Project Report

On Multivariate: Cluster Analysis

Submitted by

MR. Chirat	Suwannachote	665020010-1
Miss. Wanissara	Chongchai	665020059-1
MR. Sitthatka	Jarutsang	665020060-6

Present to

Dr.Prem Junsawang

This report is a part of the Multivariate Analysis Subject

Semester 1 2023

Khon Kean University

introduction

This report is part of the course SC637703 Multivariate Analysis to gain knowledge about data preparation using the Principal Component Analysis method along with Clustering under public data and to study it to benefit future learning.

The creator hopes that this report will be of benefit to readers. If there are any suggestions or errors, we apologize here.

Research team

Table of Contents.

	Page
Introduction of Clustering	1
Difference between Clustering and Classification	1
Clustering Analysis	1
Classification Analysis	2
Types of data and measures of distance	2
Types of data	2
measures of distance	3
K-Means Clustering (Non-hierarchical Method)	4
What is K-Means Clustering	4
Theory of K-Means	4
Hierarchical Clustering	5
What is Hierarchical Clustering	5
Theory of Hierarchical Clustering	6
DBScan (Density-Based Spatial Clustering of Applications with Noise)	7
What is DBScan	7
Theory of DBScan	7
Test in R	8
Data preparation	8
Clustering With Euclidean distance	12
Clustering With Manhattan distance	25
Reference	39

Cluster Analysis

1. Introduction of Cluster Analysis

Clustering is the process of grouping similar data together based on similar characteristics or properties. By making data within the same group more similar and data between groups more different. Using this grouping allows us to understand the structure and behavior of data more easily without having to know in advance the groups or the rules used to group them. This is a useful process in many tasks and brands such as business data analysis, data discovery, object classification, image classification, and customer surveys.

Clustering can be done using methods such as K-Means Clustering, Hierarchical Clustering, DBSCAN, and there are many ways to evaluate the cost-effectiveness of clustering.

Clustering is a useful tool for researching and making sense of highly dimensional or complex data. It helps us find relationships and sequences in data that may not be directly formal and is an important part of Data analysis and decision-making span many disciplines. However, the selection and evaluation of clustering methods is important to obtain quality results and use them in understanding and decision-making.

2. Difference between Clustering and Classification

Clustering and Classification have different goals. It can be explained as follows.

2.1 Clustering Analysis

Clustering is a process used to group similar data together without prior knowledge of the groups or grouping rules. Clustering is often used to discover hidden structures in data or to create a new type or group of data that has no history in the data may be called "unsupervised learning".

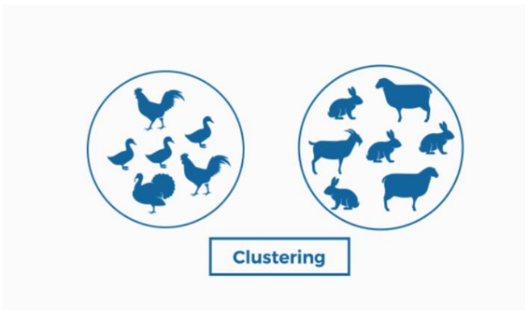


Figure 1 Clustering.

2.2 Classification Analysis

Classification It is a process used to assign categories or classes to data based on learning from training data with answers or labels given in each data sample.

Classification is often used in applications where data is needed to make predictions or identify the correct answer to new data. By using models learned from training data. Examples of classification methods are Support Vector Machines (SVM), Decision Trees, and Neural Networks.

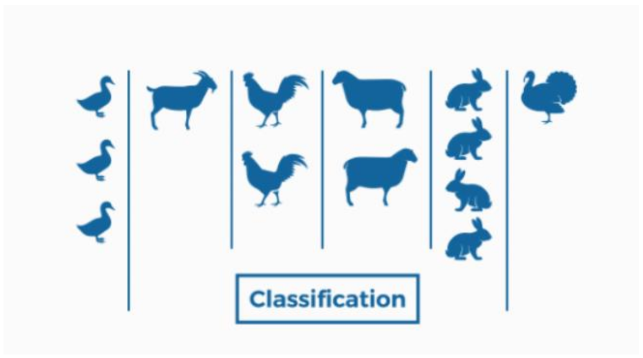


Figure 2 Classification.

It should be noted that Clustering is often used to explore data structures or discover unknown data in advance, while Classification It is often used to predict or classify data into different classes or categories that are known in advance from training data. The choice of these processes depends on the purpose and nature of the data we want to deal with.

3 Types of data and measures of distance

3.1 Types of data

The data used in cluster analysis can be interval, ordinal or categorical. However, the most common type of data is Interval data because it is possible to find distances for grouping. However, we can use non-interval data, but it is very complicated. and may not be able to find the distance information.

3.2 measures of distance

Finding the distance between the data can be found in many ways, including Manhattan distance or Euclidean distance. We would like to present a method for finding the distance of data using Euclidean distance and Manhattan distance.

3.2.1 Euclidean distance is a method of calculating the distance between two points in a dimensional space using the decimal of a square triangle from the decimal of the dimensional difference between the points. which is like Pythagoras. Using Euclidean distance is a common way to measure distance in various situations.

3.2.2 The Manhattan distance is a method of calculating the distance between two points in Euclidean space. This distance calculation is called "Manhattan" because of the method of walking in New York (New York City) that must follow a grid of streets or a grid. It's called Manhattan walking because it follows vertical and horizontal streets. Without turning your temples down or right when walking on a map of New York City.

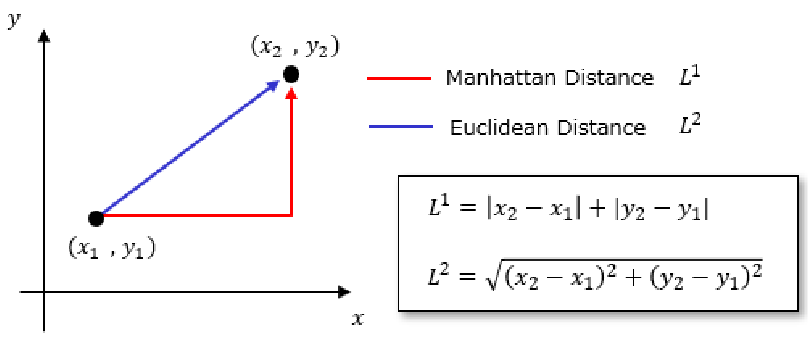


Figure 3 Different between Euclidean distance and Manhattan distance.

4 K-Means Clustering (Non-hierarchical Method)

4.1 What is K-Means Clustering

K-Means is a method of grouping data in grouping (Clustering) that uses computation and analysis to divide data into groups or clusters that are similar together.

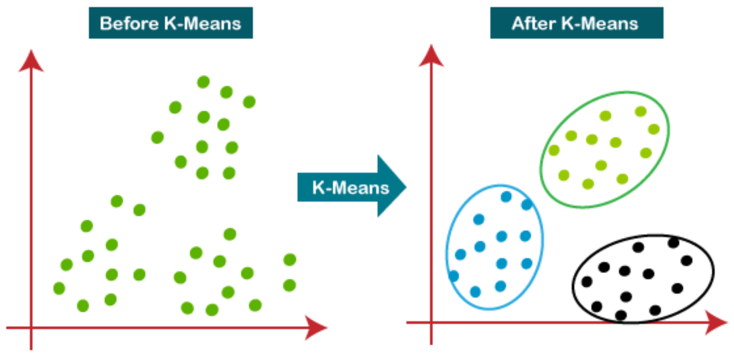


Figure 4 Before and After K-means.

4.2 Theory of K-Means

- Selecting the number of clusters (K):

  The first step is to select the number of clusters (K) where we want to divide the data into K different clusters. Choosing a K value is an important decision. Because it affects the results of grouping. To select groups (K), we recommend the Elbow Method. The Elbow method uses the SSE (Sum of Squared Errors) of each K value to find the appropriate K value. The best method is to find the K value that gives SSE. rapid decline and after that it slowly decreased.

- Starting at a random point:

  In this step, we randomly select K starting points (centroids) from the data, each of which is the centroid of each cluster.

- Dividing data in a cluster:

  Each data point is assigned to the cluster with the closest centroid by calculating the distance between the data point and each centroid. and make it a member of the nearest cluster.

- Calculating the new centroid:

After dividing the data into each cluster, we calculate the new centroid of each cluster by taking the new centroid from all the data in that cluster, counting the new centroid as the cluster centroid.

- Looping:

Steps 3 and 4 are repeated until the cluster centroid does not change further, or the specified stopping condition is reached.

- Termination:

Once the centroid calculation and clustering is complete, we will have K clusters that divide the data into groups based on their similarity.

5 Hierarchical Clustering

5.1 What is Hierarchical Clustering

Hierarchical Clustering is a method of clustering data that allows data to be arranged into an original mean sequential structure and broken down into deeper levels of clusters.

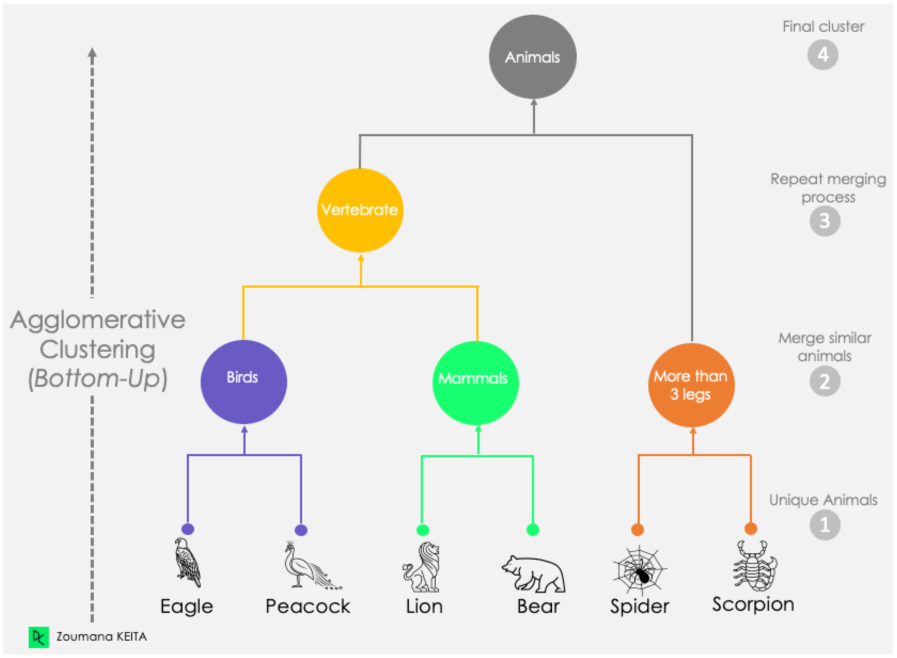


Figure 5 a dendrogram of Hierarchical Clustering.

5.2 Theory of Hierarchical Clustering

In hierarchical clustering, Objects are categorized into a hierarchy like a tree-shaped structure which is used to interpret hierarchical clustering models. The algorithm is as follows:

- Initialization:

  Start by giving each data a separate cluster. That means each data will have its own cluster at startup.

- Calculation of distance:

  Calculate the distance between all clusters. This is used to calculate distances by the Euclidean distance or Manhattan distance between the centroids of each cluster.

- Cluster Agglomeration:

  Clusters with the least distance between centroids are combined to form a new cluster. and will create a new sequence structure. This new cluster will be a child of the two old clusters.

- Repetition:

  The process of calculating distance and merging clusters is repeated until a single root cluster remains and we obtain a sequence structure of clusters that indicates the similarity of all data in the initial data.

- Creating a grouping table (Dendrogram):

  As we calculate and combine clusters throughout the iterative process, We can create a clustering table or dendrogram that shows the sequence structure of the clusters based on their degree of similarity. This helps in analysis and decision making in data grouping.

6 DBScan (Density-Based Spatial Clustering of Applications with Noise)

6.1 What is DBScan.

Using DBScan (Density-Based Spatial Clustering of Applications with Noise) is a data clustering method that focuses on the density of data in a space and can group high-density data together.

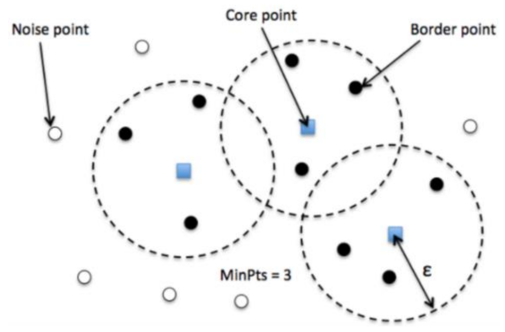


Figure 6 DBScan.

6.2 Theory of DBScan.

In DBScan clustering, dependence on distance-curve of dimensionality is more. The algorithm is as follows:

- Defining parameters:

  Before using DBScan, we need to define two important parameters:

  Radius (Epsilon, ε): The radius of the circle about each data point. It is a parameter that determines the maximum distance between points in a cluster.

  Minimum amount of data in a cluster (MinPts): The minimum amount of data that must be within the radius **ε** for it to be considered a dense or similarity point.

- Centroid:

  To create a centroid, it randomly form all datapoint that you use.

- Default configuration:

  Begin by selecting a data point and checking whether this data point can be connected to any data points in radius **ε**. If it can be connected, these data points are grouped into the same cluster.

- Expanding the cluster:

  If a data point is added to a cluster and can be connected to another data point in radius **ε**, it is added to the same cluster.

- Creating a new cluster:

  If it cannot connect any data points in an existing cluster with a new data point in radius **ε**, a new cluster is created and starts connecting this new data point.

- Checking for too far apart data points (Noise):

  Data points that cannot be connected to any cluster within radius **ε** are considered dense points or noise and will not be clustered into any cluster.

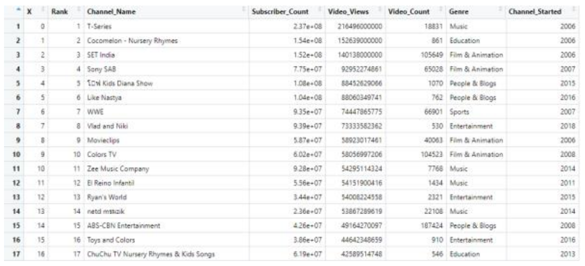
7. Test in R

To test in R, we propose three Clustering methods and different distance. Here are the steps:

1. **Data preparation**
2. **Clustering with Euclidean distance**
   - 2.1. K-Means (Euclidean distance)
   - 2.2. Hierarchical Clustering (Euclidean distance)
   - 2.3. DBSCAN (Euclidean distance)
3. **Clustering with Manhattan distance**
   - 3.1. K-Means (Manhattan distance)
   - 3.2. Hierarchical Clustering (Manhattan distance)
   - 3.3. DBSCAN (Manhattan distance)

7.1 Data preparation

To Clustering we used the dataset from Kaggle which is Top 300 YouTube Channels.

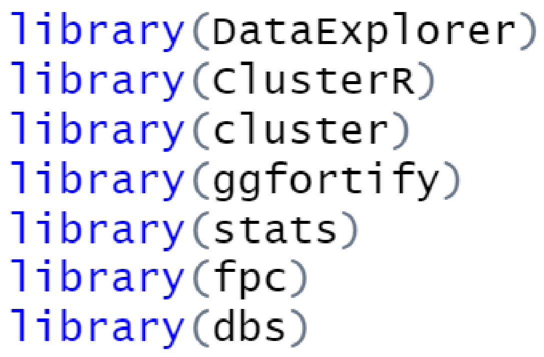


	X	Rank	Channel_Name	Subscriber_Count	Video_Views	Video_Count	Genre	Channel_Started
1	0	1	T-Series	2.37e+08	216496000000	18831	Music	2006
2	1	2	Cocomelon - Nursery Rhymes	1.54e+08	152639000000	861	Education	2006
3	2	3	SET India	1.52e+08	140138000000	105649	Film & Animation	2006
4	3	4	Sony SAB	7.75e+07	92952274861	65028	Film & Animation	2007
5	4	5	โDท Kids Diana Show	1.08e+08	88452629066	1070	People & Blogs	2015
6	5	6	Like Nastya	1.04e+08	88060349741	762	People & Blogs	2016
7	6	7	WWE	9.35e+07	74447865775	66901	Sports	2007
8	7	8	Vlad and Niki	9.39e+07	73333582362	530	Entertainment	2018
9	8	9	Movieclips	5.87e+07	58923017461	40063	Film & Animation	2006
10	9	10	Colors TV	6.02e+07	58056997206	104523	Film & Animation	2008
11	10	11	Zee Music Company	9.28e+07	54295114324	7768	Music	2014
12	11	12	El Reino Infantil	5.56e+07	54151900416	1434	Music	2011
13	12	13	Ryan's World	3.44e+07	54008224558	2321	Entertainment	2015
14	13	14	netd müzik	2.36e+07	53867289619	22108	Music	2014
15	14	15	ABS-CBN Entertainment	4.26e+07	49164270097	187424	People & Blogs	2008
16	15	16	Toys and Colors	3.86e+07	44642348659	910	Entertainment	2016
17	16	17	ChuChu TV Nursery Rhymes & Kids Songs	6.19e+07	42589514748	546	Education	2013

Figure 7 Top 300 Youtube Channel data.

We have used a total of 7 packages consisting of:

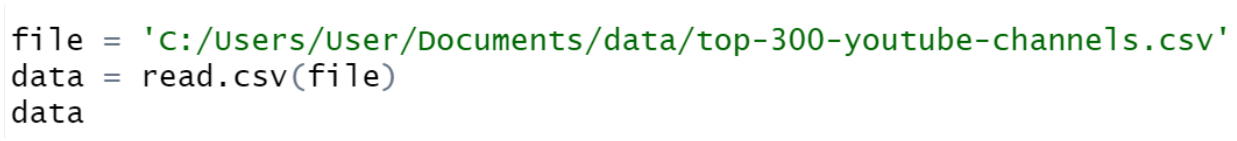
- **library(DataExplorer):** DataExplorer is an R package used for data exploration and basic data analysis. such as graphing and statistics used for creating snapshots of data. Finding lost data and analysis of relationships between variables So you can understand your data better.
- **library(ClusterR):** ClusterR is a package that focuses on cluster analysis, which is used for grouping data based on similarity. This can be used for finding structure in data. or for grouping customers or users based on similar behaviors.
- **library(cluster):** The Cluster package is a package that focuses on clustering and grouping data analysis. It includes functions for calculating distances between data points and for clustering using methods such as k-means and hierarchical clustering.
- **library(ggfortify)**: ggfortify is a package focused on working with ggplot2 to display graphs and clustering data in a graph format, so you can create beautiful graphs to show the results of clustering data.
- **library(stats):** The stats package is part of R and provides functions for calculating basic statistics, such as calculating averages. Calculating the median Calculation of standard deviation values, etc.
- **library(fpc):** The fpc package is a package focused on analyzing and evaluating the results of clustering data. This includes calculating clustering quality indicators such as Silhouette values and Dunn values.
- **library(dbs):** The dbs package is a package focused on grouped data analysis. And there is a function for calculating the distance between data points. and clustering data using methods such as k-means and hierarchical clustering. You should install the package and run it with the library.



```
library(DataExplorer)
library(ClusterR)
library(cluster)
library(ggfortify)
library(stats)
library(fpc)
library(dbs)
```

Figure 8 package to use for Test by R.

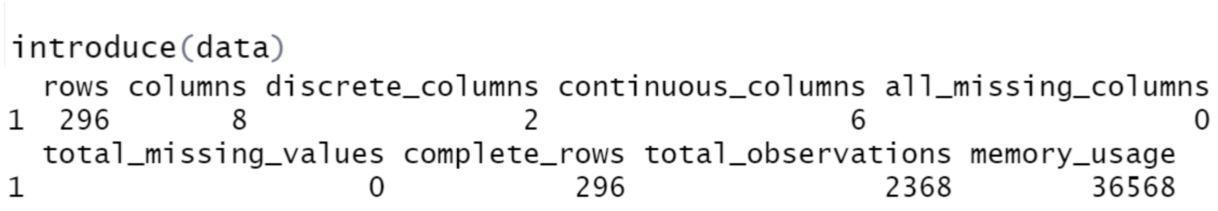
After installing and running the library, we will retrieve the data from the path that the data resides in and retrieve the data by defining the variable named data.



```
file = 'C:/Users/User/Documents/data/top-300-youtube-channels.csv'
data = read.csv(file)
data
```

Figure 9 Command for read data.

Then use the command introduce(data) for checking overall information

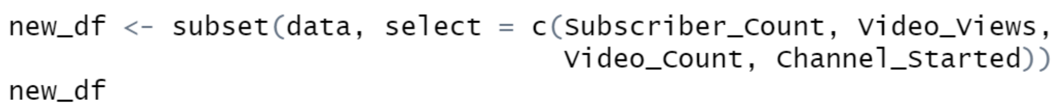


```
introduce(data)
  rows columns discrete_columns continuous_columns all_missing_columns
1  296       8                2                  6                   0
  total_missing_values complete_rows total_observations memory_usage
1                    0           296               2368        36568
```

Figure 10 Explore data with command introduce.

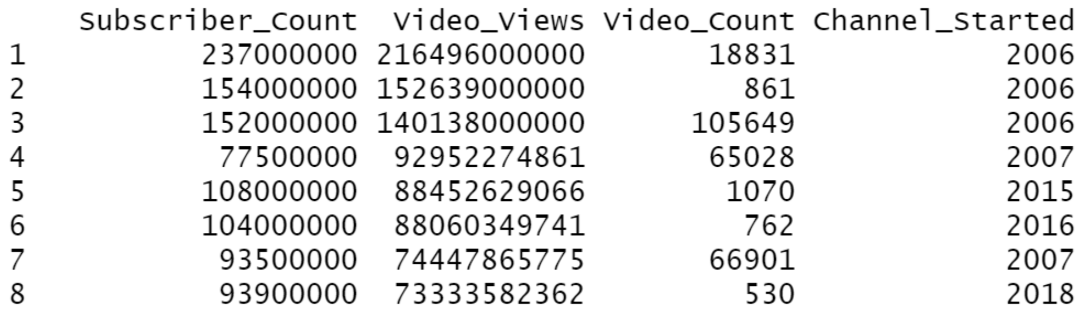
When we looked, we found a total of 296 rows and 8 columns of data. We also found that there were no missing values for this data, which allowed us to put this data to use.

We select data to be used for clustering including Subscriber_count, Video_Views, Video_count and Channel_Started with variable named new_df.



```
new_df <- subset(data, select = c(Subscriber_Count, Video_Views,
                                  Video_Count, Channel_Started))
new_df
```

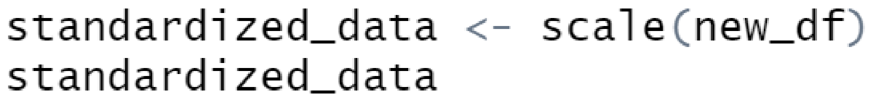
Figure 11 Set variable that interest.



```
  Subscriber_Count  Video_Views Video_Count Channel_Started
1        237000000 216496000000       18831            2006
2        154000000 152639000000         861            2006
3        152000000 140138000000      105649            2006
4         77500000  92952274861       65028            2007
5        108000000  88452629066        1070            2015
6        104000000  88060349741         762            2016
7         93500000  74447865775       66901            2007
8         93900000  73333582362         530            2018
```

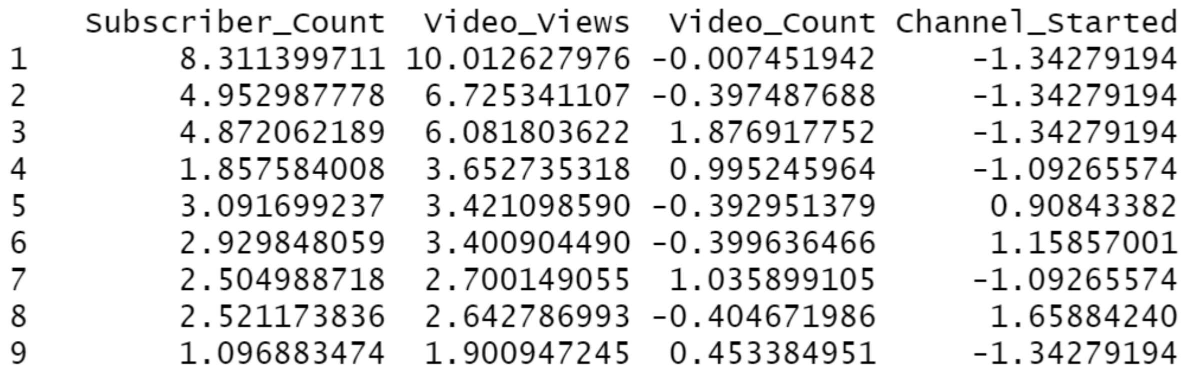
Figure 12 Eyes ball data.

Then we will standardize, which is the process of adjusting the data to a range or scale with a mean equal to 0 and a standard deviation equal to 1 so that the data are not too far apart or make any one feature stand out. Use the Scale command on the remaining data to standardize the data and store it as a variable standardized_data.



```
standardized_data <- scale(new_df)
standardized_data
```

Figure 13 Command to set standardize.



```
  Subscriber_Count  Video_Views  Video_Count Channel_Started
1      8.311399711 10.012627976 -0.007451942     -1.34279194
2      4.952987778  6.725341107 -0.397487688     -1.34279194
3      4.872062189  6.081803622  1.876917752     -1.34279194
4      1.857584008  3.652735318  0.995245964     -1.09265574
5      3.091699237  3.421098590 -0.392951379      0.90843382
6      2.929848059  3.400904490 -0.399636466      1.15857001
7      2.504988718  2.700149055  1.035899105     -1.09265574
8      2.521173836  2.642786993 -0.404671986      1.65884240
9      1.096883474  1.900947245  0.453384951     -1.34279194
```

Figure 14 Set data is variable test.

7.2 Clustering with Euclidean distance.

7.2.1 K-Means (Euclidean distance)

First step for K-means, you need to select the number of clusters (K) and We use Elbow method to find the best of K. So, we use a loop to find the best option by going from number 1 to number 20 and storing the variable in sse with a length of 20. This vector will be used to store the SSE values. for different values of K. and plot from number 1 to number 20 compared to the sse value of each item.

```
sse <- numeric(20)
for (k in 1:20) {
  kmeans_model <- kmeans(standardized_data, centers = k)
  sse[k] <- kmeans_model$tot.withinss
}

ggplot(data.frame(K = 1:20, SSE = sse), aes(x = K, y = SSE)) +
  geom_line() +
  geom_point() +
  labs(x = "จำนวนกลุ่ม (K)", y = "SSE") +
  theme_minimal()
```

Figure 15 Command looping to get best K.

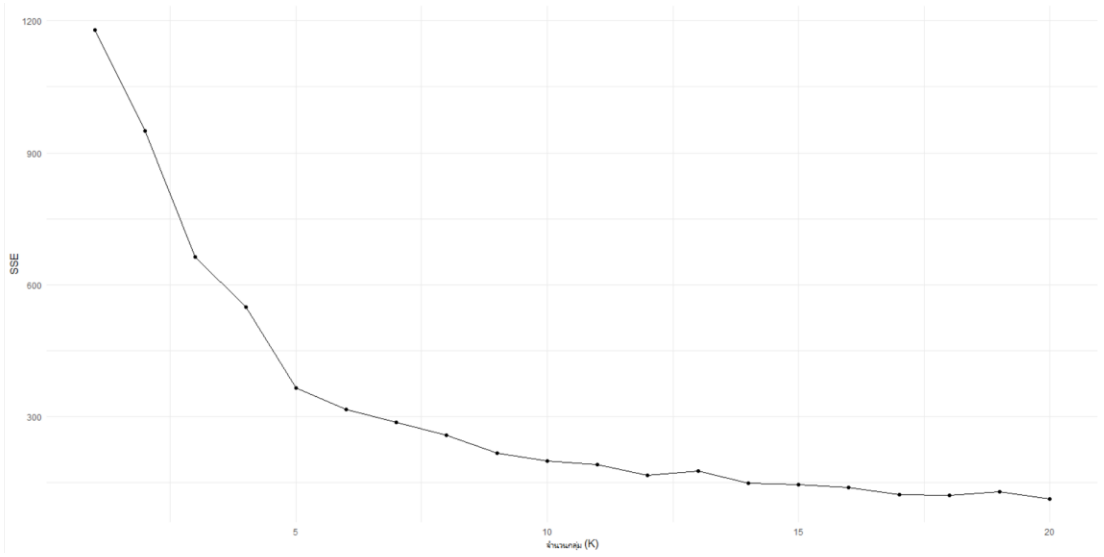
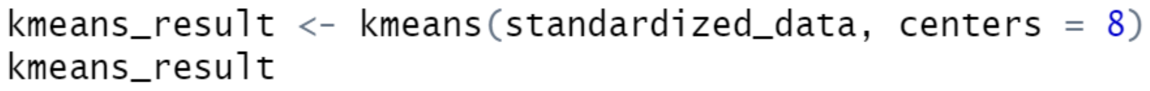


Figure 16 Graph of sse with K = 1-20.

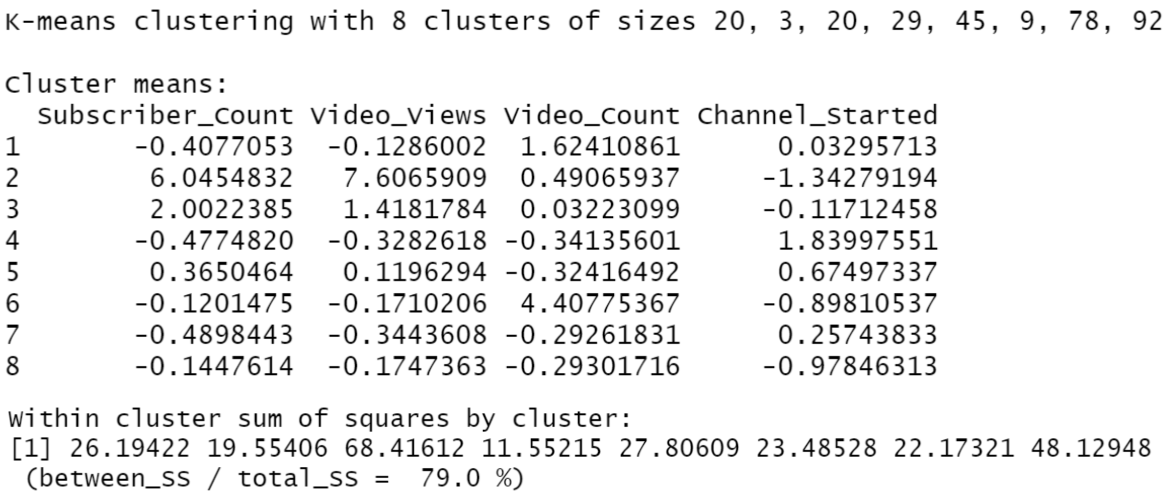
In decision rule, to find the K value that gives SSE rapid decline and after that it slowly decreased. We choose k = 8 to do clustering.

Next, you need to selecting the number of clusters (K) and we choose k = 8 and use command kmeans(test, center = k) to get result of K-means.



```
kmeans_result <- kmeans(standardized_data, centers = 8)
kmeans_result
```

Figure 17 Set parameter for Test K-means.



```
K-means clustering with 8 clusters of sizes 20, 3, 20, 29, 45, 9, 78, 92

Cluster means:
  Subscriber_Count Video_Views Video_Count Channel_Started
1       -0.4077053  -0.1286002  1.62410861      0.03295713
2        6.0454832   7.6065909  0.49065937     -1.34279194
3        2.0022385   1.4181784  0.03223099     -0.11712458
4       -0.4774820  -0.3282618 -0.34135601      1.83997551
5        0.3650464   0.1196294 -0.32416492      0.67497337
6       -0.1201475  -0.1710206  4.40775367     -0.89810537
7       -0.4898443  -0.3443608 -0.29261831      0.25743833
8       -0.1447614  -0.1747363 -0.29301716     -0.97846313

Within cluster sum of squares by cluster:
[1] 26.19422 19.55406 68.41612 11.55215 27.80609 23.48528 22.17321 48.12948
 (between_SS / total_SS =  79.0 %)
```

Figure 18 Output of K-means.

In model, the 8 clusters are made which are of 20,3,20,29,45,9,78 and 92 sizes respectively. Within the cluster, the sum of squares is 79.0%.

Next, it plots. We recommend plotting every pair. We can plot a total of 6 pairs as follows.

1. Subscriber_Count and Video_Views.
2. Subscriber_Count and Video_Count.
3. Subscriber_Count and Channel_Started.
4. Video_Views and Video_Count.
5. Video_Views and Channel_Started.
6. Video_Count and Channel_Started.

1. Subscriber_Count and Video_Views.

```
plot(standardized_data[, c(1, 2)],
     col = kmeans_result$cluster,
     main = "K-means with 8 clusters")
```

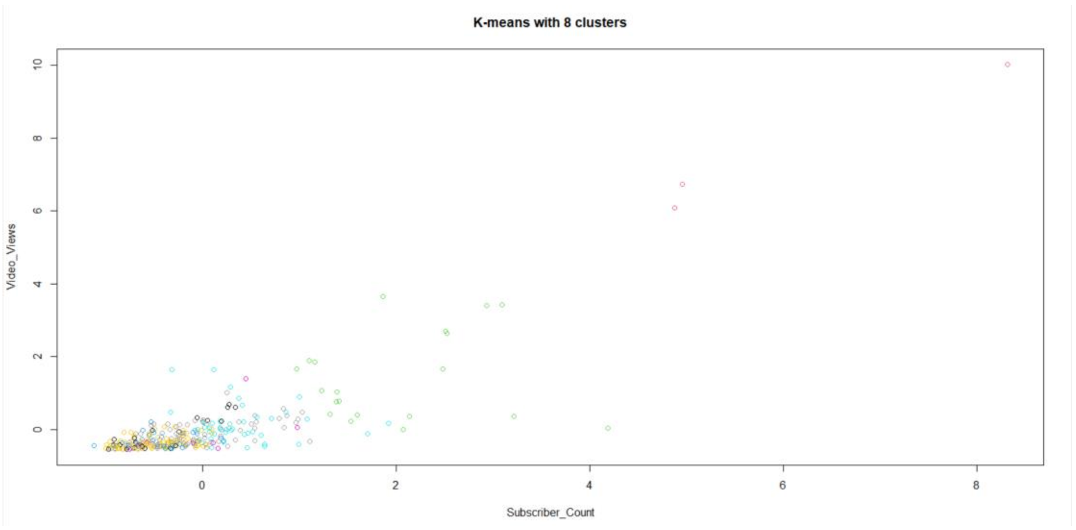


Figure 19 plot with Subscriber_Count and Video_Views.

2. Subscriber_Count and Video_Count.

```
plot(standardized_data[, c(1, 3)],
     col = kmeans_result$cluster,
     main = "K-means with 8 clusters")
```

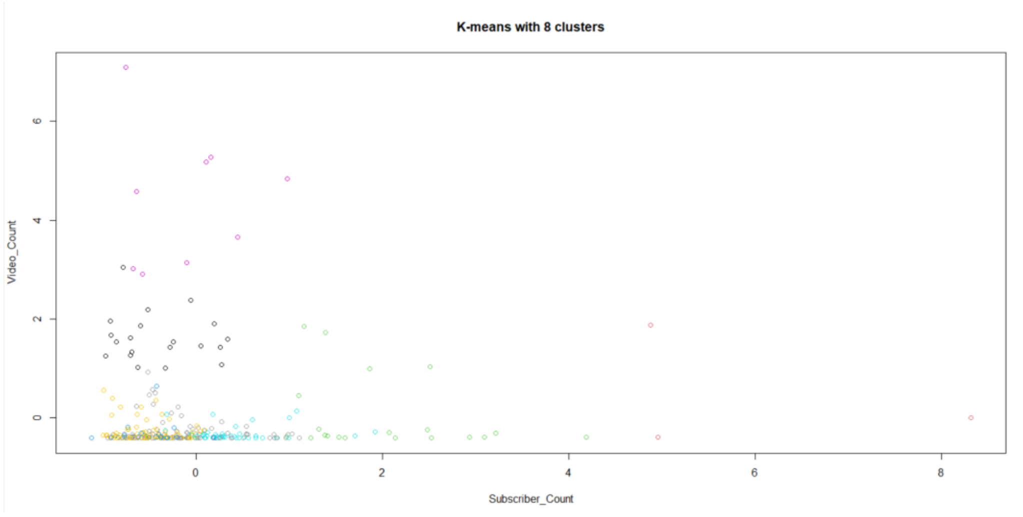


Figure 20 plot with Subscriver_count and Video_Count.

3. Subscriber_Count and Channel_Started.

```
plot(standardized_data[, c(1, 4)],
     col = kmeans_result$cluster,
     main = "K-means with 8 clusters")
```

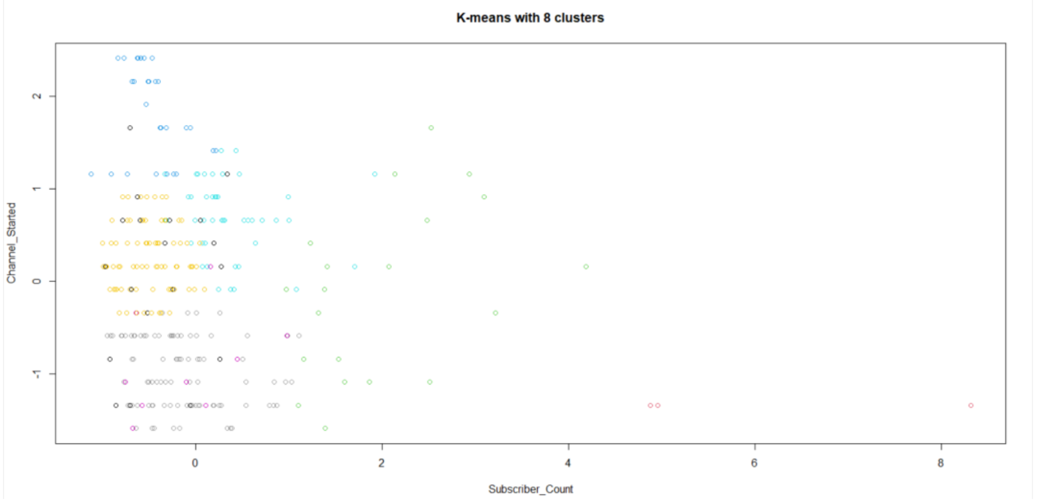


Figure 21 plot with Subscriber_Count and Channel_Started.

4. Video_Views and Video_Count.

```
plot(standardized_data[, c(2, 3)],
     col = kmeans_result$cluster,
     main = "K-means with 8 clusters")
```

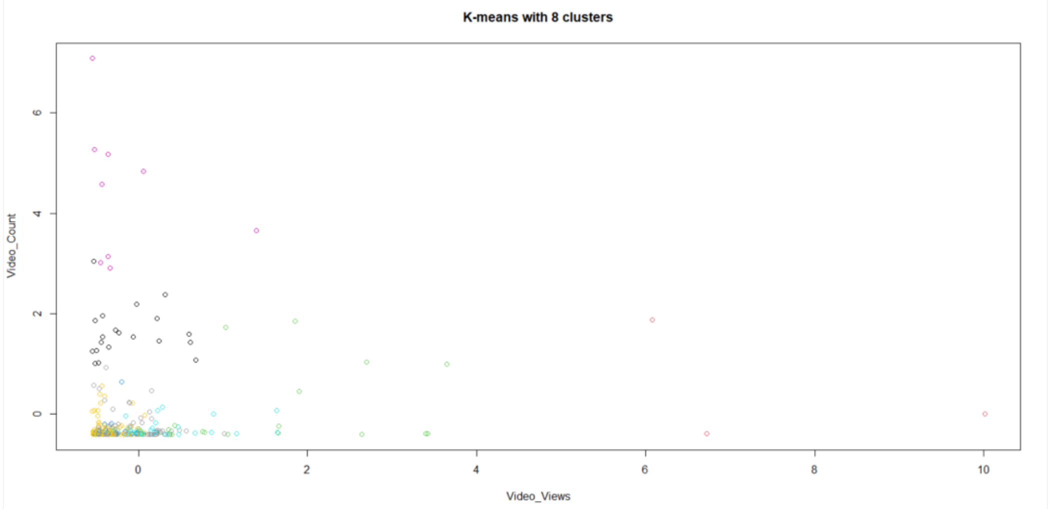


Figure 22 plot with Video_Views and Video_Count.

5. Video_Views and Channel_Started.

```
plot(standardized_data[, c(2, 4)],
     col = kmeans_result$cluster,
     main = "K-means with 8 clusters")
```

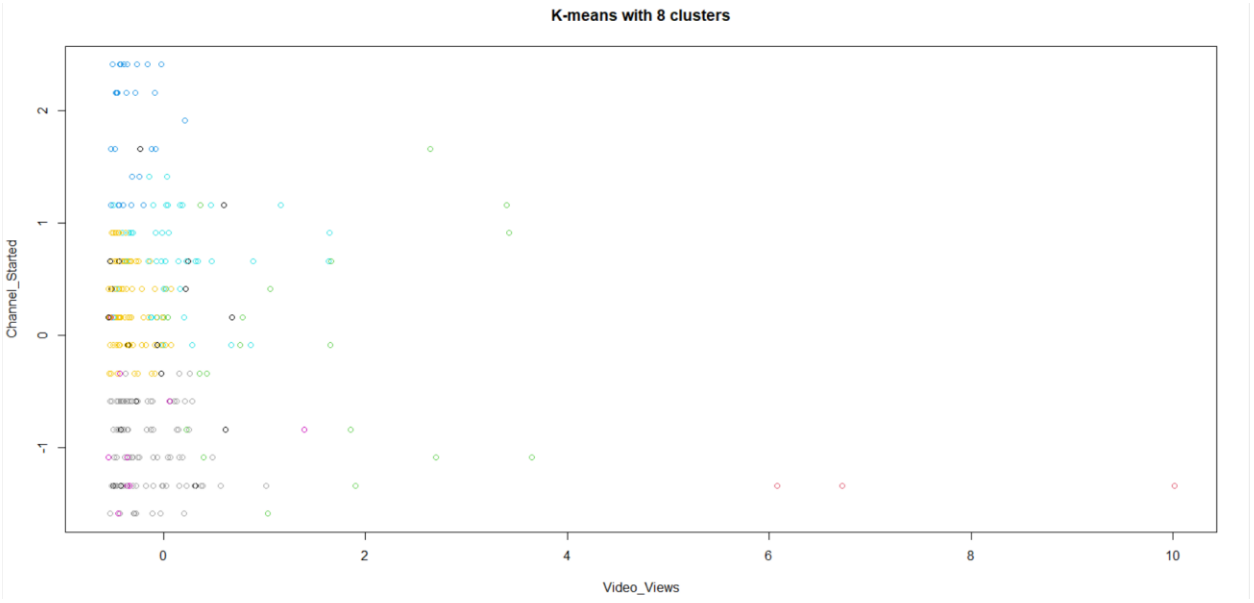


Figure 23 plot with Video_Views and Channel_Started.

6. Video_Count and Channel_Started.

```
plot(standardized_data[, c(3, 4)],
     col = kmeans_result$cluster,
     main = "K-means with 8 clusters")
```

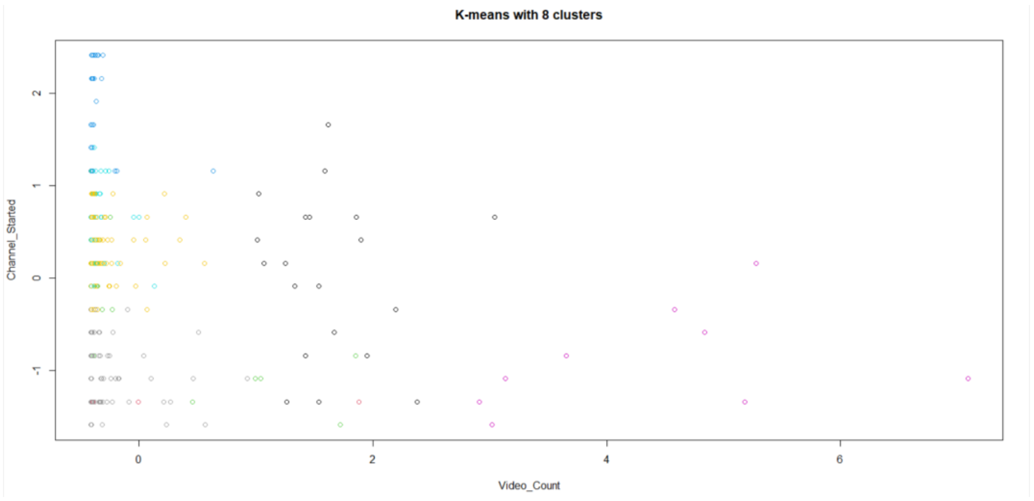


Figure 24 plot with Video_Count and Channel_Started.

To show with PC for see overall by 2 dimensions of K-means with 8 group by autoplot.

```
autoplot(kmeans_result,standardized_data,frame=TRUE)
```

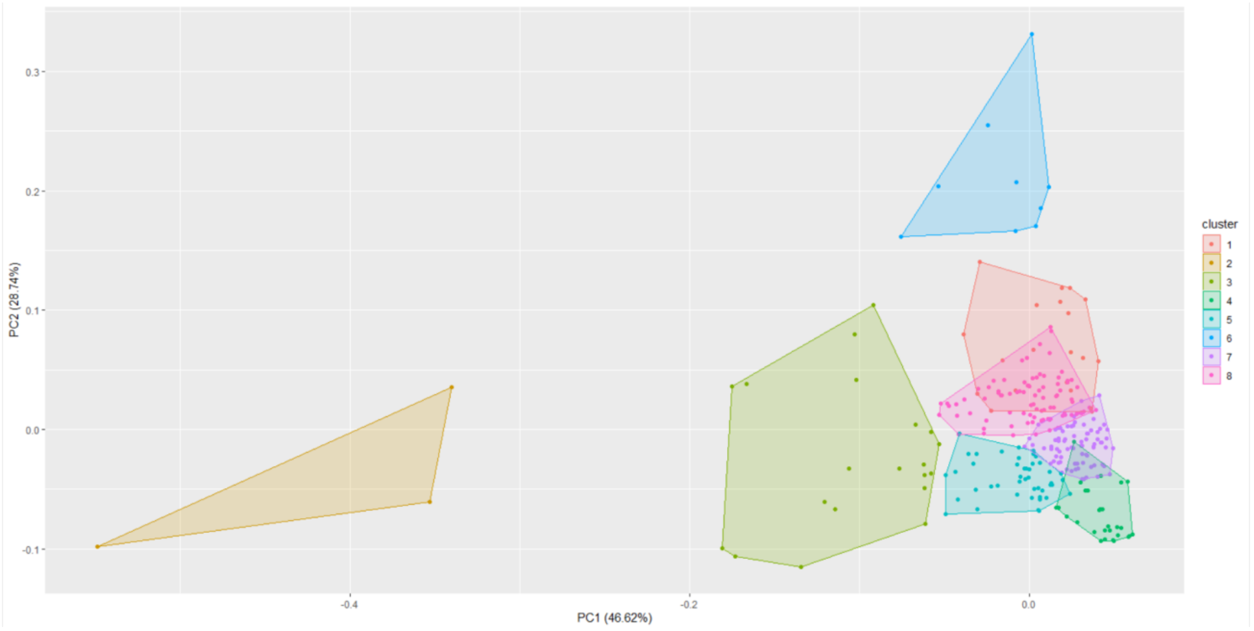


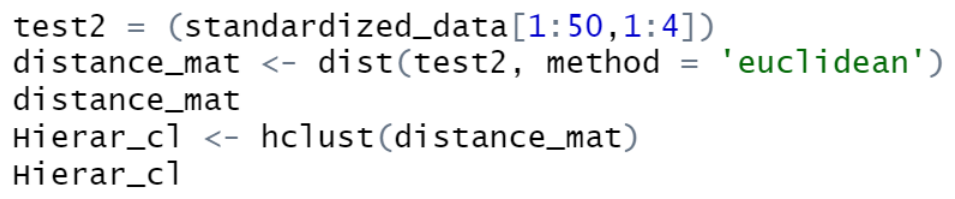
Figure 25 Graph of K-means with autoplot by PC.

For all steps, we can conclude that when we choose k = 8 or 8 Group with data Top 300 YouTube Channels it should be 8 clusters are of 20,3,20,29,45,9,78 and 92 sizes respectively. Within the cluster, the sum of squares is 79.0%.

.

7.2.2 Hierarchical Clustering (Euclidean distance)

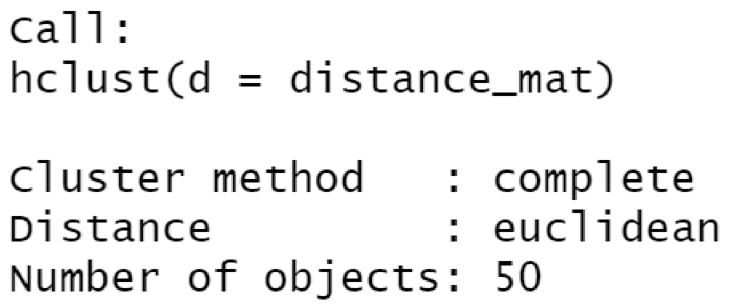
First step for Hierarchical Clustering, you need to select data that you want to know, and we choose 50 datapoint to use. Next, it's calculated distances by Euclidean with command dist and keep it to distance_mat. When you finish calculated distances, you with use command hclust to clustering by choose distance_mat and keep it with variable Heirar_cl.



```
test2 = (standardized_data[1:50,1:4])
distance_mat <- dist(test2, method = 'euclidean')
distance_mat
Hierar_cl <- hclust(distance_mat)
Hierar_cl
```

Figure 26 Set data and Parameter for Heirarchical Clustering.

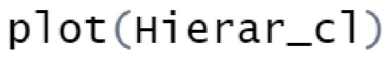
When you run Heirar_cl, you will get Number of objects, Method to use cluster and method of Distance.



```
Call:
hclust(d = distance_mat)

Cluster method   : complete
Distance         : euclidean
Number of objects: 50
```

Figure 27 Output for Heirarchical Clustering.

And We plot with plot and parameter is Heirar_cl. You wll get a dendrogram for all datapoint.



```
plot(Hierar_cl)
```

Figure 28 plot with command plot.

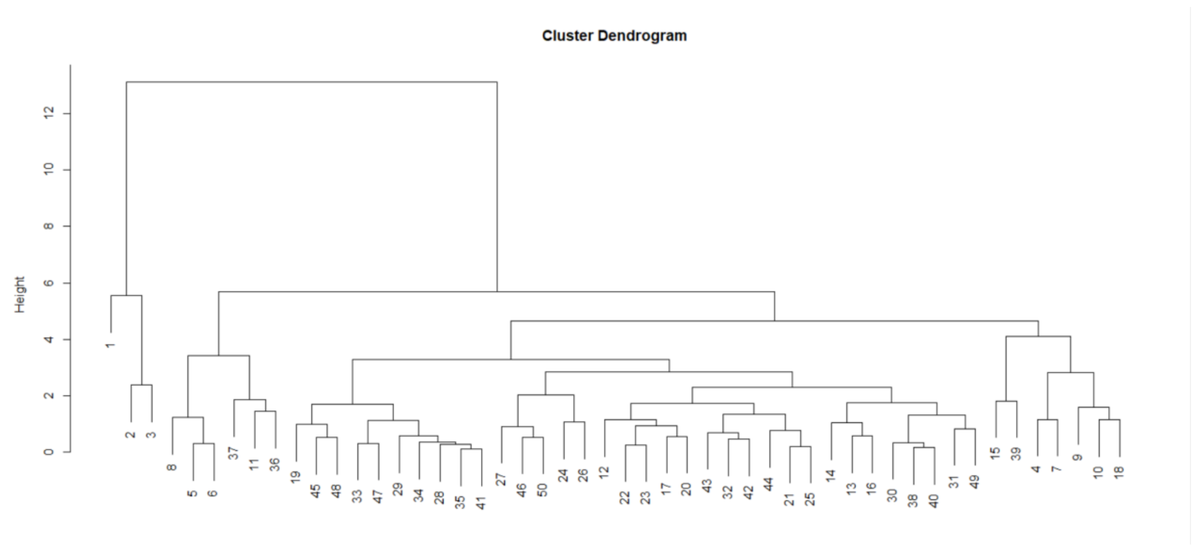


Figure 29 dendrogram.

To cutting Tree is cut where k, when k is number of groups that you interest. In this case I use 4 and each category represents its number of clusters. When we create a line to see the cutting point.

```
plot(Hierar_cl)
abline(h = 5, col = "green")
```

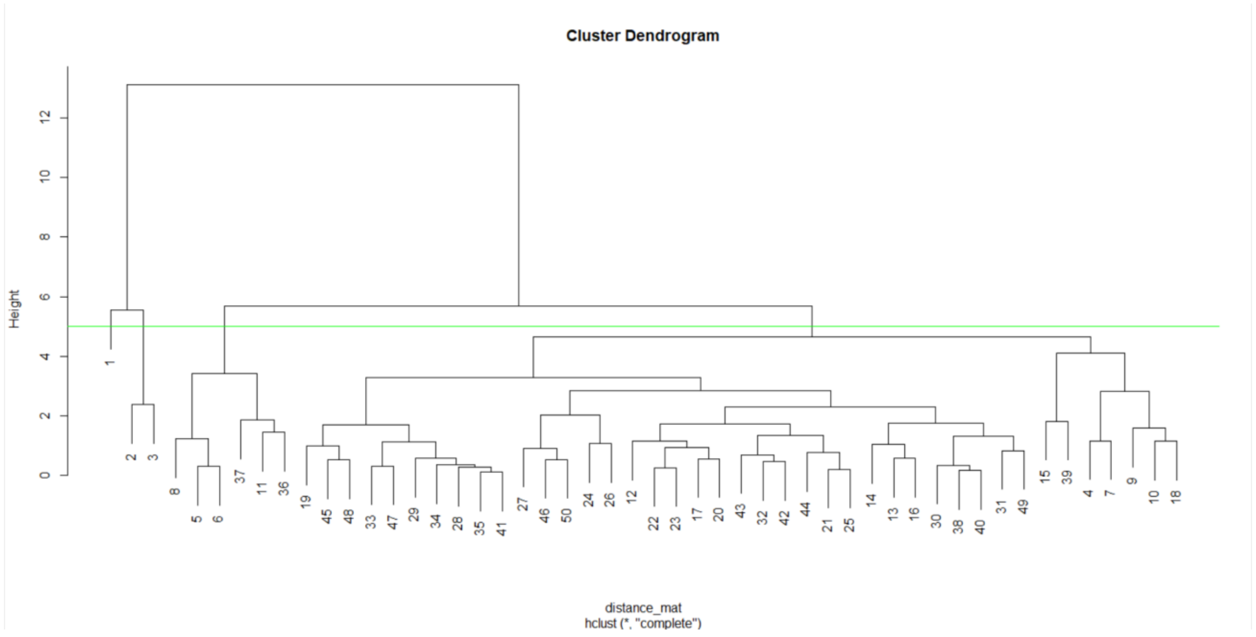


Figure 30 cutting point in dendrogram.

Now I fit group = 4 that we interest and plot with parameter is Hierar_cl and border group with green.

```
fit <- cutree(Hierar_cl, k = 4 )
fit
plot(Hierar_cl)
table(fit)
rect.hclust(Hierar_cl, k = 4, border = "green")
```

Figure 31 set clustering k = 3 and plot with border.

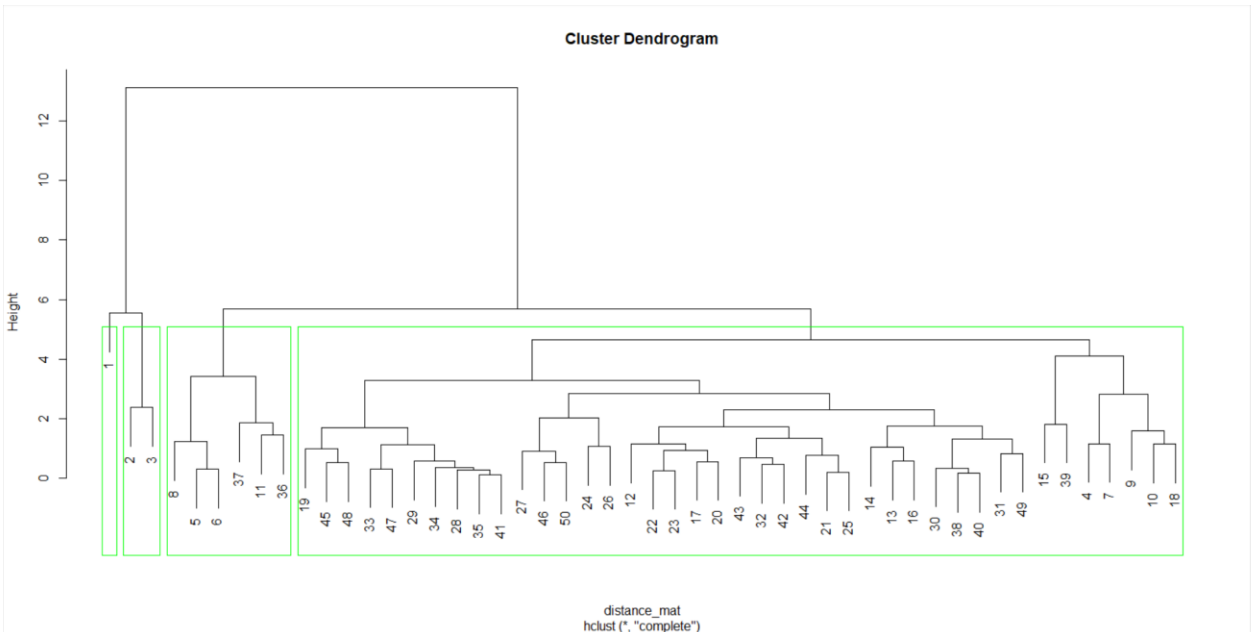
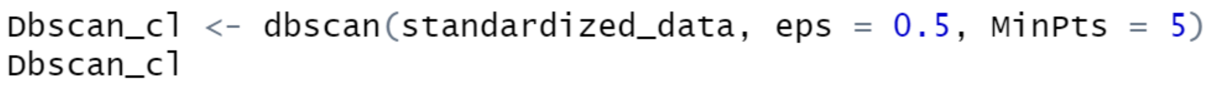


Figure 32 Dendrogram with clustering.

For all steps, we can conclude that when we choose k = 4 or 3 Group with data Top 300 YouTube Channels with 50 datapoint it should be 1 datapoint are cluster 1, 2 datapoint are cluster 2, 6 datapoint are cluster 3 and 41 datapoints are cluster 4.

7.2.3. DBSCAN (Euclidean distance)

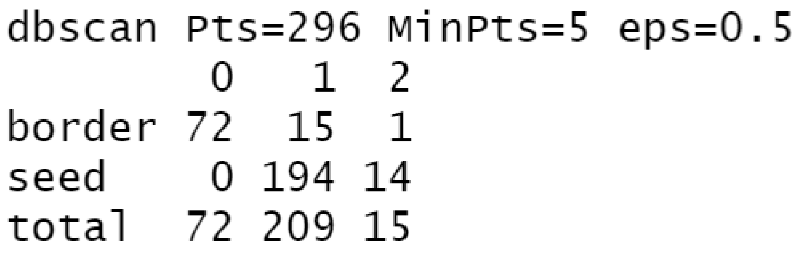
First step for DBScan, you need to set 2 parameters as follows: eps and Minpts.



```
Dbscan_cl <- dbscan(standardized_data, eps = 0.5, MinPts = 5)
Dbscan_cl
```

Figure 33 Set parameter for DBScan.

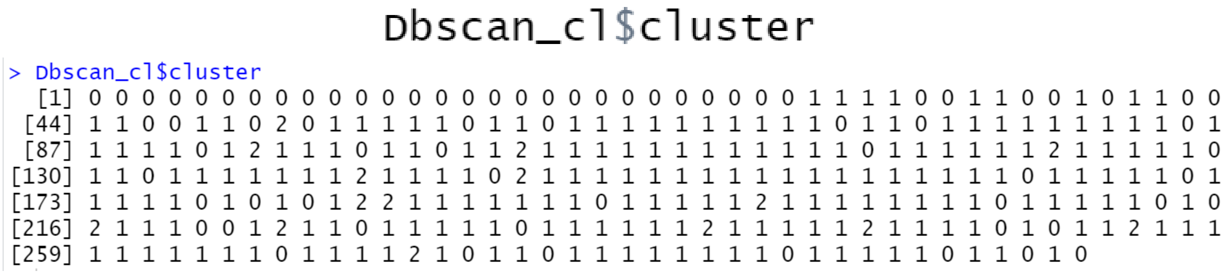
In the model, there are Pts with Minimum points are 5 and eps is 0.5.



```
dbscan Pts=296 MinPts=5 eps=0.5
         0   1  2
border  72  15  1
seed     0 194 14
total   72 209 15
```

Figure 34 Show a DBScan and output.

To show cluster identification, we use Dbscan_cl$cluster.



```
Dbscan_cl$cluster
```

```
> Dbscan_cl$cluster
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 0 1 1 0 0
 [44] 1 1 0 0 1 1 0 2 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 0 1
 [87] 1 1 1 1 0 1 2 1 1 1 0 1 1 0 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 2 1 1 1 1 1 0
[130] 1 1 0 1 1 1 1 1 1 1 2 1 1 1 1 0 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1
[173] 1 1 1 1 0 1 0 1 0 1 2 2 1 1 1 1 1 1 1 0 1 1 1 1 1 2 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0
[216] 2 1 1 1 0 0 1 2 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 0 1 0 1 1 2 1 1 1
[259] 1 1 1 1 1 1 1 0 1 1 1 1 2 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 0 1 0
```

Figure 35 Output with Cluster by DBScan.

Next, It plot. We recommend plotting every pair. We can plot a total of 6 pairs as follows.

1. Subscriber_Count and Video_Views.
2. Subscriber_Count and Video_Count.
3. Subscriber_Count and Channel_Started.
4. Video_Views and Video_Count.
5. Video_Views and Channel_Started.
6. Video_Count and Channel_Started.

1. Subscriber_Count and Video_Views.

```
plot(Dbscan_cl, standardized_data[, c(1,2)], main = "DBScan")
```

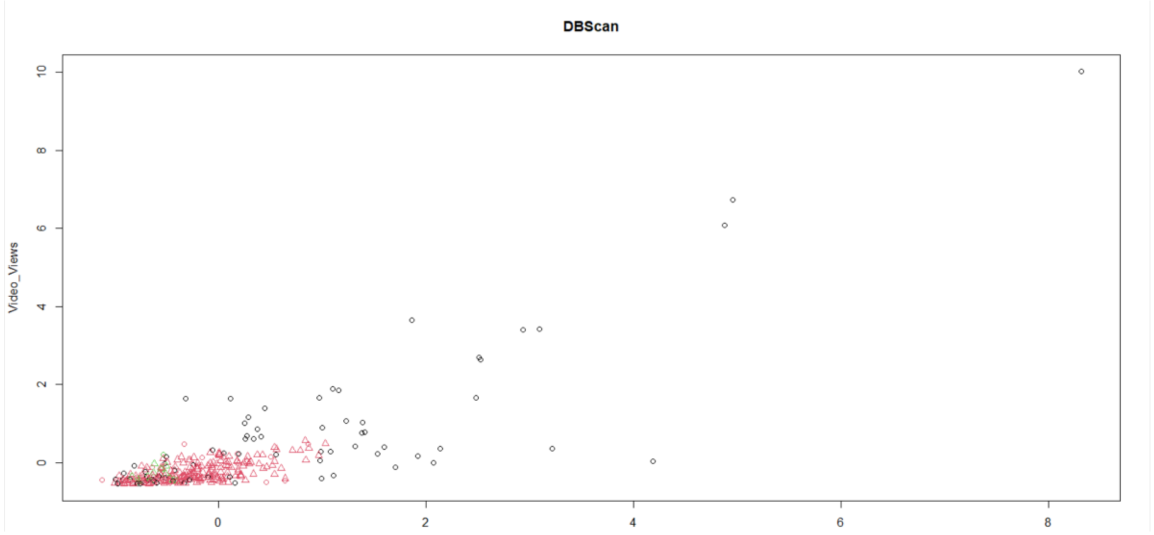


Figure 36 plot with Subscriber_Count and Video_Views.

2. Subscriber_Count and Video_Count.

```
plot(Dbscan_cl, standardized_data[, c(1,3)], main = "DBScan")
```

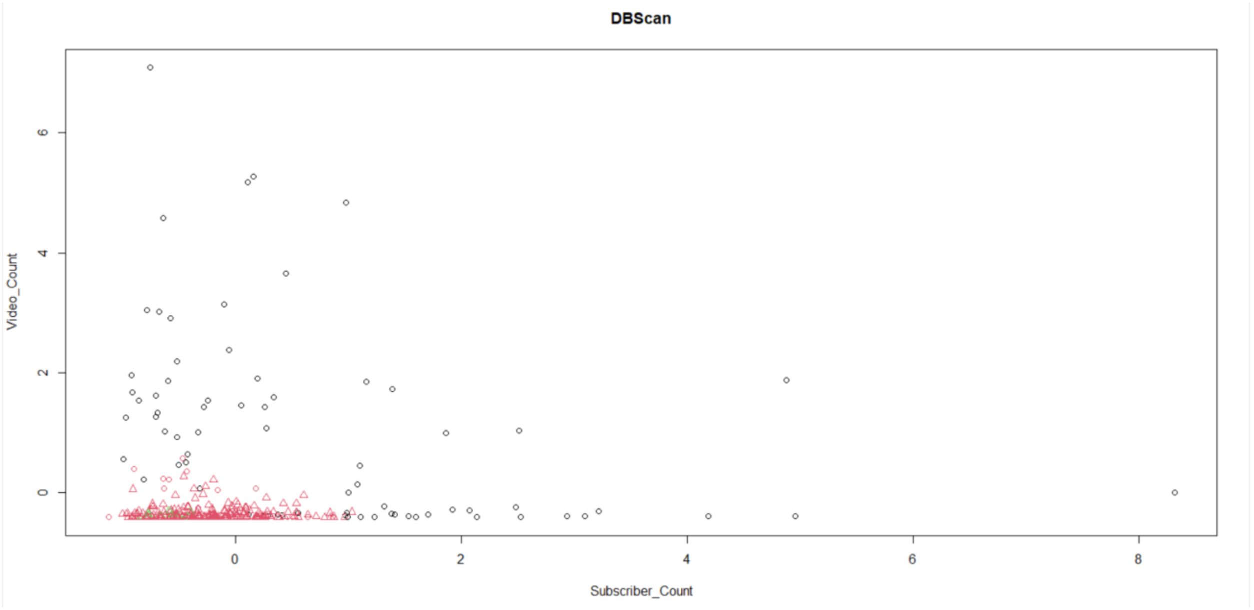


Figure 37 plot with Subscriver_count and Video_Count.

3. Subscriber_Count and Channel_Started.

```
plot(Dbscan_cl, standardized_data[, c(1,4)], main = "DBScan")
```

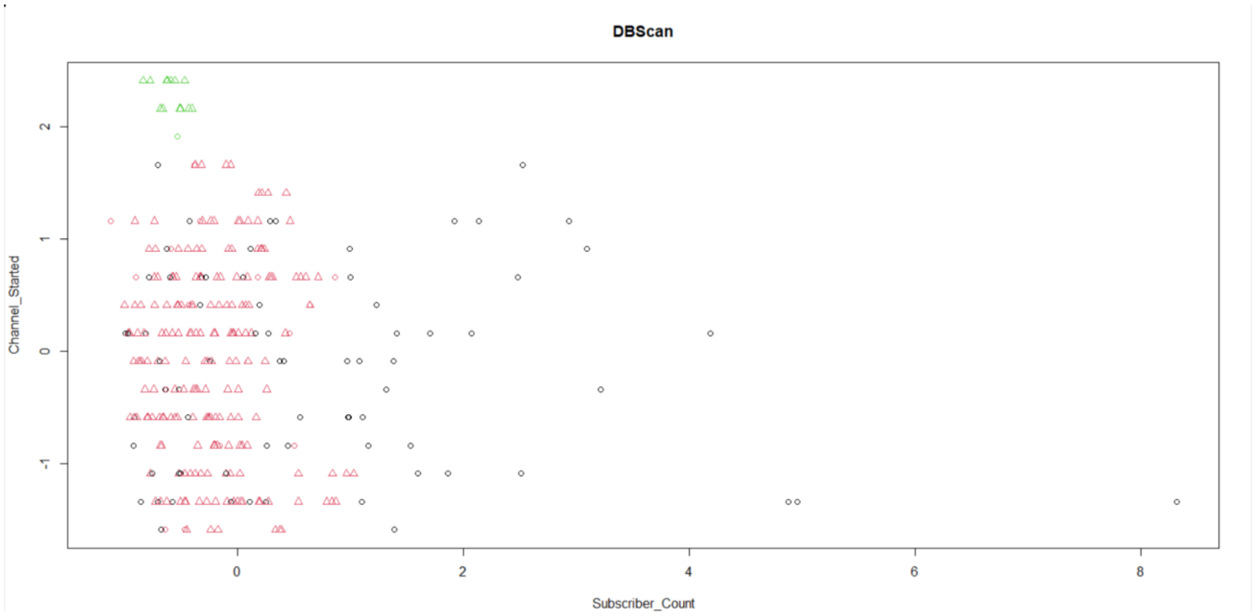


Figure 38 plot with Subscriber_Count and Channel_Started.

4. Video_Views and Video_Count.

```
plot(Dbscan_cl, standardized_data[, c(2,3)], main = "DBScan")
```

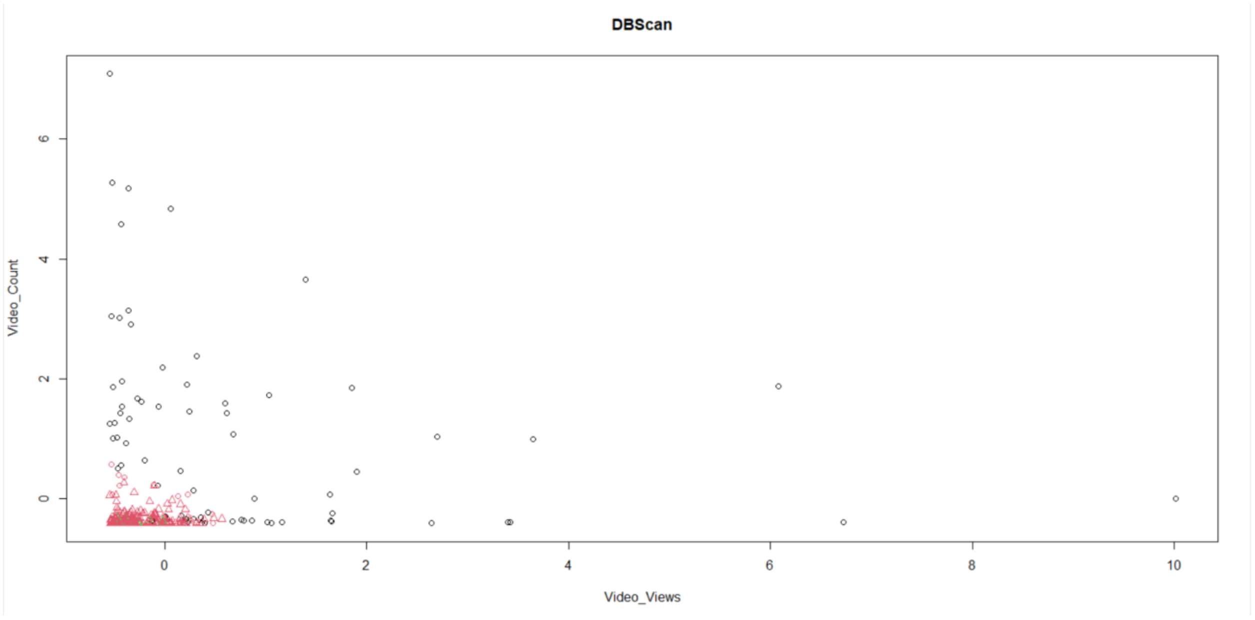


Figure 39 plot with Video_Views and Video_Count.

5. Video_Views and Channel_Started.

```
plot(Dbscan_cl, standardized_data[, c(2,4)], main = "DBScan")
```

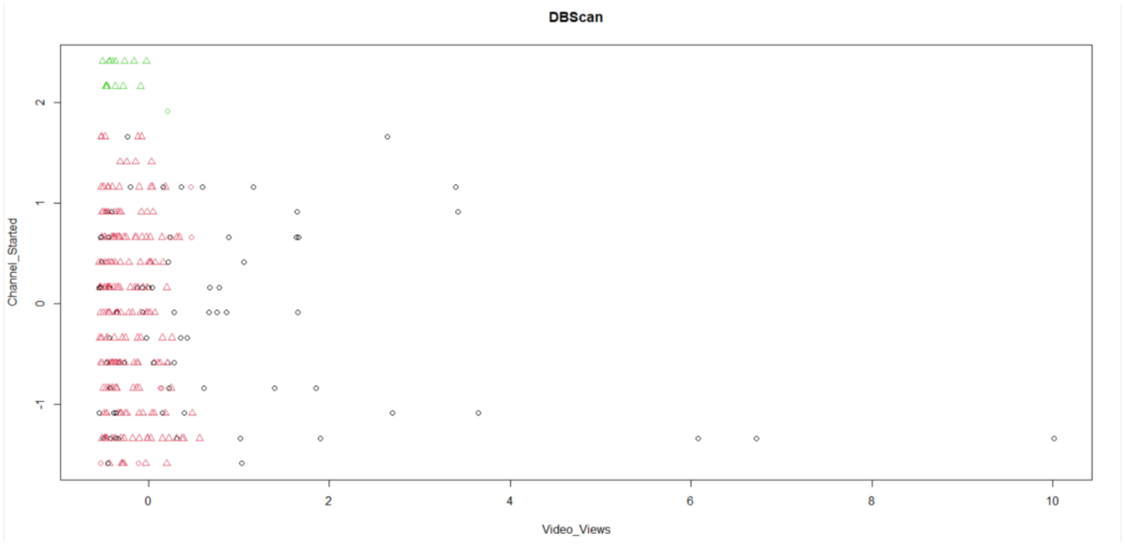


Figure 40 plot with Video_Views and Channel_Started.

6. Video_Count and Channel_Started.

```
plot(Dbscan_cl, standardized_data[, c(3,4)], main = "DBScan")
```

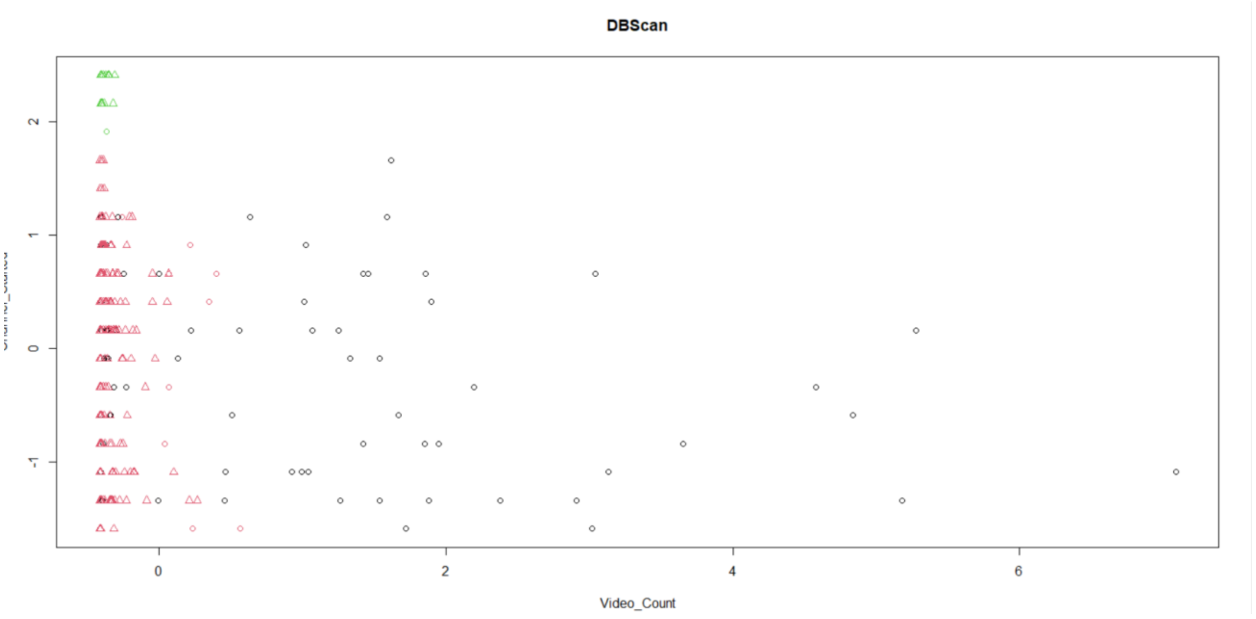


Figure 41 plot with Video_Count and Channel_Started.

For all steps, we can concluded that when we set 2 parameter including eps = 0.5 and Minpts = 5 in Top 300 YouTube Channels data it should be 209 datapoint are cluster 1, 15 datapoint are cluster 2 and 72 datapoint are noise point.

7.3 Clustering with Manhattan distance

7.3.1 K-Means (Manhattan distance)

First step for K-means, you need to select the number of clusters (K) and We use Elbow method to find the best of K. So, we use a loop to find the best option by going from number 1 to number 20 and storing the variable in sse with a length of 20. This vector will be used to store the SSE values. for different values of K. and plot from number 1 to number 20 compared to the sse value of each item.

```
sse <-numeric(20)
for (k in 1:20) {
  kmeans_model <- kmeans(standardized_data, centers = k,algorithm = "Lloyd")
  sse[k] <- kmeans_model$tot.withinss
}

ggplot(data.frame(K = 1:20, SSE = sse), aes(x = K, y = SSE)) +
  geom_line() +
  geom_point() +
  labs(x = "จำนวนกลุ่ม (K)", y = "SSE") +
  theme_minimal()
```

Figure 42 command looping to find best K.

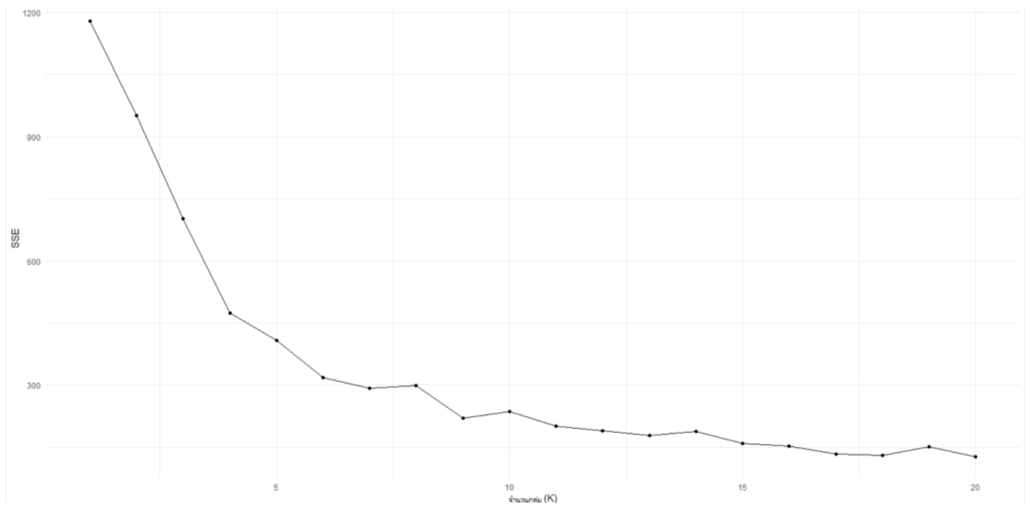
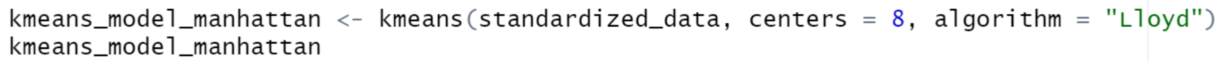


Figure 43 Graph with sse K = 1-20.

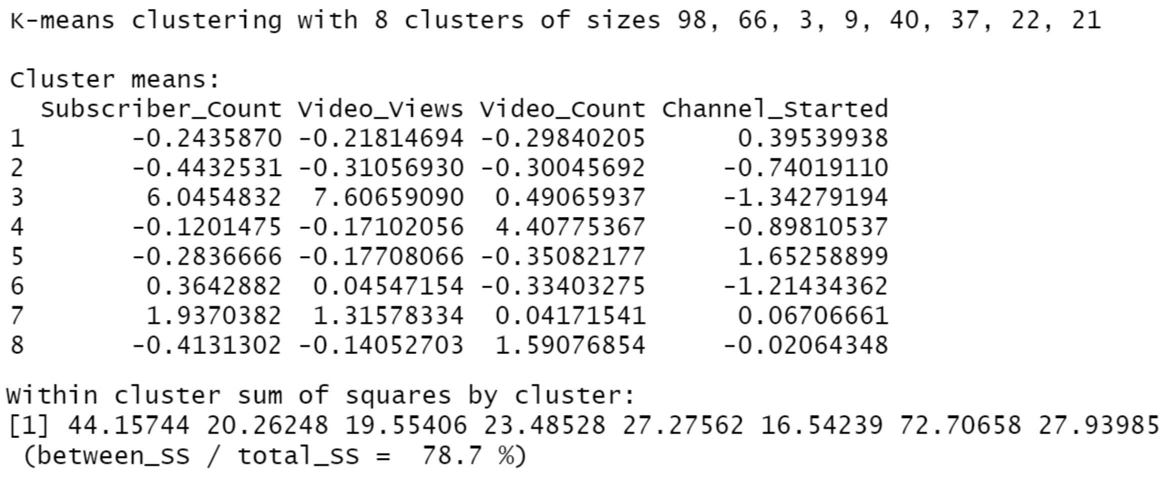
In decision rule, To find the K value that gives SSE rapid decline and after that it slowly decreased. We choose k = 8 to do clustering.

Next, you need to selecting the number of clusters (K) and we choose k = 8 and use command kmeans(test, center = k,algorithm =”Lloyd”) to get result of K-means with Manhattan distance.



```
kmeans_model_manhattan <- kmeans(standardized_data, centers = 8, algorithm = "Lloyd")
kmeans_model_manhattan
```

Figure 44 Set parameter for Test K-means with Manhattan.



```
K-means clustering with 8 clusters of sizes 98, 66, 3, 9, 40, 37, 22, 21

Cluster means:
  Subscriber_Count Video_Views Video_Count Channel_Started
1       -0.2435870 -0.21814694 -0.29840205      0.39539938
2       -0.4432531 -0.31056930 -0.30045692     -0.74019110
3        6.0454832  7.60659090  0.49065937     -1.34279194
4       -0.1201475 -0.17102056  4.40775367     -0.89810537
5       -0.2836666 -0.17708066 -0.35082177      1.65258899
6        0.3642882  0.04547154 -0.33403275     -1.21434362
7        1.9370382  1.31578334  0.04171541      0.06706661
8       -0.4131302 -0.14052703  1.59076854     -0.02064348

Within cluster sum of squares by cluster:
[1] 44.15744 20.26248 19.55406 23.48528 27.27562 16.54239 72.70658 27.93985
 (between_SS / total_SS =  78.7 %)
```

Figure 45 Output of K-means with Manhattan.

In model, the 8 clusters are made which are of 98,66,3,9,40,37,22 and 21 sizes respectively. Within the cluster, the sum of squares is 78.7%.

Next, It plot. We recommend plotting every pair. We can plot a total of 6 pairs as follows.

1. Subscriber_Count and Video_Views.
2. Subscriver_count and Video_Count.
3. Subscriber_Count and Channel_Started.
4. Video_Views and Video_Count.
5. Video_Views and Channel_Started.
6. Video_Count and Channel_Started.

1. Subscriber_Count and Video_Views.

```
plot(standardized_data[, c(1, 2)],
     col = kmeans_model_manhattan$cluster,
     main = "K-means with 8 clusters (Manhattan)")
```

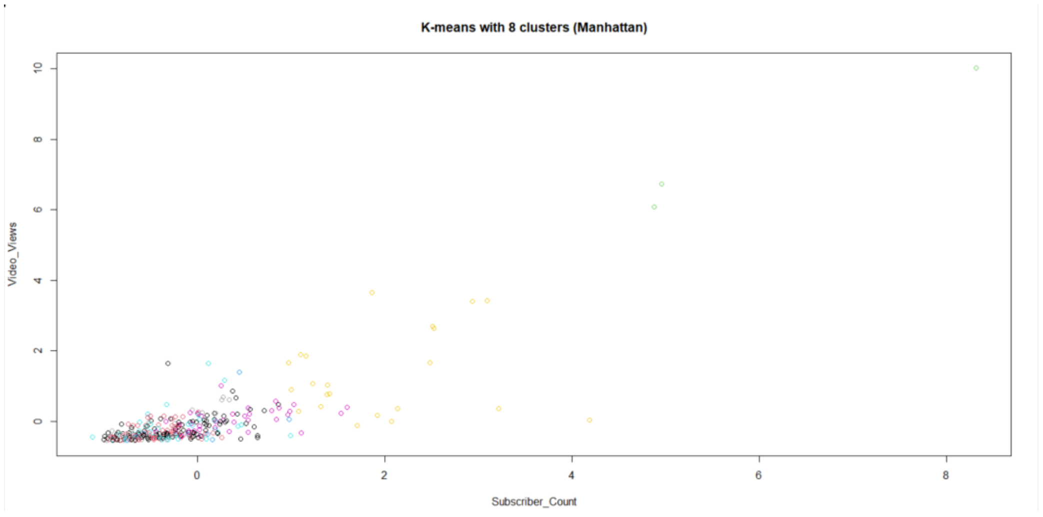


Figure 46 Subscriber_Count and Video_Views with Manhattan.

2. Subscriber_Count and Video_Count.

```
plot(standardized_data[, c(1, 3)],
     col = kmeans_model_manhattan$cluster,
     main = "K-means with 8 clusters (Manhattan)")
```

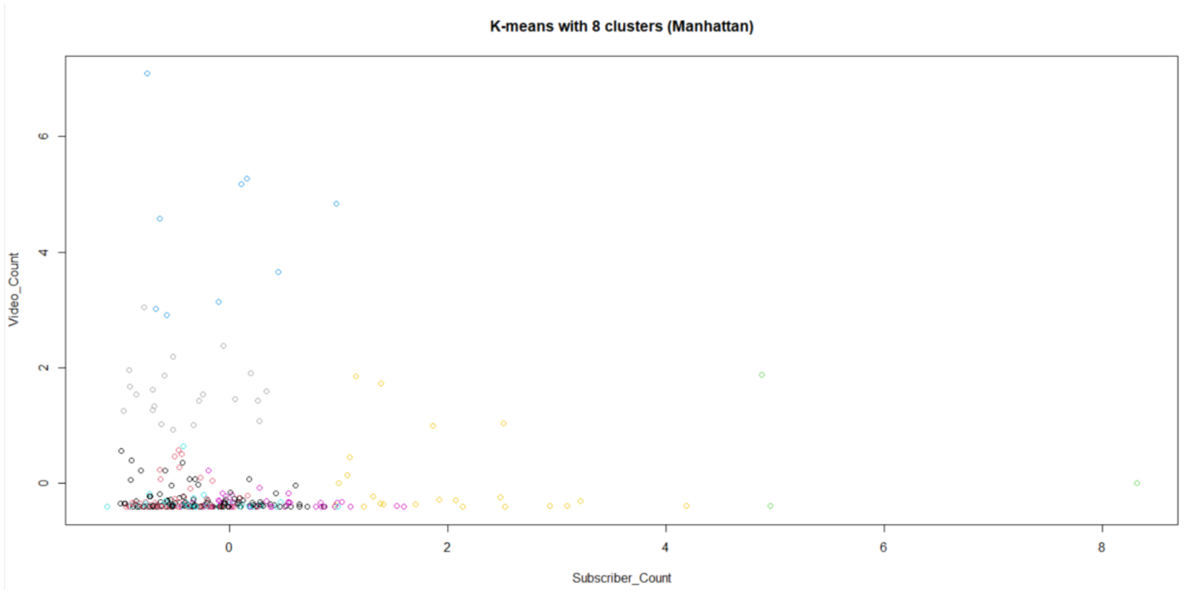


Figure 47 Subscriver_count and Video_Count with Manhattan.

3. Subscriber_Count and Channel_Started.

```
plot(standardized_data[, c(1, 4)],
     col = kmeans_model_manhattan$cluster,
     main = "K-means with 8 clusters (Manhattan)")
```

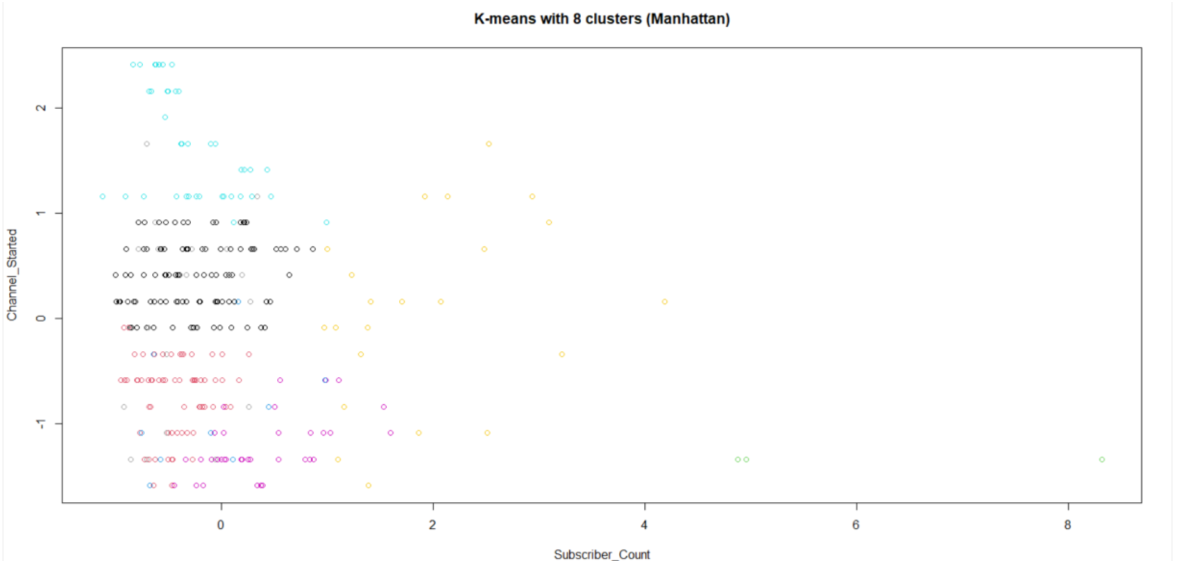


Figure 48 plot with Subscriber_Count and Channel_Started with Manhattan.

4. Video_Views and Video_Count.

```
plot(standardized_data[, c(2, 3)],
     col = kmeans_model_manhattan$cluster,
     main = "K-means with 8 clusters (Manhattan)")
```

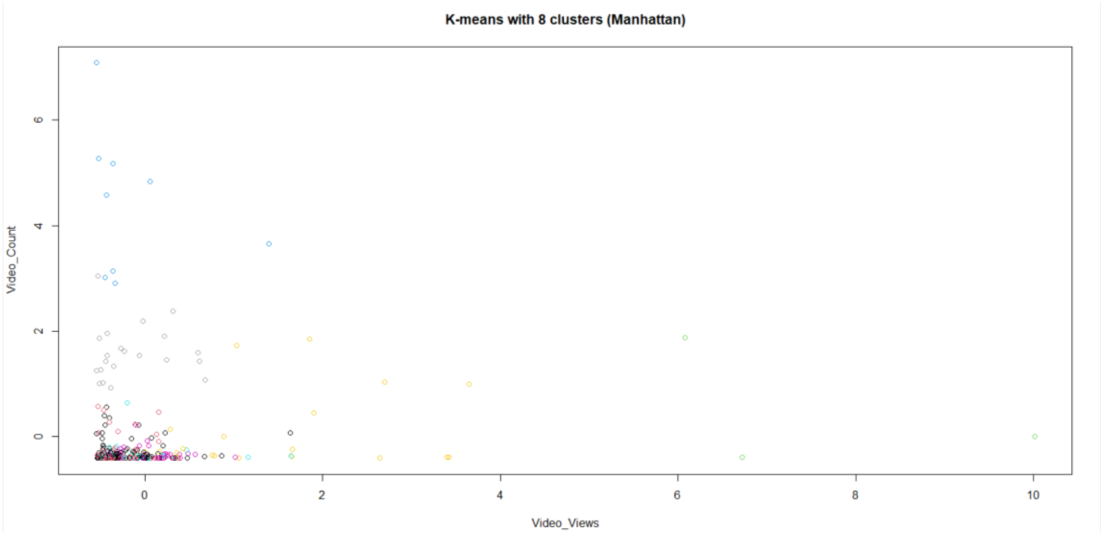


Figure 49 plot with Video_Views and Video_Count with Manhattan.

5. Video_Views and Channel_Started.

```
plot(standardized_data[, c(2, 4)],
     col = kmeans_model_manhattan$cluster,
     main = "K-means with 8 clusters (Manhattan)")
```

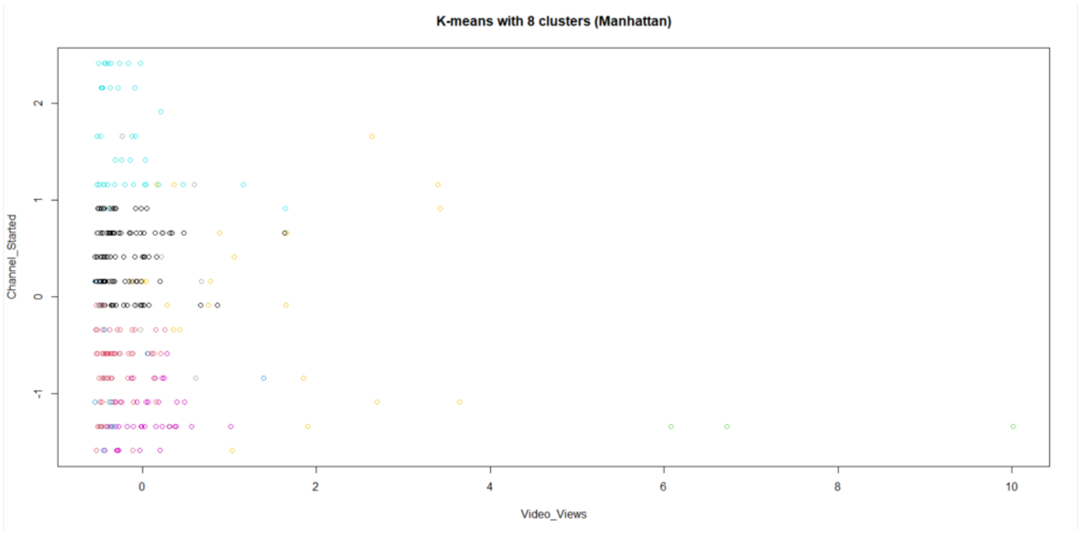


Figure 50 plot with Video_Views and Channel_Started with Manhattan.

6. Video_Count and Channel_Started.

```
plot(standardized_data[, c(3, 4)],
     col = kmeans_model_manhattan$cluster,
     main = "K-means with 8 clusters (Manhattan)")
```

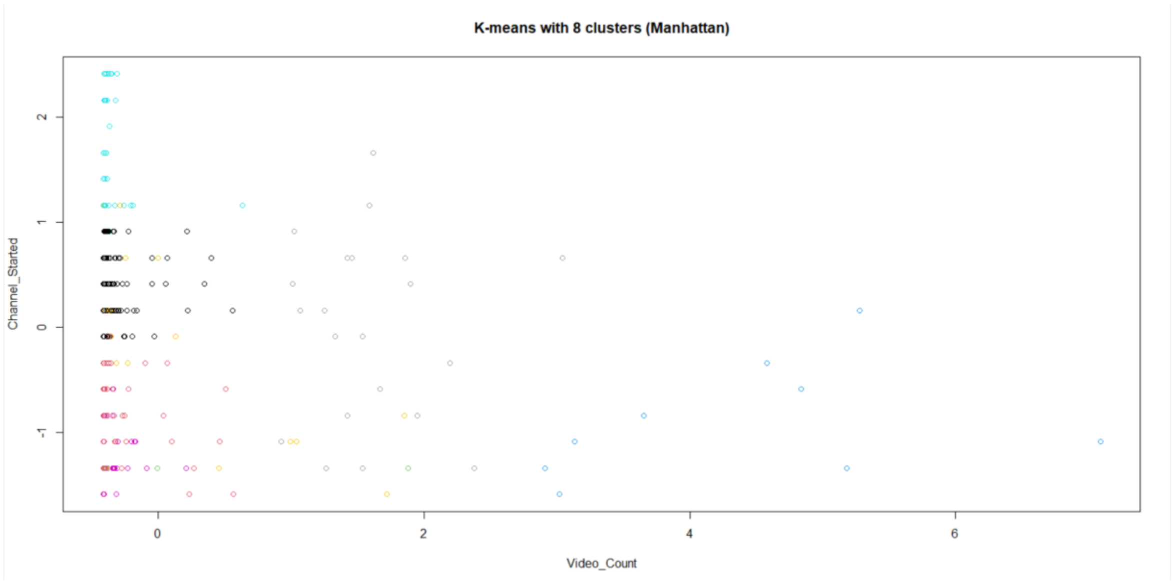


Figure 51 plot with Video_Count and Channel_Started with Manhattan.

To show with PC for see overall by 2 Dimension of K-means with 8 group by autoplot.

```
autoplot(kmeans_model_manhattan,standardized_data,frame=TRUE)
```

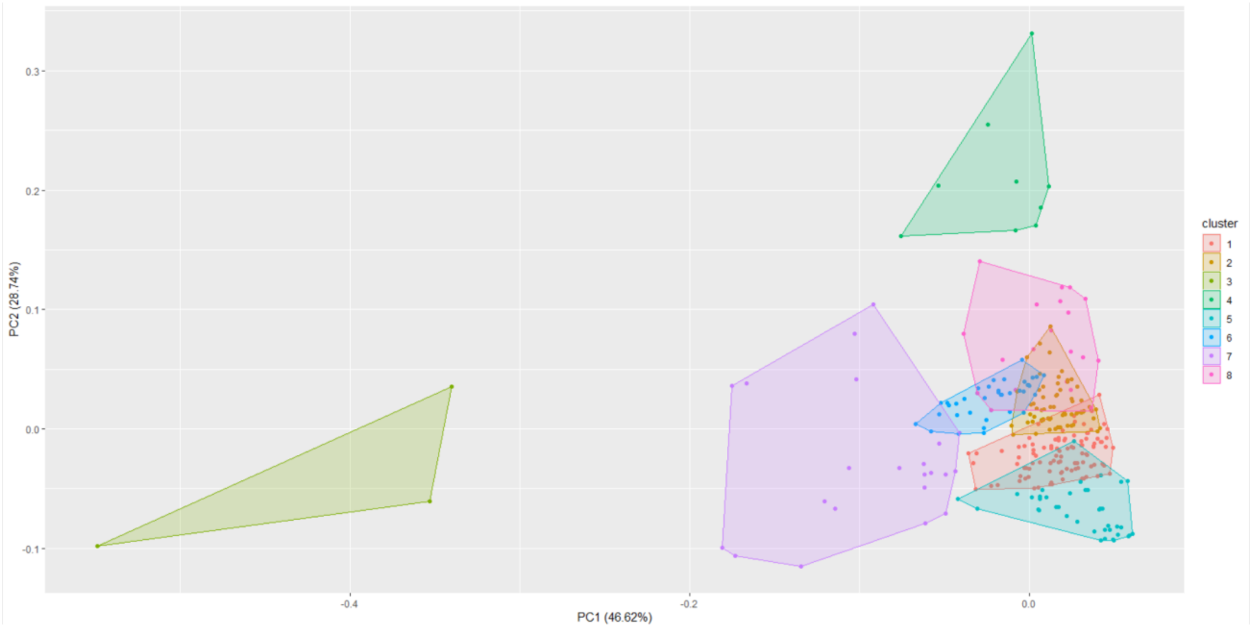
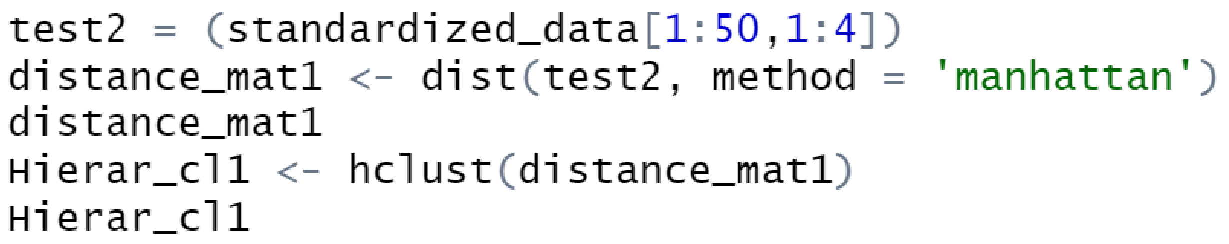


Figure 52 Graph of K-means with autoplot by Manhattan.

For all steps, we can conclude that when we choose k = 8 or 8 Group with data Top 300 YouTube Channels it should be 8 clusters are of 98,66,3,9,40,37,22 and 21 sizes respectively. Within the cluster, the sum of squares is 78.7%.

7.3.2 Hierarchical Clustering (Manhattan distance)

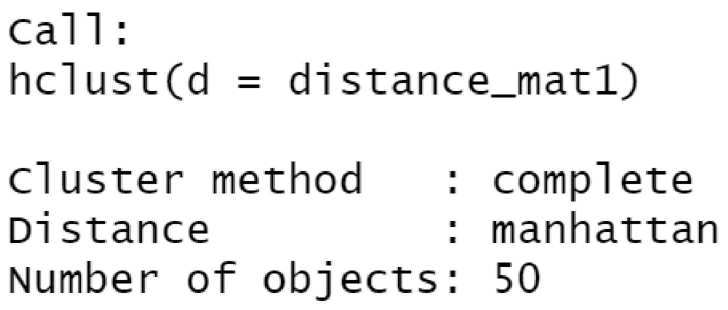
First step for Hierarchical Clustering, you need to select data that you want to know, and we choose 50 datapoint to use. Next, it's calculated distances by Manhattan with command dist and keep it to distance_mat1. When you finish calculated distances, you use command hclust to clustering by choose distance_mat1 and keep it with variable Heirar_cl1.



```
test2 = (standardized_data[1:50,1:4])
distance_mat1 <- dist(test2, method = 'manhattan')
distance_mat1
Hierar_cl1 <- hclust(distance_mat1)
Hierar_cl1
```

Figure 53 Set data and Parameter for Heirarchical Clustering and Manhattan.

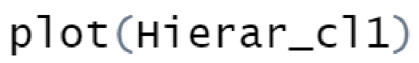
When you run Heirar_cl1, you will get Number of objects, Method to use cluster and method of Distance.



```
Call:
hclust(d = distance_mat1)

Cluster method   : complete
Distance         : manhattan
Number of objects: 50
```

Figure 54 Output for Heirarchical Clustering by Manhattan.

And We plot with plot and parameter is Heirar_cl1. You will get a dendrogram for all datapoint.



```
plot(Hierar_cl1)
```

Figure 55 plot with command plot.

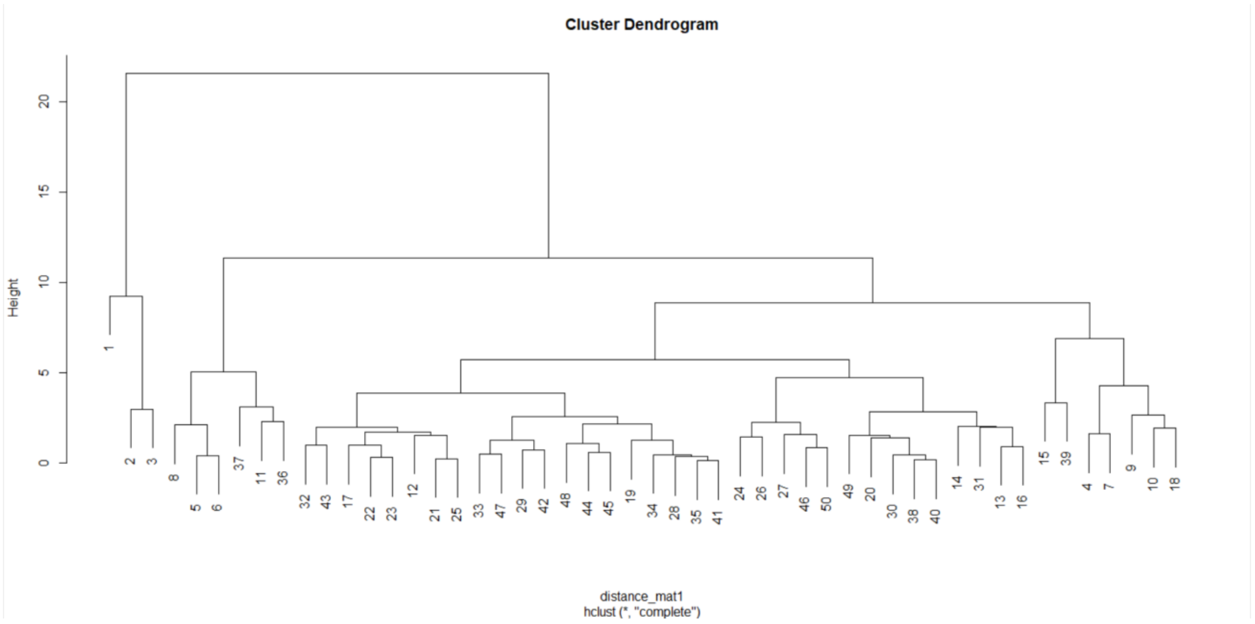


Figure 56 dendrogram with Manhattan.

To cutting Tree is cut where k when k is number of groups that you interest. In this case I use 4 and each category represents its number of clusters. When we create a line to see the cutting point.

```
plot(Hierar_cl1)
abline(h = 9, col = "green")
```

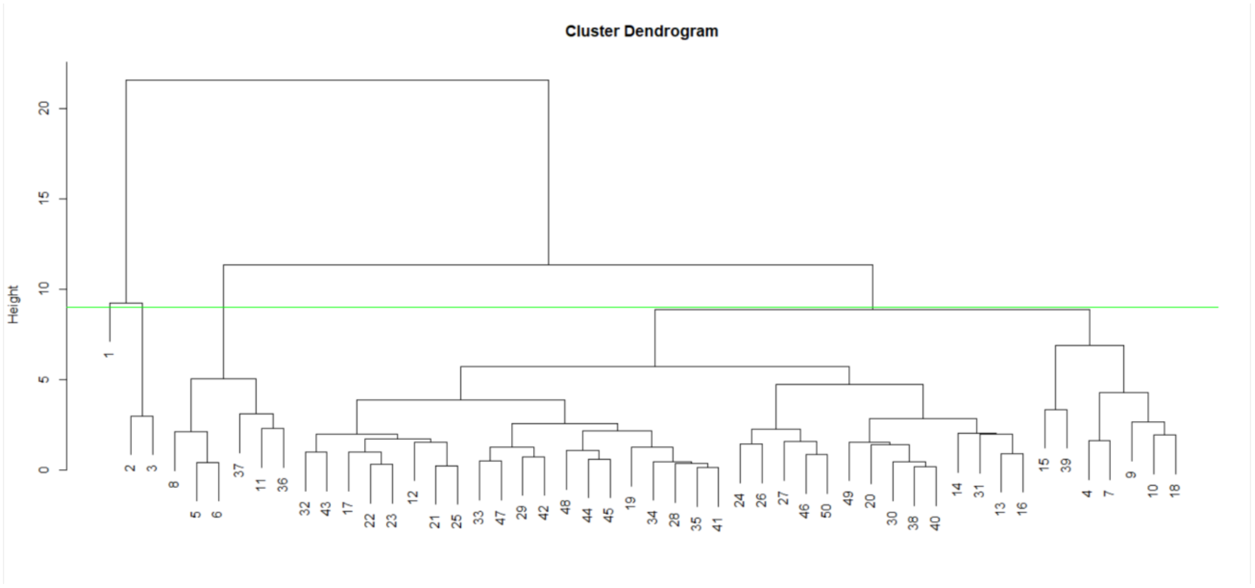
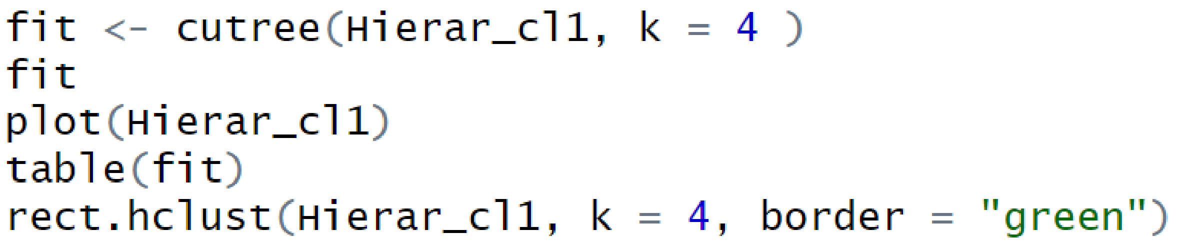


Figure 57 cutting point in dendrogram with Manhattan.

Now I fit group = 4 that we interest and plot with parameter is Hierar_cl1 and border group with green.



```
fit <- cutree(Hierar_cl1, k = 4 )
fit
plot(Hierar_cl1)
table(fit)
rect.hclust(Hierar_cl1, k = 4, border = "green")
```

Figure 58 set clustering k = 3 and plot with border by Manhattan.

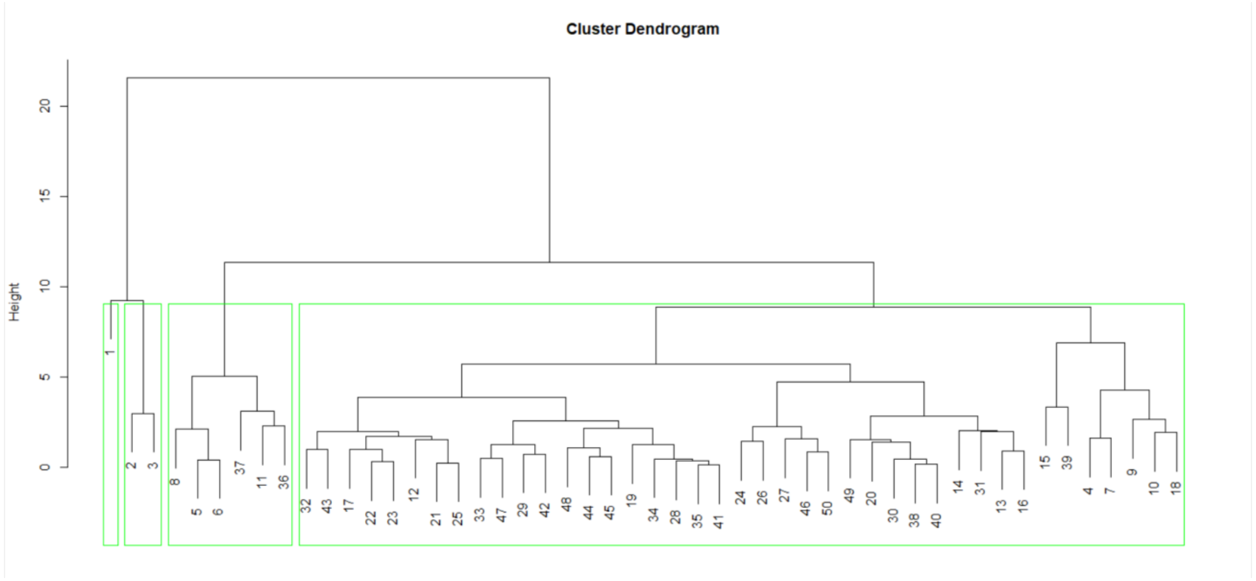
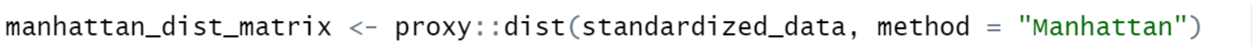


Figure 59 Dendrogram with clustering using Manhattan.

For all steps, We can conclude that when we choose k = 4 or 3 Group with data Top 300 YouTube Channels with 50 datapoint it should be 1 datapoint are cluster 1, 2 datapoint are cluster 2, 6 datapoint are cluster 3 and 41 datapoints are cluster 4.

7.3.3. DBSCAN (Manhattan distance)

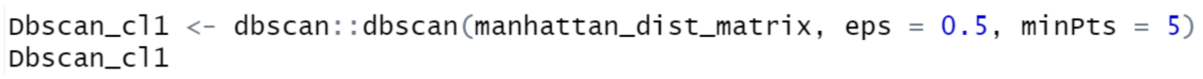
First step for DBScan, you need find a distance of Manhattan using proxy::dist by method Manhattan and keep it to variable Manhattan_dist_matrix.



```
manhattan_dist_matrix <- proxy::dist(standardized_data, method = "Manhattan")
```

Figure 60 Set parameter Manhattan.

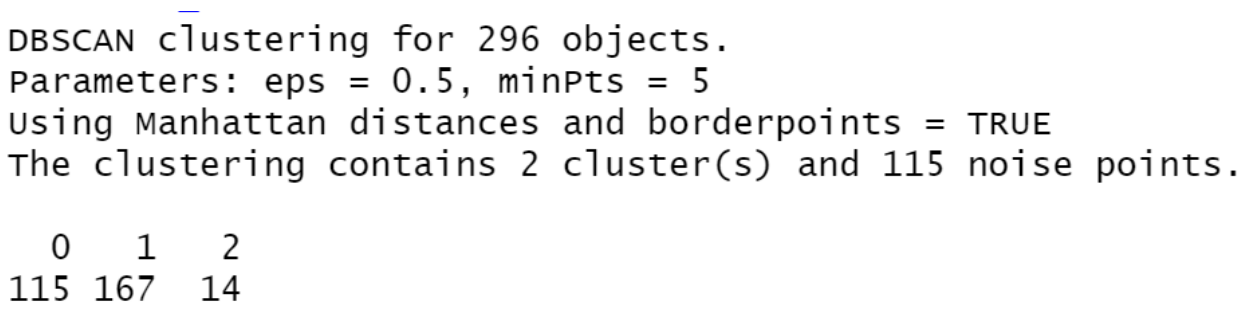
Next, you need to set 3 parameters as follows: eps, Minpts and distance.



```
Dbscan_cl1 <- dbscan::dbscan(manhattan_dist_matrix, eps = 0.5, minPts = 5)
Dbscan_cl1
```

Figure 61 Set parameter for DBScan.

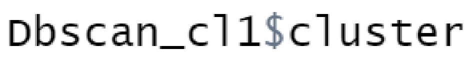
In the model, there are Pts with Minimum points are 5, eps is 0.5 and distance.



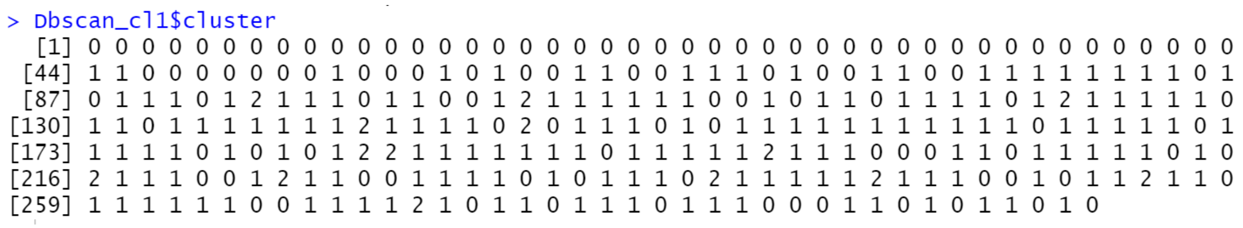
```
DBSCAN clustering for 296 objects.
Parameters: eps = 0.5, minPts = 5
Using Manhattan distances and borderpoints = TRUE
The clustering contains 2 cluster(s) and 115 noise points.

  0   1   2 
115 167  14 
```

Figure 62 Show a DBScan and output with Manhattan.

To show cluster identification, we use Dbscan_cl1$cluster.



```
Dbscan_cl1$cluster
```



```
> Dbscan_cl1$cluster
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [44] 1 1 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 1 0 0 1 1 1 0 1 0 0 1 1 0 0 1 1 1 1 1 1 1 1 0 1
 [87] 0 1 1 1 0 1 2 1 1 1 0 1 1 0 0 1 2 1 1 1 1 1 1 0 0 1 0 1 1 0 1 1 1 1 0 1 2 1 1 1 1 1 0
[130] 1 1 0 1 1 1 1 1 1 1 2 1 1 1 1 0 2 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1
[173] 1 1 1 1 0 1 0 1 0 1 2 2 1 1 1 1 1 1 1 0 1 1 1 1 1 2 1 1 1 0 0 0 1 1 0 1 1 1 1 1 1 0 1 0
[216] 2 1 1 1 0 0 1 2 1 1 0 0 1 1 1 1 0 1 0 1 1 1 0 2 1 1 1 1 1 2 1 1 1 0 0 1 0 1 1 2 1 1 0
[259] 1 1 1 1 1 1 0 0 1 1 1 1 2 1 0 1 1 0 1 1 1 0 1 1 1 0 0 0 1 1 0 1 0 1 1 0 1 0
```

Figure 63 Output with Cluster by DBScan using Manhattan.

Next, It plot. We recommend plotting every pair. We can plot a total of 6 pairs as follows.

1. Subscriber_Count and Video_Views.
2. Subscriber_Count and Video_Views.
3. Subscriver_count and Video_Count.
4. Subscriber_Count and Channel_Started.
5. Video_Views and Video_Count.
6. Video_Views and Channel_Started.
7. Video_Count and Channel_Started.

1. Subscriber_Count and Video_Views.

```
plot(Dbscan_cl1, standardized_data[, c(1,2)], main = "DBScan with manhattan")
```

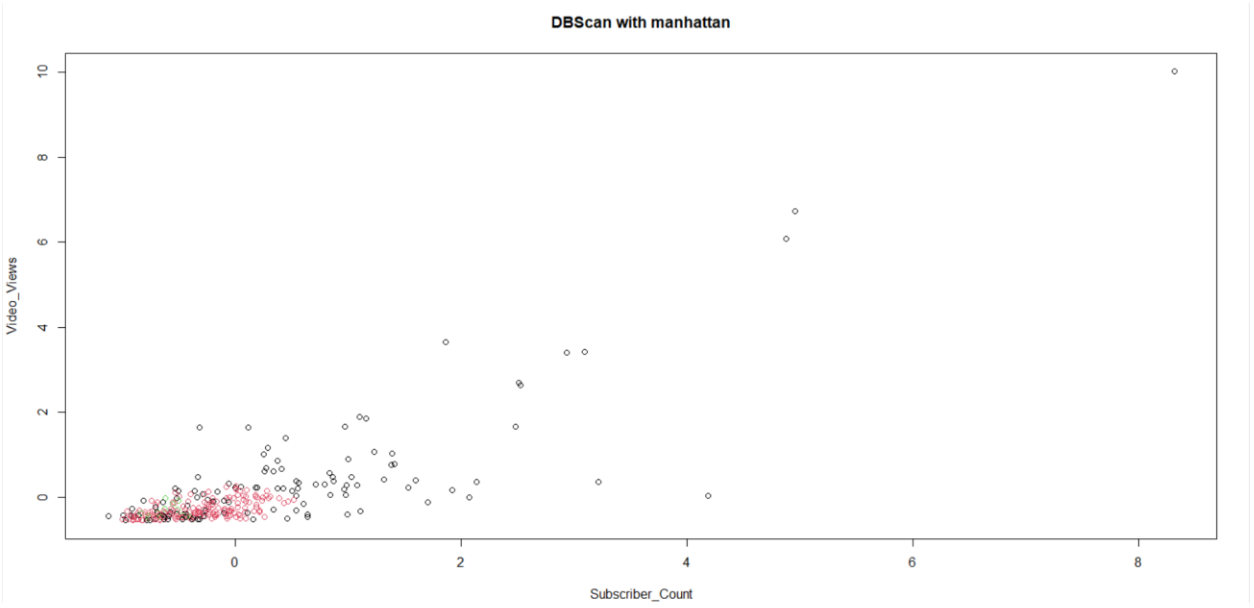


Figure 64 plot with Subscriber_Count and Video_Views with Manhattan.

2. Subscriber_Count and Video_Count.

```
plot(Dbscan_cl1, standardized_data[, c(1,3)], main = "DBScan with manhattan")
```

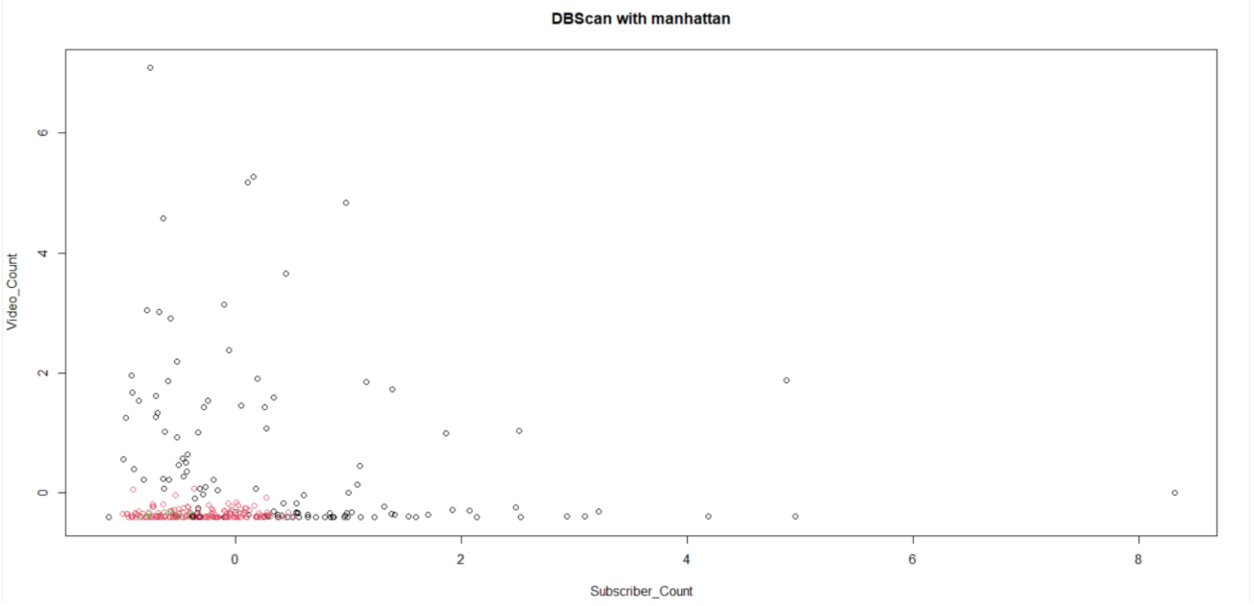


Figure 65 plot with Subscriver_count and Video_Count with Manhattan.

3. Subscriber_Count and Channel_Started.

```
plot(Dbscan_cl1, standardized_data[, c(1,4)], main = "DBScan with manhattan")
```

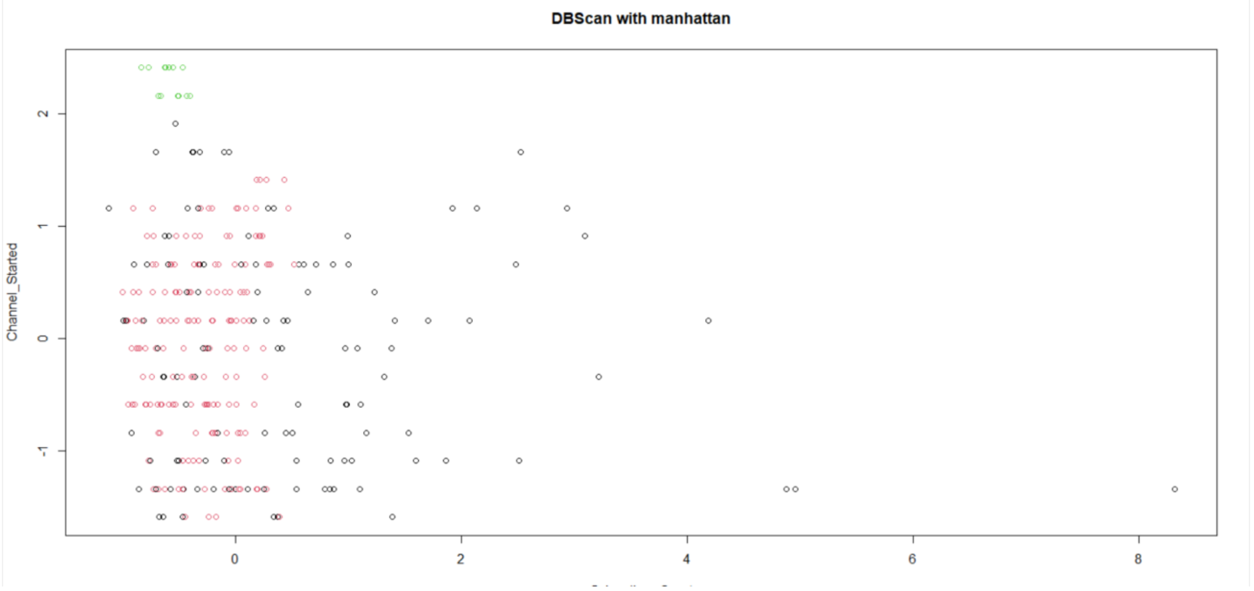


Figure 66 Subscriber_Count and Channel_Started with Manhattan.

4. Video_Views and Video_Count.

```
plot(Dbscan_cl1, standardized_data[, c(2,3)], main = "DBScan with manhattan")
```

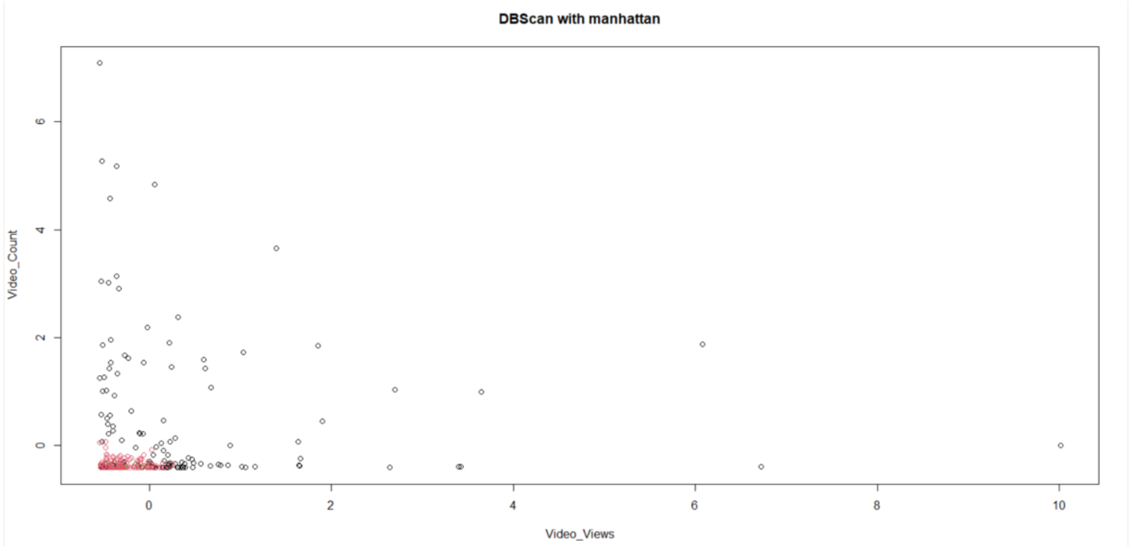


Figure 67 Video_Views and Video_Count with Manhattan.

5. Video_Views and Channel_Started.

```
plot(Dbscan_cl1, standardized_data[, c(2,4)], main = "DBScan with manhattan")
```

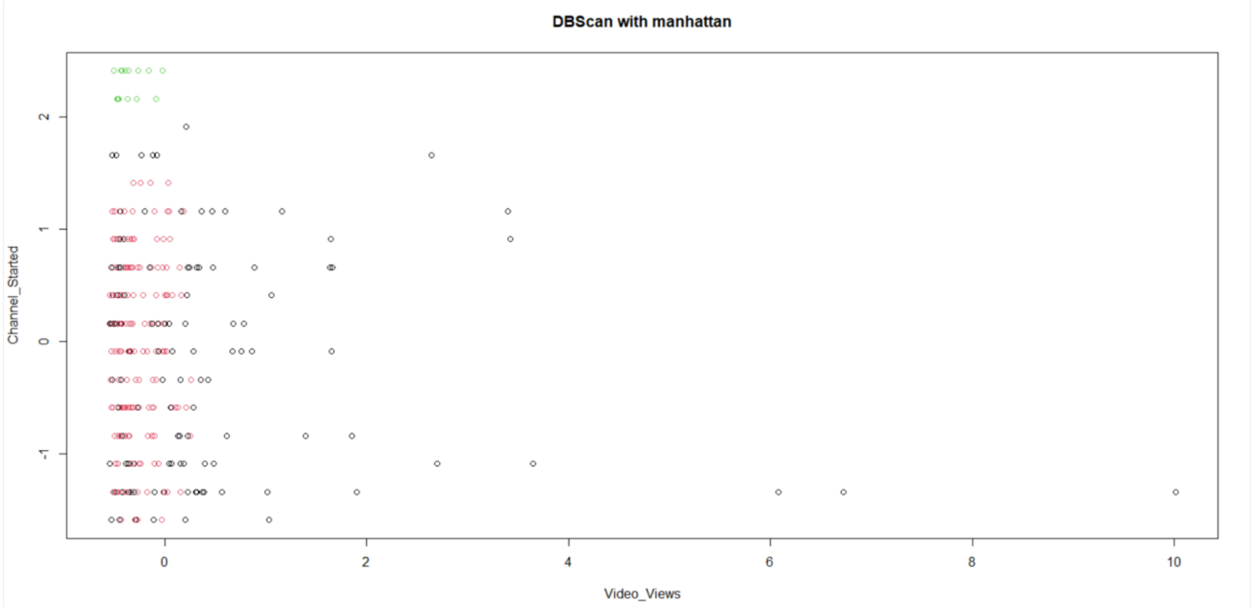


Figure 68 Video_Views and Channel_Started with Manhattan.

6. Video_Count and Channel_Started.

```
plot(Dbscan_cl1, standardized_data[, c(3,4)], main = "DBScan with manhattan")
```

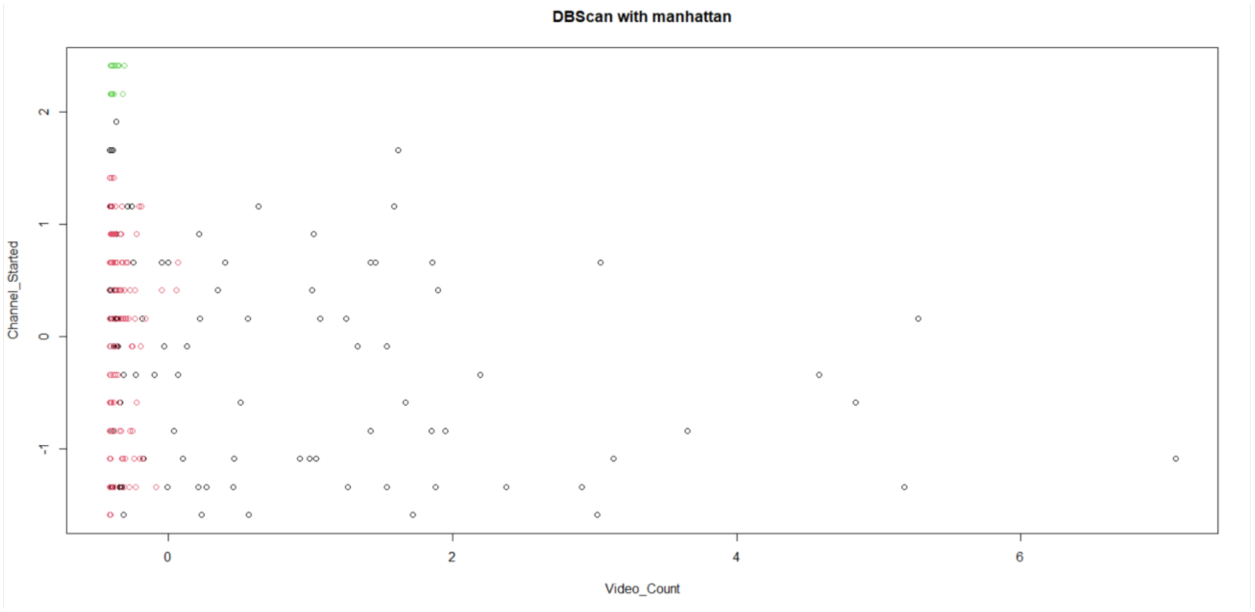


Figure 69 Video_Count and Channel_Started with Manhattan.

For all steps, We can concluded that when we set 3 parameter including eps = 0.5, Minpts = 5 and Manhattan distance in Top 300 YouTube Channels data it should be 167 datapoint are cluster 1, 14 datapoint are cluster 2 and 115 datapoint are noise point.

8. Reference

geeksforgeeks.org. (2020). **DBScan Clustering in R Programming.** Retrieved September 27, 2023, https://www.geeksforgeeks.org/dbscan-clustering-in-r-programming/

geeksforgeeks.org. (2020). **K-Means Clustering in R Programming.** Retrieved September 27, 2023, from https://www.geeksforgeeks.org/k-means-clustering-in-r-programming/

geeksforgeeks.org. (2021). **K Hierarchical Clustering in R Programming.** Retrieved September 27, 2023, from https://www.geeksforgeeks.org/hierarchical-clustering-in-r-programming/

Mathematics Learning Support Centre. (2007). **Cluster Analysis**. Retrieved September 27, 2023, from https://www.statstutor.ac.uk/resources/uploaded/clusteranalysis.pdf