Task1: Hierarchical clustering

1.1

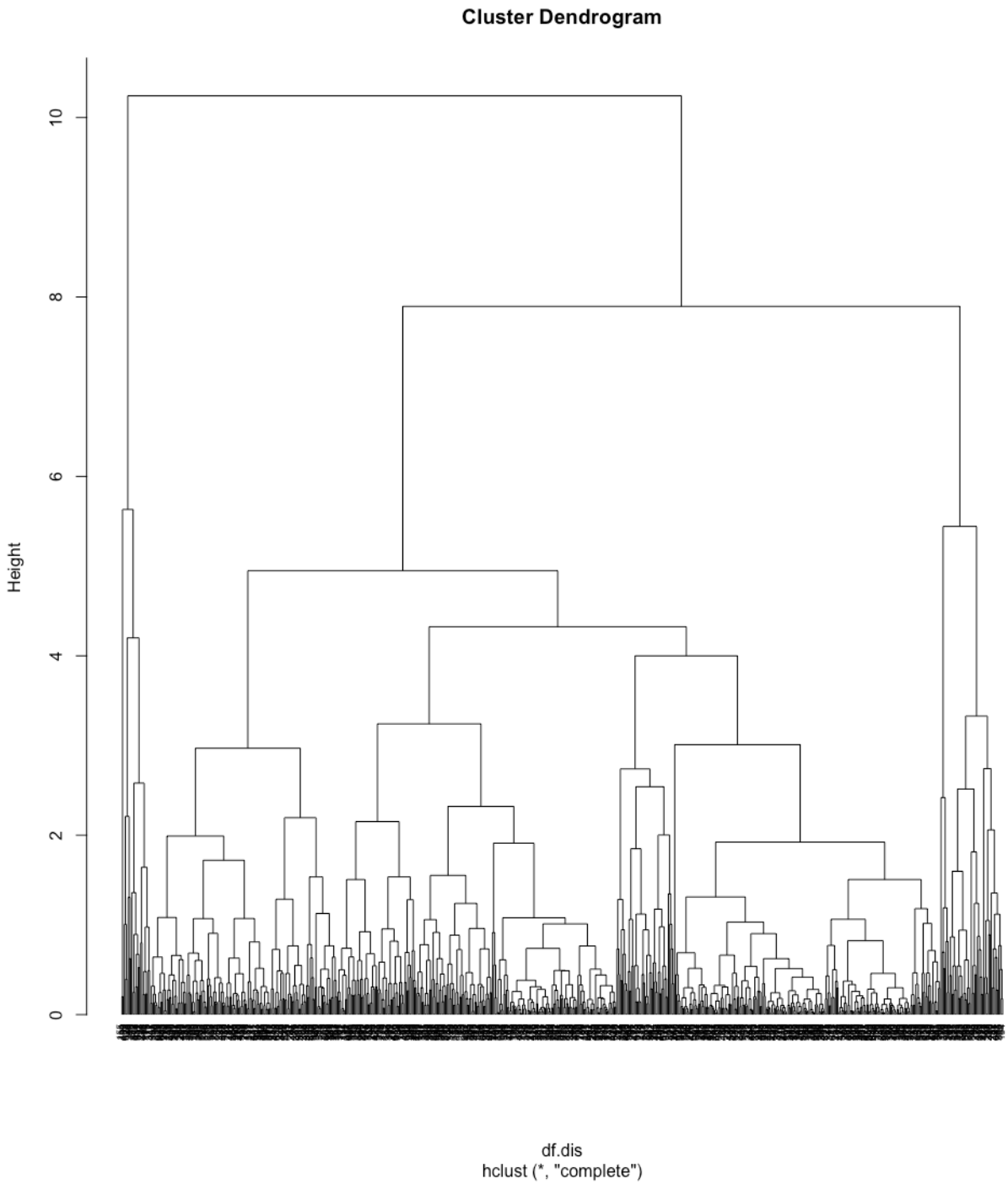
Firstly, we do data preparation. Load the data and use scale function to standardize the data

```
df = read.csv(paste(folder,"xfang7.csv", sep=''),header = FALSE)
df = scale(df)
```

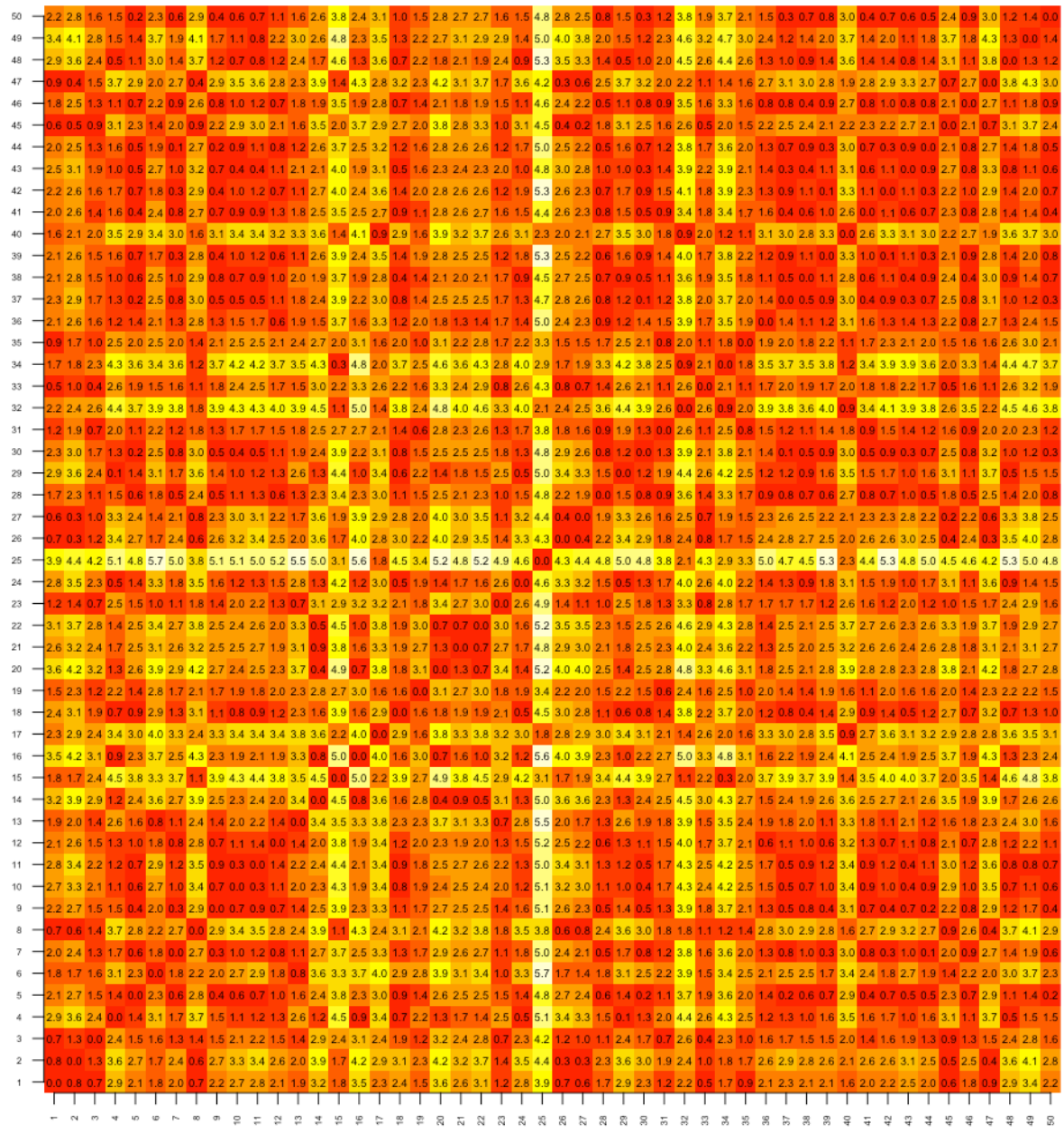
Then, we use complete linkage criteria to do hierarchical clustering.

```
df.dis = dist(df, method = "euclidean")
hc.complete = hclust(df.dis,method = "complete")
plot(hc.complete,cex=0.6,hang= -1)
```

the dendrogram showed as follows:



the graph distance matrix shown below. we just selected the 50 observations for good display

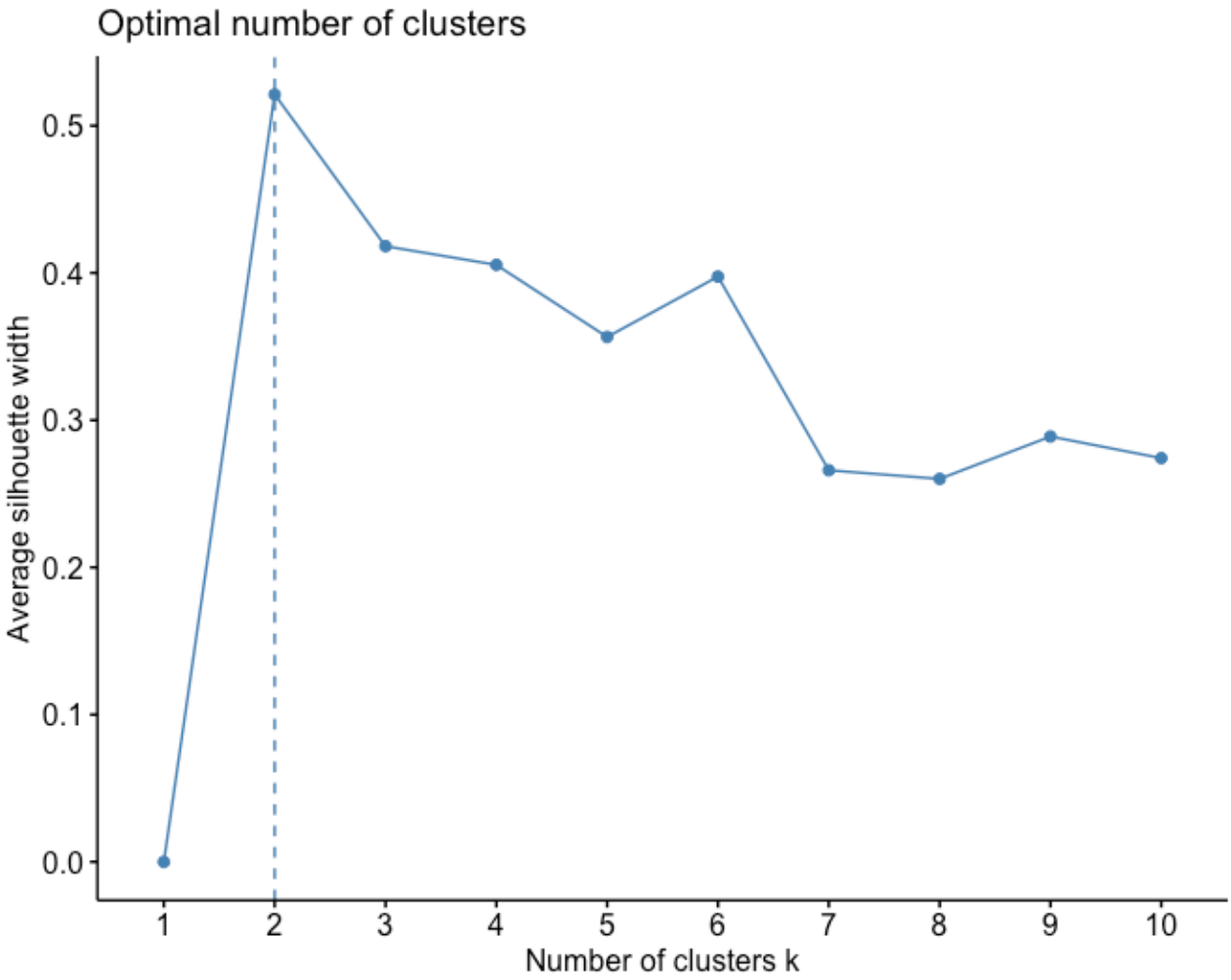


1.2

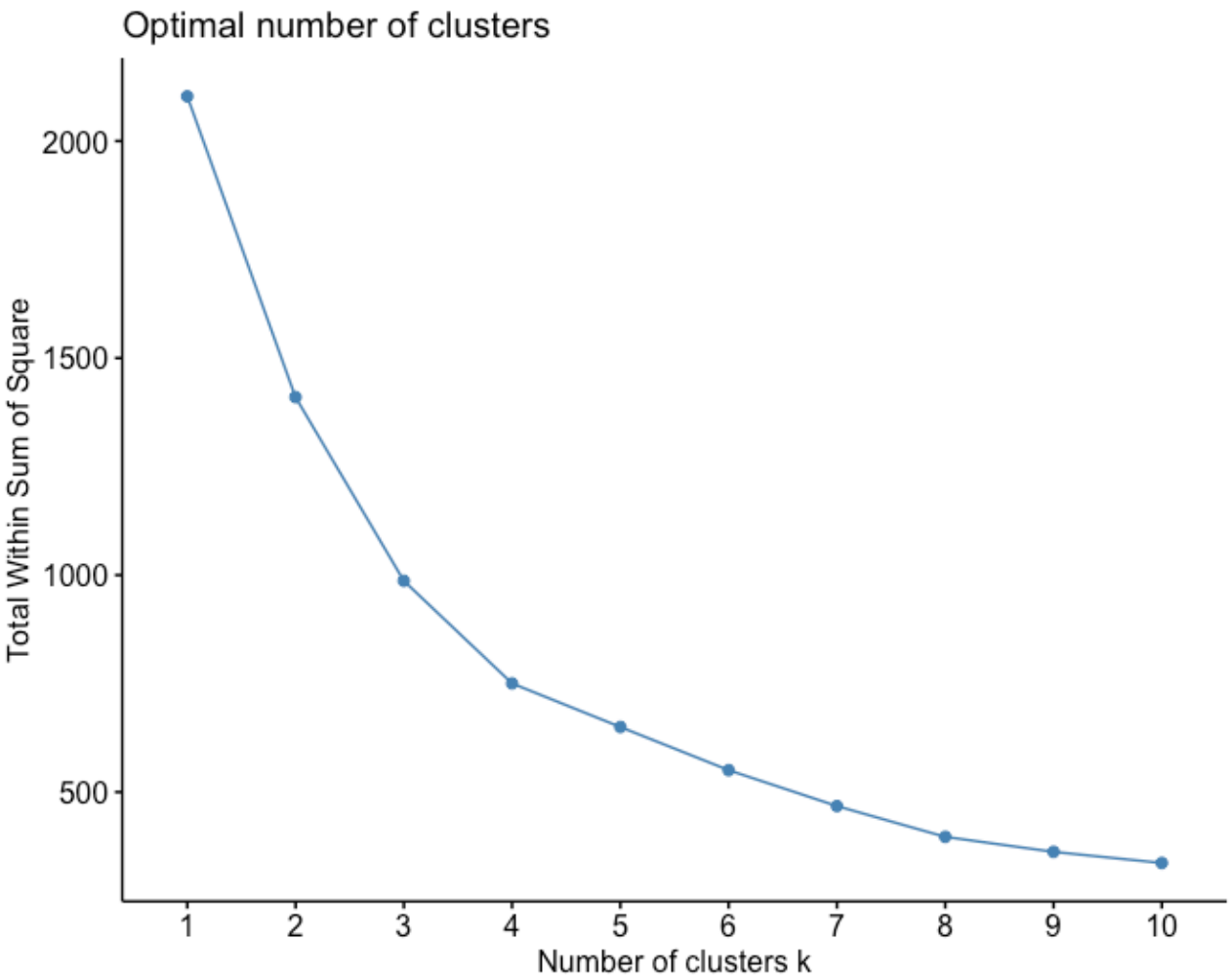
There are some approaches to determine the number of cluster. such as elbow method and average silhouette method. Here we use both them to see how it determined

```
# Average silhouette method for hierarchical clustering
fviz_nbclust(df, hcut, method = "silhouette",
             hc_method = "complete")

# Elbow method for hierarchical clustering
fviz_nbclust(df, hcut, method = "wss") + geom_vline(xintercept = 4, linetype =
2)
```

the result from average silhouette method shows that k= 2 is optimal



and elbow method result shows the elbow is not clear.

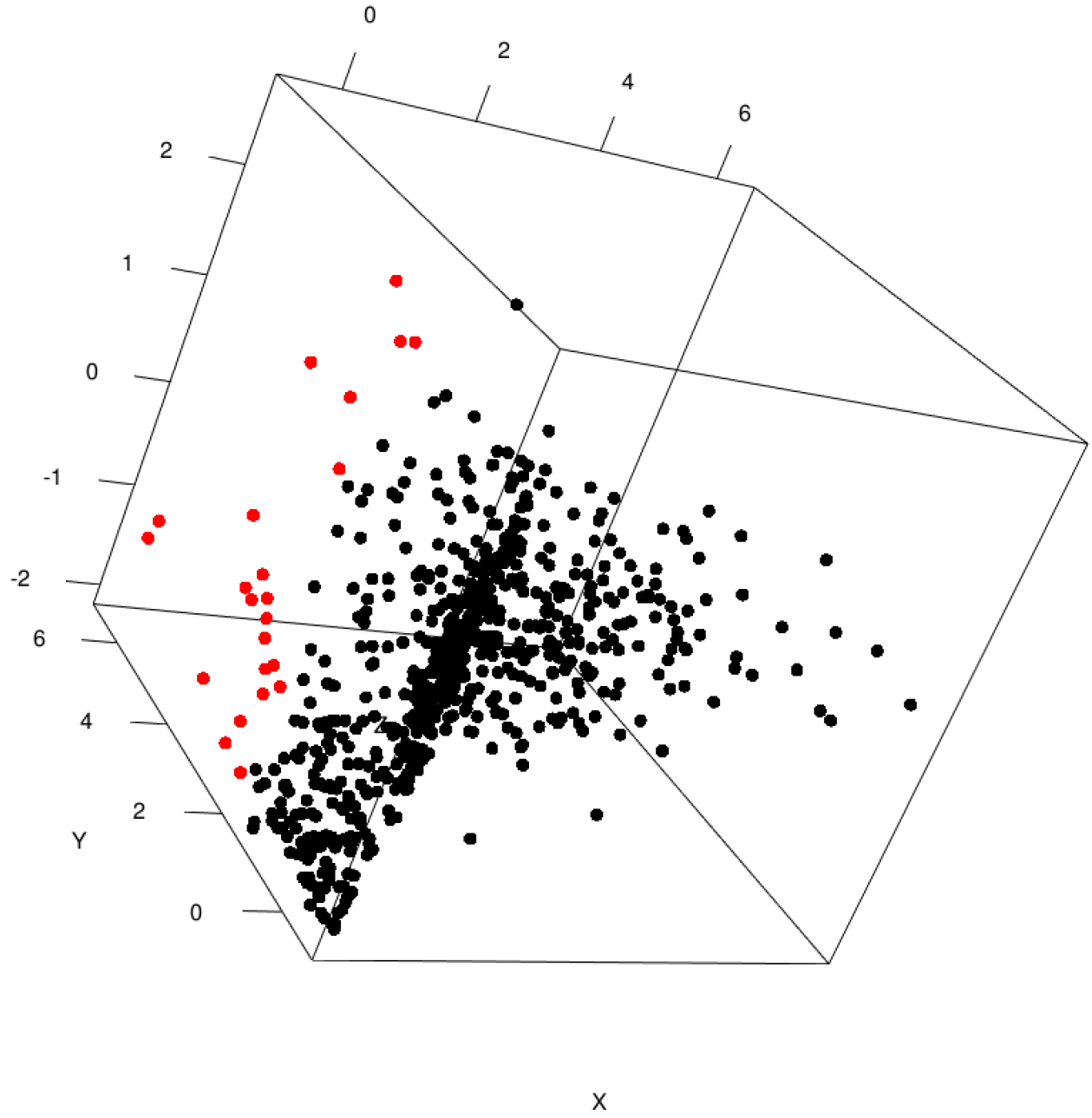


1.3

from 1.2, when can plot when k = 2 to see the result

```
cut2 = cutree(hc.complete,k = 2)
library(rgl)
plot3d(df,xlab = "X",ylab = "Y",zlab = "Z",col=cut2,size =8)
```

the results when k = 2 shows below



Task 2: K-means clustering

2.1

We can define the function to do the cluster varying k, for example, below, we set k.max = 15, and vary k from 2 to 15 to apply the function of kmeans.

```
set.seed(123)
k.max <- 15 # Maximal number of clusters
wss <- sapply(2:k.max,
              function(k){kmeans(df, k, nstart=10)})
```

2.2

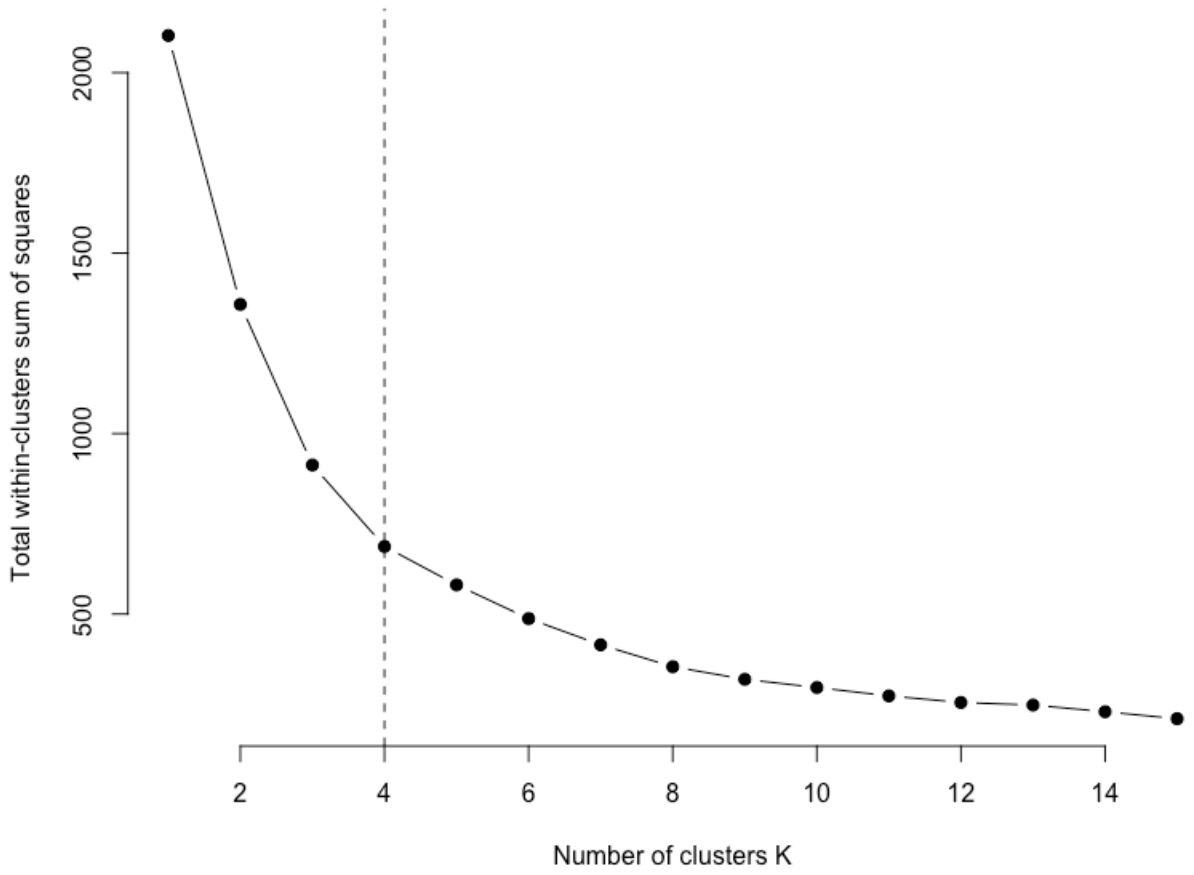
the elbow method works below

> 1. Compute clustering algorithm (e.g., k-means clustering) for different values of k. For instance, by varying k from 1 to 10 clusters
> 2. For each k, calculate the total within-cluster sum of square (wss)
> 3. Plot the curve of **wss** according to the number of clusters k.
> 4. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.

Elbow method for k-means clustering R code shown below

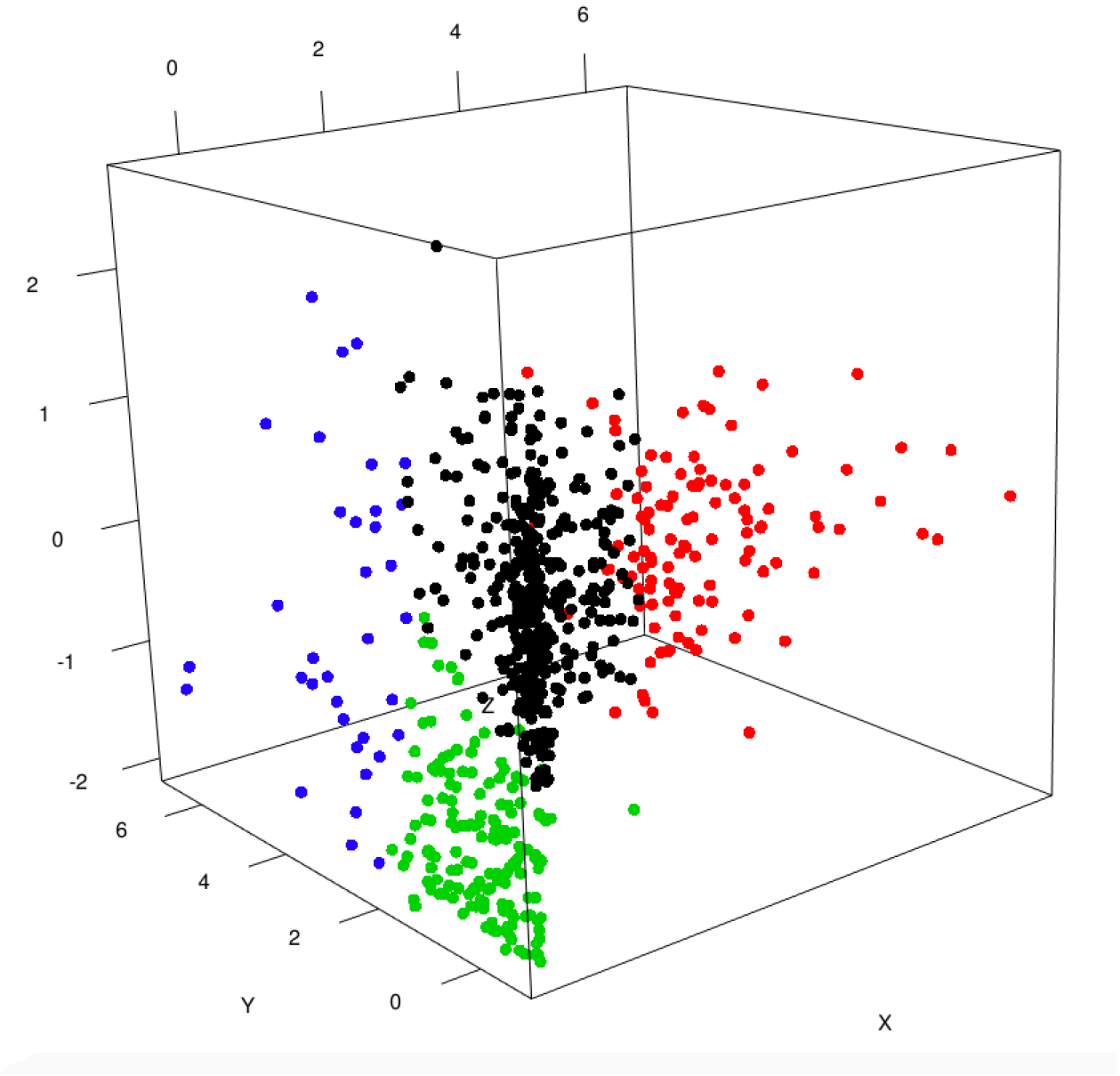
```
set.seed(123)
k.max <- 15 # Maximal number of clusters
wss <- sapply(1:k.max,
              function(k){kmeans(df, k, nstart=10 )$tot.withinss})
plot(1:k.max, wss,
     type="b", pch = 19, frame = FALSE,
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")
abline(v = 4, lty =2)
```

And the result shows below, k = 4 is optimal



2.3

we know from 2.2 k = 4 is optimal for k-means. the 3D plot shows below



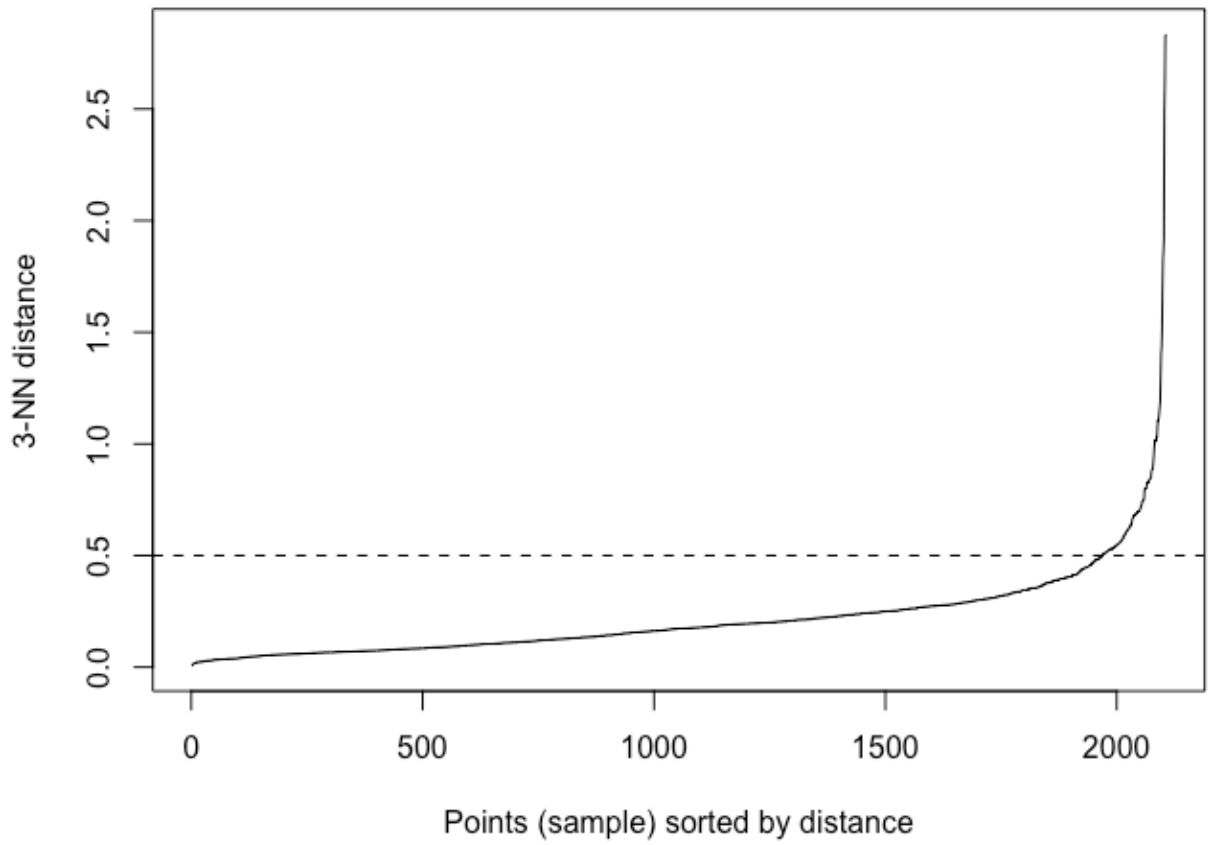
Task 3: DBSCAN Clustering

3.1

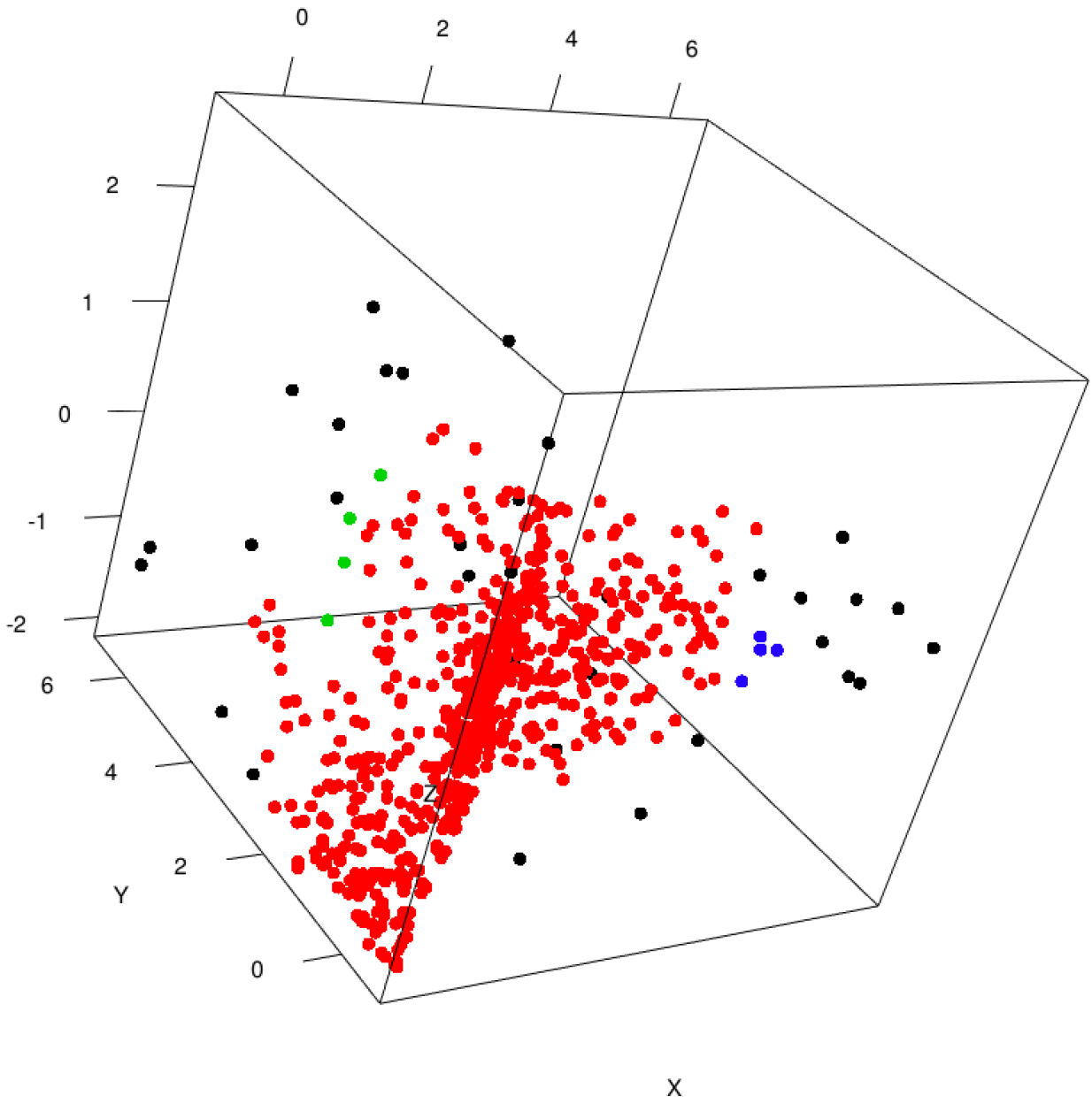
we compute the k-nearest neighbor distances in a matrix of points. plot a k-distance graph (with k=minPts) and look for an elbow in this graph.

```
dbscan::kNNdistplot(df, k =  3)
abline(h = 0.5, lty = 2)
```

the result shows below . So ϵ = 0.5 here

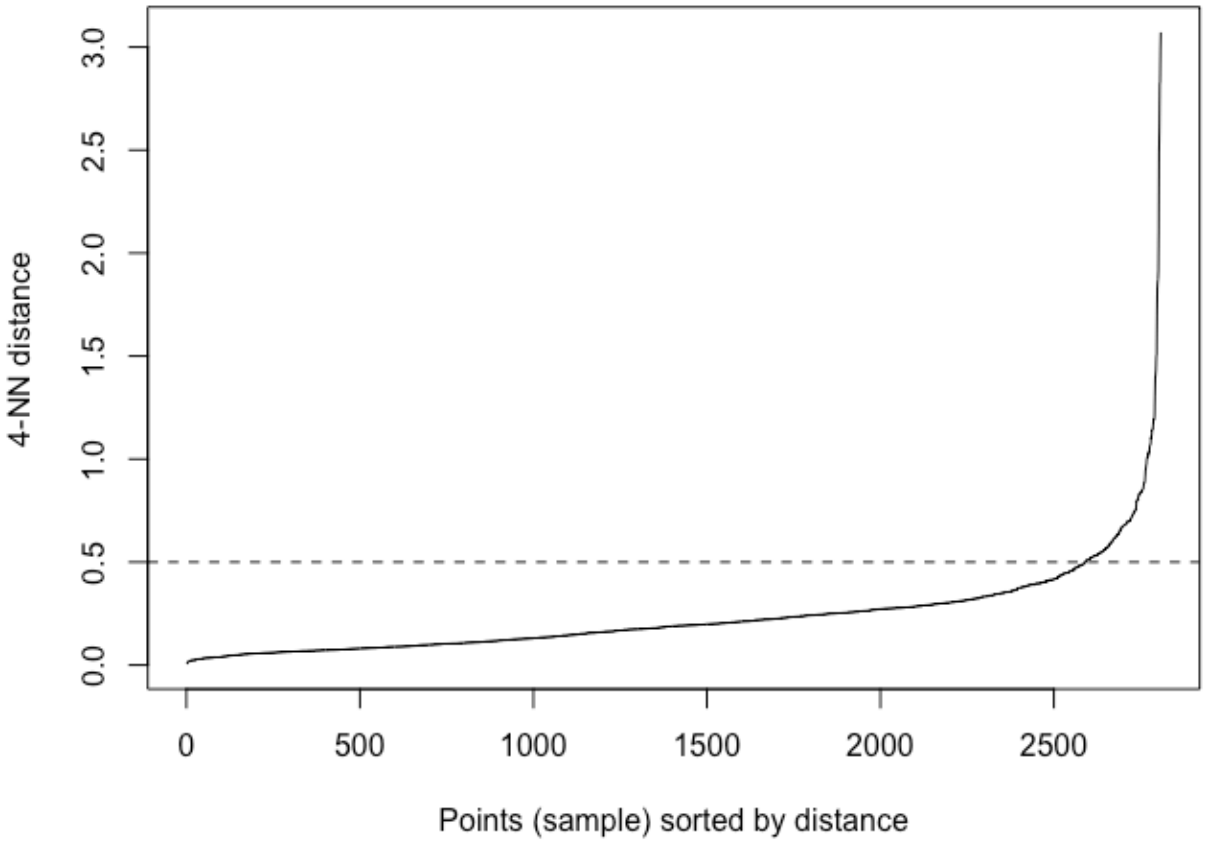


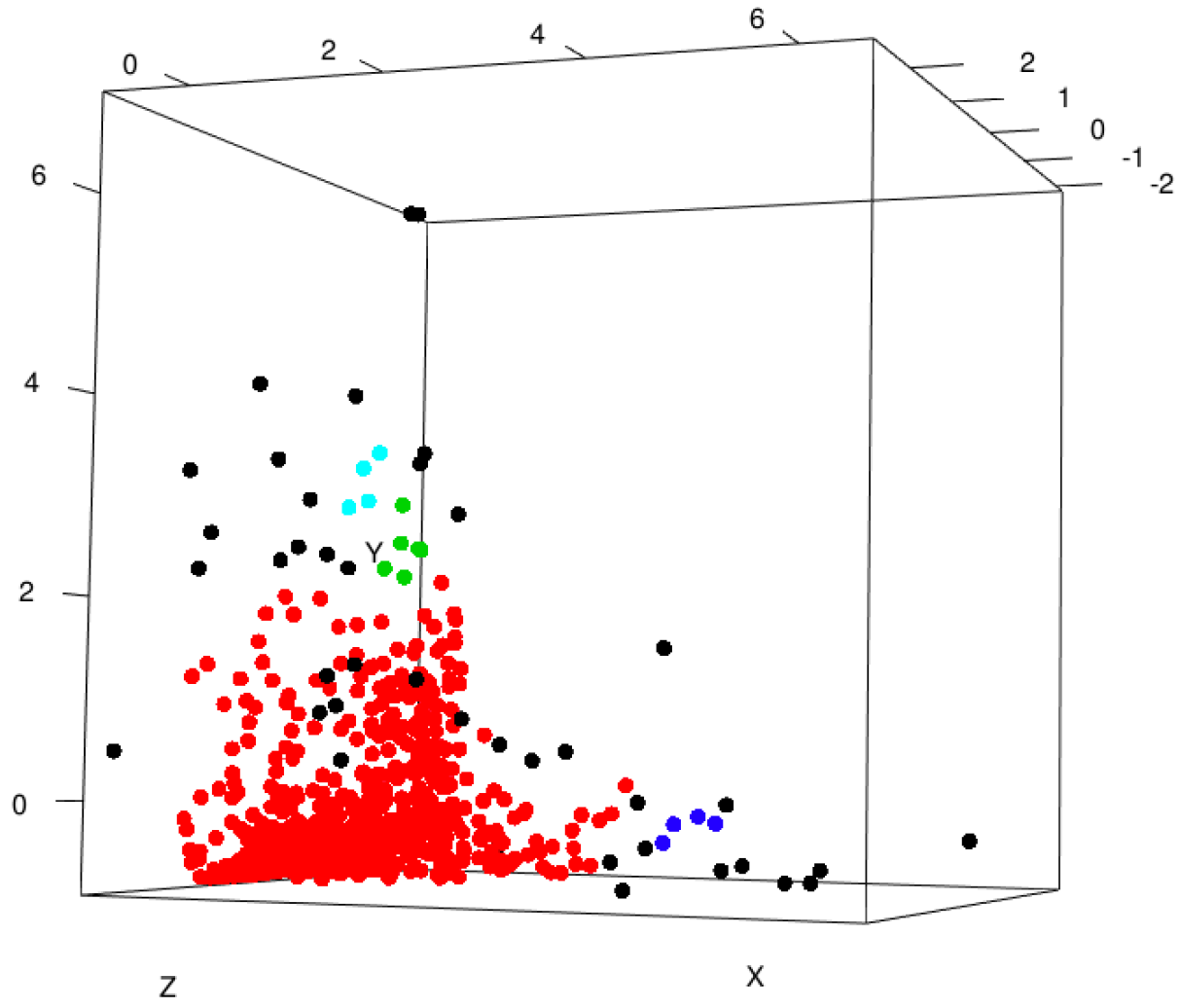
And the 3D plot shows below



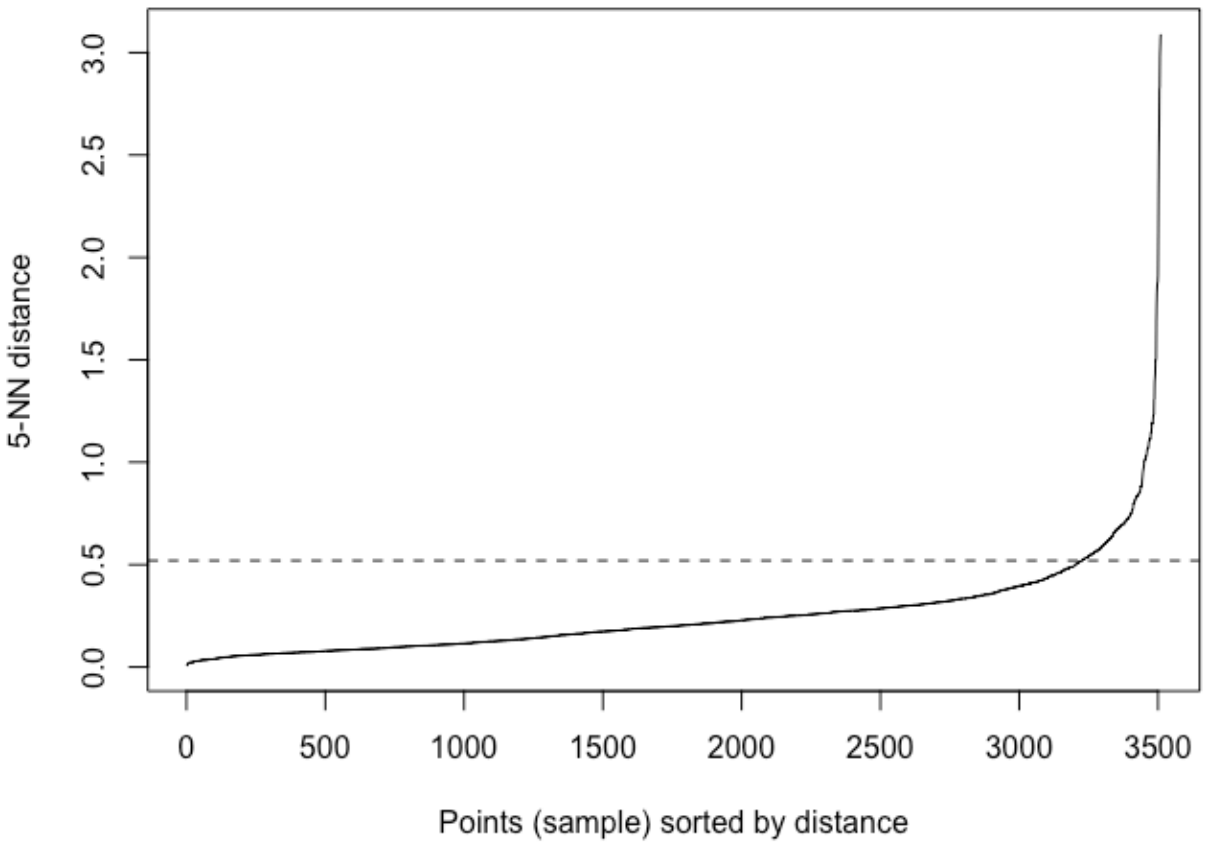
3.2

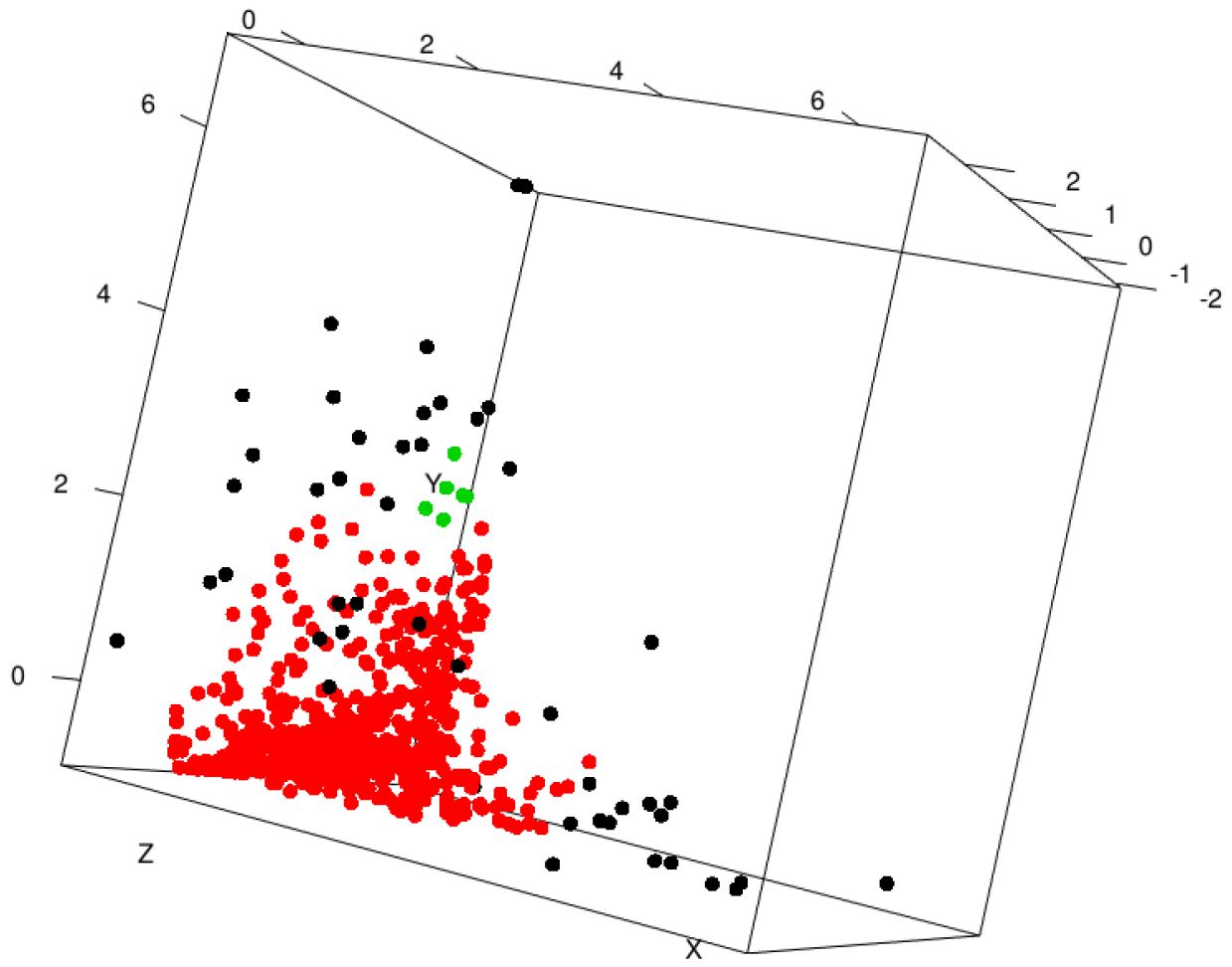
we vary minPts in the code above and gets the result when k = 4


4-NN distance
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
500
1000
1500
2000
2500
Points (sample) sorted by distance



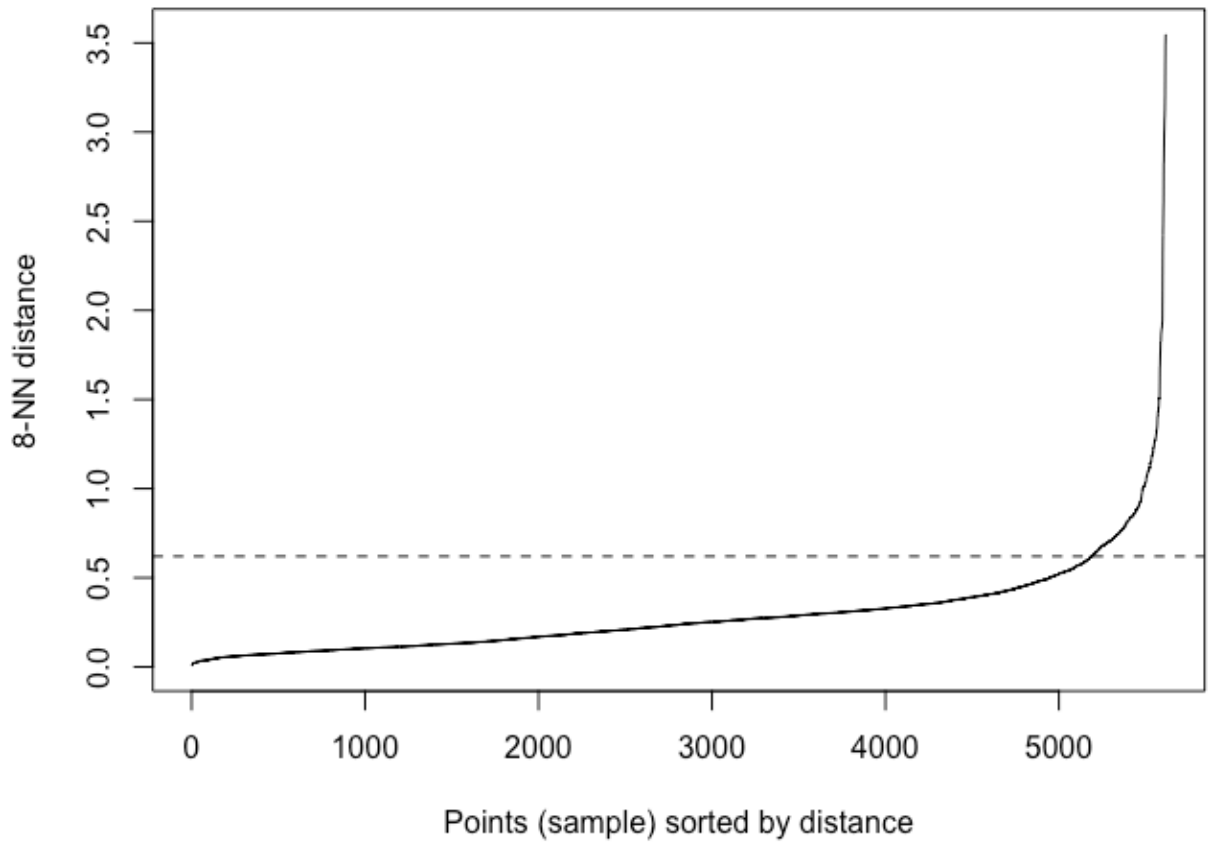
when k = 5;

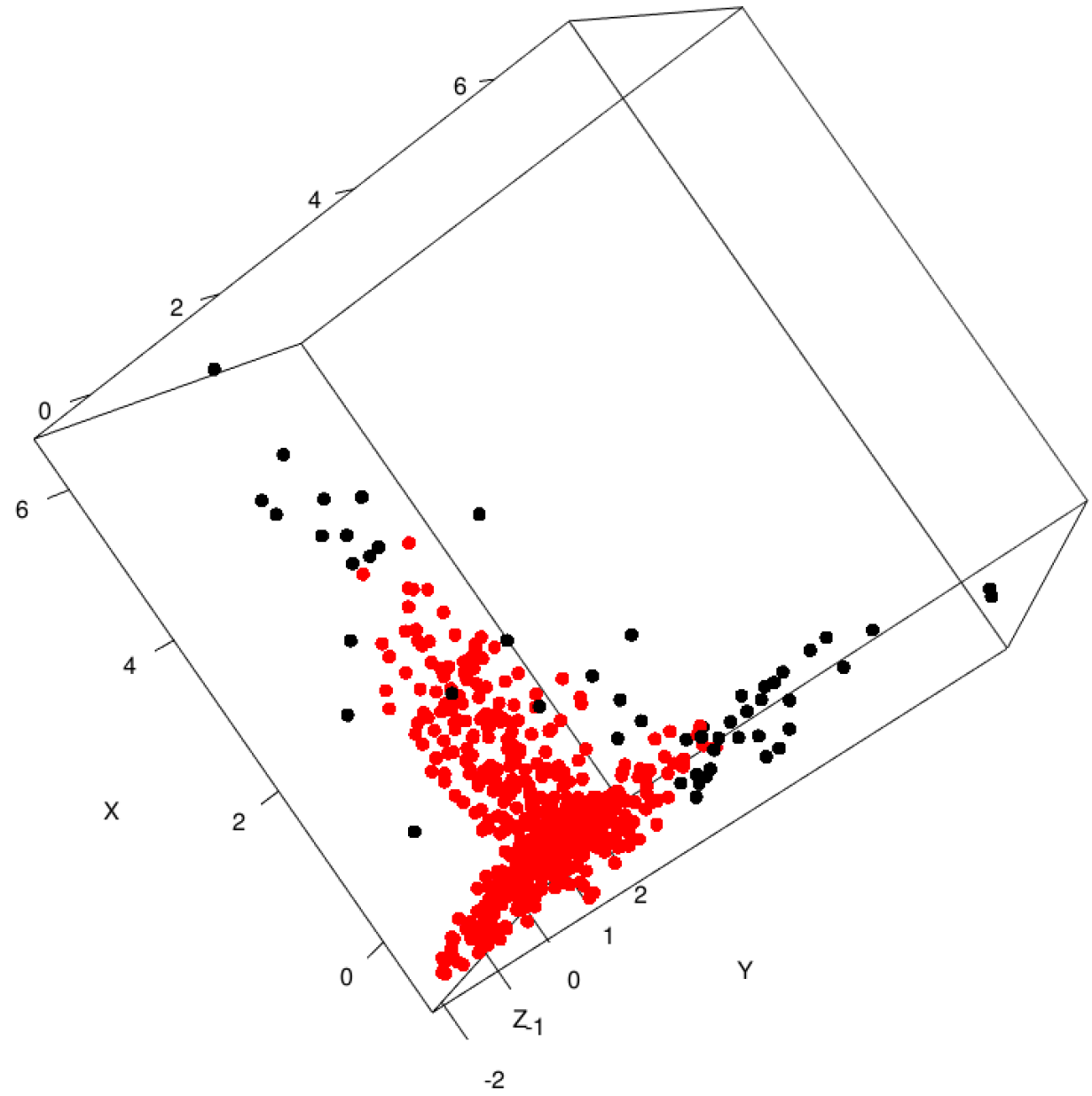

5-NN distance
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
500
1000
1500
2000
2500
3000
3500
Points (sample) sorted by distance



It seems the 3D results not so good. we do extra tests.

for example when k = 8, shows belows is better than above.


8-NN distance
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
1000
2000
3000
4000
5000
Points (sample) sorted by distance



3.3

From the finite minPts and ϵ we vary above. we select minPts = 8 and ϵ = 0.6 to be best clustering we use latter. Other varying combination was saved in appendix of this pdf

Task 4: Result discuss

In this project, we use hierarchical , k-means, and DBSCAN clustering methods to cluster the given data set. And we use elbow method and silhouette scores approaches to determine the optimal number of clusters. And use dbscan package to find the best epsilon for given minPts. From the result, we see that in hierarchical clustering, it suggests that k = 2 is optimal. In k-means, it suggests k = 4 is optimal. In DBSCAN, we test various conditions can choose minPts 8 and epsilon 0.6.

In this section, we can do cluster validatio to evaluate the goodness of clustering algorithm results. we'll describe the two commonly used indices for assessing the goodness of clustering: the silhouette width and the Dunn index. The latter one is refered from https://www.datanovia.com/en/lessons/cluster-validation-statistics-must-know-methods/.

We first analyse Dunn index.

> If the data set contains compact and well-separated clusters, the diameter of the clusters is expected to be small and the distance between the clusters is expected to be large. Thus, Dunn index should be maximized.

we can use `cluster.stats` to check it. we use it to check the three models

```
cluster.stats(df.dis,cut2)$dunn
cluster.stats(df.dis,km.res$cluster)$dunn
cluster.stats(df.dis,db$cluster+1)$dunn
```

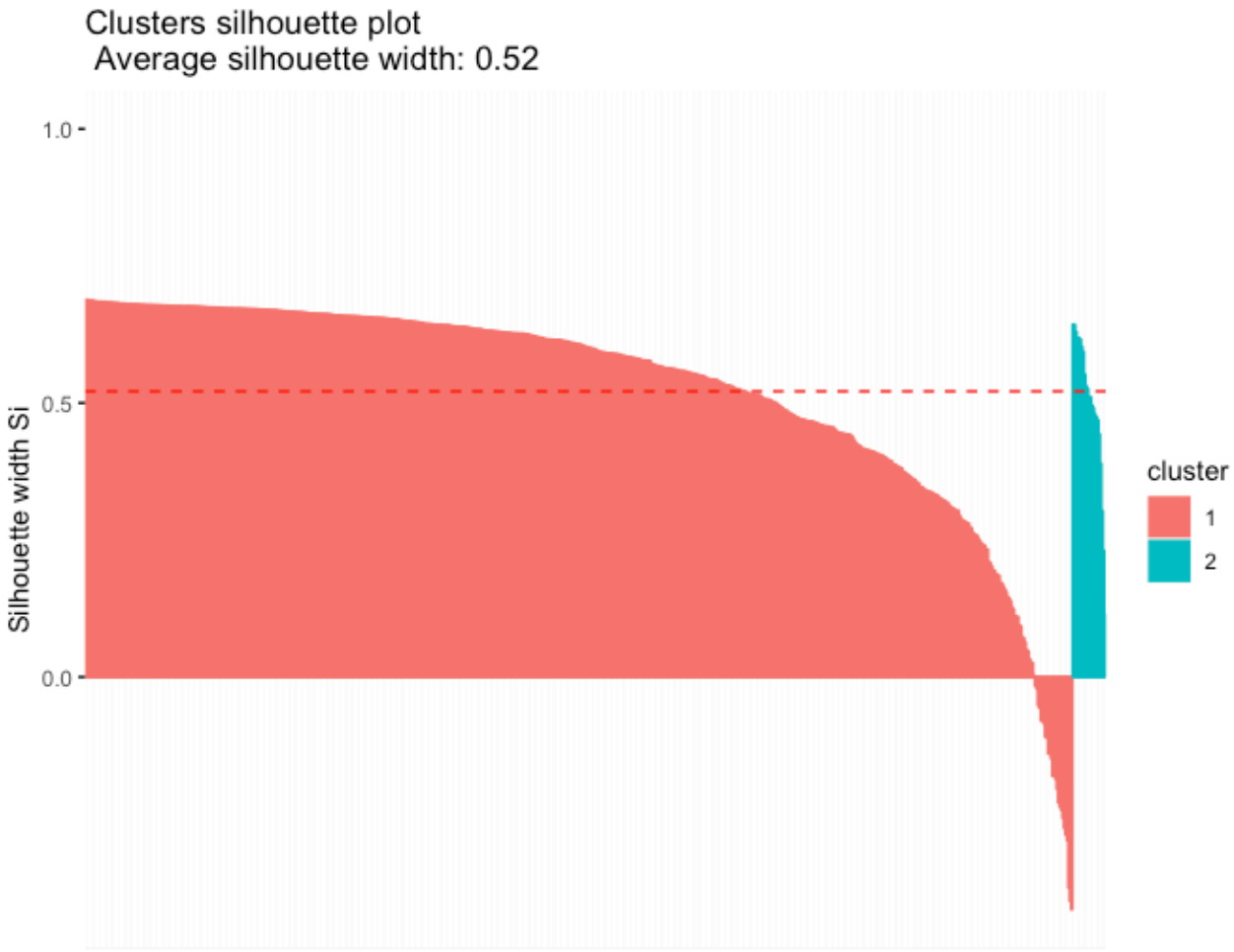
the results show below. We knows that the hierarchical cluster is best for this data set.

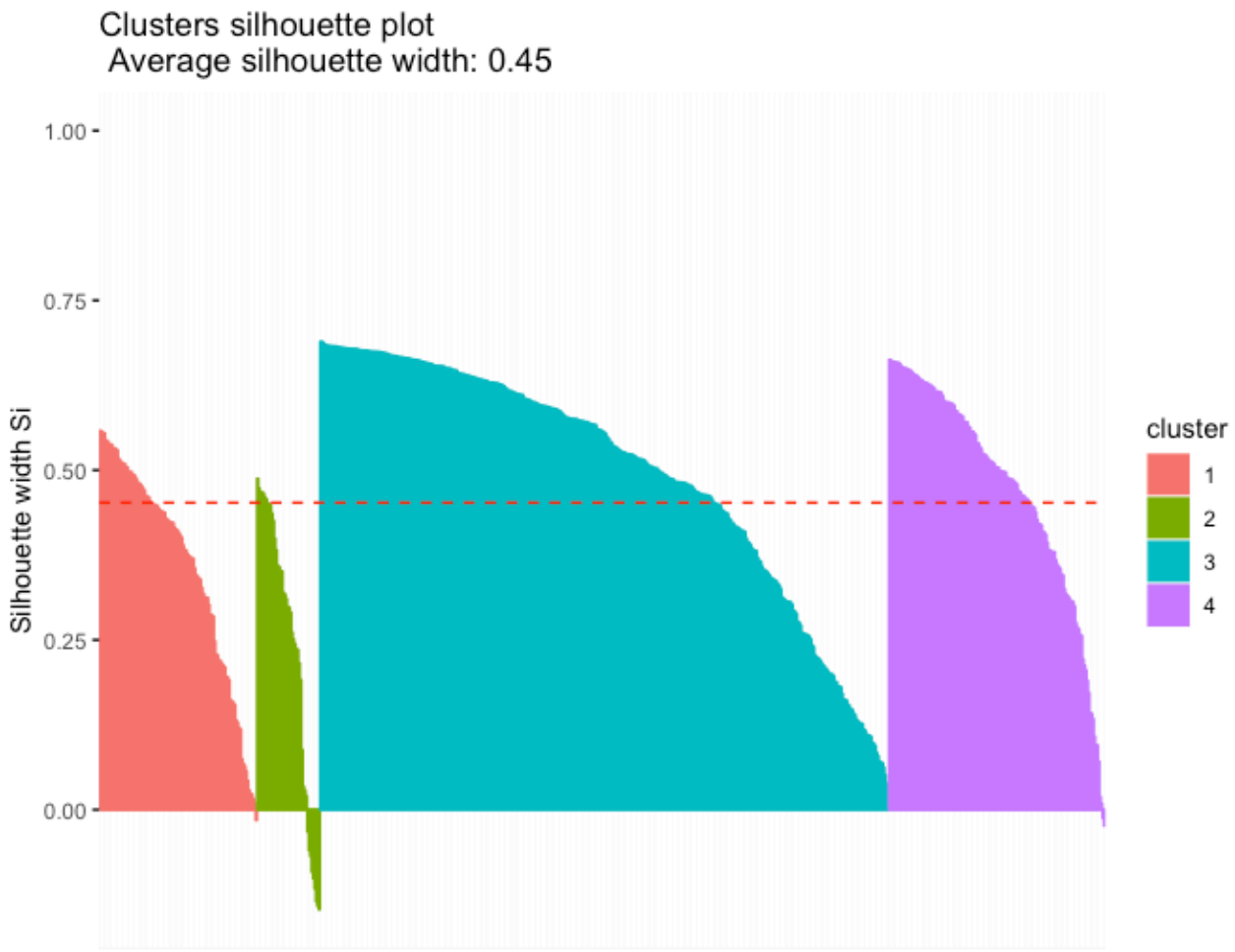
```
> cluster.stats(df.dis,cut2)$dunn
[1] 0.05932289
> cluster.stats(df.dis,km.res$cluster)$dunn
[1] 0.02226372
> cluster.stats(df.dis,db$cluster+1)$dunn
[1] 0.02272832
>
```

Then, we use silhouette width to do the validation. We use the function `eclust()` to provide silhouette informatio. Notes that DBSCAN is not applied in this function. so we check the result of hierarchical and k-means models.

```
hc.res <- eclust(df, "hclust", k = 2, hc_method = "complete", graph = FALSE)
fviz_silhouette(hc.res)
kmm.res <- eclust(df, "kmeans", k = 4, graph = FALSE)
fviz_silhouette(kmm.res)
```

As the result shows , the average silhouette width is 0.52 , larger than 0.45 in k-means method. consistent with the conclusion we use Dunn index above. So the best clustering of the dataset is hierarchical cluster where k = 2.


Clusters silhouette plot
Average silhouette width: 0.52
Silhouette width Si
1.0
0.5
0.0
cluster
1
2



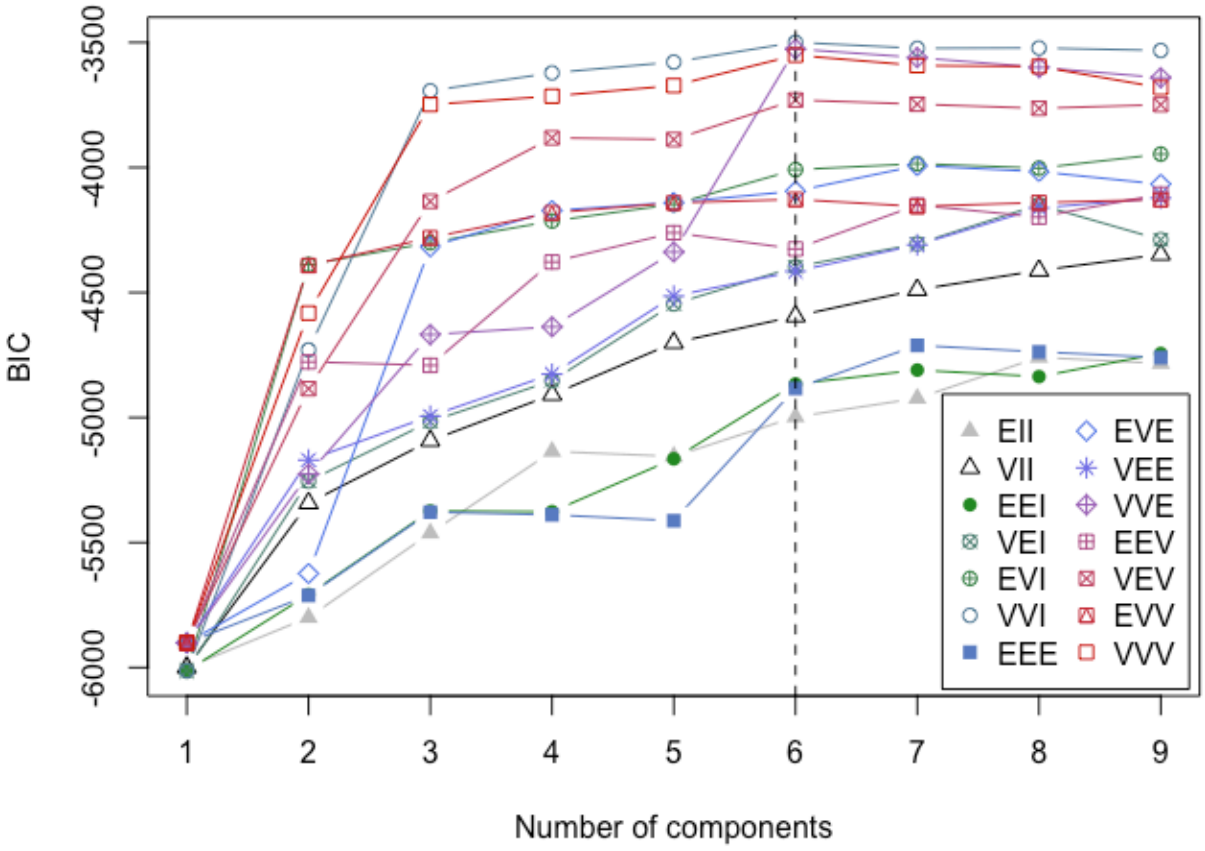
Extra Credit Task

1

The *Mclust* package uses maximum likelihood to fit , with different covariance matrix parameterizations, for a range of k components. The best model is selected using the Bayesian Information Criterion or *BIC*. A large BIC score indicates strong evidence for the corresponding model. So we can `Mclust` to do it and plot the BIC vs k, and selected the k with BIC largest

```
mod = Mclust(df)
plot(mod, what = "BIC", ylim = range(mod$BIC[,-(1:2)], na.rm =
TRUE),legendArgs = list(x = "bottomright"))
abline(v = 6, lty = 2)
```

The results show below , So we select k =6



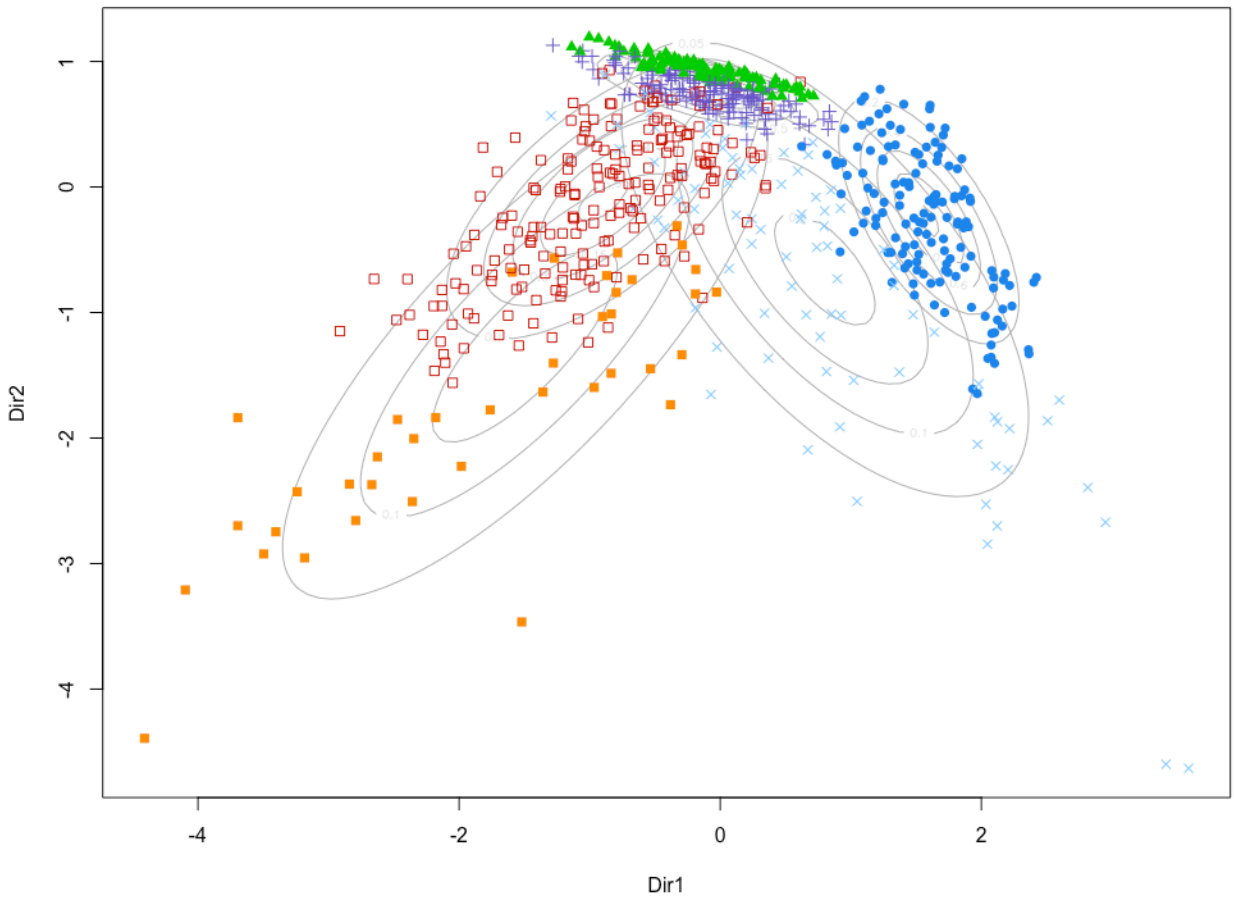
We can `summary(mod)` to get the information of it . From the data below, it can be seen that it selected a model with 6 components. The optimal selected model name is VVI model.

```
> summary(mod)
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust VVI (diagonal, varying volume and shape) model with 6 components:

 log.likelihood   n df       BIC       ICL
      -1615.336 702 41 -3499.383 -3725.036

Clustering table:
  1   2   3   4   5   6
139 197  99 144  39  84
> mod$modelName
[1] "VVI"
> mod$G
[1] 6
>
```

2

We use `MclustDR()` to do the data onto a suitable dimension reduction subspace. and use contour plot to do projection subspace .

```
drmod = MclustDR(mod,lambda = 1)
plot(drmod, what = "contour")
```



3

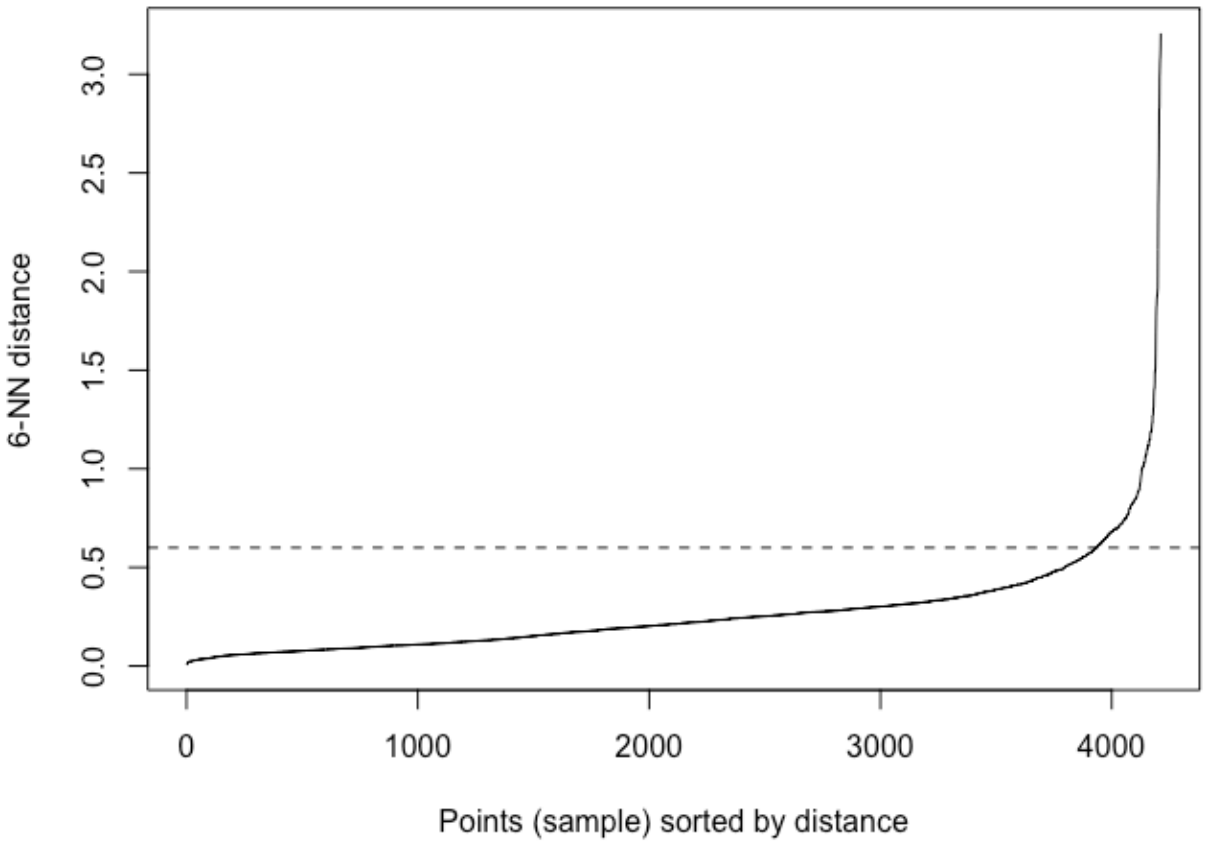
In this section. We learn to use Gaussian decomposition method to cluster the dataset. In this method we assume that the data comes from a mixture of k normal (Gaussian) distributions. **mclust** is a powerful package for this cluster method. We vary k and select the best value that has the highest likelihood. The Mclust package also provide fucntion using maximum likelihood to fit all these models. The best model is selected using the Bayesian Information Criterion or BIC. We find in this case we select k = 6. To visualise the clustering structure and the geometric characteristics, we project the data onto a suitable dimension reduction subspace and mclust provide `` `MclustDR` `` for us to use. To compare this method with the above three methods. We also use the same validation in Task 4 to calculate the Dunn index. As shown below, the dunn is 0.003518739, the smallest among these four model results. So it not the least model we would choose for this dataset.

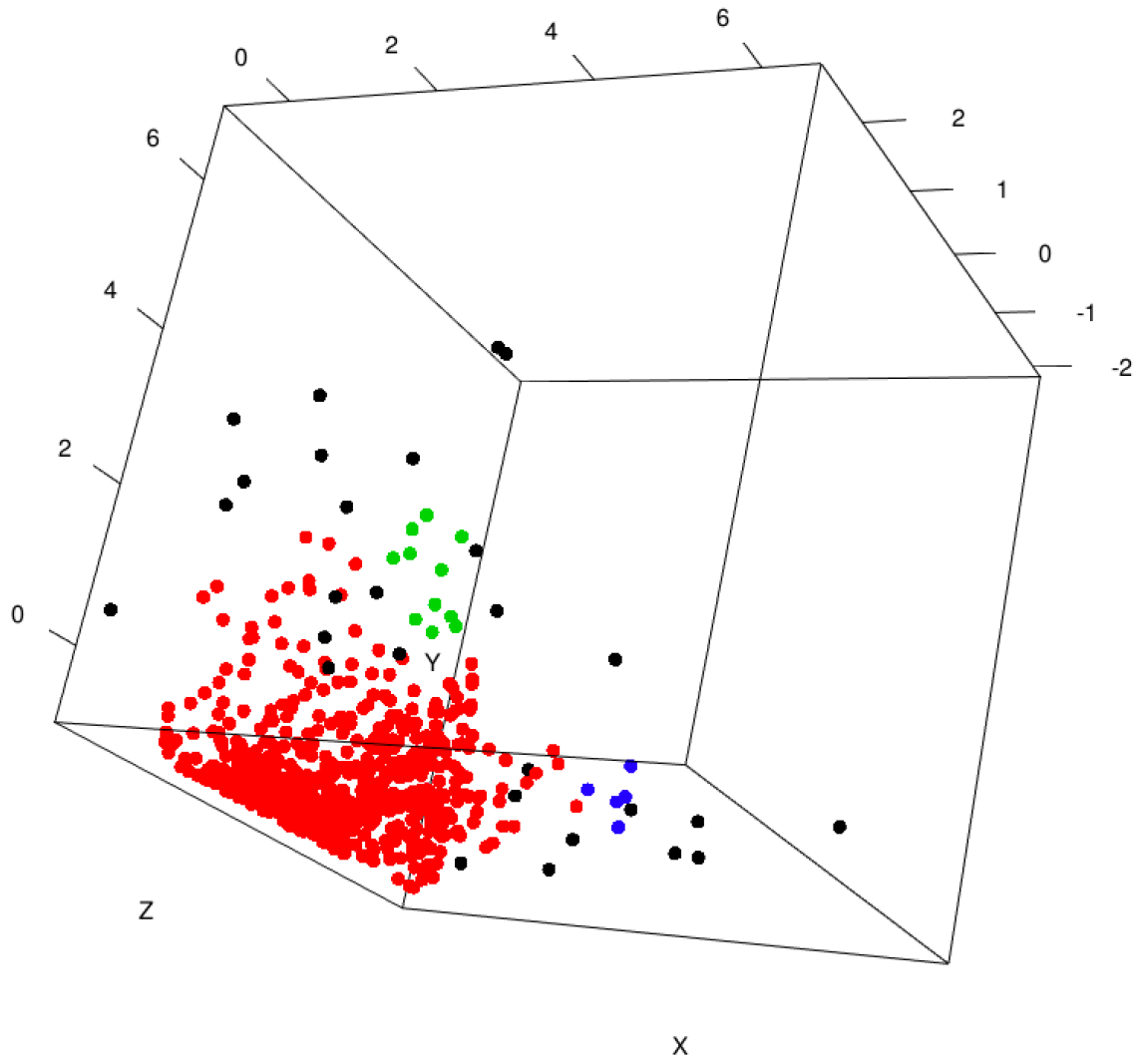
```
cluster.stats(df.dis,mod$classification)$dunn
[1] 0.003518739
```

Appendix for Task 3.3

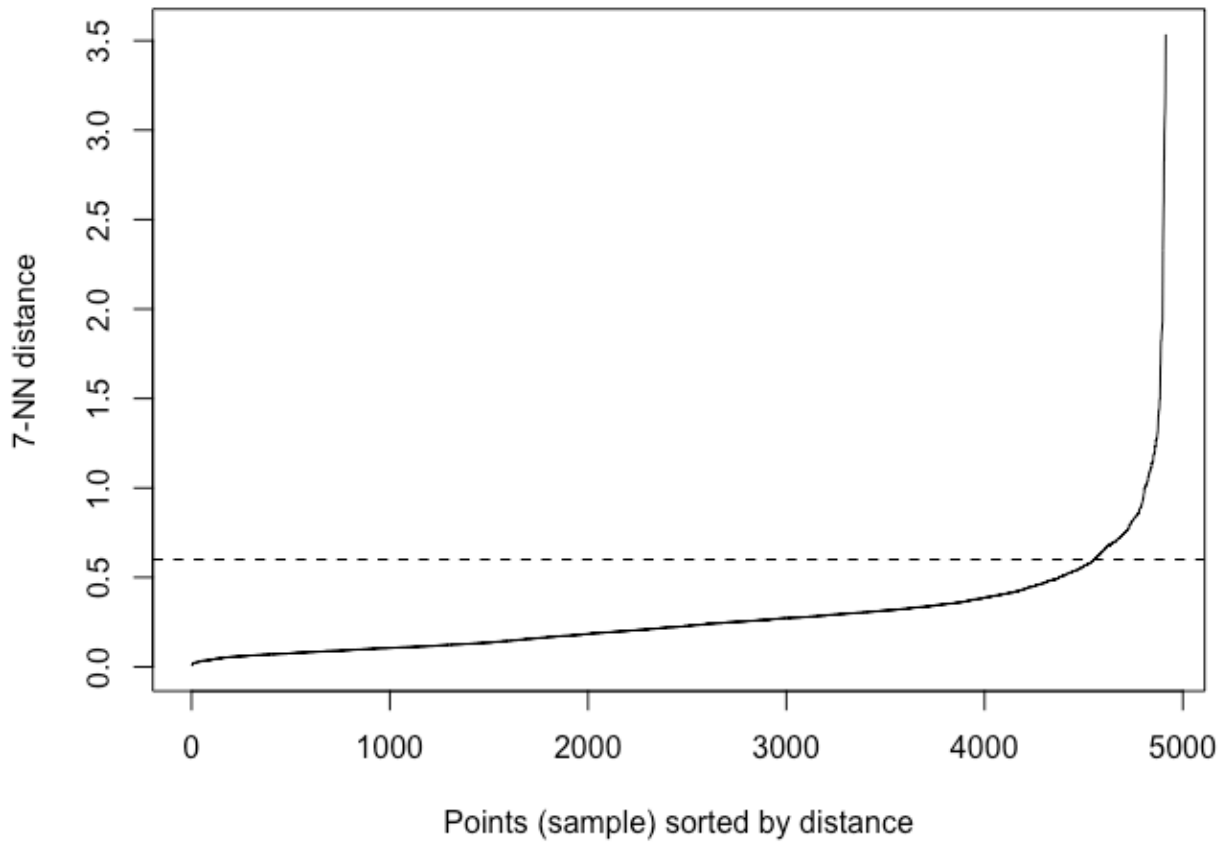
Besides the minpts = 3, 4,5,15 we analyse in Task 3.3, we also repeat for minpts = 6,7,8,

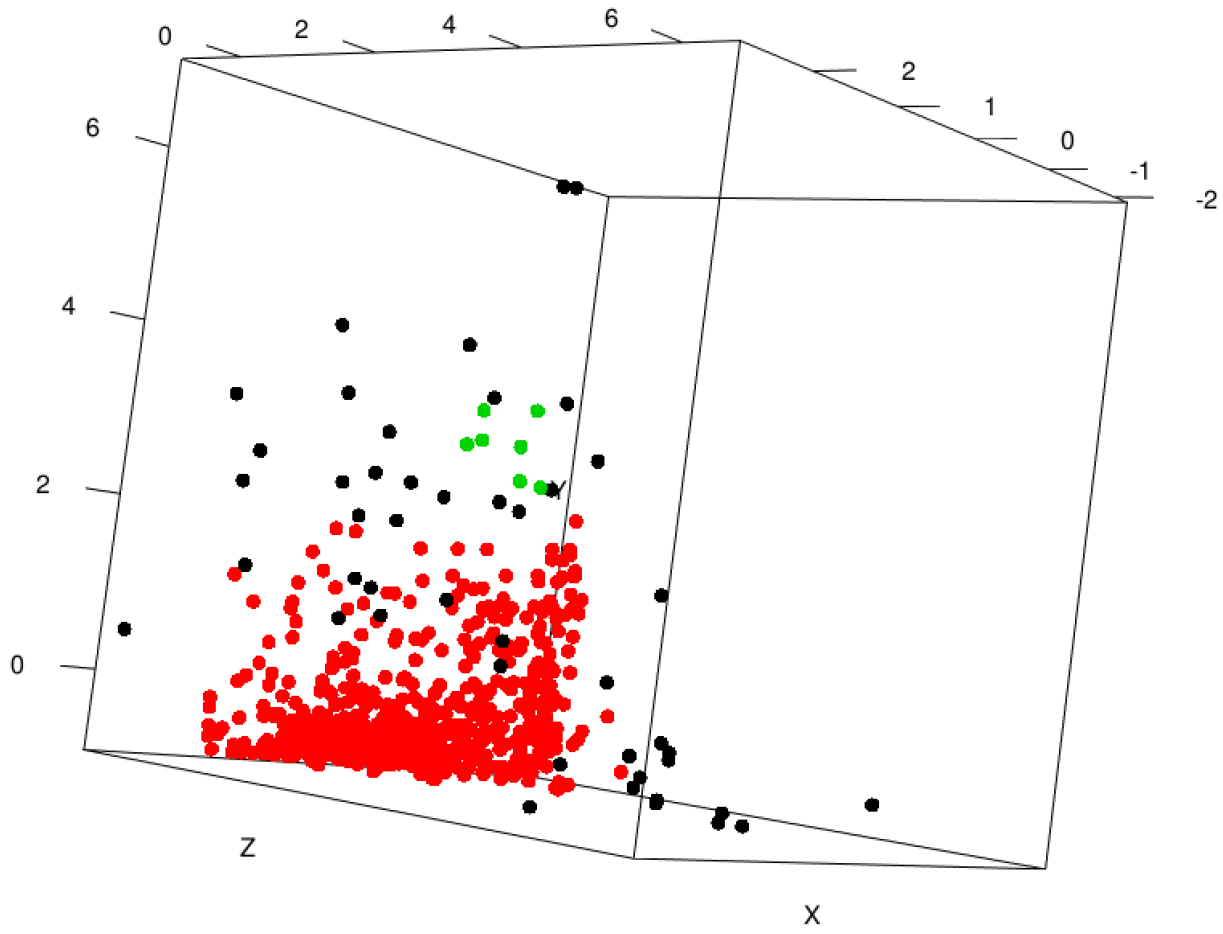
- when k = 6;



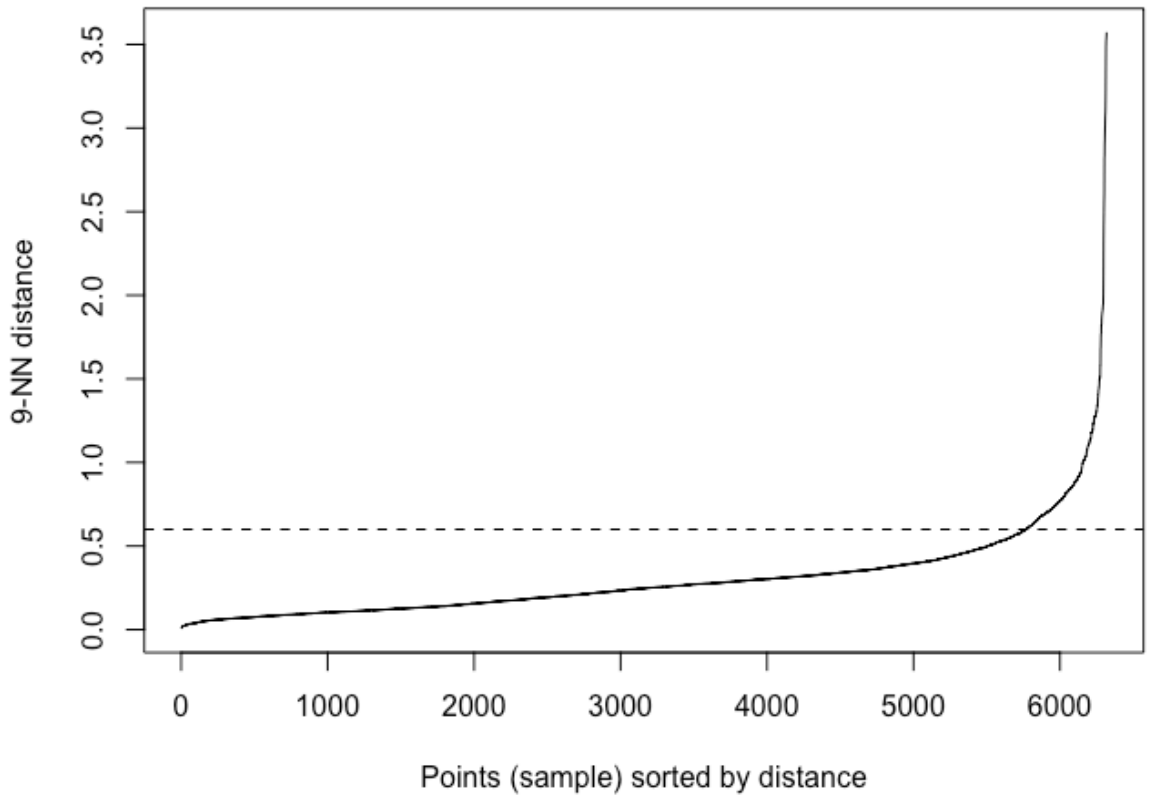


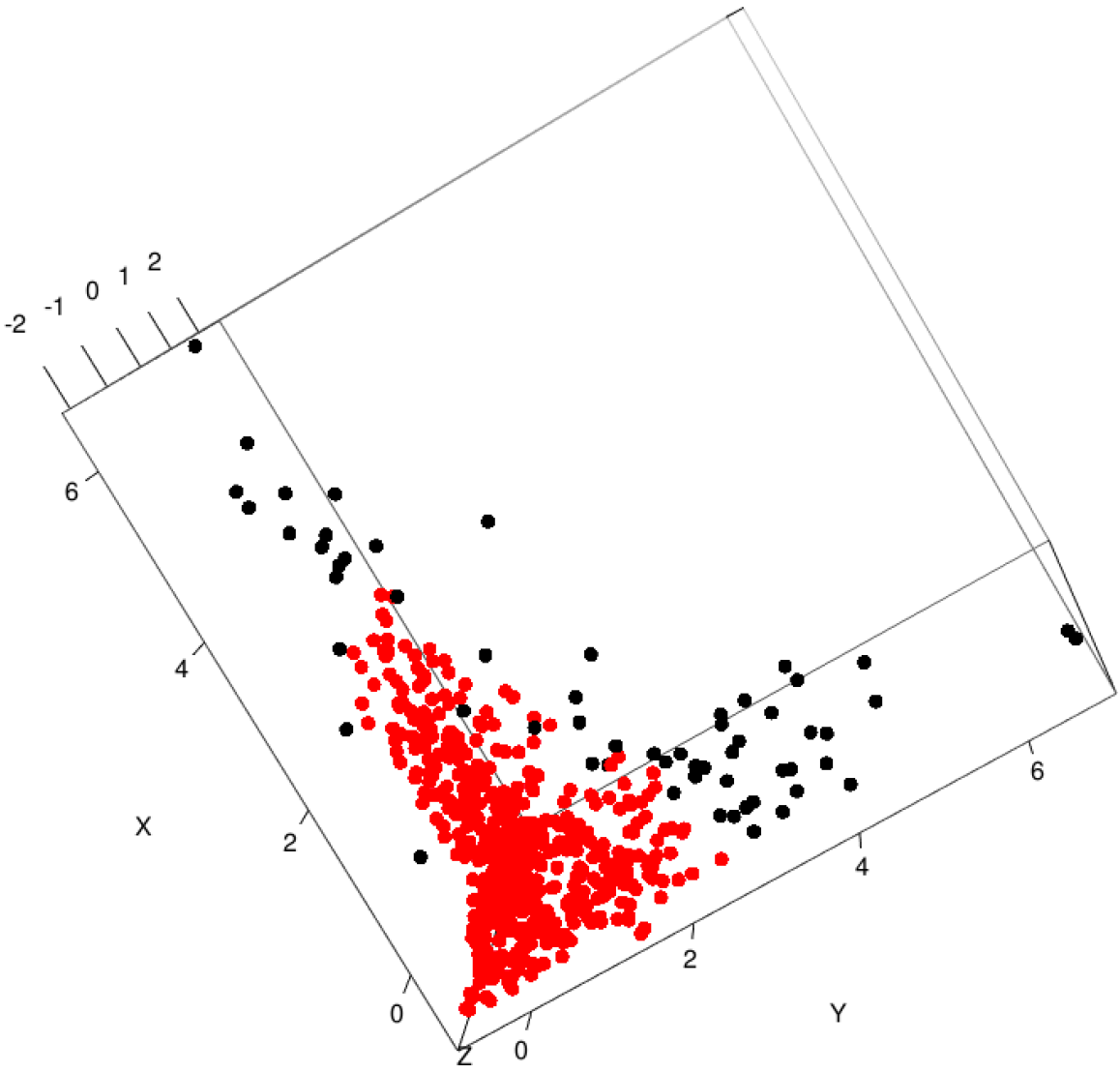
- when k = 7


7-NN distance
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
1000
2000
3000
4000
5000
Points (sample) sorted by distance



- when k = 9


9-NN distance
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
1000
2000
3000
4000
5000
6000
Points (sample) sorted by distance


-2
-1
0
1
2
6
4
2
0
X
Z
0
2
4
6
Y