	53	16.4
Nebraska	4.3	102	62	16.5
Nevada	12.2	252	81	46.0
New Hampshire	2.1	57	56	9.5
New Jersey	7.4	159	89	18.8
New Mexico	11.4	285	70	32.1
New York	11.1	254	86	26.1
North Carolina	13.0	337	45	16.1
North Dakota	0.8	45	44	7.3
Ohio	7.3	120	75	21.4
Oklahoma	6.6	151	68	20.0

	Murder	Assault	UrbanPop	Rape
Oregon	4.9	159	67	29.3
Pennsylvania	6.3	106	72	14.9
Rhode Island	3.4	174	87	8.3
South Carolina	14.4	279	48	22.5
South Dakota	3.8	86	45	12.8
Tennessee	13.2	188	59	26.9
Texas	12.7	201	80	25.5
Utah	3.2	120	80	22.9
Vermont	2.2	48	32	11.2
Virginia	8.5	156	63	20.7
Washington	4.0	145	73	26.2
West Virginia	5.7	81	39	9.3
Wisconsin	2.6	53	66	10.8
Wyoming	6.8	161	60	15.6

In [27]:
```
summary(df)
```

```
     Murder          Assault         UrbanPop          Rape      
 Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   : 7.30  
 1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st Qu.:15.07  
 Median : 7.250   Median :159.0   Median :66.00   Median :20.10  
 Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean   :21.23  
 3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd Qu.:26.18  
 Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.   :46.00  
```

Data Preprocessing

Scaling data - To avoid biased results

In [29]:
```
df <- scale(USArrests) # Scaling the data
head(df, n = 15)
```

	Murder	Assault	UrbanPop	Rape
Alabama	1.24256408	0.7828393	-0.52090661	-0.003416473
Alaska	0.50786248	1.1068225	-1.21176419	2.484202941
Arizona	0.07163341	1.4788032	0.99898006	1.042878388
Arkansas	0.23234938	0.2308680	-1.07359268	-0.184916602
California	0.27826823	1.2628144	1.75892340	2.067820292
Colorado	0.02571456	0.3988593	0.86080854	1.864967207
Connecticut	-1.03041900	-0.7290821	0.79172279	-1.081740768
Delaware	-0.43347395	0.8068381	0.44629400	-0.579946294
Florida	1.74767144	1.9707777	0.99898006	1.138966691
Georgia	2.20685994	0.4828549	-0.38273510	0.487701523

	Murder	Assault	UrbanPop	Rape
Hawaii	-0.57123050	-1.4970423	1.20623733	-0.110181255
Idaho	-1.19113497	-0.6090884	-0.79724965	-0.750769945
Illinois	0.59970018	0.9388312	1.20623733	0.295524916
Indiana	-0.13500142	-0.6930840	-0.03730631	-0.024769429
Iowa	-1.28297267	-1.3770485	-0.58999237	-1.060387812

In [30]:
```
summary(df)
```

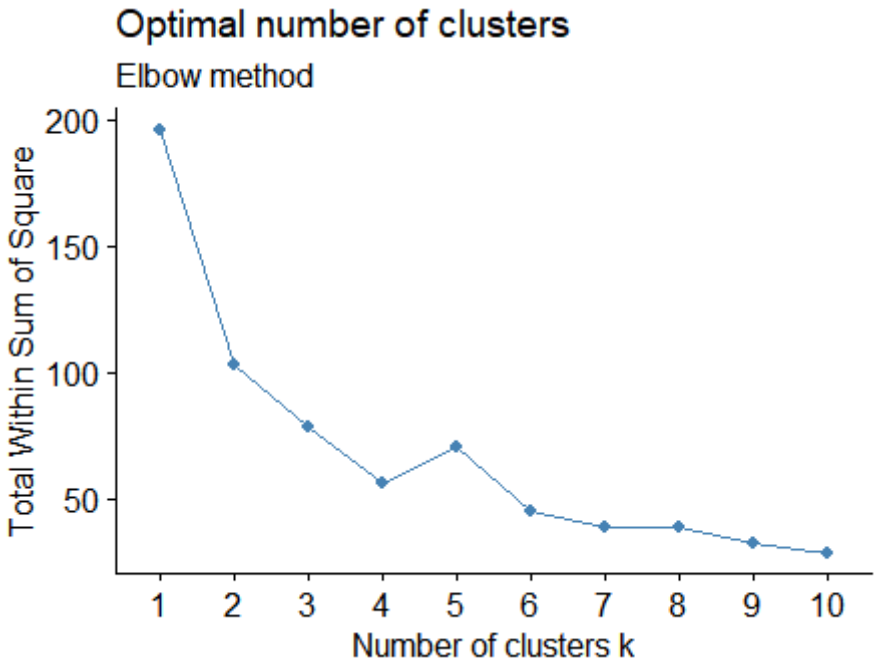
```
     Murder           Assault          UrbanPop           Rape       
Min.   :-1.6044   Min.   :-1.5090   Min.   :-2.31714   Min.   :-1.4874  
1st Qu.:-0.8525   1st Qu.:-0.7411   1st Qu.:-0.76271   1st Qu.:-0.6574  
Median :-0.1235   Median :-0.1411   Median : 0.03178   Median :-0.1209  
Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000  
3rd Qu.: 0.7949   3rd Qu.: 0.9388   3rd Qu.: 0.84354   3rd Qu.: 0.5277  
Max.   : 2.2069   Max.   : 1.9948   Max.   : 1.75892   Max.   : 2.6444  
```

STEP 1 : FINDING NUMBER OF K VALUES

To find the appropriate suitable K value:

1. **Compute k-means clustering using different values of clusters k.**
2. **Next, the wss (within sum of square) is drawn according to the number of clusters.**
3. **The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.**

After Performing Elbow Method



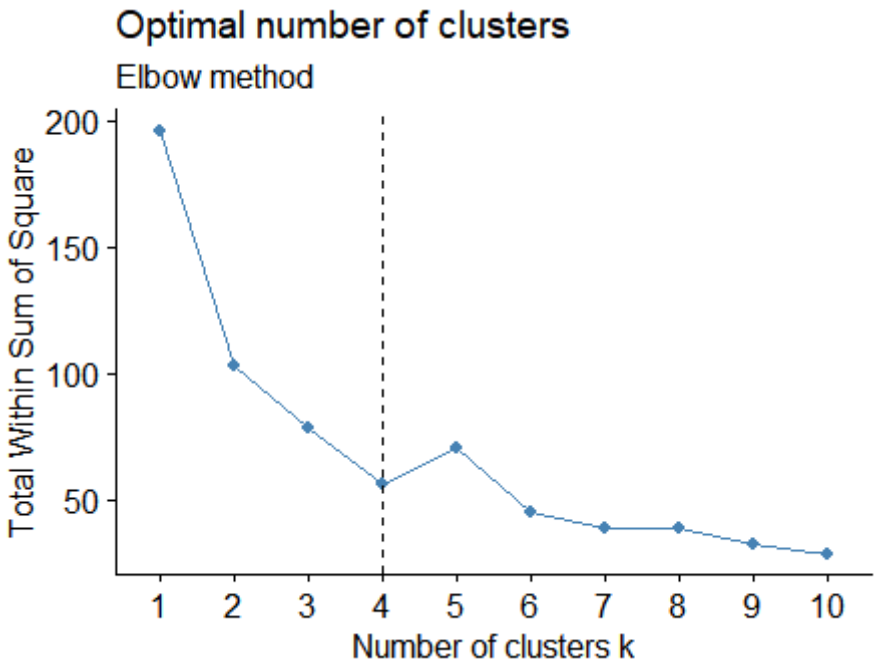
The Elbow method looks at the total within-cluster sum of square (WSS) as a

function with respect to the number of clusters.

The location of a knee in the plot is usually considered as an indicator of the appropriate number of clusters because it means that adding another cluster does not improve much better the partition

So Here k = 4 is a best fit and it points out position of elbow

code : fviz_nbclust(df, kmeans, method = "wss") + geom_vline(xintercept = 4, linetype = 3)+ labs(subtitle = "Elbow method")



Compute k-means with k = 4

In [31]:
```
set.seed(27)
kmcluster <- kmeans(df, 4, nstart = 25)
```

set.seed() function in order to set a key for random number generator.As k-means clustering algorithm starts with k randomly selected centroids
Syntax <- kmeans("dataframe_name",Number of clusters,nstart = no. of different random starting choices)

In [36]:
```
kmcluster
```

```
K-means clustering with 4 clusters of sizes 16, 13, 8, 13

Cluster means:
      Murder    Assault   UrbanPop        Rape
1 -0.4894375 -0.3826001  0.5758298 -0.26165379
2  0.6950701  1.0394414  0.7226370  1.27693964
3  1.4118898  0.8743346 -0.8145211  0.01927104
4 -0.9615407 -1.1066010 -0.9301069 -0.96676331

Clustering vector:
       Alabama         Alaska        Arizona       Arkansas     California
```

```
         3          2          2          3          2 
  Colorado Connecticut   Delaware    Florida    Georgia 
         2          1          1          2          3 
    Hawaii      Idaho   Illinois    Indiana       Iowa 
         1          4          2          1          4 
    Kansas   Kentucky  Louisiana      Maine   Maryland 
         1          4          3          4          2 
Massachusetts   Michigan  Minnesota Mississippi   Missouri 
         1          2          4          3          2 
   Montana   Nebraska     Nevada New Hampshire New Jersey 
         4          4          2          4          1 
New Mexico   New York North Carolina North Dakota       Ohio 
         2          2          3          4          1 
  Oklahoma     Oregon Pennsylvania Rhode Island South Carolina 
         1          1          1          1          3 
South Dakota  Tennessee      Texas       Utah    Vermont 
         4          3          2          1          4 
  Virginia Washington West Virginia  Wisconsin    Wyoming 
         1          1          4          4          1 

Within cluster sum of squares by cluster:
[1] 16.212213 19.922437  8.316061 11.952463
 (between_SS / total_SS =  71.2 %)

Available components:

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

In [44]:
```
table(kmcluster$tot.withinss)
```

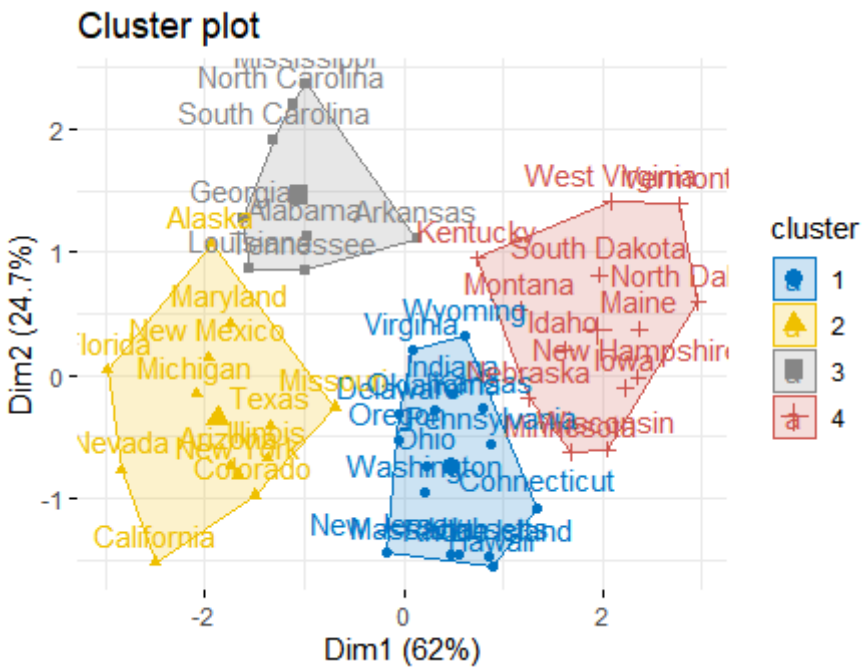
```
56.4031734582928
               1 
```

In [43]:
```
table(kmcluster$cluster)
```

```
 1  2  3  4 
16 13  8 13 
```

Plotting & Visulaization

Code :fviz_cluster(kmcluster, data = df)

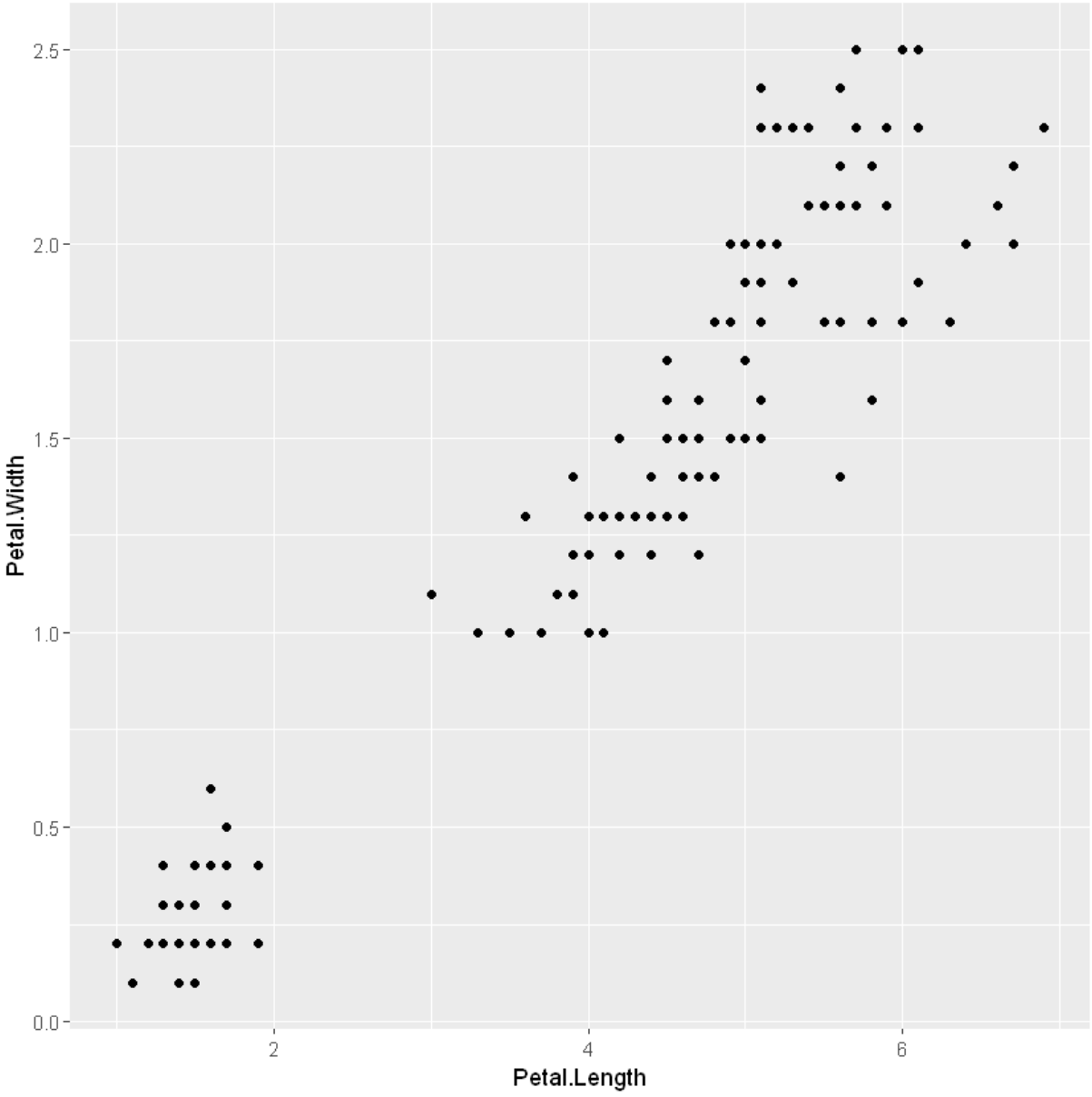


Iris Dataset

In [8]:
```
head(iris)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

In [9]:
```
ggplot(iris, aes(Petal.Length, Petal.Width)) + geom_point()
```



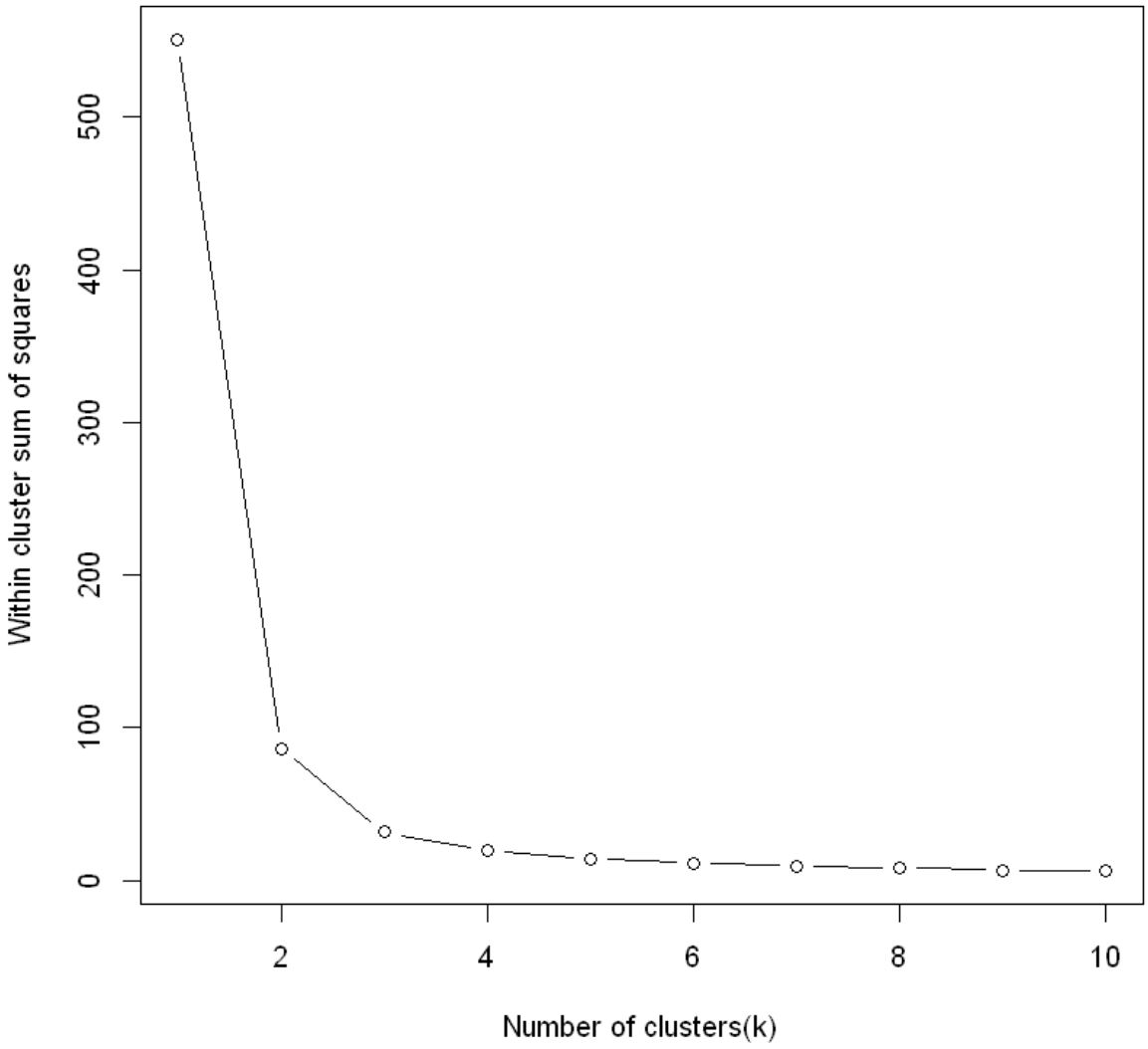
To Find optimal number of clusters

Elbow Method

In [49]:
```
set.seed(27)
k.max <- 10

#nstart       - No. of  random sets should be chosen?
#Total within - cluster sum of squares, i.e. sum(withinss).
#sapply       - takes list, vector or data frame as input and gives output in vector
#iter. max is the number of times the algorithm will repeat the cluster assignment a
#nstart is the number of times the initial starting points are re-sampled.

wss<- sapply(1:k.max,function(k){kmeans(iris[,3:4],k,nstart = 20,iter.max = 20)$tot.
plot(1:k.max,wss, type= "b", xlab = "Number of clusters(k)",ylab = "Within cluster s
```



Here we can see best fit at k = 3 and it points out position of elbow

In [51]:
```
set.seed(20)
irisCluster <- kmeans(iris[, 1:4], 3, nstart = 20)
irisCluster
```

```
K-means clustering with 3 clusters of sizes 50, 62, 38

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.006000    3.428000     1.462000    0.246000
2     5.901613    2.748387     4.393548    1.433871
3     6.850000    3.073684     5.742105    2.071053
```

```
Clustering vector:
  [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 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [75] 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3
[112] 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
[149] 3 2

Within cluster sum of squares by cluster:
[1] 15.15100 39.82097 23.87947
 (between_SS / total_SS =  88.4 %)

Available components:

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

In [11]:
```
irisCluster$cluster
```

1. 1
2. 1
3. 1
4. 1
5. 1
6. 1
7. 1
8. 1
9. 1
10. 1
11. 1
12. 1
13. 1
14. 1
15. 1
16. 1
17. 1
18. 1
19. 1
20. 1
21. 1
22. 1
23. 1
24. 1
25. 1
26. 1
27. 1
28. 1
29. 1
30. 1
31. 1
32. 1
33. 1
34. 1

35. 1
36. 1
37. 1
38. 1
39. 1
40. 1
41. 1
42. 1
43. 1
44. 1
45. 1
46. 1
47. 1
48. 1
49. 1
50. 1
51. 2
52. 2
53. 3
54. 2
55. 2
56. 2
57. 2
58. 2
59. 2
60. 2
61. 2
62. 2
63. 2
64. 2
65. 2
66. 2
67. 2
68. 2
69. 2
70. 2
71. 2
72. 2
73. 2
74. 2
75. 2
76. 2
77. 2
78. 3
79. 2
80. 2
81. 2

82. 2
83. 2
84. 2
85. 2
86. 2
87. 2
88. 2
89. 2
90. 2
91. 2
92. 2
93. 2
94. 2
95. 2
96. 2
97. 2
98. 2
99. 2
100. 2
101. 3
102. 2
103. 3
104. 3
105. 3
106. 3
107. 2
108. 3
109. 3
110. 3
111. 3
112. 3
113. 3
114. 2
115. 2
116. 3
117. 3
118. 3
119. 3
120. 2
121. 3
122. 2
123. 3
124. 2
125. 3
126. 3
127. 2
128. 2

129. 3
130. 3
131. 3
132. 3
133. 3
134. 2
135. 3
136. 3
137. 3
138. 3
139. 2
140. 3
141. 3
142. 3
143. 2
144. 3
145. 3
146. 3
147. 2
148. 3
149. 3
150. 2

In [12]:
```
irisCluster$tot.withinss
```

78.851441426146

In [13]:
```
iris$Species
```

1. setosa
2. setosa
3. setosa
4. setosa
5. setosa
6. setosa
7. setosa
8. setosa
9. setosa
10. setosa
11. setosa
12. setosa
13. setosa
14. setosa
15. setosa
16. setosa
17. setosa
18. setosa
19. setosa

20. setosa
21. setosa
22. setosa
23. setosa
24. setosa
25. setosa
26. setosa
27. setosa
28. setosa
29. setosa
30. setosa
31. setosa
32. setosa
33. setosa
34. setosa
35. setosa
36. setosa
37. setosa
38. setosa
39. setosa
40. setosa
41. setosa
42. setosa
43. setosa
44. setosa
45. setosa
46. setosa
47. setosa
48. setosa
49. setosa
50. setosa
51. versicolor
52. versicolor
53. versicolor
54. versicolor
55. versicolor
56. versicolor
57. versicolor
58. versicolor
59. versicolor
60. versicolor
61. versicolor
62. versicolor
63. versicolor
64. versicolor
65. versicolor
66. versicolor

67. versicolor
68. versicolor
69. versicolor
70. versicolor
71. versicolor
72. versicolor
73. versicolor
74. versicolor
75. versicolor
76. versicolor
77. versicolor
78. versicolor
79. versicolor
80. versicolor
81. versicolor
82. versicolor
83. versicolor
84. versicolor
85. versicolor
86. versicolor
87. versicolor
88. versicolor
89. versicolor
90. versicolor
91. versicolor
92. versicolor
93. versicolor
94. versicolor
95. versicolor
96. versicolor
97. versicolor
98. versicolor
99. versicolor
100. versicolor
101. virginica
102. virginica
103. virginica
104. virginica
105. virginica
106. virginica
107. virginica
108. virginica
109. virginica
110. virginica
111. virginica
112. virginica
113. virginica

114. virginica
115. virginica
116. virginica
117. virginica
118. virginica
119. virginica
120. virginica
121. virginica
122. virginica
123. virginica
124. virginica
125. virginica
126. virginica
127. virginica
128. virginica
129. virginica
130. virginica
131. virginica
132. virginica
133. virginica
134. virginica
135. virginica
136. virginica
137. virginica
138. virginica
139. virginica
140. virginica
141. virginica
142. virginica
143. virginica
144. virginica
145. virginica
146. virginica
147. virginica
148. virginica
149. virginica
150. virginica

▶ **Levels:**

In [14]:
```
table(irisCluster$cluster, iris$Species)
```

```
  setosa versicolor virginica
1     50          0         0
2      0         48        14
3      0          2        36
```

In [15]:
```
irisCluster$cluster <- as.factor(irisCluster$cluster)
```

In [16]:
```
irisCluster$cluster
```

1. 1
2. 1
3. 1
4. 1
5. 1
6. 1
7. 1
8. 1
9. 1
10. 1
11. 1
12. 1
13. 1
14. 1
15. 1
16. 1
17. 1
18. 1
19. 1
20. 1
21. 1
22. 1
23. 1
24. 1
25. 1
26. 1
27. 1
28. 1
29. 1
30. 1
31. 1
32. 1
33. 1
34. 1
35. 1
36. 1
37. 1
38. 1
39. 1
40. 1
41. 1
42. 1
43. 1
44. 1
45. 1

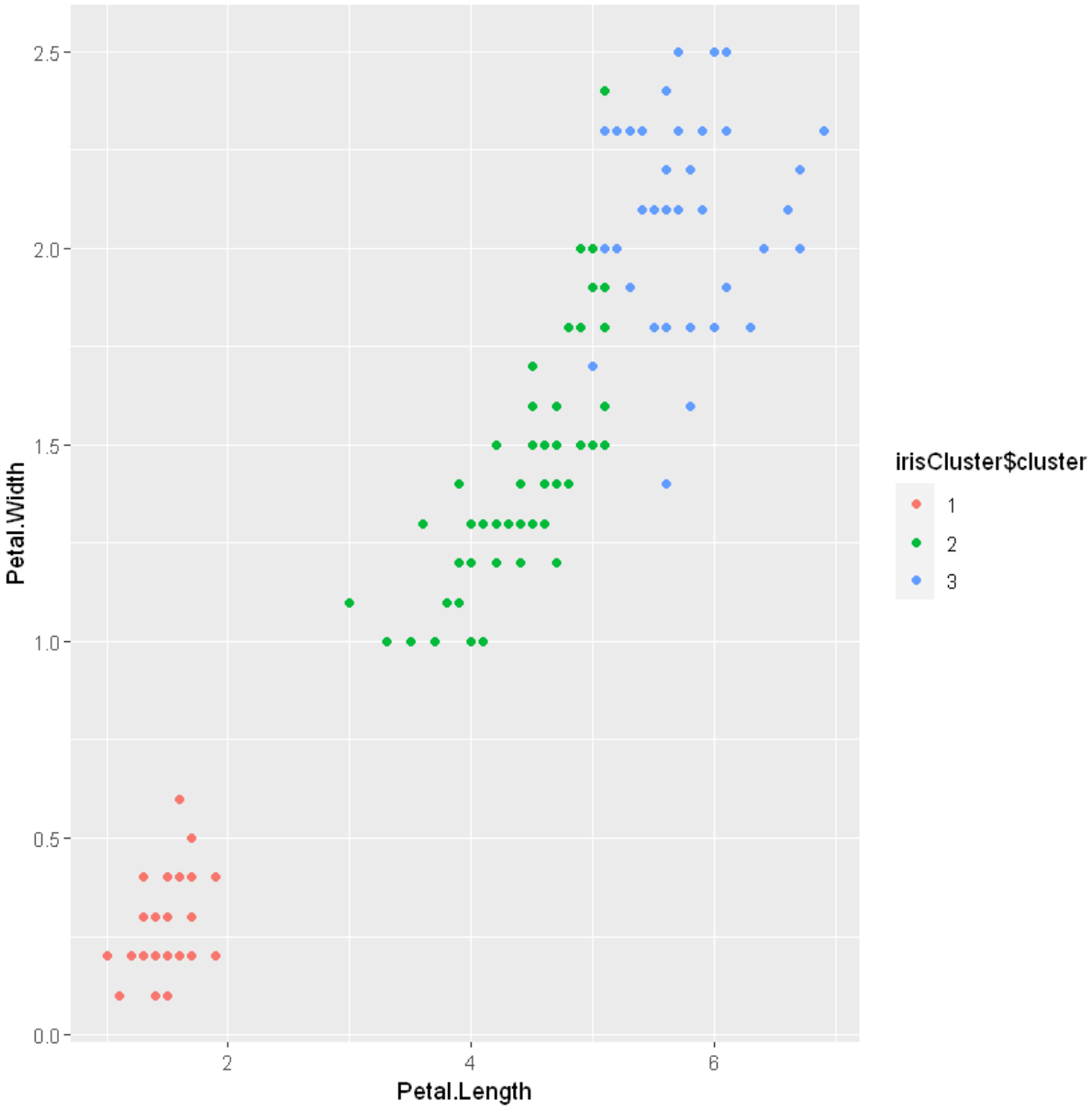
46. 1
47. 1
48. 1
49. 1
50. 1
51. 2
52. 2
53. 3
54. 2
55. 2
56. 2
57. 2
58. 2
59. 2
60. 2
61. 2
62. 2
63. 2
64. 2
65. 2
66. 2
67. 2
68. 2
69. 2
70. 2
71. 2
72. 2
73. 2
74. 2
75. 2
76. 2
77. 2
78. 3
79. 2
80. 2
81. 2
82. 2
83. 2
84. 2
85. 2
86. 2
87. 2
88. 2
89. 2
90. 2
91. 2
92. 2

93. 2
94. 2
95. 2
96. 2
97. 2
98. 2
99. 2
100. 2
101. 3
102. 2
103. 3
104. 3
105. 3
106. 3
107. 2
108. 3
109. 3
110. 3
111. 3
112. 3
113. 3
114. 2
115. 2
116. 3
117. 3
118. 3
119. 3
120. 2
121. 3
122. 2
123. 3
124. 2
125. 3
126. 3
127. 2
128. 2
129. 3
130. 3
131. 3
132. 3
133. 3
134. 2
135. 3
136. 3
137. 3
138. 3
139. 2

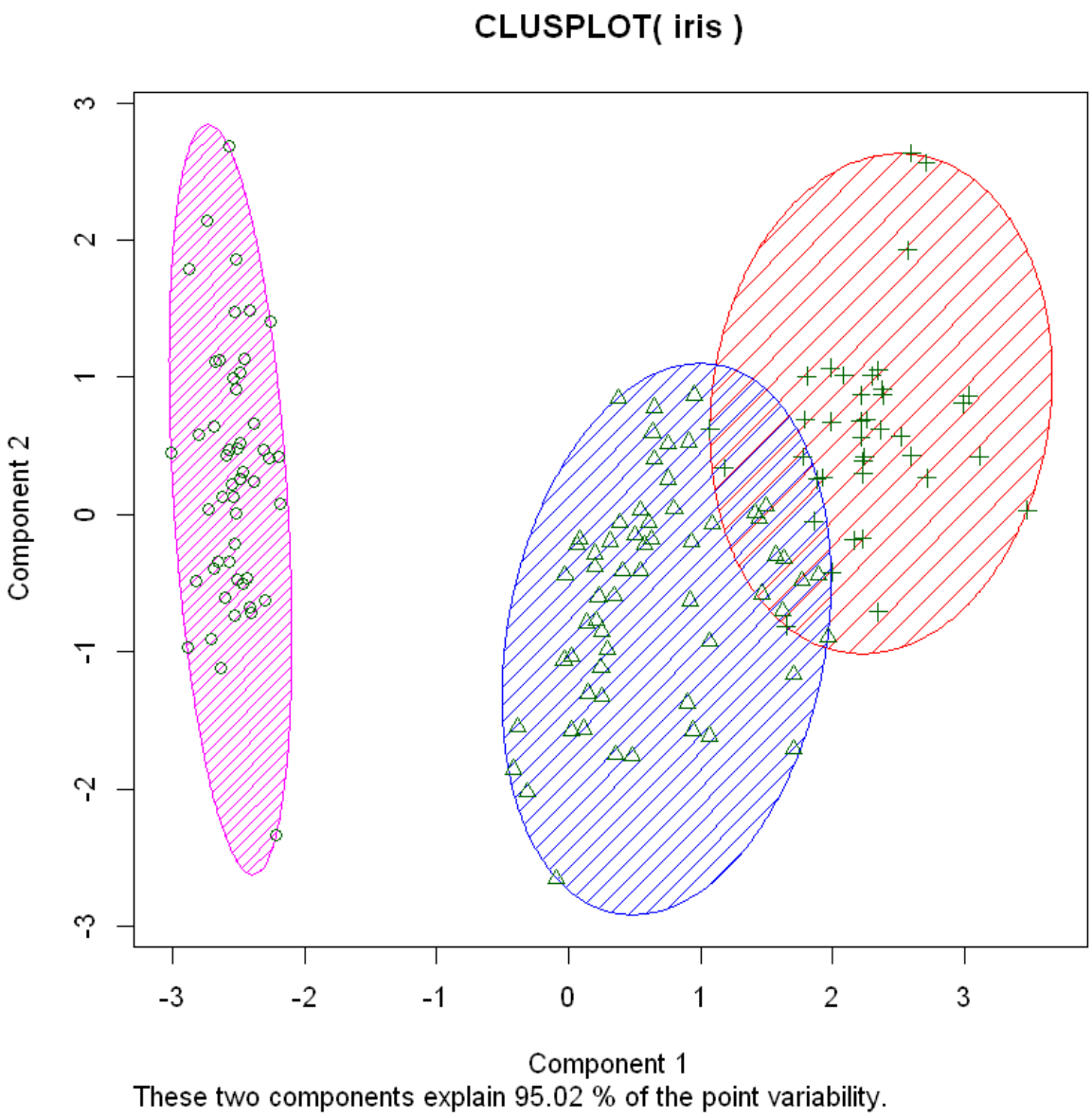
140. 3
141. 3
142. 3
143. 2
144. 3
145. 3
146. 3
147. 2
148. 3
149. 3
150. 2

▶ **Levels:**

In [17]:
```
ggplot(iris, aes(Petal.Length, Petal.Width, color = irisCluster$cluster)) + geom_poi
```



In [24]:
```
clusplot(iris, irisCluster$cluster, color=T, shade=T,lines=0)
```



Some Takeaways

- **Scale/standardize the data when applying kmeans algorithm.**
- **Elbow method for selecting number of clusters**
- **Kmeans gives more weight to the bigger clusters.**
- **Kmeans may still cluster the data even if it can't be clustered**
- **Different initial partitions can result in different final clusters.**
- **It can not handle noisy data and outliers.**
- **It is not suitable to identify clusters with non-convex shapes(irregular shapes - elliptical).**

****THANK YOU****

In [ ]:

In [ ]: