Exam 1.

Nisarg Negi

2022-10-14

This is an exam. You cannot discuss it with anybody. You should do the work yourself. Questions 1, 2 and 3 can be done with pen and paper. If you choose so, you can submit the scans along with other material (you also can embed them to your knitted pdf).

The submitted files must include pdf-s with your answers along with all R scripts. For example:

- Student A submitted:
  - Exam_1.pdf - final report containing all answers
  - Exam_1.Rmd - R-markdown files with student solutions
- Student B submitted:
  - Exam_1_Q1_Q2.pdf - scanned paper answers to questions 1 and 2
  - Exam_1_Q3_Q4.pdf - answers to questions 3 and 4
  - Exam_1.Rmd - R-markdown files with student solutions

No pdf report - no grade. If you experience difficulties with knitting, combine your answers in Word and any other editor and produce pdf-file for grading.

No R scripts - 50 % reduction in grade if relative code present in pdf- report, 100% reduction if no such code present.

Late submissions will result in a points reduction: 10% first hour, 50% second hour, 100% six hours. Only valid and documented reason to miss the exam or be late is acceptable. Sleeping in, lack of preparation, ennui, grogginess, inability to knit, etc. are not acceptable excuses. University policy allows multiple midterm exams on same day.

Reports longer than 25 pages are not going to be graded.

Exam must be submitted through UBLearns by 11:59pm 2022 October 14th.

Question 1. Clustering with Pen and Paper (25 Points).

Consider following 6 points:

	1	**2**	**3**	**4**	**5**	**6**
x	1	4	3	4	3	5
y	1	1	4	5	7	7

Q1.1 Perform single K-Means clustering with **Manhattan distance** using pen and paper. Show you work in readable manner. (20 points)

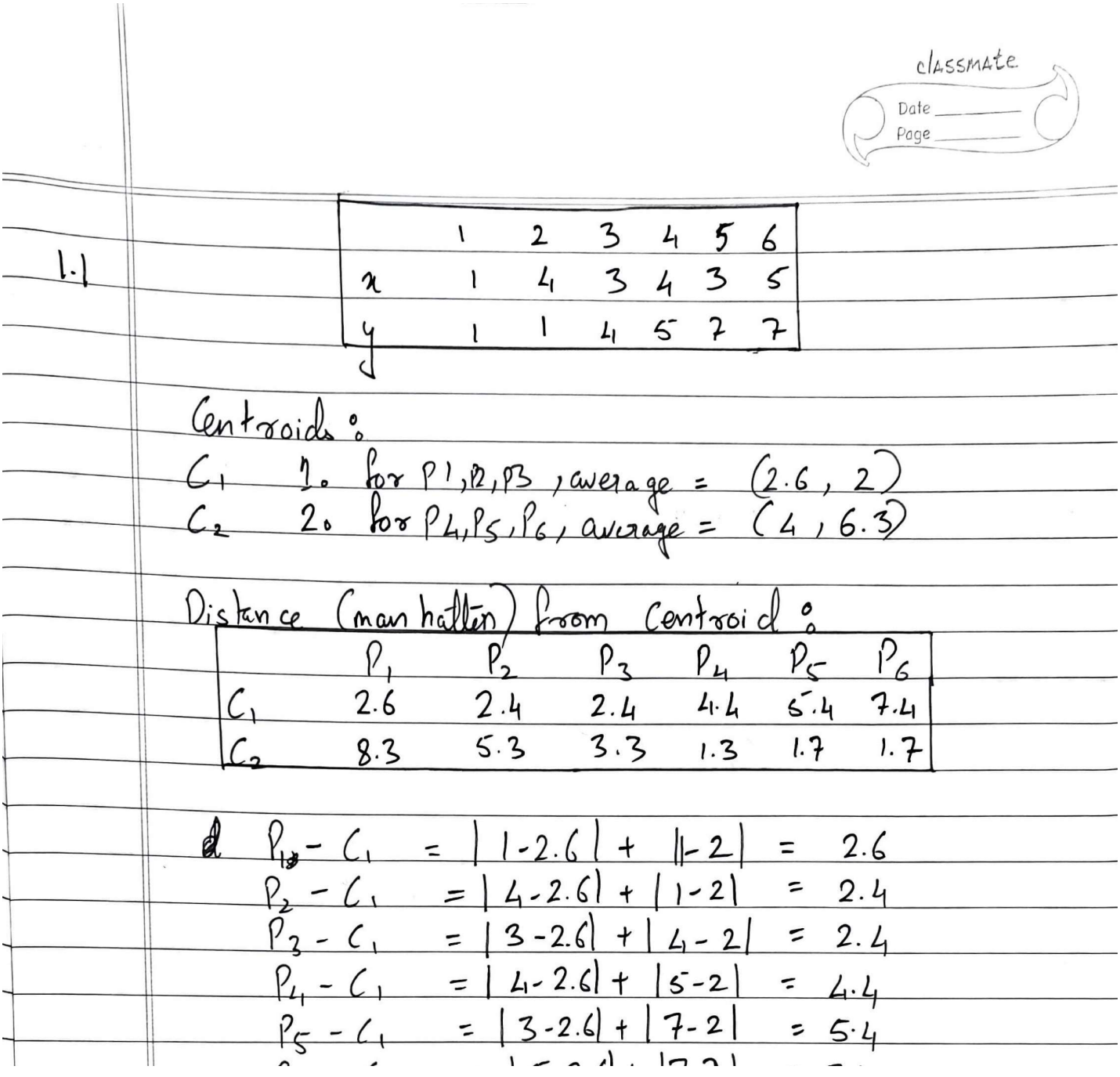
You can use rmarkdown instead of paper, simple calculator or simple vector calculations in R.

You can set points as p1 <- c(1,1), p2 <- c(4,1) and use regular math operation on them (+-*/). You also can use math function like sum, abs and so on and output function like print and cat

You will need a random sample without replacement, either of size two or six, use one generated below:

5 6 3 4 1 2

```
hint:
If you used pen and paper you can take a photo of it and add to Rmd using
following markdown command:
![Note](filename.jpeg)
```



	1	2	3	4	5	6
x	1	4	3	4	3	5
y	1	1	4	5	7	7

	P_1	P_2	P_3	P_4	P_5	P_6
C_1	2.6	2.4	2.4	4.4	5.4	7.4
C_2	8.3	5.3	3.3	1.3	1.7	1.7

$P_6 - C_1 = |5-2.6| + |7-2| = 7.4$

$P_1 - C_2 = |1-4| + |1-6.3| = 8.3$
$P_2 - C_2 = |4-4| + |1-6.3| = 5.3$
$P_3 - C_2 = |3-4| + |4-6.3| = 3.3$
$P_4 - C_2 = |4-4| + |5-6.3| = 1.3$
$P_5 - C_2 = |3-4| + |7-6.3| = 1.7$
$P_6 - C_2 = |5-4| + |7-6.3| = 1.7$

As we can see Points P_1, P_2, P_3 are closer to C_1 & Points P_4, P_5 & P_6 are closer to C_2
Therefore the distribution

P_1	P_2	P_3	P_4	P_5	P_6
C_1	C_1	C_1	C_2	C_2	C_2

Answer to 1.1

Q1.2 One of the students decided to check himself and make following clustering with R (5 points):

That is the student got that first two points belong to first cluster and 4 other points to second cluster. But the students results in Q1.1 are different. Can you explain why?

> hits: in general there are two reasons besides that student mess-up in Q1.1. assume that student is correct in Q1.1. Out of thouse two reasons only one actually have effect in this small example. Though if it is done not properly can play too.

```
clus$cluster
```

```
## [1] 1 1 2 2 2 2
```

```
print(clus$centers)
```

```
##      x    y
## 1 2.50 1.00
## 2 3.75 5.75
```

The centroids for the kmeans algorithm are different from the theoretical centers as we can see above. Our centroids we mean of the first half of the values and the second half. They were (2.6,2) & (4,6.3). The centroids for the above kmeans used with seed value 1 are (2.50,1) & (3.75,5.75). I believe that it might differ due to two

reasons: 1. The algorithm to find the centroids for R are better or more mordern as the ones we used in our theoretical approach. 2. The seed values might affect the centroid algorithm.

Question 2 (25 Points)

Q2.1 Just like in Q.1.1 but this time perform hierarchical clustering with complete linkage on same data. (20 points)

After building whole tree, don't forget to cut it to get two clusters. Draw dendogram by hand (show proper hights) or using ascii graphics

2.1

	1	2	3	4	5	6
x	1	4	3	4	3	5
y	1	1	4	5	7	7

Distance Matrix

	P_1	P_2	P_3	P_4	P_5	P_6
P_1	0					
P_2	3	0				
P_3	3.60	3.16	0			
P_4	5	4	1.41	0		
P_5	6.32	6.08	3	2.23	0	
P_6	7.21	6.08	3.60	2.23	2	0

Shortest distance
1.41
$P_4 - P_3 \rightarrow P_{43}$

$P_1 - P_{43} = \max(P_1 - P_3, P_1 - P_4) = 5$
$P_2 - P_{43} = \max(P_2 - P_3, P_2 - P_4) = 4$
$P_5 - P_{43} = \max(P_5 - P_3, P_5 - P_4) = 3$
$P_6 - P_{43} = \max(P_6 - P_3, P_6 - P_4) = 3.6$

stage 1:

	P_1	P_2	P_{43}	P_5	P_6
P_1	0				
P_2	3	0			
P_{43}	5	4	0		
P_5	6.32	6.08	3	0	
P_6	7.21	6.08	3.60	2	0

shortest distance
2
$P_5 - P_6 \rightarrow P_{56}$

$P_1 - P_{56} = \max(P_1 - P_5, P_1 - P_6) = 7.21$
$P_2 - P_{56} = \max(P_2 - P_5, P_2 - P_6) = 6.08$
$P_{34} - P_{56} = \max(P_{34} - P_5, P_{34} - P_6) = 3.60$

stage 2:

	P_1	~~P_{34}~~	~~P_{56}~~
P_1	3		
P_{34}	4		
P_{56}	7.21		

shortest distance = 3
$P_1 - P_2 \Rightarrow P_{12}$

	P_1	P_2	P_{34}	P_{56}
P_1	0			
P_2	3	0		
P_{34}	5	4	0	
P_{56}	7.21	6.08	3.6	0

$P_{34} - P_{12} = \max(P_{34} - P_1, P_{34} - P_2) = 5$

$P_{56} - P_{12} = \max(P_{56} - P_1, P_{56} - P_2) = 7.21$

	P_{12}	P_{34}	P_{56}
stage 3: P_{12}	0		
P_{34}	5	0	
P_{56}	7.21	3.6	0

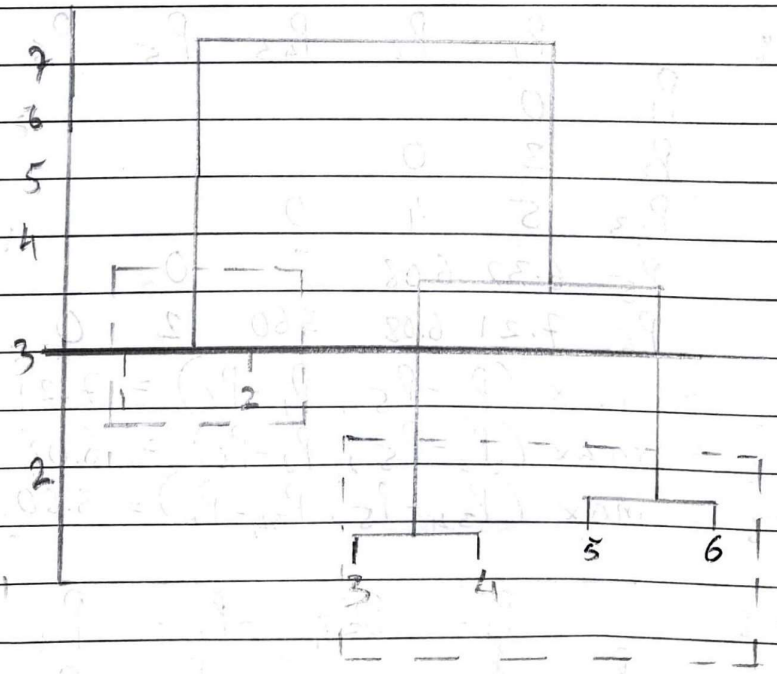
shortest distance = 3.6

$P_{12} - P_{5634}$ *

$P_{12} - P_{5634} = \max(P_{12} - P_{34}, P_{12} - P_{56}) = 7.21$

	P_{12}	P_{5634}
P_{12}	0	
P_{5634}	7.21	0

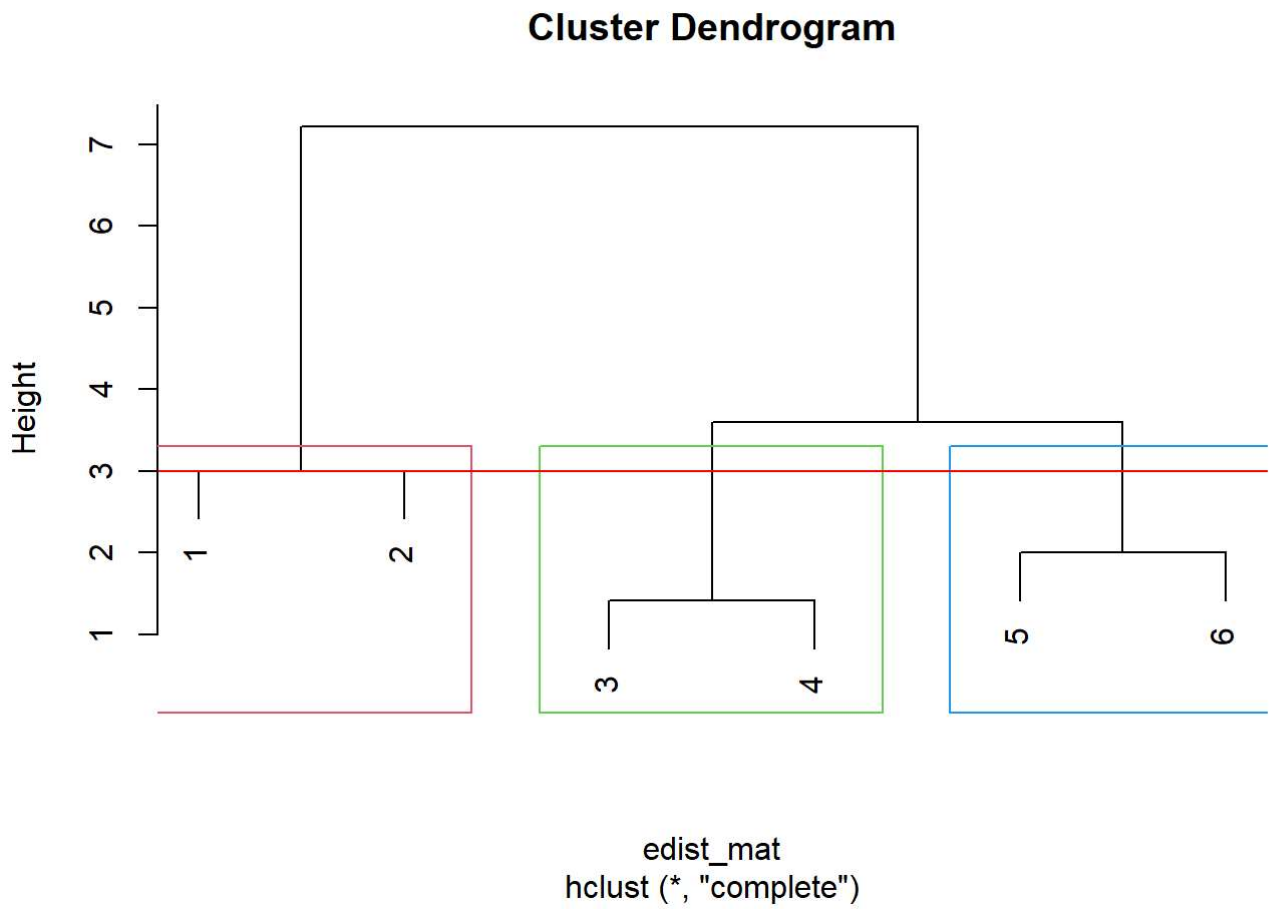
Last cluster : $P_{12} - P_{5634}$



Answer to 2.1, Page2

Q2.1 Do you expect hclust provides same answer as yours? Check it. (5 points)

```
x <- c(1,4,3,4,3,5)
y <- c(1,1,4,5,7,7)
df_0 <- data.frame(x,y)
#df_0_sc <- as.data.frame(scale(df_0))
edist_mat <- dist(df_0, method = 'euclidean')
hclust_com <- hclust(edist_mat, method = 'complete')
plot(hclust_com)
rect.hclust(hclust_com , k = 3, border = 2:6)
abline(h = 3, col = 'red')
```



The above plot matches our manual plot so we can tell that hclust gives the same answer as our manual calculated plot.

Question 3 (10 points)

Compare and contrast k-means and hierarchical clustering in terms of input, output, speed, cluster characteristics/ability to separate manifold structures.

K-Means	Hierarchical Clustering Algorithm(HCA)
We need to pre-defined the number of clusters or K.	HCA does not need any pre-defined value of K, we can stop at any number of clusters.
It forms spherical shaped clusters based on the distance of each data point	HCA follows divisive or agglomerative methods
We choose median or mean as centroid for our clusters	We begin with n clusters and the algorithm works till only one cluster is left
K-Mean is faster as it requires less computation	JCA required more computation and is slower than K-mean
It is suited for large datasets. Eg Sports performance of athletes	It is suited when the cluster needs to be arranged in a hierarchical manner. Eg DNA sequencing
It is a division of data based on non-overlapping clusters	Nested clusters arranged as a tree
Works well when the data is spread in a hyper spherical manner	Doesn't work as well as K means when the data is spread hyper spherical
Convergence is guaranteed	Convergence is not guaranteed
Results are not reproducible as K-means start at random points	Results are reproducible

K-Means vs Hierarchical clustering

Question 4 (40 points)

Clustering is often used to reexamine existing classifications. Sometime misclassification can occur in the original design. For example, one class might consist of two sufficiently different groups, or two classes are essentially the same.

In this exercise, you are presented with a seed dataset containing measurements from multiple wheat subspecies. Your task is to identify a number of subspecies.

`seeds.csv` constists of 200 records and reports 7 measurements for each seed:

- `length` - length of kernel
- `width` - width of kernel
- `asymmetry` - asymmetry coefficient
- `groove` - length of kernel groove
- `area` - area A
- `perimeter` - perimeter P
- `compactness` - compactness C = 4*pi*A/P^2

Q4.1 Read and Visualize Dataset (5 points)

```
# read dataset
df = fread("seeds.csv",stringsAsFactors = FALSE)
print(c("na values:",sum(is.na(df))))
```

```
## [1] "na values:" "0"
```

```
print(c("null values:",sum(is.null(df))))
```

```
## [1] "null values:" "0"
```

```
head(df)
```

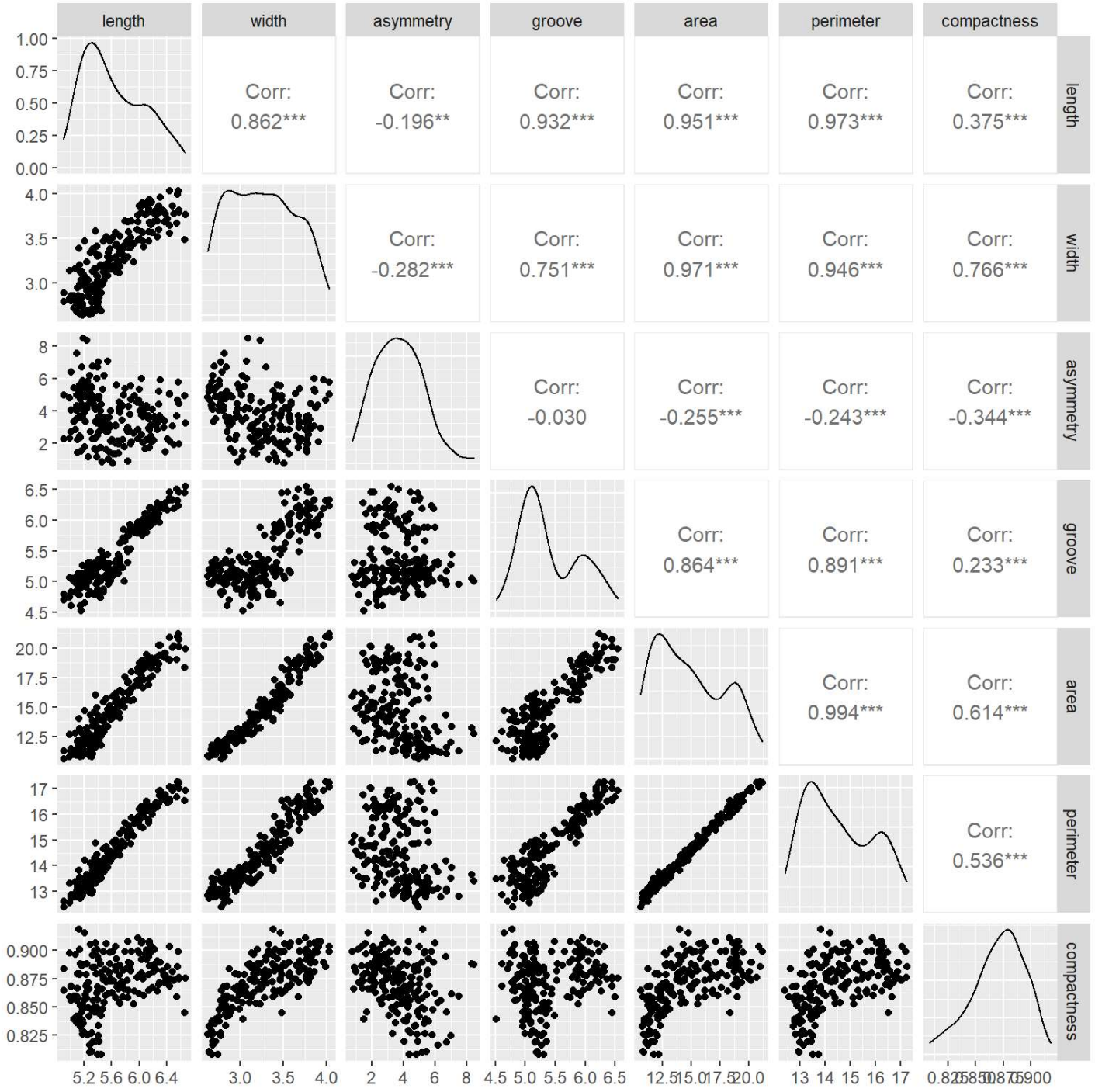
length	width	asymmetry	groove	area	perimeter	compactness
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
5.541	3.073	7.035	5.440	13.32	13.94	0.8613
5.609	3.158	2.217	5.132	14.01	14.29	0.8625
5.319	2.897	4.924	5.270	12.44	13.59	0.8462
6.513	3.773	1.910	6.185	20.16	17.03	0.8735
5.350	2.810	4.271	5.308	12.02	13.33	0.8503
5.138	3.201	2.461	4.783	13.16	13.55	0.9009

6 rows

```
summary(df)
```

```
##      length          width         asymmetry          groove
##  Min.   :4.899   Min.   :2.630   Min.   :0.7651   Min.   :4.519
##  1st Qu.:5.263   1st Qu.:2.930   1st Qu.:2.6183   1st Qu.:5.045
##  Median :5.524   Median :3.237   Median :3.6250   Median :5.223
##  Mean   :5.631   Mean   :3.258   Mean   :3.7143   Mean   :5.409
##  3rd Qu.:6.000   3rd Qu.:3.562   3rd Qu.:4.7860   3rd Qu.:5.878
##  Max.   :6.675   Max.   :4.033   Max.   :8.4560   Max.   :6.550
##       area         perimeter      compactness
##  Min.   :10.59   Min.   :12.41   Min.   :0.8081
##  1st Qu.:12.25   1st Qu.:13.45   1st Qu.:0.8566
##  Median :14.34   Median :14.29   Median :0.8734
##  Mean   :14.85   Mean   :14.56   Mean   :0.8707
##  3rd Qu.:17.33   3rd Qu.:15.74   3rd Qu.:0.8875
##  Max.   :21.18   Max.   :17.25   Max.   :0.9183
```

```
# Examine dataset, for example by plotting it with GGally:ggpairs
data<-dplyr::select(df,length,width,asymmetry,groove,area,perimeter,compactness)
ggpairs(data)
```



Comment on observation, any distinct clustering?

The data which forms hyper spherical clusters are good for K-mean clustering, so from the above we can tell that the following are good for K Mean clustering: -length-asymmetry:3 clusters visible -width-asymmetry:3 clusters visible -asymmetry-groove:2 clusters visible -asymmetry-area:2 clusters visible -asymmetry-perimeter:2 clusters visible -asymmetry-compactness:2 clusters visible

Q4.2 Run PCA (8 Points)

Q4.2.1 Run PCA (don't forget to scaled data), save as `pc_out` , we will use `pc_out$x[,1]` and `pc_out$x[,2]` later for plotting

```
df_scale <- as.data.frame(scale(df))
df_scale
```

length <dbl>	width <dbl>	asymmetry <dbl>	groove <dbl>	area <dbl>	perimeter <dbl>	compac
-0.201804468	-0.4871928932	2.21342739	0.06232685	-0.522654134	-0.473615569	-0.397(
-0.049389254	-0.2637822549	-0.99805230	-0.56013329	-0.287239712	-0.207389842	-0.346(
-0.699395316	-0.9497843326	0.80632203	-0.28123881	-0.822892816	-0.739841296	-1.033(
1.976836541	1.3526594225	-1.20268581	1.56795286	1.811019262	1.876777280	0.117(
-0.629911909	-1.1784516918	0.37105922	-0.20444178	-0.966188551	-0.937608979	-0.860(
-1.105088755	-0.1507627555	-0.83541198	-1.26545339	-0.577242985	-0.770267093	1.272(
-0.340771282	-0.2690389758	-0.38548488	-1.08558666	-0.365711186	-0.382338177	0.218(
-0.488703696	-1.4255175742	0.78899151	-0.11551890	-1.037836418	-0.846331587	-2.146(
-0.103182859	0.0358508366	0.17375785	-0.53588160	-0.143943977	-0.123718899	0.184(
1.647350709	1.4262535152	-0.50146609	1.56795286	1.589252053	1.633370901	0.306(

1-10 of 200 rows Previous **1** 2 3 4 5 6 … 20 Next

```
pc_out <- prcomp(df_scale)
pc_out
```

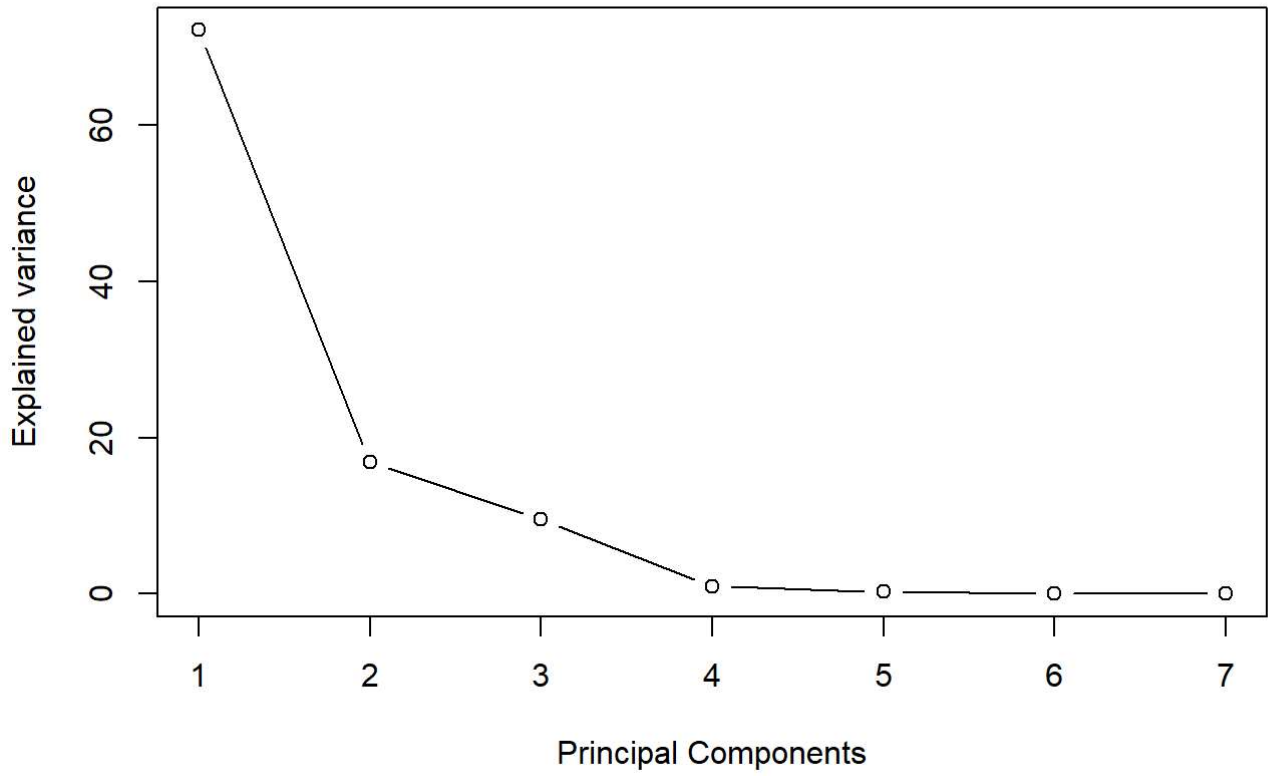
```
## Standard deviations (1, .., p=7):
## [1] 2.24824359 1.08840367 0.81857786 0.25782495 0.13444617 0.07312961 0.02847917
##
## Rotation (n x k) = (7 x 7):
##                    PC1         PC2         PC3         PC4         PC5
## length       0.4224390  0.21054814 -0.20929501  0.27957654 -0.76376099
## width        0.4321888 -0.11334615  0.21744663  0.19127128  0.46128464
## asymmetry   -0.1313456  0.71254540  0.68090375  0.09927373 -0.03360331
## groove       0.3856353  0.38601837 -0.20648132 -0.80470471  0.09474757
## area         0.4434049  0.03053263  0.02791147  0.19013824  0.21222821
## perimeter    0.4405006  0.08739006 -0.05756447  0.29323567  0.18723595
## compactness  0.2795410 -0.52680332  0.63131244 -0.32512704 -0.33716649
##                    PC6          PC7
## length      -0.26464436  0.043357115
## width       -0.70844688 -0.042629661
## asymmetry    0.02000234 -0.003594224
## groove      -0.04333067 -0.035161796
## area         0.41929322  0.738024799
## perimeter    0.47830958 -0.667221101
## compactness  0.14560864 -0.072034718
```

Q4.2.2 Make scree plot/percentage variance explained plot. Comment on percentage variance explained (will two first components cover enough varience in dataset)

```
#screeplot(pc_out, type = "line", main = "Scree plot")
exp_var = 100 * pc_out$sdev^2 / sum(pc_out$sdev^2)
print(exp_var)
```

```
## [1] 72.20856085 16.92317912  9.57242456  0.94962436  0.25822534  0.07639915
## [7]  0.01158662
```

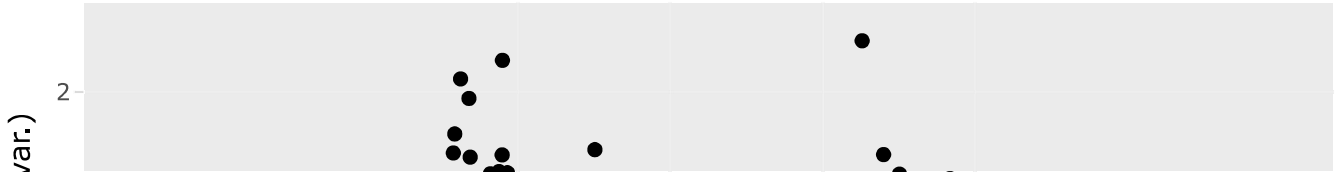
```
# Scree plot
plot(exp_var, xlab = "Principal Components",
    ylab = "Explained variance",
     type = "b")
```

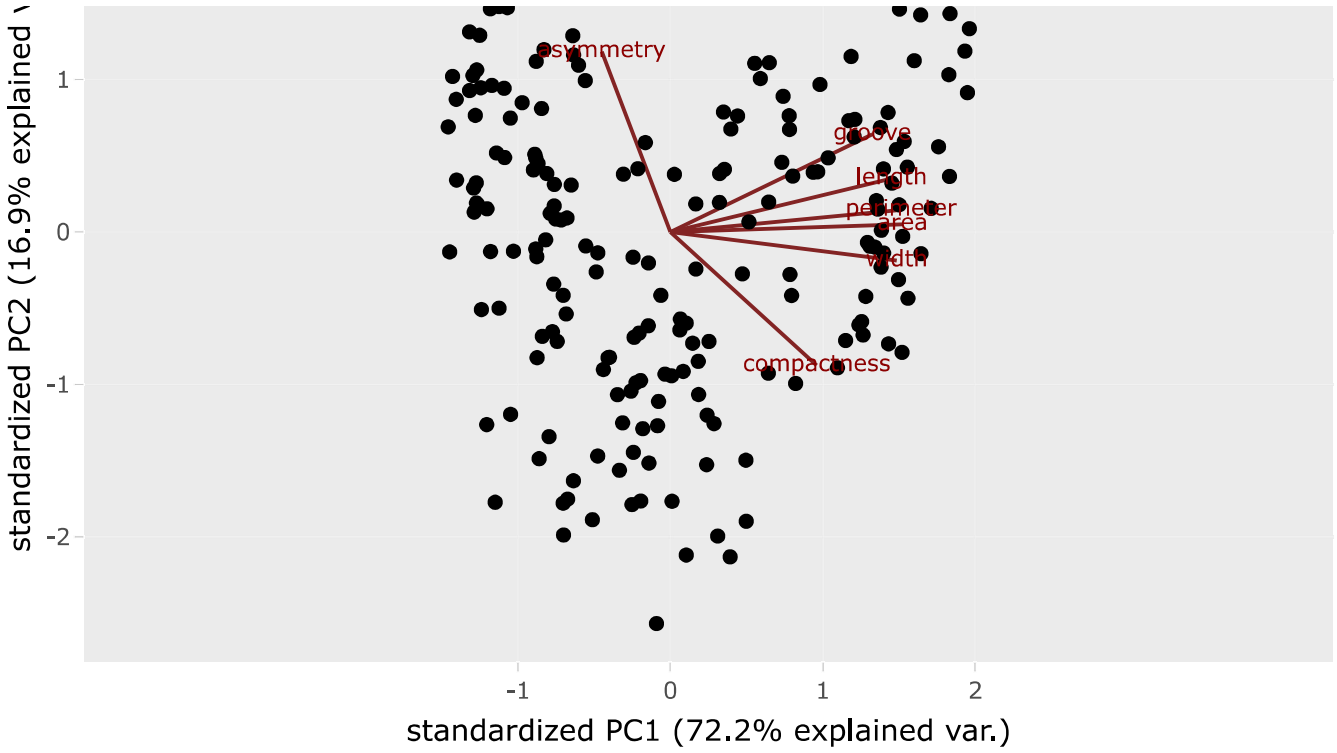


Percent variance explained as a cumulative sum of PC1 & PC2 is more than 80% which is enough for our analysis.

Q4.2.3 Make biplot. Comment on biplots (clusters?, possible meaning of PC?)

```
ggplotly(ggbiplot(pc_out,scale=1))
```



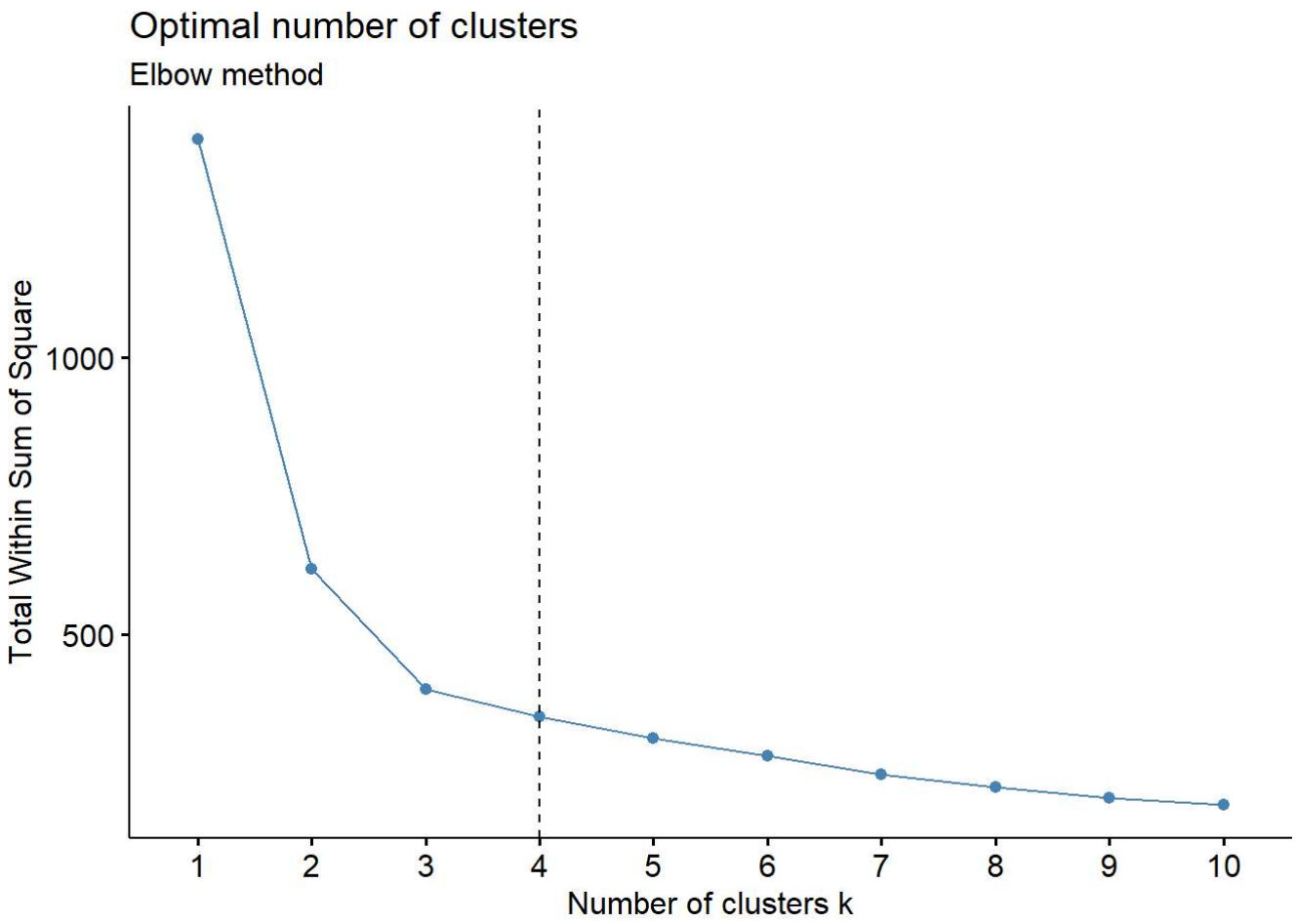


We can observe that the PC1 explaince 72% variance and PC2 explains 16.9% variance which sums up to give us more than 80% variance explained. Further I can see that there are 3 clusters, one from “compactness”, one from “asymmetry” abd third from all the other components.

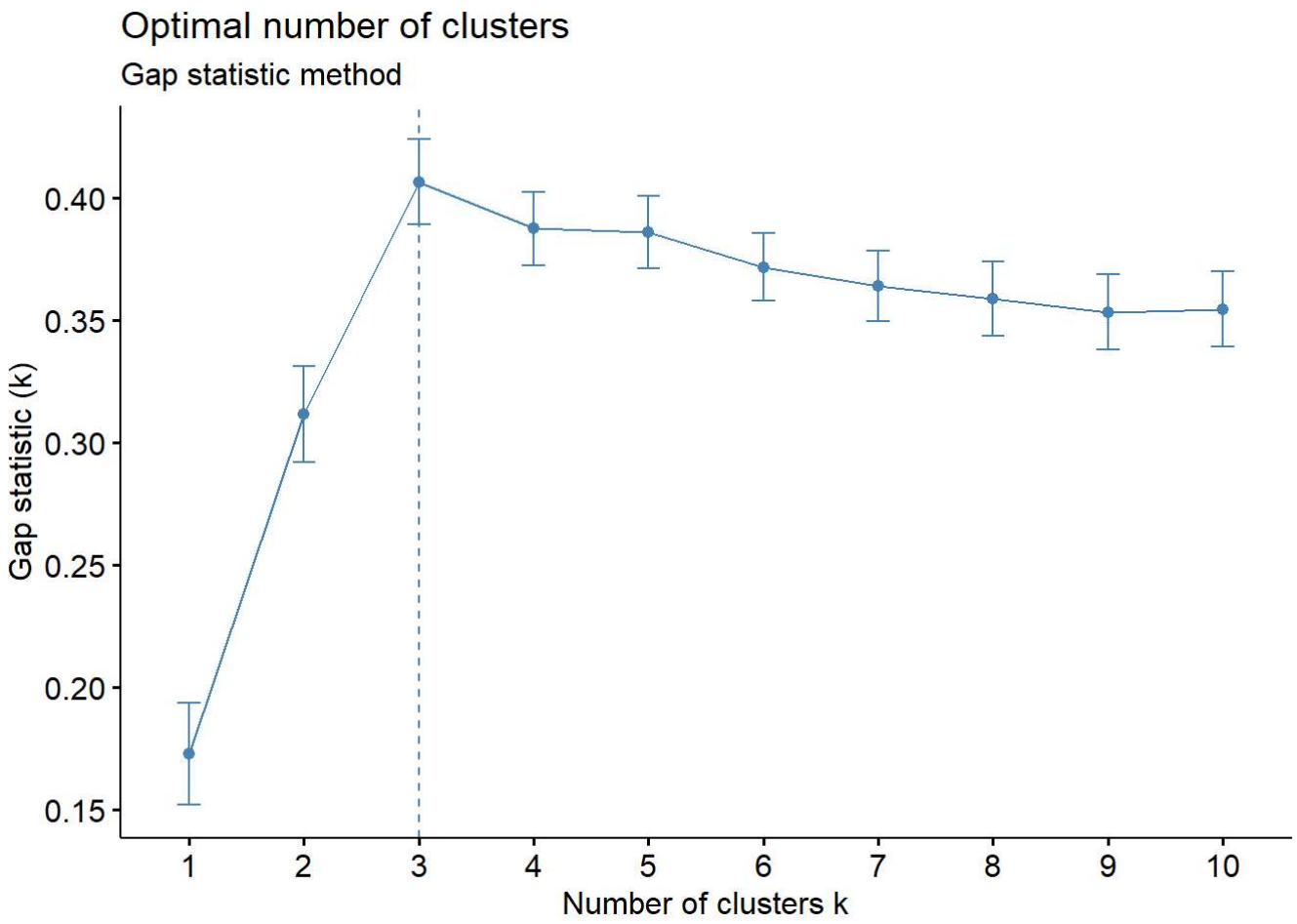
Q4.3 Perform Clustering of your choise. Select number of clusters. (20 points)

looks like k-means should do the job, biplot has nice packed shape.

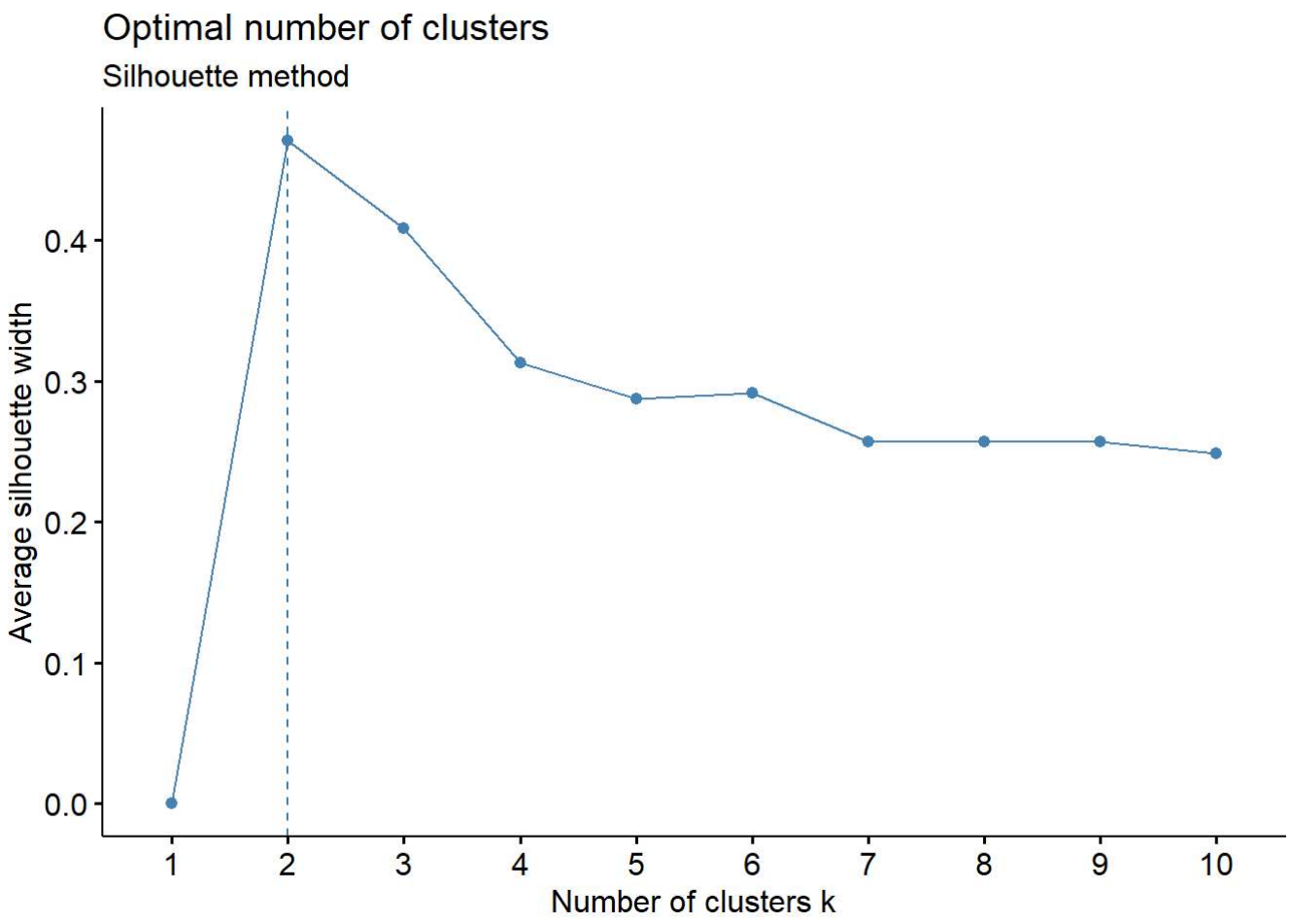
```
#elbow method
#compute from k=2 to k=20
#k.max <- 20
#data <- df_scale
#wss <- sapply(1:k.max,
#              function(k){kmeans(data, k, nstart=50,iter.max = 20 )$tot.withinss})
#wss
#plot(1:k.max, wss,
#     type="b", pch = 19, frame = FALSE,
#     xlab="Number of clusters K",
#     ylab="Total within-clusters sum of squares")
fviz_nbclust(df_scale, kmeans, method = "wss") +
    geom_vline(xintercept = 4, linetype = 2)+
  labs(subtitle = "Elbow method")
```



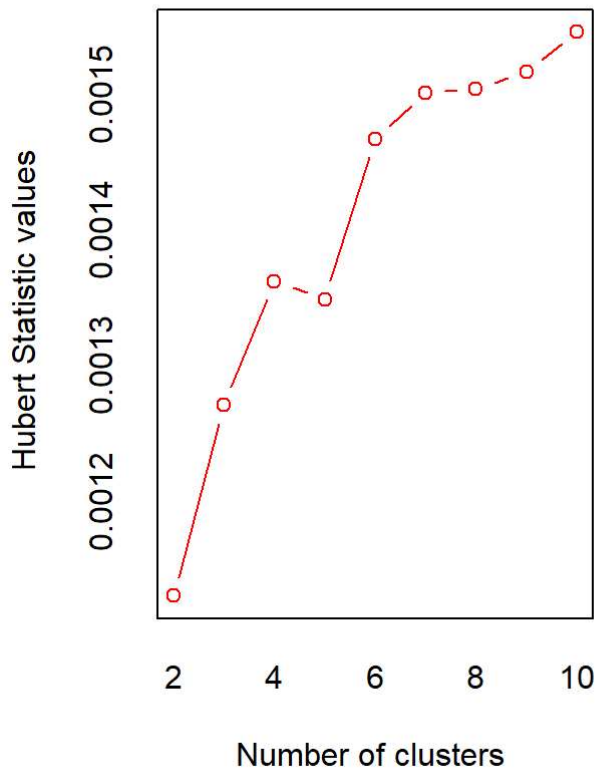

```
#Gap statistics
#gap_kmeans <- clusGap(df_scale, kmeans, nstart = 20, K.max = 10, B = 100)
#plot(gap_kmeans, main = "Gap Statistic: kmeans")
set.seed(72)
fviz_nbclust(df_scale, kmeans, nstart = 25,  method = "gap_stat", nboot = 50)+
  labs(subtitle = "Gap statistic method")
```

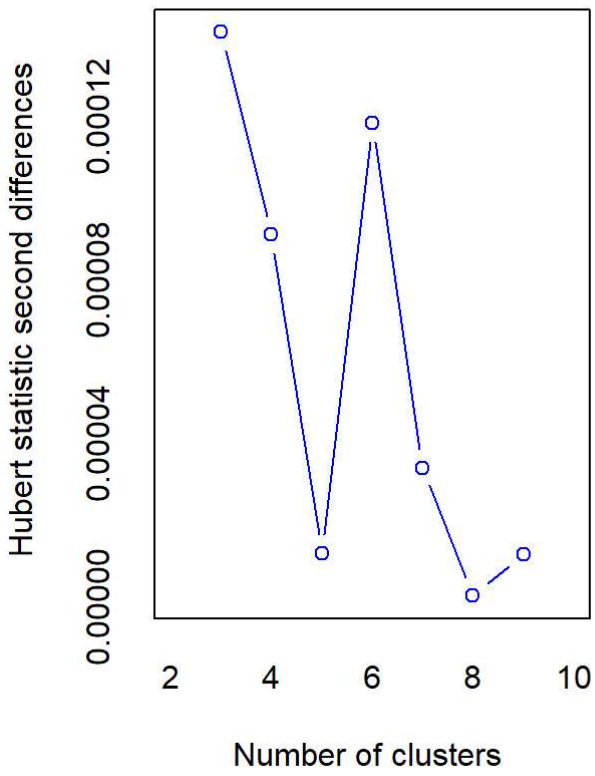



```
#silhouette analysis
set.seed(72)
fviz_nbclust(df_scale, kmeans, method='silhouette')+
  labs(subtitle = "Silhouette method")
```

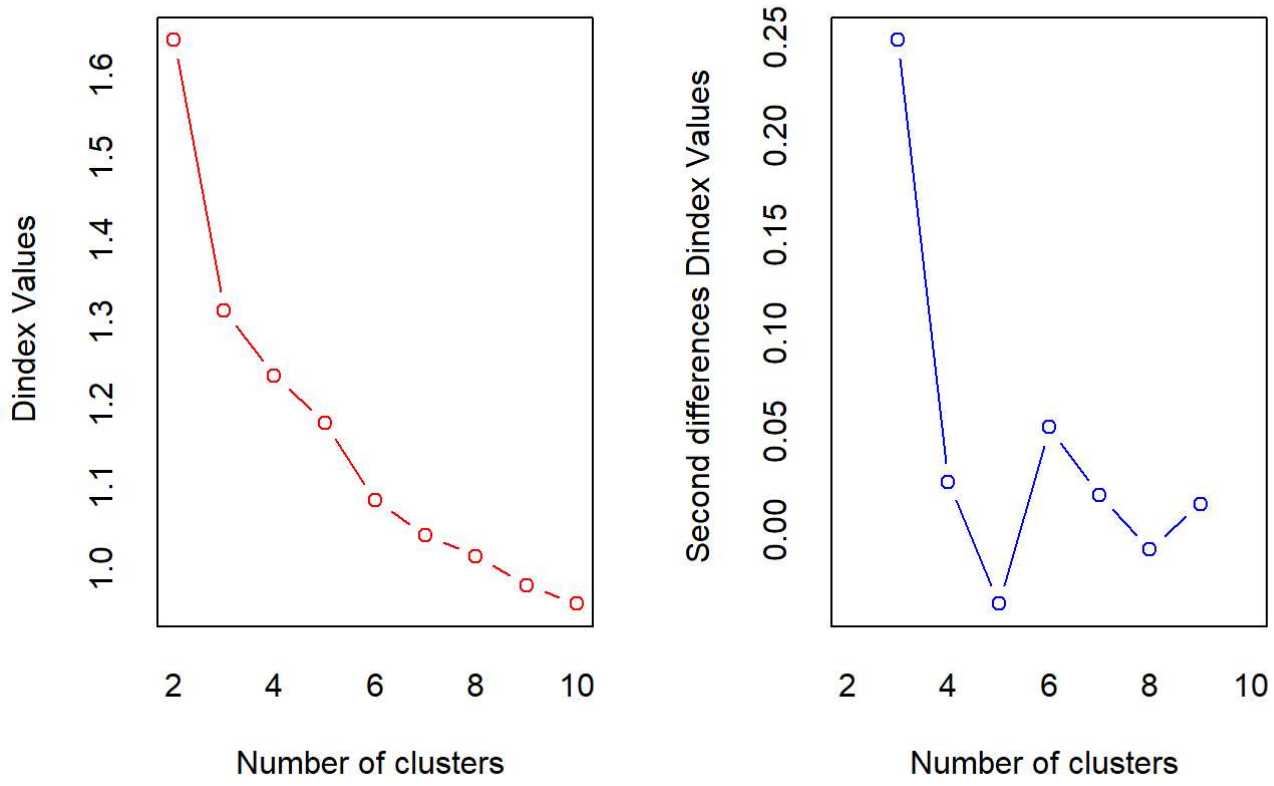



```
library("NbClust")
nb <- NbClust(df_scale, distance = "euclidean", min.nc = 2,
        max.nc = 10, method = "kmeans")
```





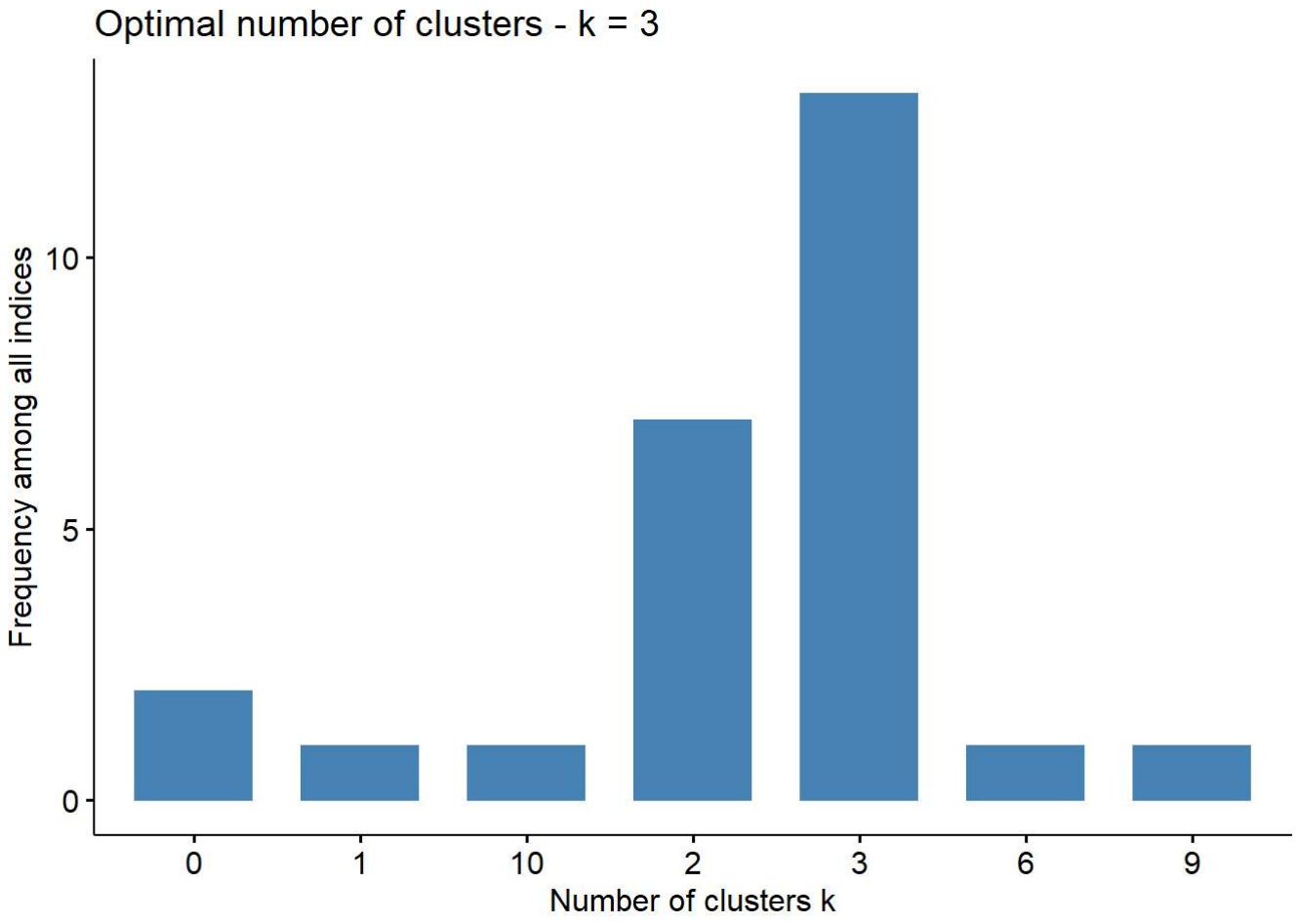

```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in
Hubert
##                 index second differences plot.
##
```



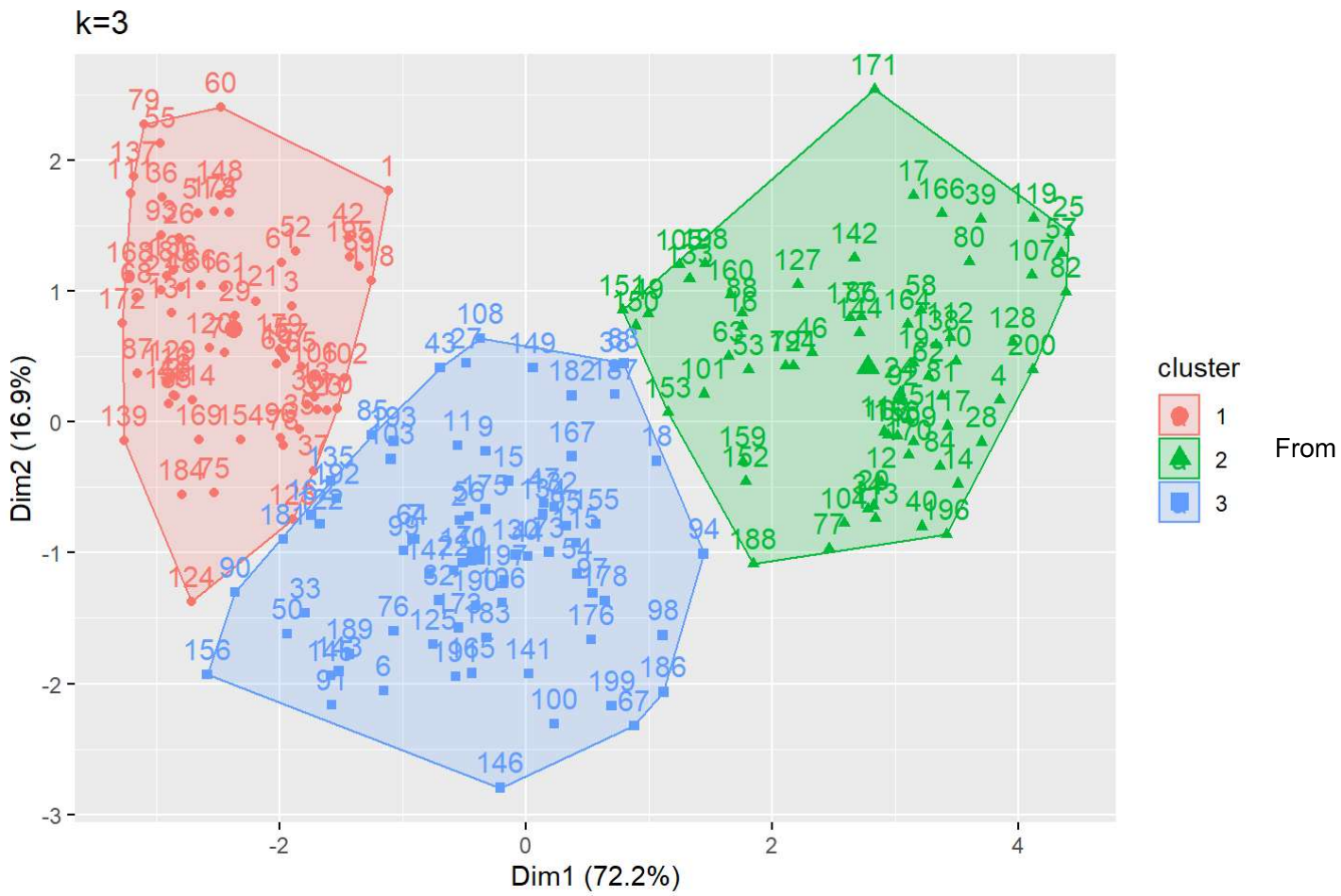

```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in D
index
##                 second differences plot) that corresponds to a significant increase of the va
lue of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 7 proposed 2 as the best number of clusters
## * 13 proposed 3 as the best number of clusters
## * 1 proposed 6 as the best number of clusters
## * 1 proposed 9 as the best number of clusters
## * 1 proposed 10 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  3
##
##
## *******************************************************************
```

```
fviz_nbclust(nb)
```

```
## Among all indices:
## ==================
## * 2 proposed  0 as the best number of clusters
## * 1 proposed  1 as the best number of clusters
## * 7 proposed  2 as the best number of clusters
## * 13 proposed  3 as the best number of clusters
## * 1 proposed  6 as the best number of clusters
## * 1 proposed  9 as the best number of clusters
## * 1 proposed  10 as the best number of clusters
##
## Conclusion
## ========================
## * According to the majority rule, the best number of clusters is  3 .
```




```
km_3 <- kmeans(df_scale, 3, nstart = 25)
fviz_cluster(km_3, data = df_scale) + ggtitle("k=3")
```



our analysis, we can tell that from elbow, gap analysis and Silhouette we are getting ideal values of k as 4,3,2 respectively. So from the above 3 itself we can tell that the ideal K should be 3 as it is the mean value(we have no majority here). I used further another library of 30 tests and found out that 13 tests resulted in K=3 with clear majority. Hence we shall used K=3 as ideal value.

Q4.4 Characterize clusters. (5 points)

```
df_scale$Cluster <- km_3$cluster
head(df_scale)
```

	length <dbl>	width <dbl>	asymmetry <dbl>	groove <dbl>	area <dbl>	perimeter <dbl>	compactne… <dbl>	Clu <dbl>
1	-0.20180447	-0.4871929	2.2134274	0.06232685	-0.5226541	-0.4736156	-0.3970116	
2	-0.04938925	-0.2637823	-0.9980523	-0.56013329	-0.2872397	-0.2073898	-0.3464261	
3	-0.69939532	-0.9497843	0.8063220	-0.28123881	-0.8228928	-0.7398413	-1.0335453	
4	1.97683654	1.3526594	-1.2026858	1.56795286	1.8110193	1.8767773	0.1172740	
5	-0.62991191	-1.1784517	0.3710592	-0.20444178	-0.9661886	-0.9376090	-0.8607117	
6	-1.10508875	-0.1507628	-0.8354120	-1.26545339	-0.5772430	-0.7702671	1.2723087	

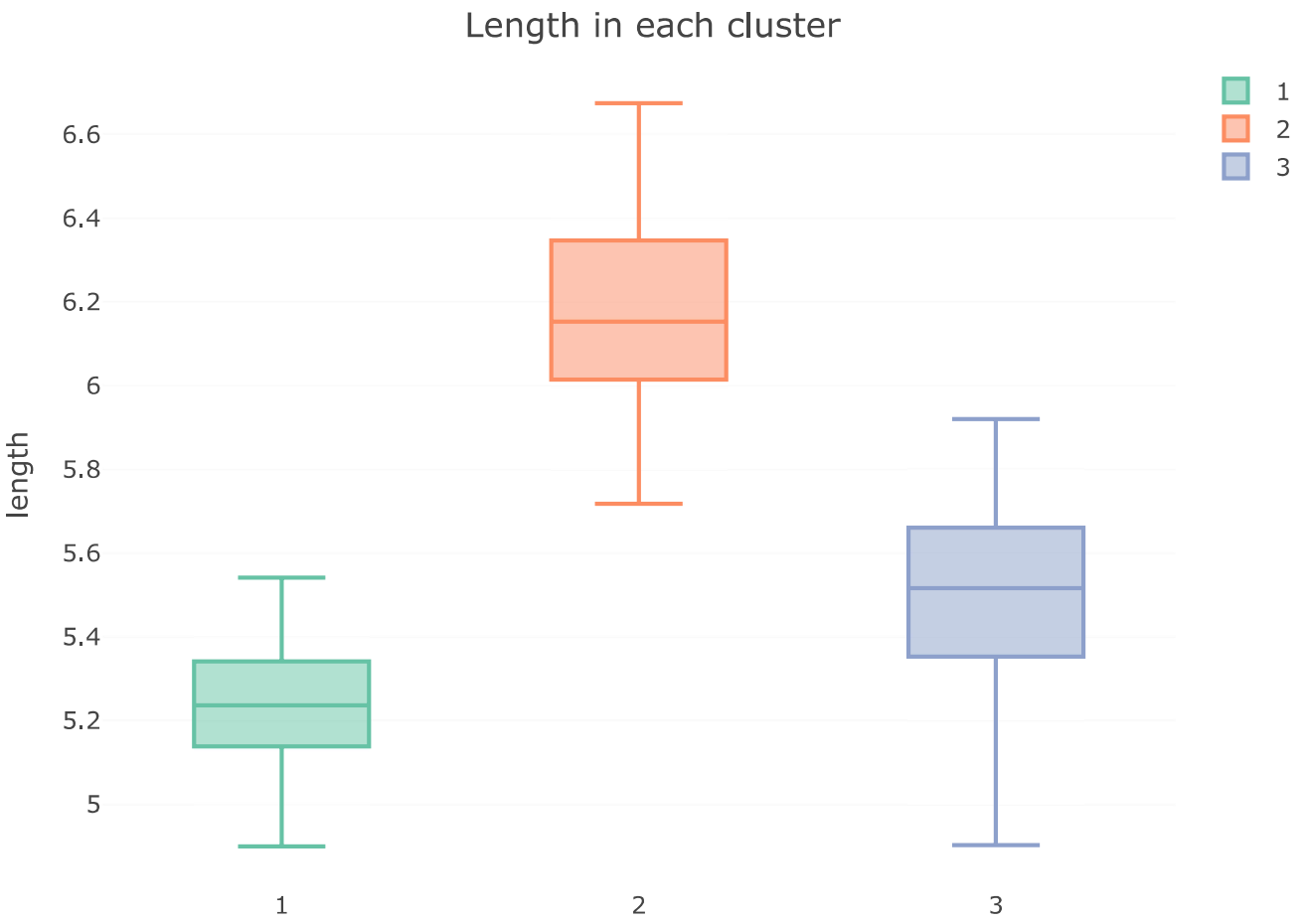
6 rows

```
describeBy(df_scale, group="Cluster")
```

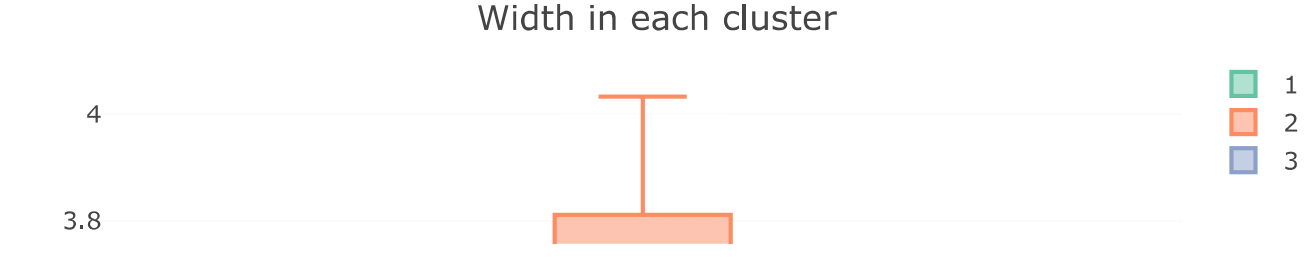
```
##
##  Descriptive statistics by group
## Cluster: 1
##             vars  n  mean   sd median trimmed  mad   min   max range  skew
## length         1 64 -0.88 0.32  -0.89   -0.88 0.33 -1.64 -0.20  1.44 -0.03
## width          2 64 -1.11 0.35  -1.15   -1.13 0.41 -1.65 -0.07  1.58  0.56
## asymmetry      3 64  0.81 0.81   0.76    0.75 0.72 -0.68  3.16  3.84  0.75
## groove         4 64 -0.58 0.34  -0.64   -0.58 0.28 -1.62  0.17  1.79 -0.21
## area           5 64 -1.04 0.24  -1.05   -1.04 0.28 -1.45 -0.52  0.94  0.31
## perimeter      6 64 -1.01 0.27  -1.02   -1.01 0.30 -1.64 -0.47  1.17  0.04
## compactness    7 64 -1.02 0.81  -0.96   -1.01 0.79 -2.64  0.74  3.38 -0.11
## Cluster        8 64  1.00 0.00   1.00    1.00 0.00  1.00  1.00  0.00   NaN
##             kurtosis   se
## length         -0.65 0.04
## width          -0.30 0.04
## asymmetry       0.47 0.10
## groove          0.12 0.04
## area           -0.97 0.03
## perimeter      -0.83 0.03
## compactness    -0.59 0.10
## Cluster          NaN 0.00
## ------------------------------------------------------------
## Cluster: 2
##             vars  n  mean   sd median trimmed  mad   min  max range  skew
## length         1 65  1.22 0.54   1.17    1.21 0.52  0.19 2.34  2.15  0.18
## width          2 65  1.15 0.44   1.15    1.15 0.47  0.34 2.04  1.70 -0.10
## asymmetry      3 65 -0.08 0.78  -0.06   -0.10 0.77 -1.49 1.52  3.02  0.16
## groove         4 65  1.28 0.47   1.21    1.27 0.52  0.15 2.31  2.15  0.16
## area           5 65  1.24 0.44   1.33    1.24 0.38  0.24 2.16  1.92 -0.16
## perimeter      6 65  1.25 0.42   1.27    1.25 0.42  0.25 2.04  1.80 -0.13
## compactness    7 65  0.56 0.62   0.51    0.57 0.70 -1.08 1.69  2.77 -0.17
## Cluster        8 65  2.00 0.00   2.00    2.00 0.00  2.00 2.00  0.00   NaN
##             kurtosis   se
## length         -0.76 0.07
## width          -0.92 0.05
## asymmetry      -0.92 0.10
## groove         -0.54 0.06
## area           -0.56 0.05
## perimeter      -0.64 0.05
## compactness    -0.65 0.08
## Cluster          NaN 0.00
## ------------------------------------------------------------
## Cluster: 3
##             vars  n  mean   sd median trimmed  mad   min  max range  skew
## length         1 71 -0.33 0.53  -0.26   -0.31 0.53 -1.63 0.65  2.28 -0.34
## width          2 71 -0.05 0.44  -0.07   -0.04 0.41 -1.00 0.85  1.85 -0.08
## asymmetry      3 71 -0.66 0.81  -0.72   -0.72 0.76 -1.97 1.98  3.95  0.75
## groove         4 71 -0.65 0.52  -0.64   -0.67 0.40 -1.80 0.95  2.75  0.39
## area           5 71 -0.20 0.40  -0.18   -0.19 0.42 -1.24 0.54  1.78 -0.33
## perimeter      6 71 -0.23 0.44  -0.21   -0.22 0.50 -1.47 0.54  2.01 -0.40
## compactness    7 71  0.41 0.69   0.44    0.42 0.70 -1.33 2.01  3.33 -0.08
## Cluster        8 71  3.00 0.00   3.00    3.00 0.00  3.00 3.00  0.00   NaN
```

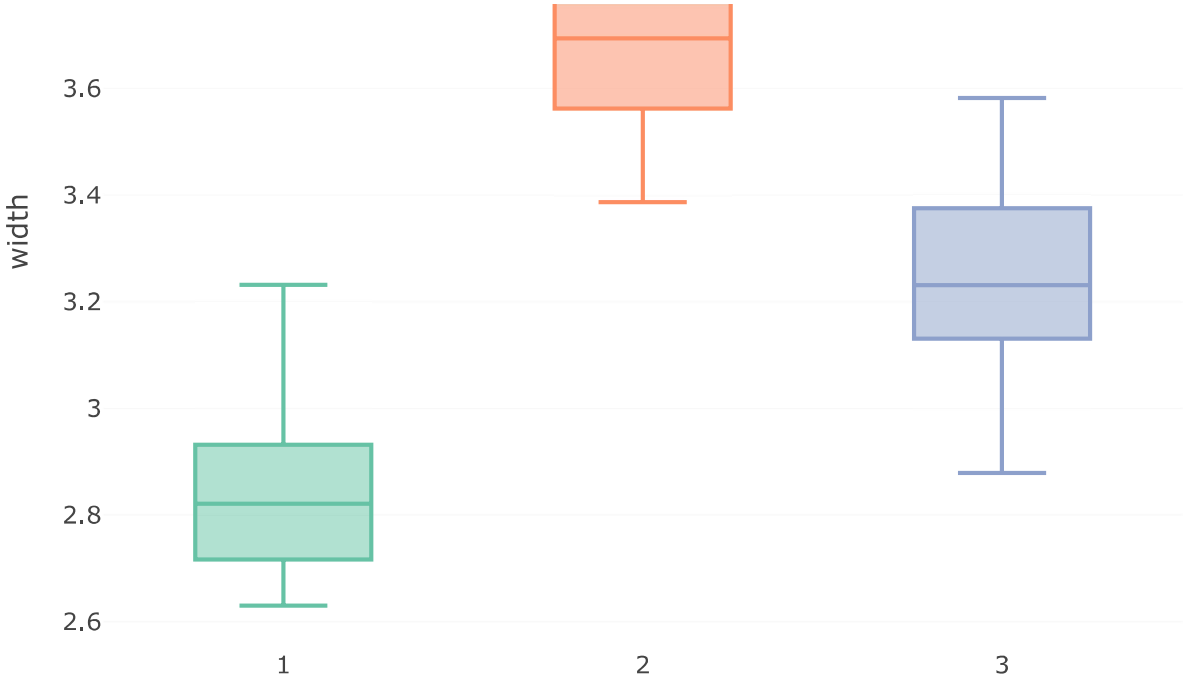
```
##             kurtosis   se
## length         -0.57 0.06
## width          -0.66 0.05
## asymmetry       0.46 0.10
## groove          0.38 0.06
## area           -0.56 0.05
## perimeter      -0.40 0.05
## compactness    -0.32 0.08
## Cluster          NaN 0.00
```

```
char_data <- cbind(data, clusters= km_3$cluster)
char_data$clusters <- as.factor(char_data$clusters)
plot_ly(char_data, y =~length ,color = ~clusters, type = "box" )  %>% layout(title = "Length in
 each cluster")
```

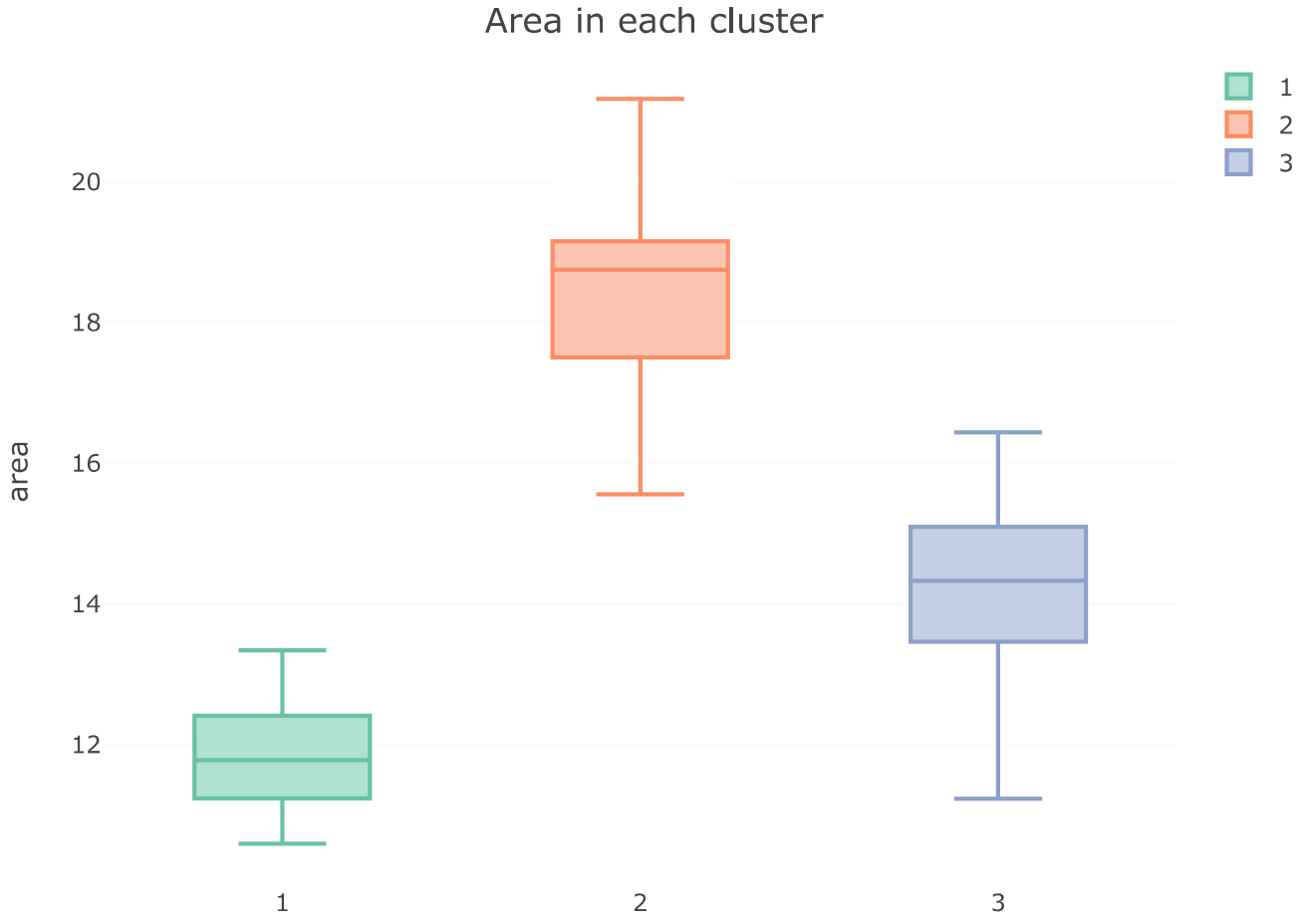



```
plot_ly(char_data, y =~width ,color = ~clusters, type = "box" )  %>% layout(title = "Width in ea
ch cluster")
```






```
plot_ly(char_data, y =~area ,color = ~clusters, type = "box" )  %>% layout(title = "Area in each cluster")
```



Q4.5 How many subspecies you think are in the set? (does it matches conclussion from Q4.3?) (2 points)

From the above analysis data we can tell that there are 3 subspecies in the given dataset with major difference in the length and area. It matches our biplot analysis explectations in 4.3