Cluster Analysis

PRESENT BY

MR. CHIRAT SUWANNACHOTE
MISS. WANISSARA CHONGCHAI
MR. SITTHATKA JARUTSANG

PRESENT TO

DR.PREM JUNSAWANG

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.

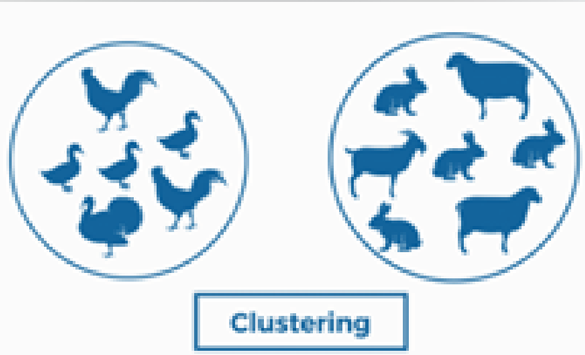
Clustering can be done using methods such as K-Means Clustering, Hierarchical Clustering, DBSCAN, and there are many ways to evaluate the cost-effectiveness of clustering.

However, the selection and evaluation of clustering methods is important to obtain quality results and use them in understanding and decision-making.

Difference between Clustering and Classification

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".

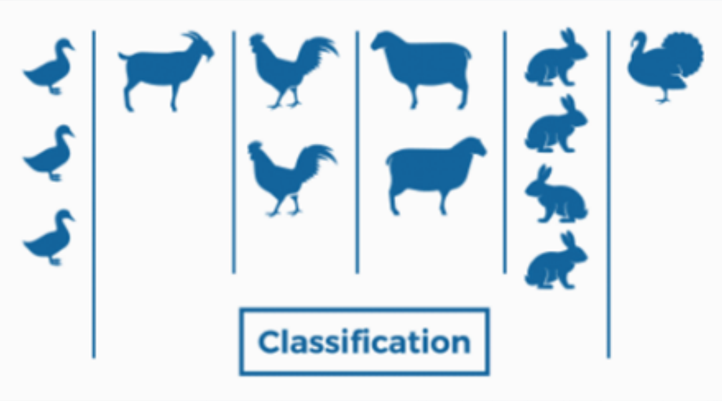


Difference between Clustering and Classification

Classification Analysis

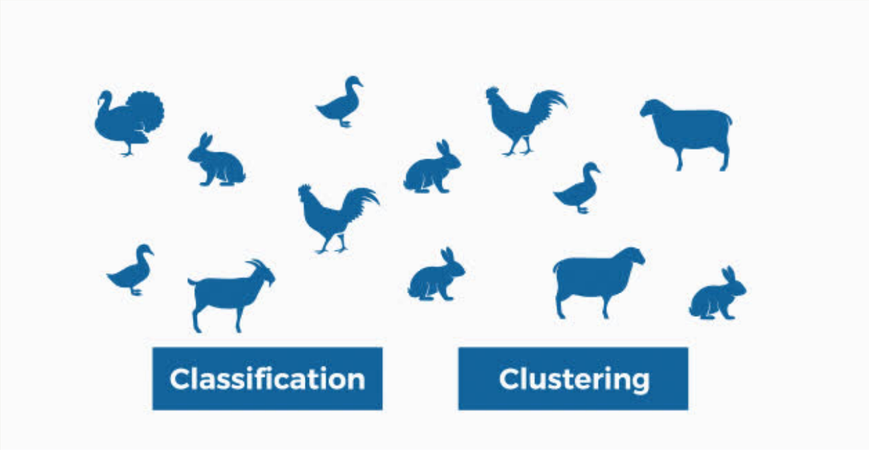
Classification It is a process used to assign categories or classes to data based on learning from training data with answers or labels given in each data sample.

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.



Difference between Clustering and 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.



Types of data and measures of distance

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.

Types of data and measures of distance

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.

Types of data and measures of distance

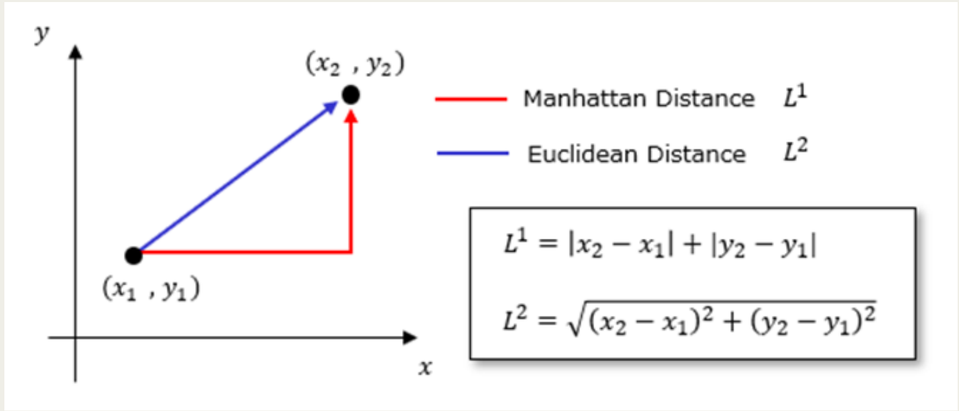
measures of distance

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.

Types of data and measures of distance

measures of distance

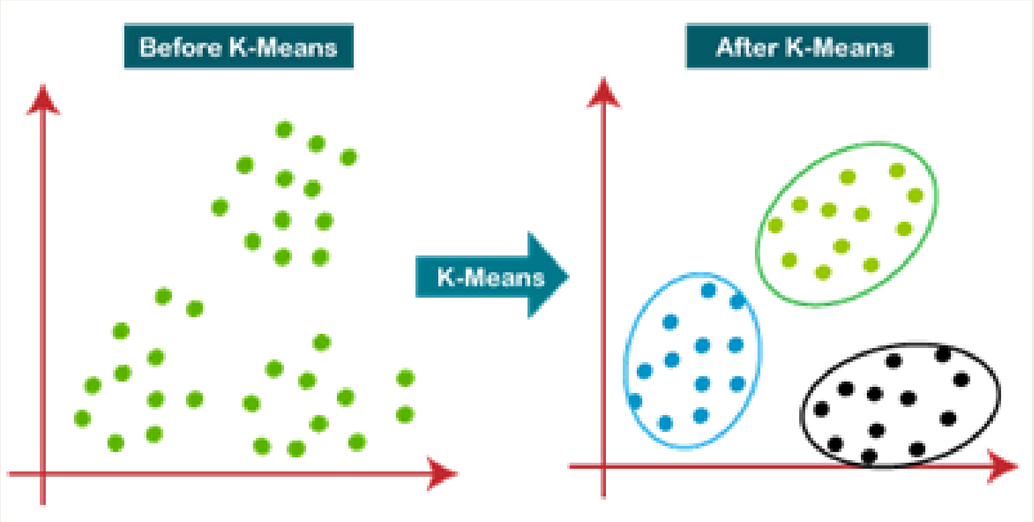
The Manhattan distance is a method of calculating the distance between two points in Euclidean space. This distance calculation is called "Manhattan" because of the method of walking in New York (New York City) that must follow a grid of streets or a grid. It's called Manhattan walking because it follows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



Clustering Methods

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.



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.

Theory of K-Means

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.

Theory of K-Means

Looping:

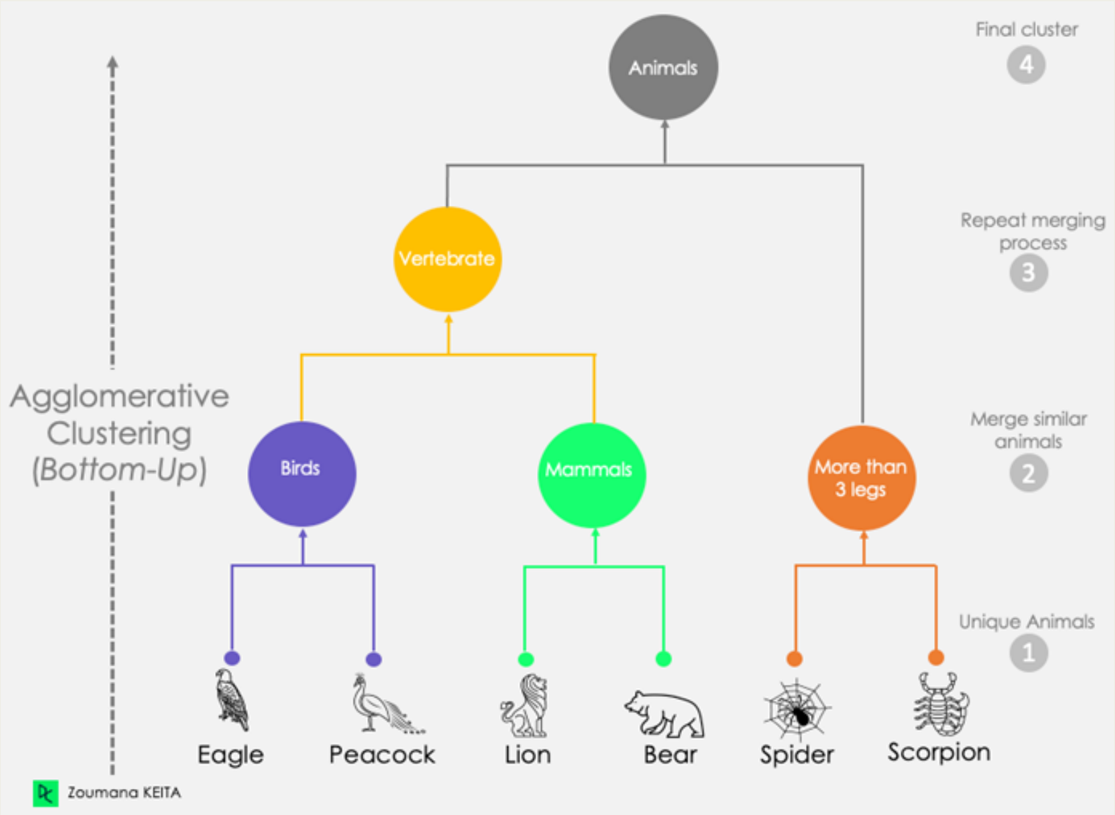
Steps 3 and 4 are repeated until the cluster centroid does not change further, or the specified stopping condition is reached.

Termination:

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

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.



Hierarchical Clustering

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.

Hierarchical Clustering

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.

Hierarchical Clustering

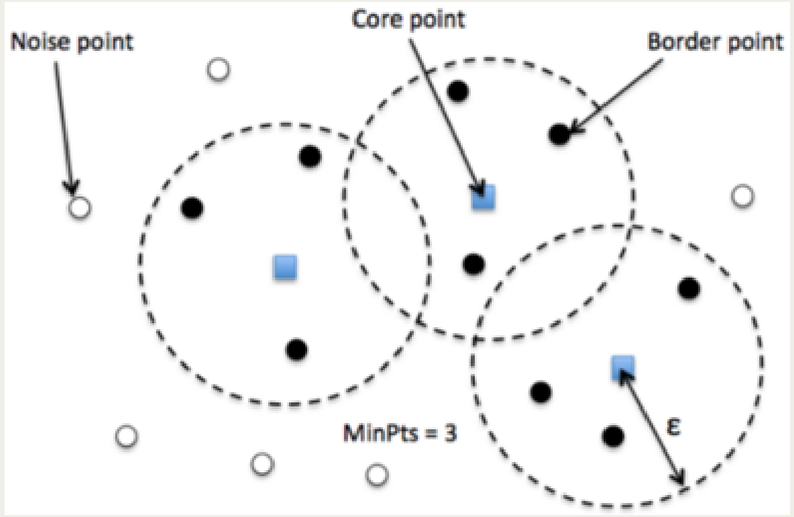
Creating a grouping table (Dendrogram):

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.

DBScan

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.



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

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.

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

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.

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

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.

Test in R

Step in Test R

1

Data preparation

2

Clustering with Euclidean distance

3

Clustering with Manhattan distance

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	ٹ□ש 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 mʊzik	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

1

Data preparation

all package to use for Test by R.

```
library(DataExplorer)
library(clusterR)
library(cluster)
library(ggfortify)
library(stats)
library(fpc)
```

1

Data preparation

Check missing value and get new data

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

```
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
```

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

1

Data preparation

transform to standardization

	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

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

1

Data preparation

eyesball data

	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

2

Clustering with Euclidean distance.

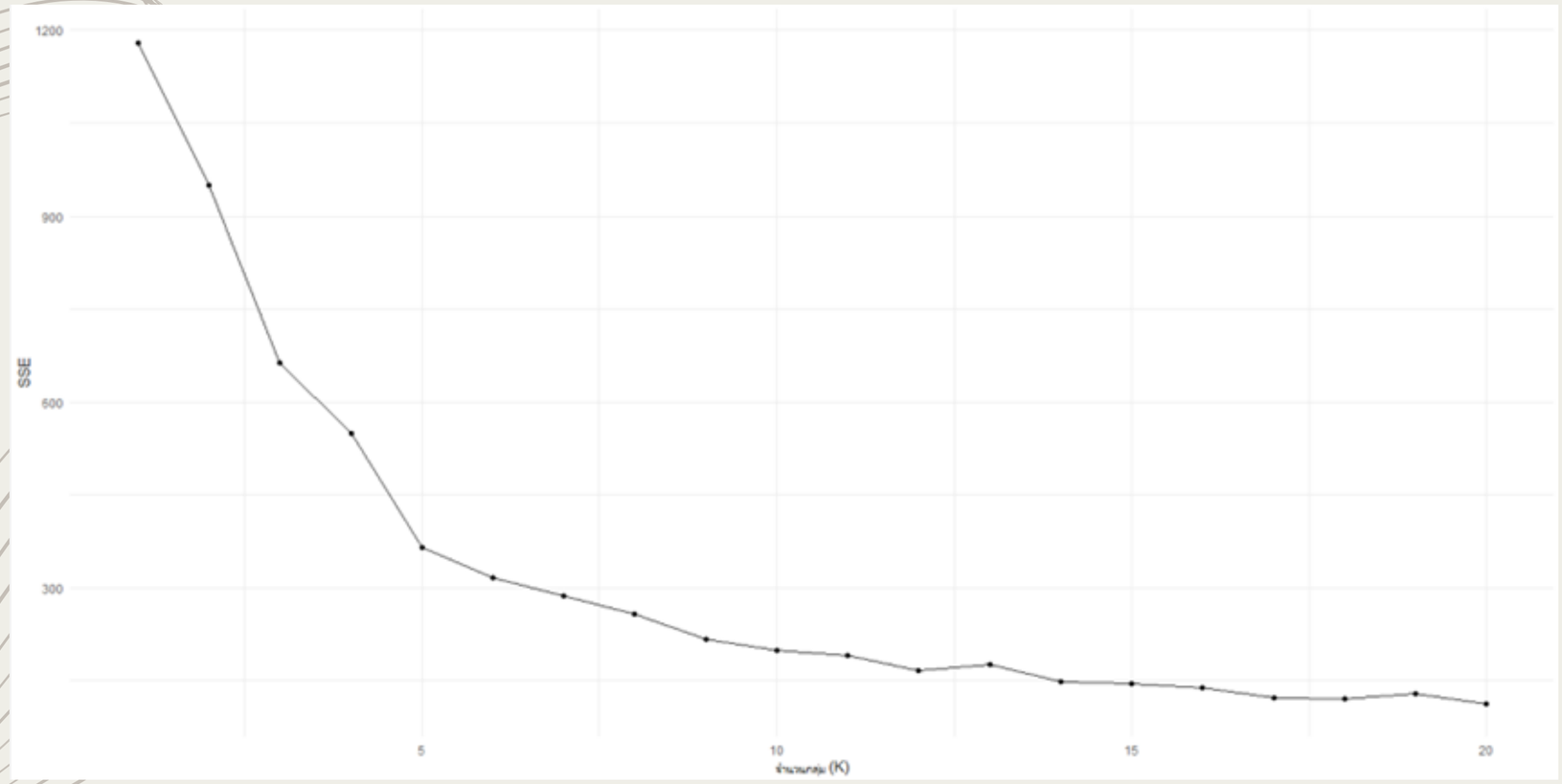
K-Means (Euclidean distance)

```r
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()
```

2

Clustering with Euclidean distance.

elbow method



2

Clustering with Euclidean distance.

cluster by K-means

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

```
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
```

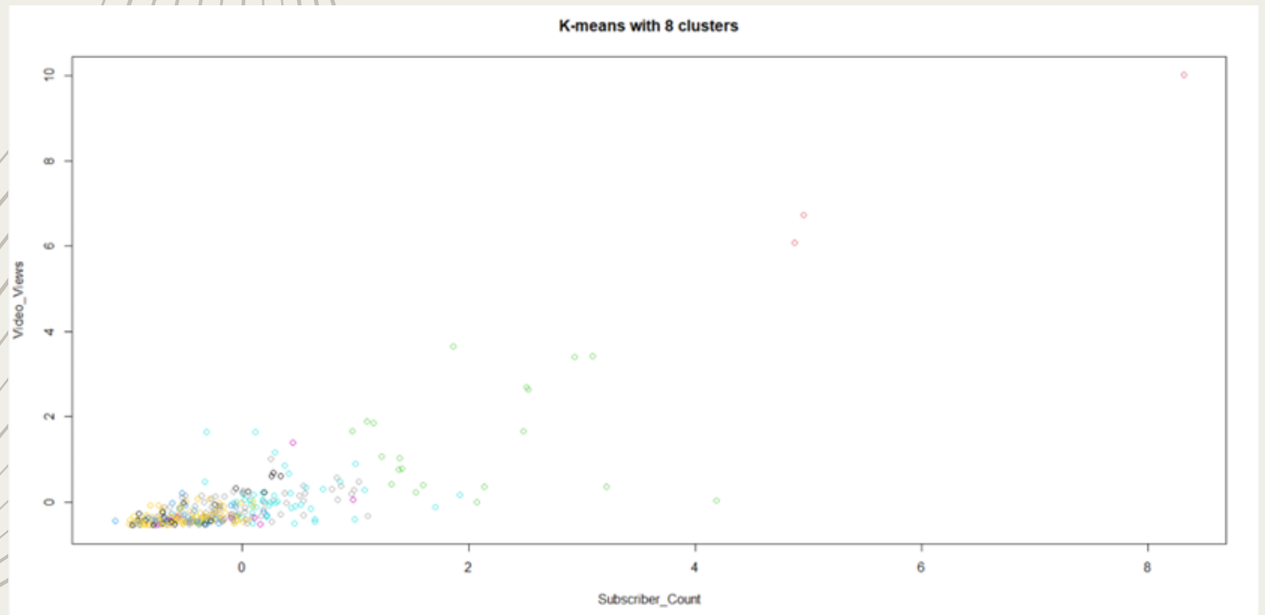
```
Within cluster sum of squares by cluster:
[1] 26.19422 19.55406 68.41612 11.55215 27.80609 23.48528 22.17321 48.12948
 (between_SS / total_SS =  79.0 %)
```

2

Clustering with Euclidean distance.

Subscriber_Count and Video_Views.

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

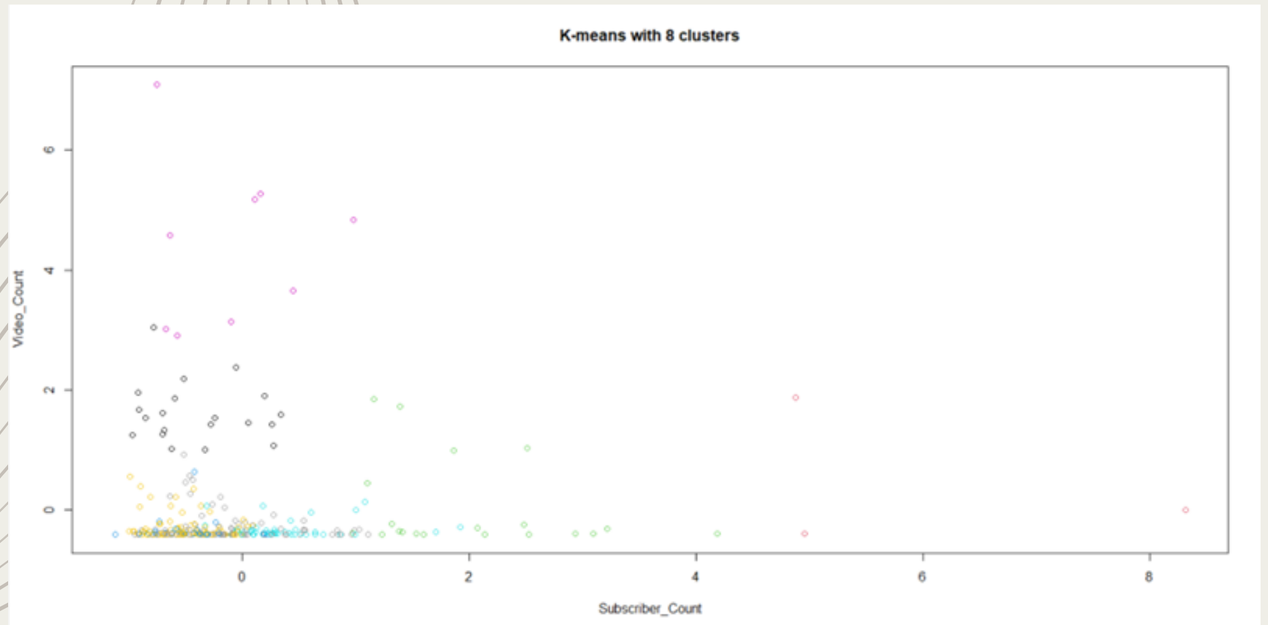


2

Clustering with Euclidean distance.

Subscriber_Count and Video_Count.

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

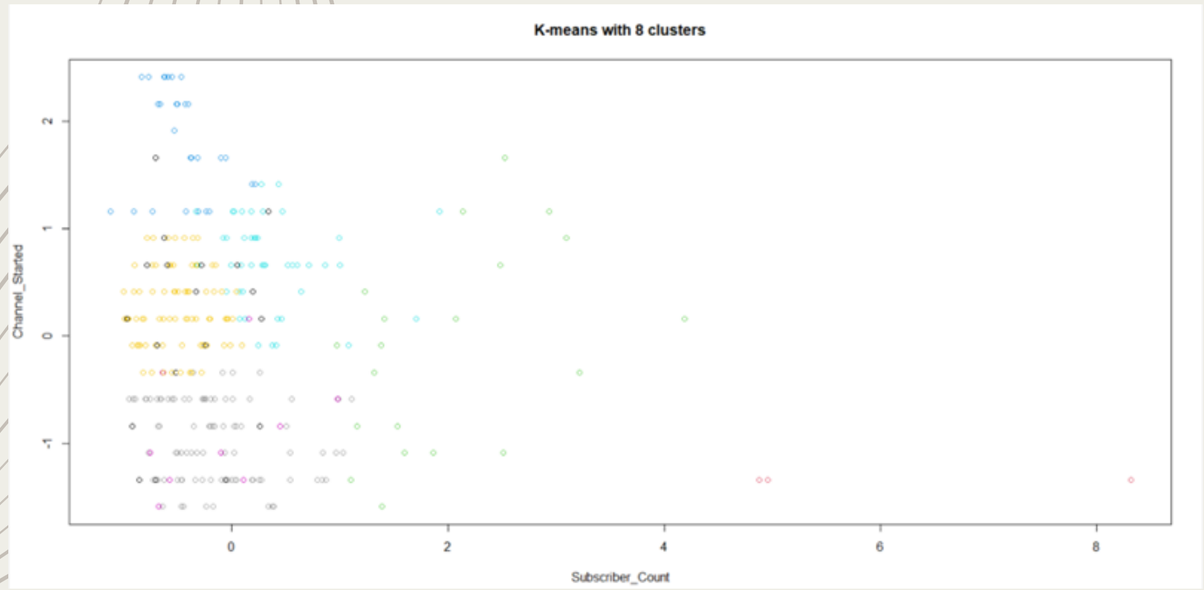


2

Clustering with Euclidean distance.

Subscriber_Count and Channel_Started.

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

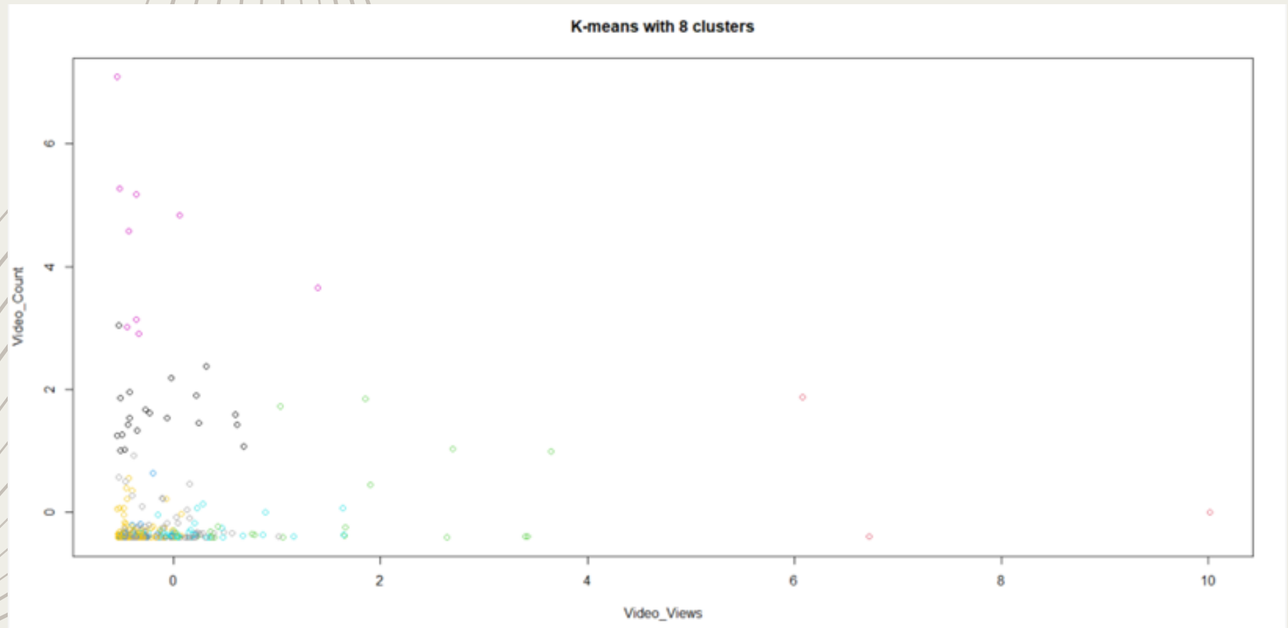


2

Clustering with Euclidean distance.

Video_Views and Video_Count.

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

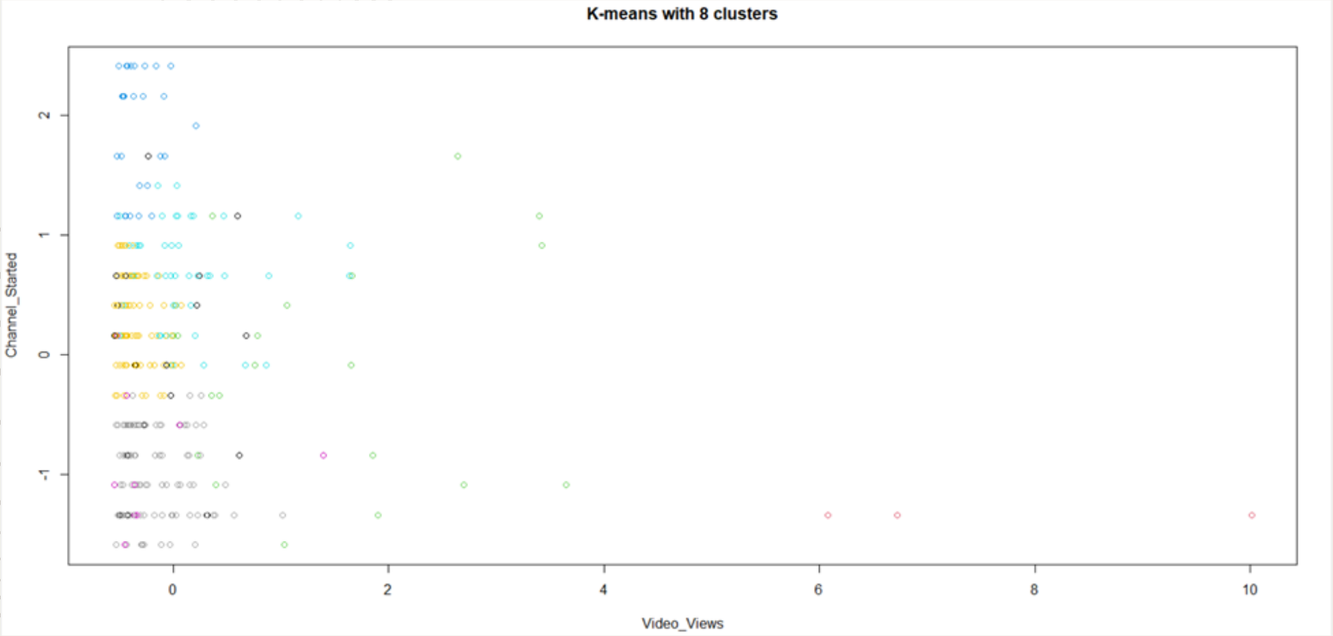


2

Video_Views and Channel_Started.

Clustering with Euclidean distance.

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

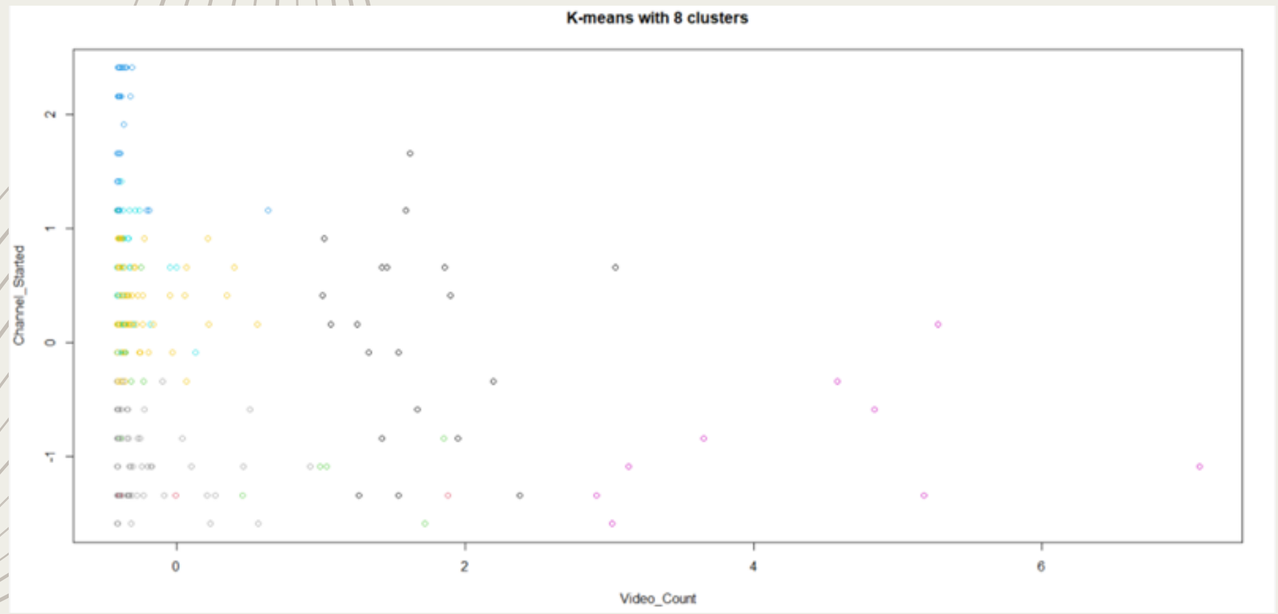


2

Video_Count and Channel_Started.

Clustering with Euclidean distance.

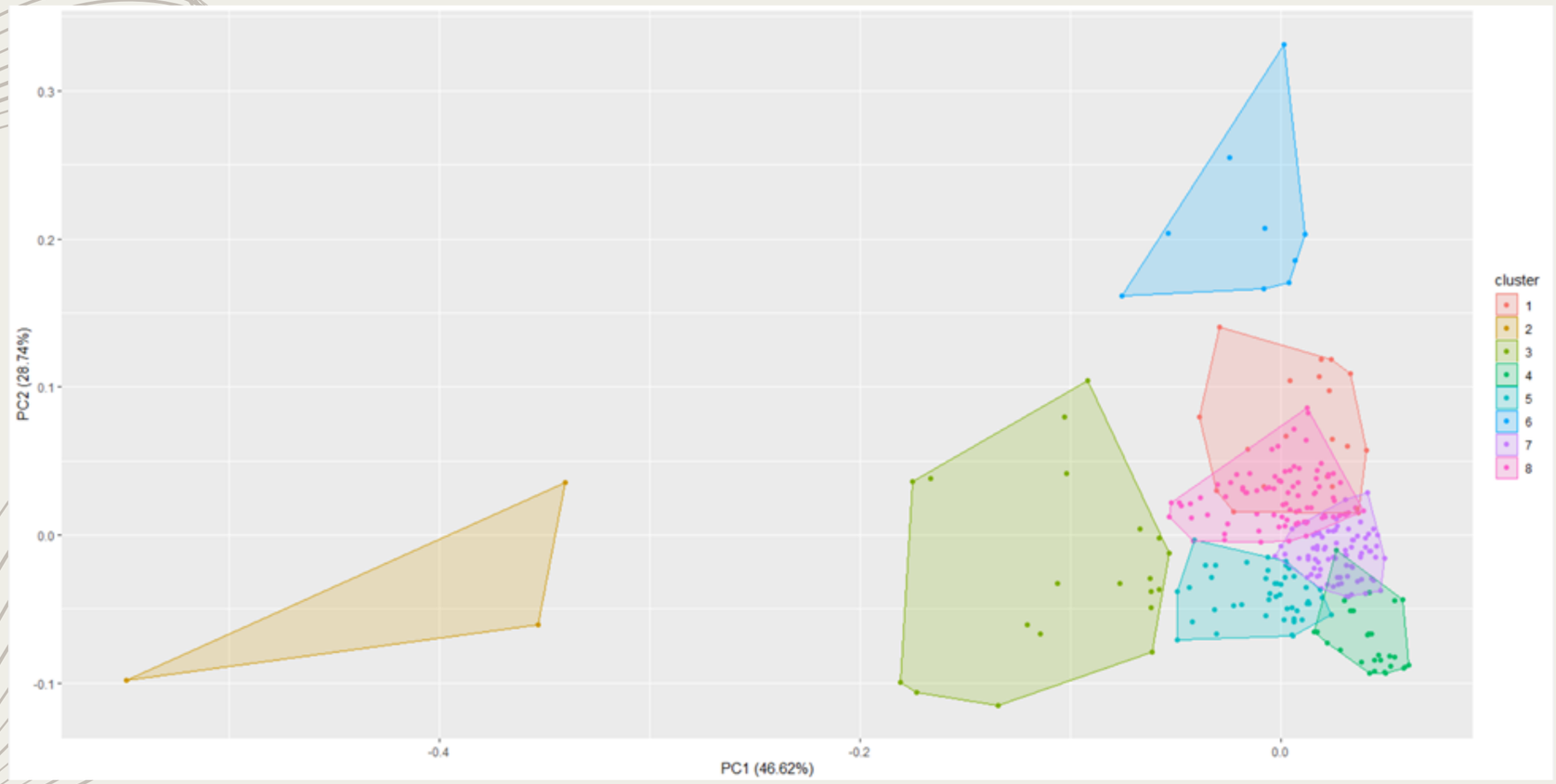
```
plot(standardized_data[, c(3, 4)],
     col = kmeans_result$cluster,
     main = "K-means with 8 clusters")
```



2

visualization

Clustering with Euclidean distance.



2

Hierarchical Clustering (Euclidean distance)

Clustering with Euclidean distance.

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

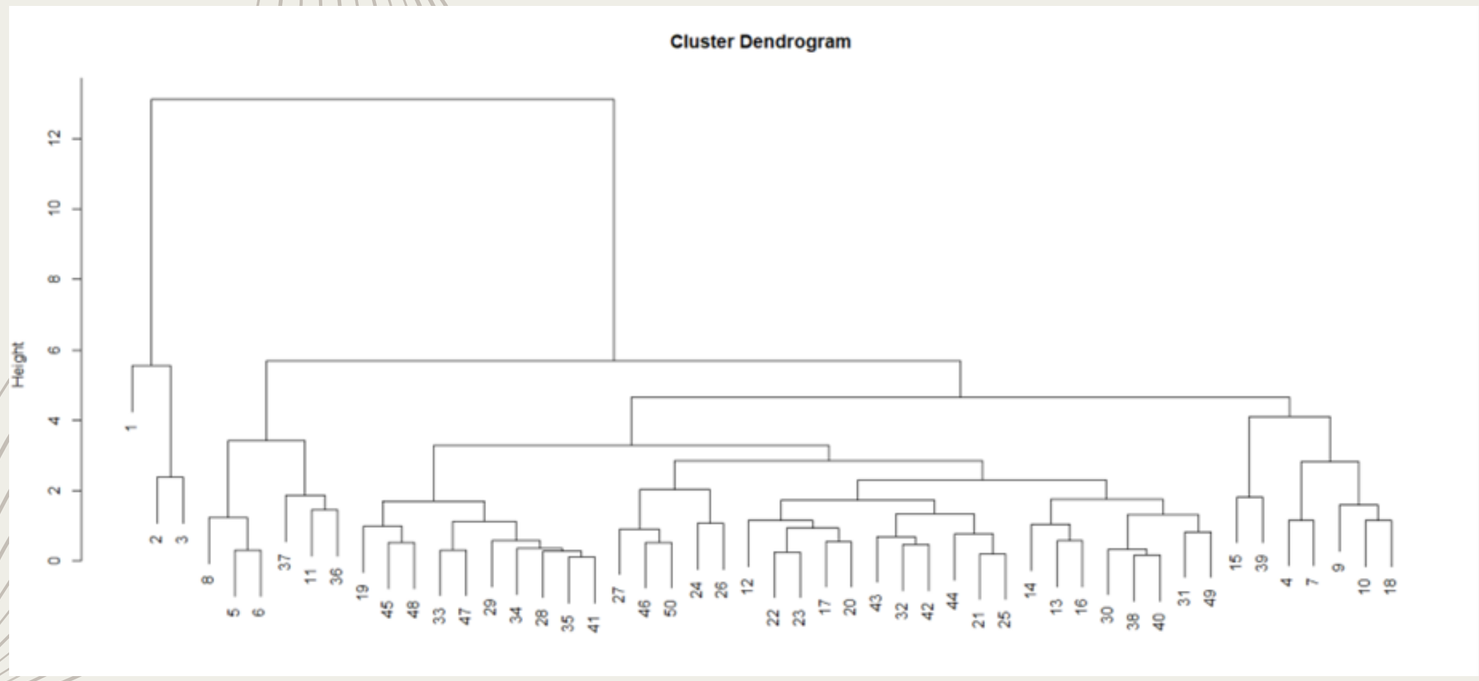
```
Call:
hclust(d = distance_mat)

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

2

Clustering with Euclidean distance.

plot dendrogram

```
plot(Hierar_cl)
```

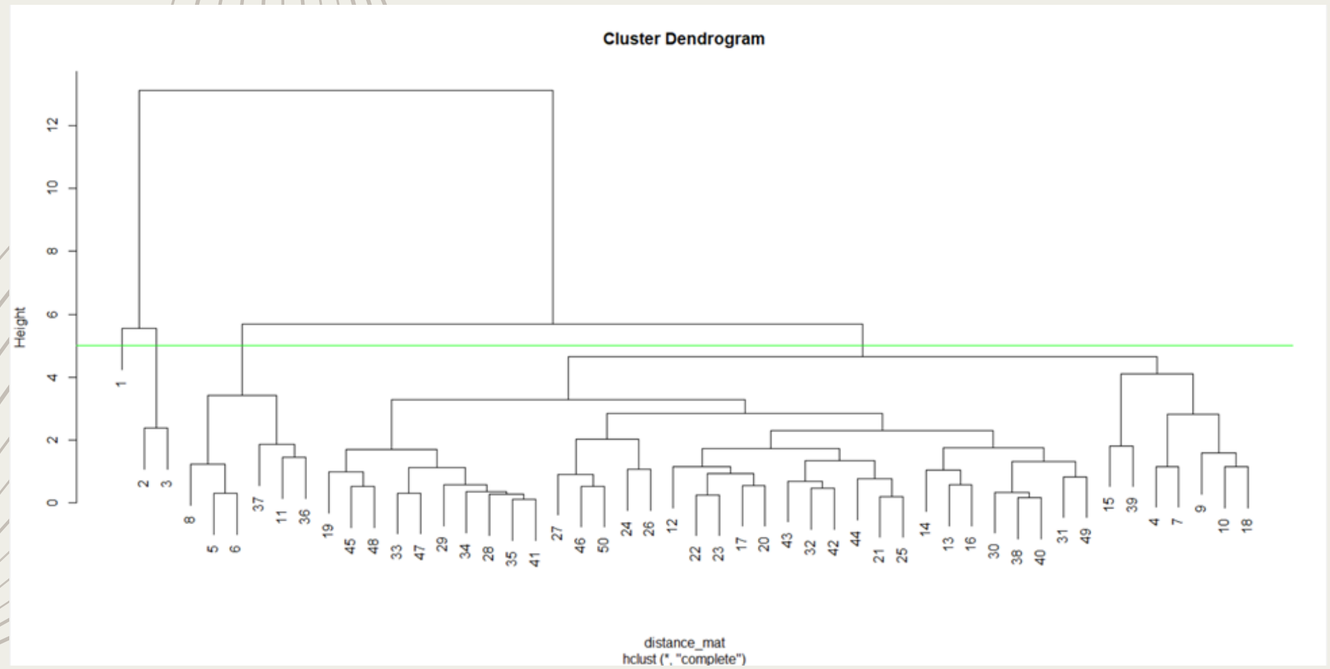


2

Clustering with Euclidean distance.

cutting point dendrogram

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

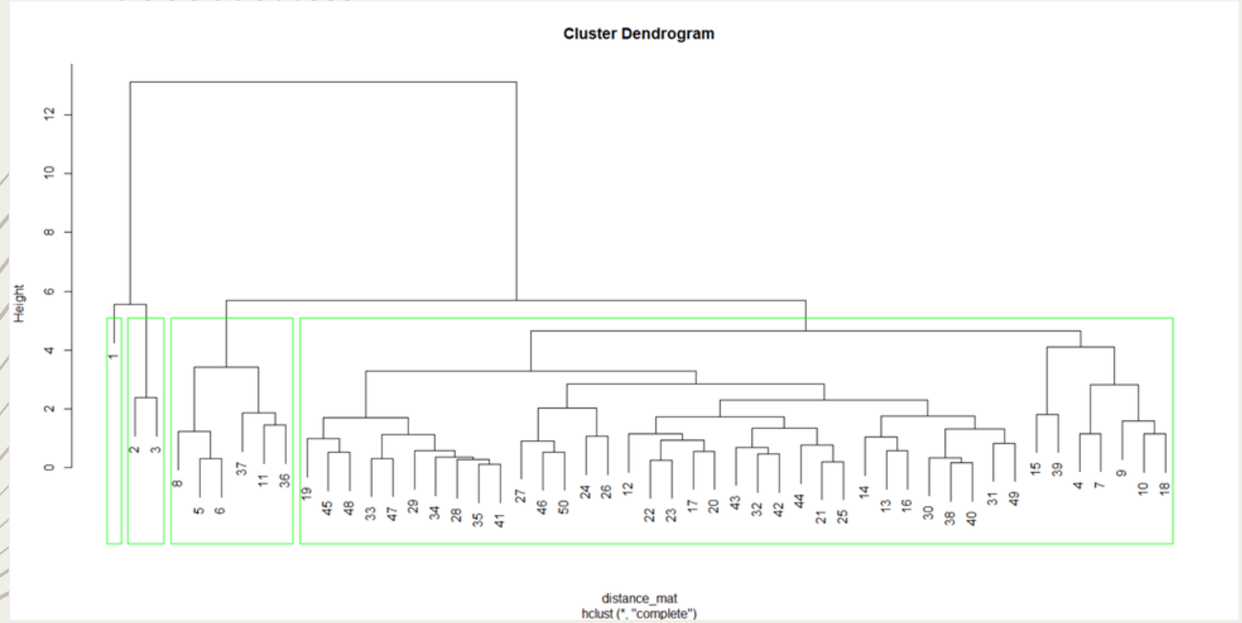


2

Clustering with Euclidean distance.

choose number group and see each group

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



2

Clustering with Euclidean distance.

DBSCAN (Euclidean distance)

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

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

2

Clustering with Euclidean distance.

see all each cluster

```
Dbscan_cl$cluster
```

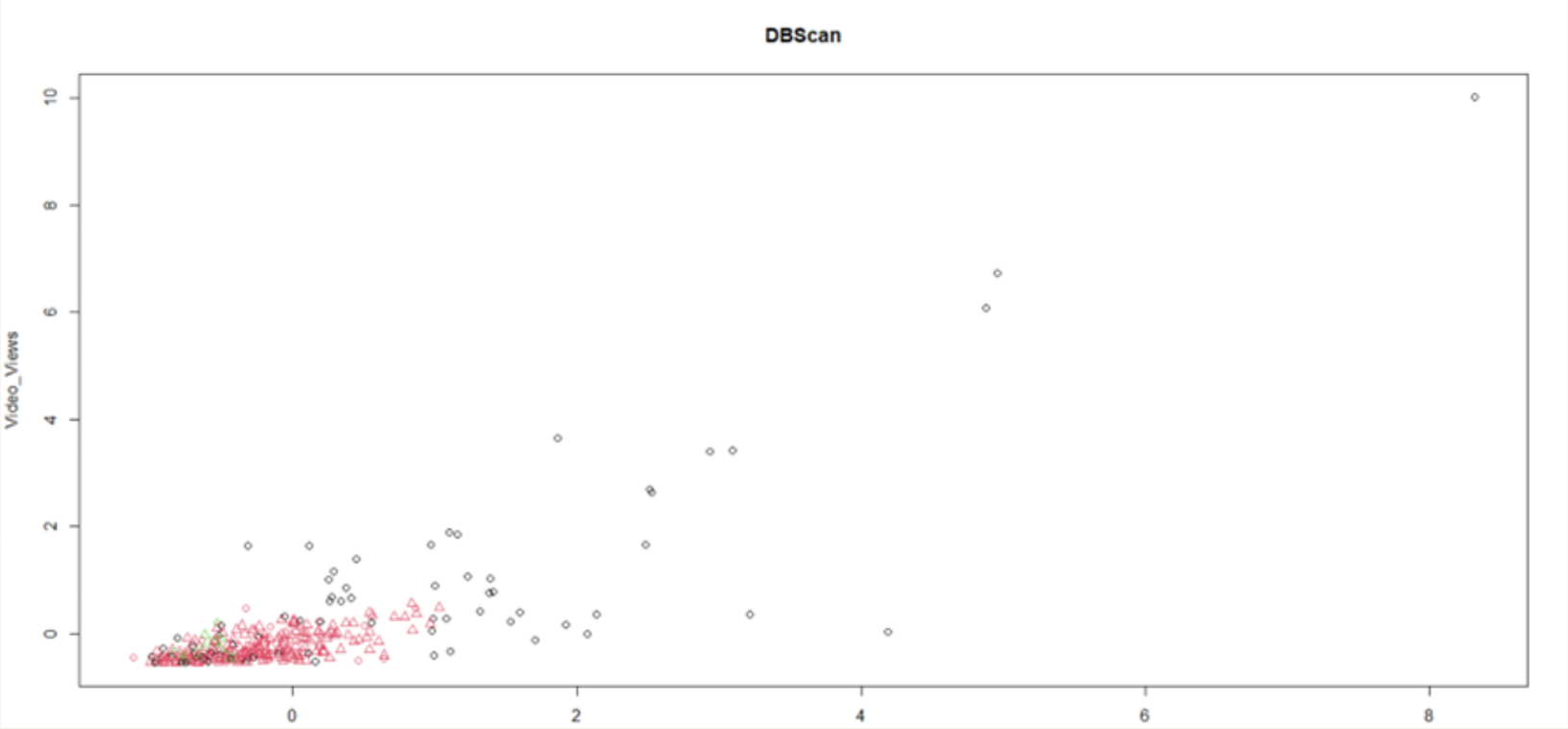
```
> Dbscan_cl$cluster
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 0 1 1 0 0
 [44] 1 1 0 0 1 1 0 2 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 0 1
 [87] 1 1 1 1 1 0 1 2 1 1 1 0 1 1 0 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 2 1 1 1 1 1 0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

2

Clustering with Euclidean distance.

Subscriber_Count and Video_Views.

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

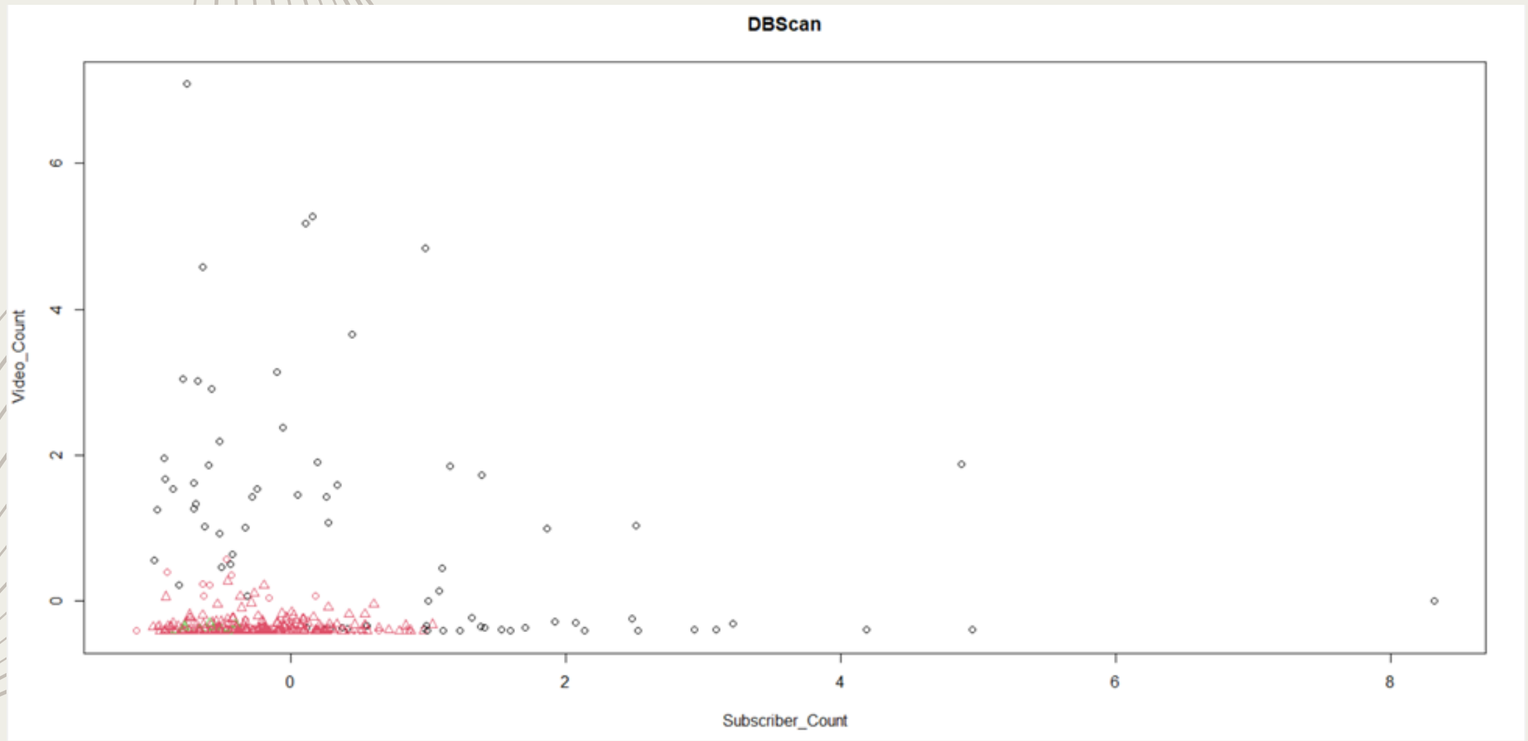


2

Clustering with Euclidean distance.

Subscriber_Count and Video_Count.

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

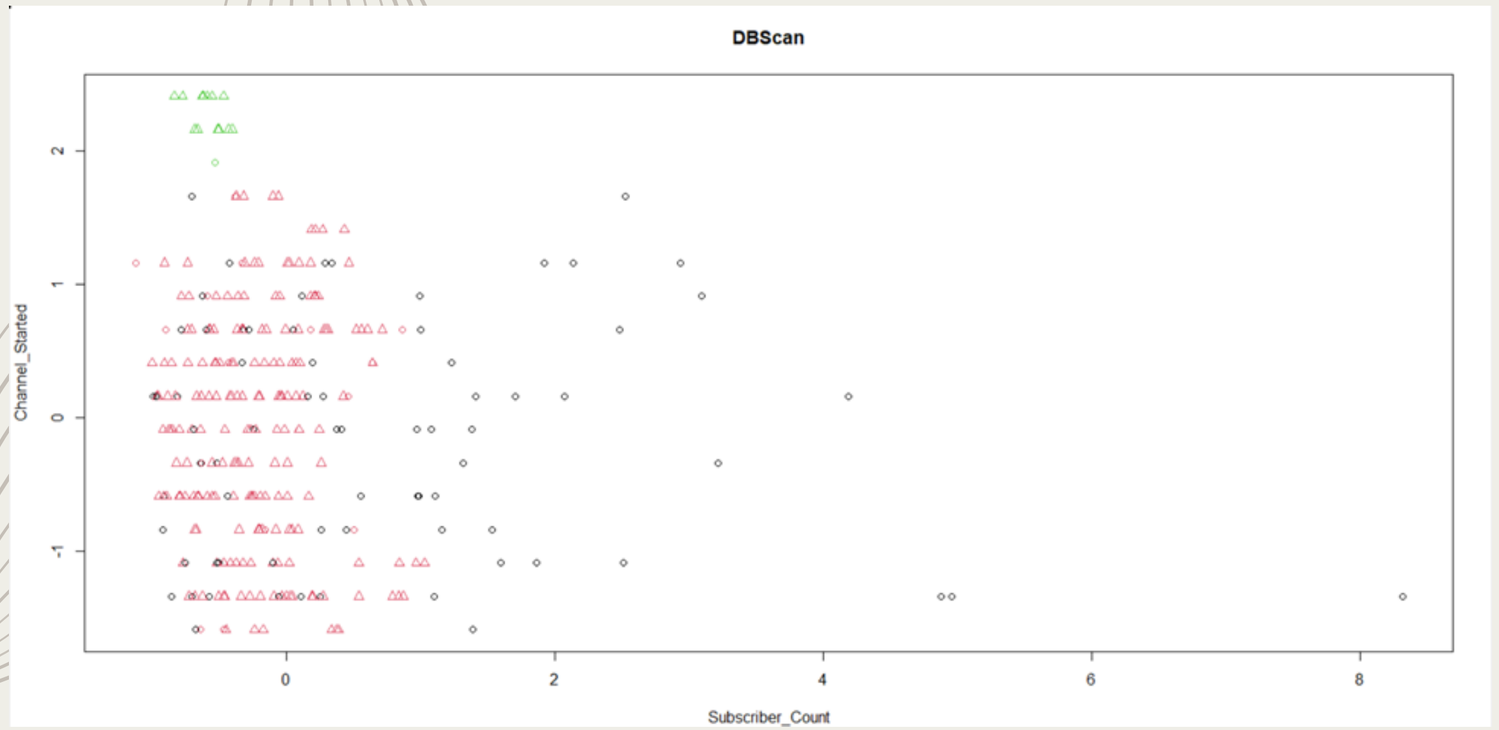


2

Clustering with Euclidean distance.

Subscriber_Count and Channel_Started.

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

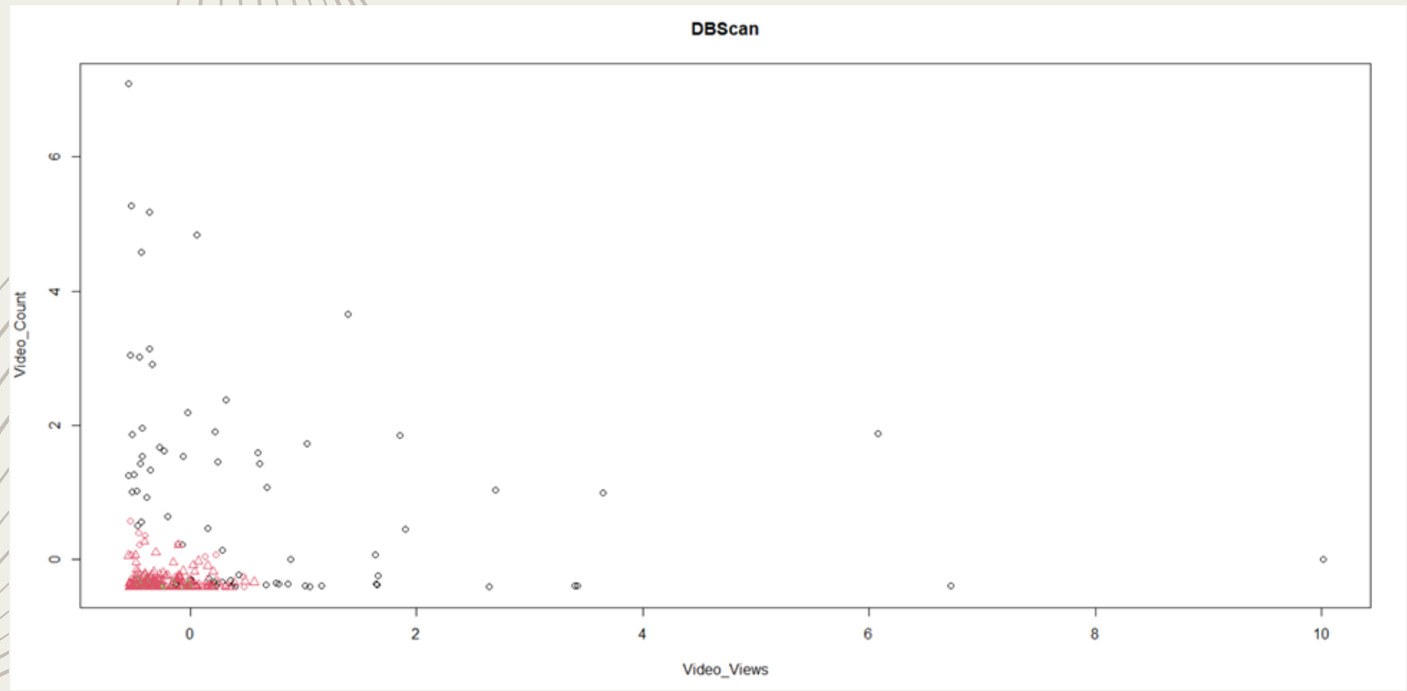


2

Clustering with Euclidean distance.

Video_Views and Video_Count.

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

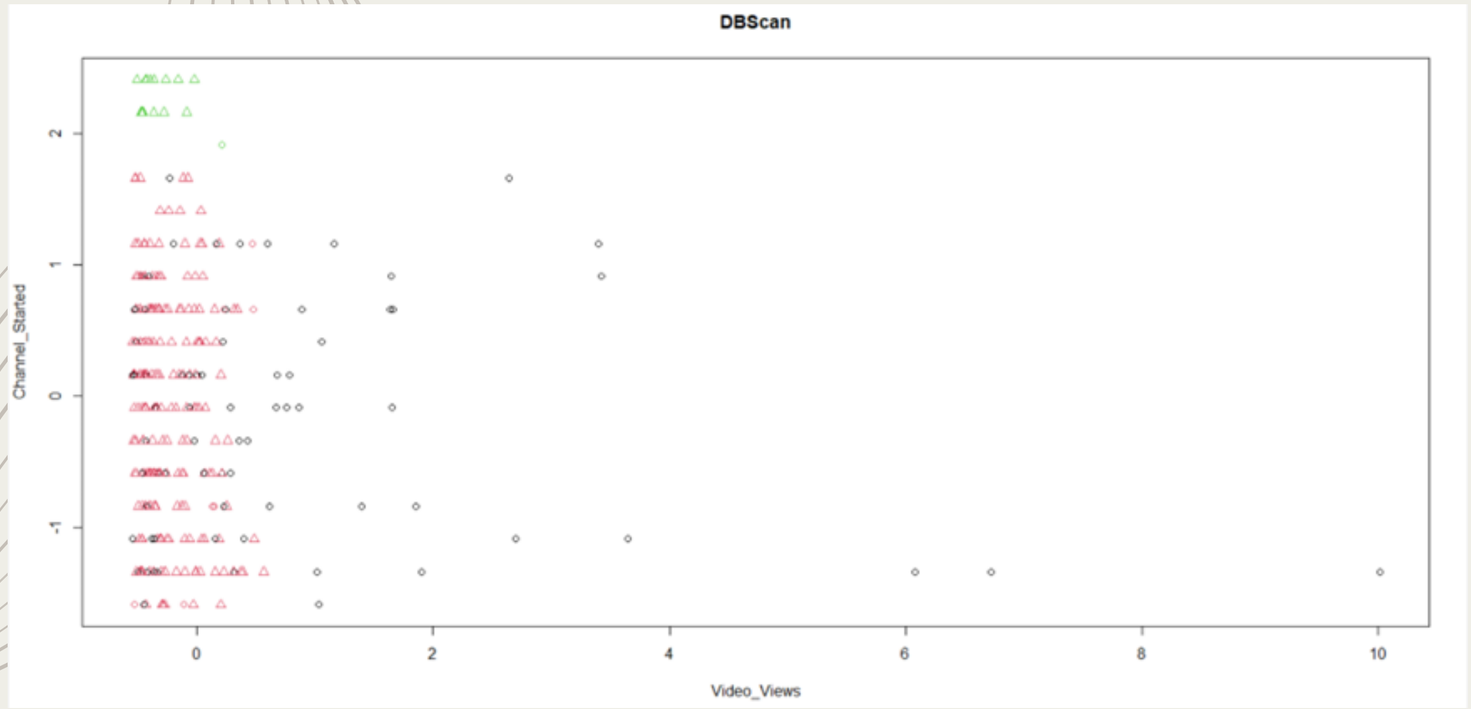


2

Clustering with Euclidean distance.

Video_Views and Channel_Started.

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



3

Clustering with Manhattan distance

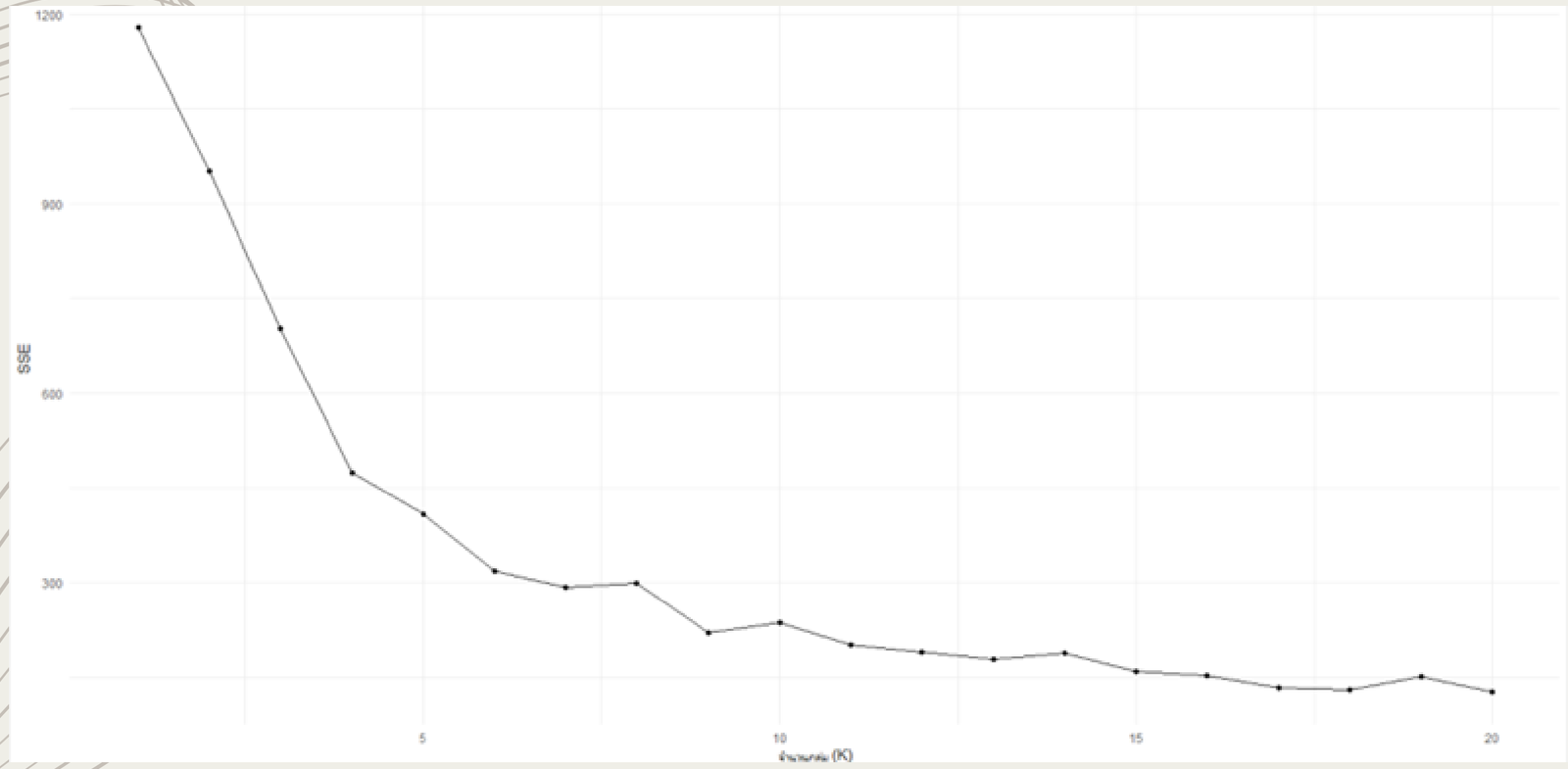
K-Means (Manhattan distance)

```
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()
```

3

Clustering with Manhattan distance

K-Means (Manhattan distance)



3

Clustering with Manhattan distance

K-Means (Manhattan distance)

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

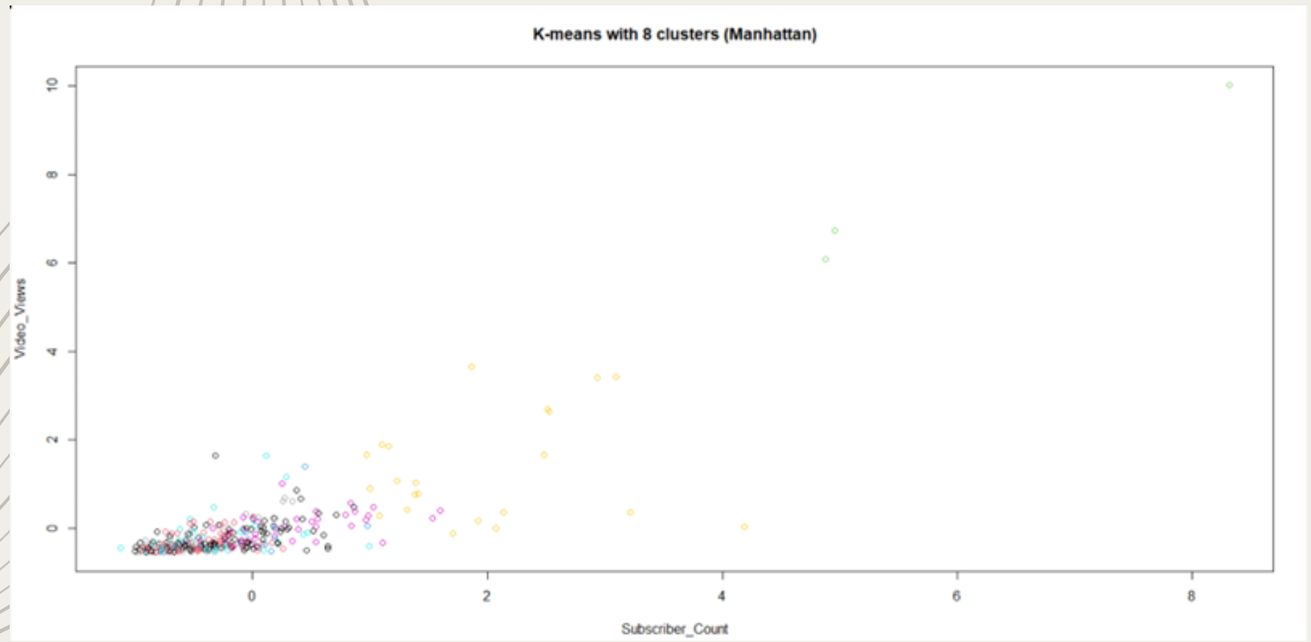
```
Within cluster sum of squares by cluster:
[1] 44.15744 20.26248 19.55406 23.48528 27.27562 16.54239 72.70658 27.93985
 (between_SS / total_SS =  78.7 %)
```

3

Clustering with Manhattan distance

Subscriber_Count and Video_Views.

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

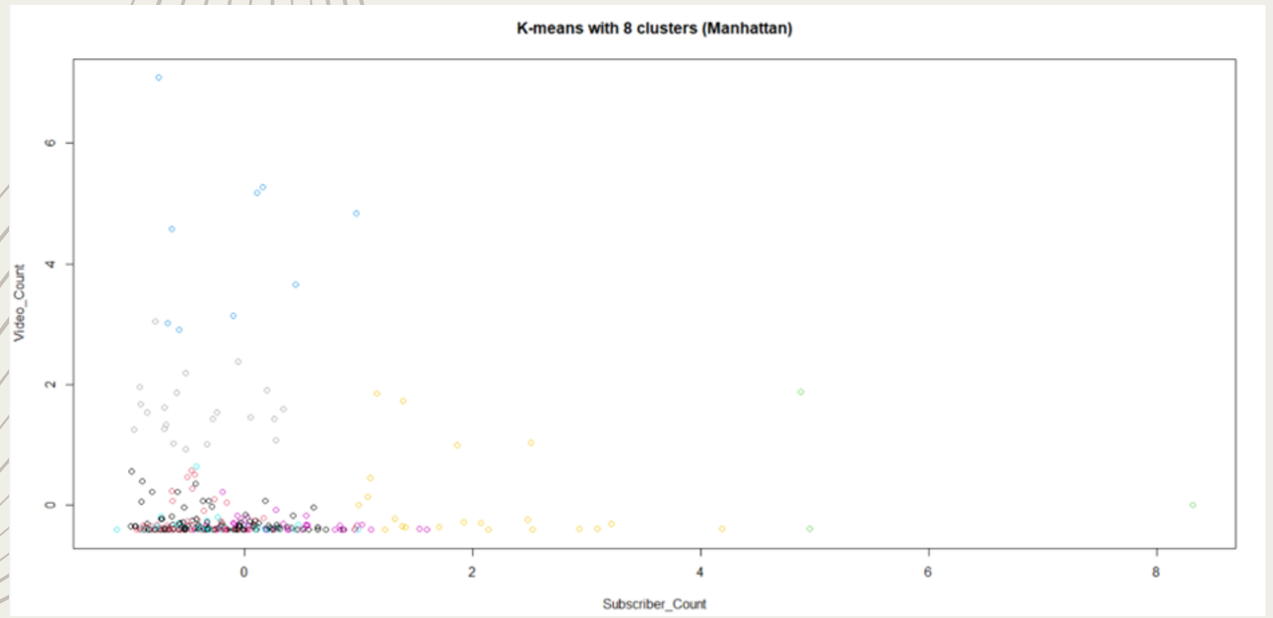


3

Clustering with Manhattan distance

Subscriber_Count and Video_Count.

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

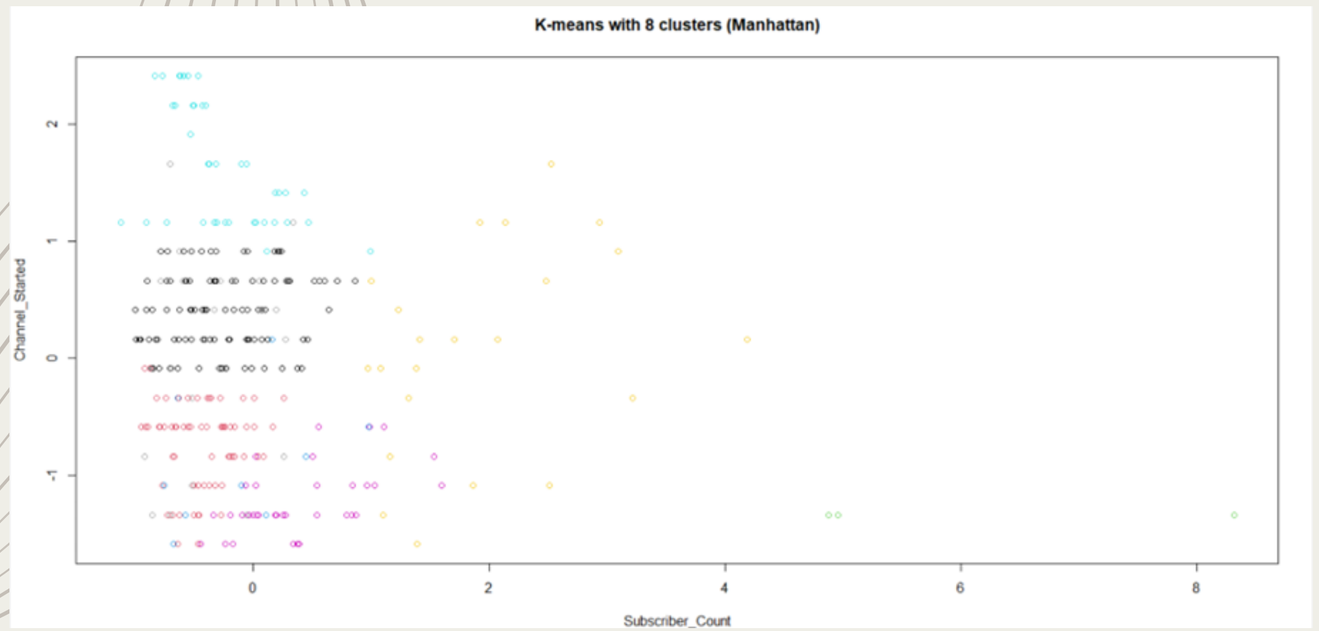


3

Clustering with Manhattan distance

Subscriber_Count and Channel_Started.

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

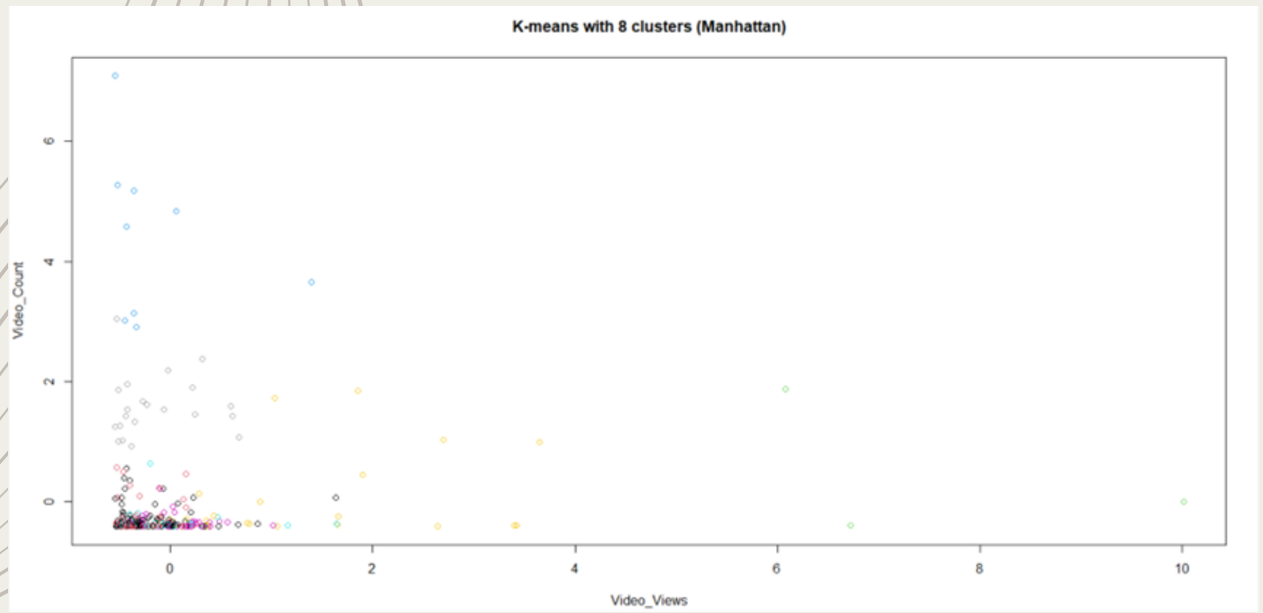


3

Clustering with Manhattan distance

Video_Views and Video_Count.

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

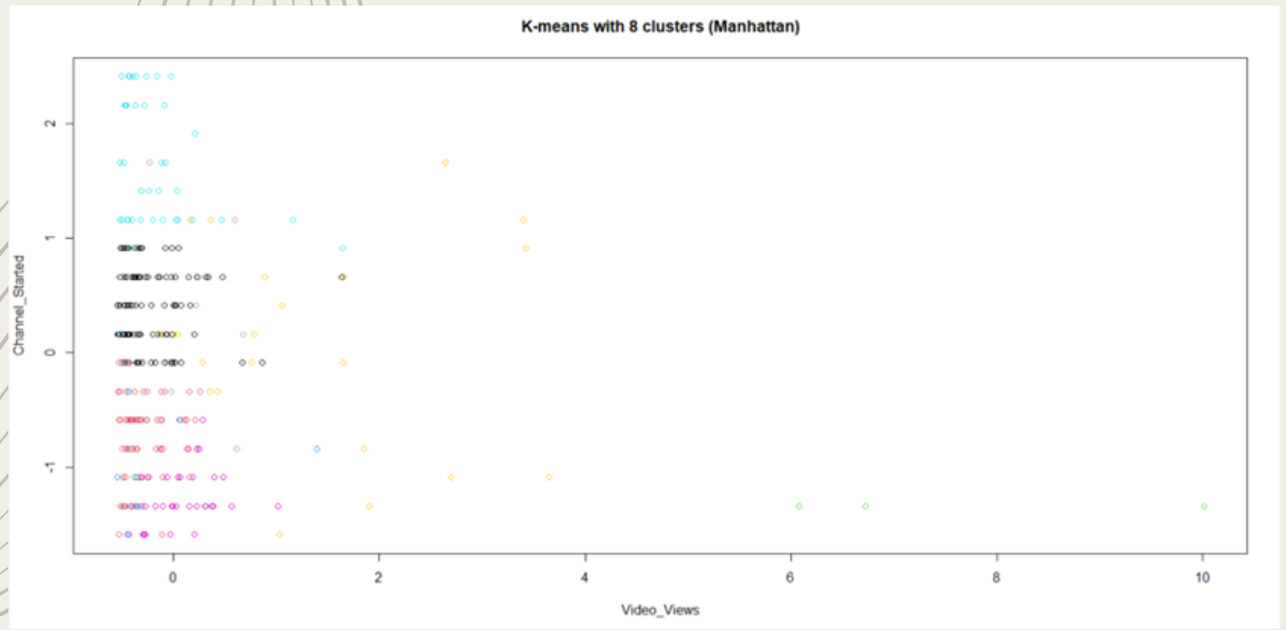


3

Clustering with Manhattan distance

Video_Views and Channel_Started.

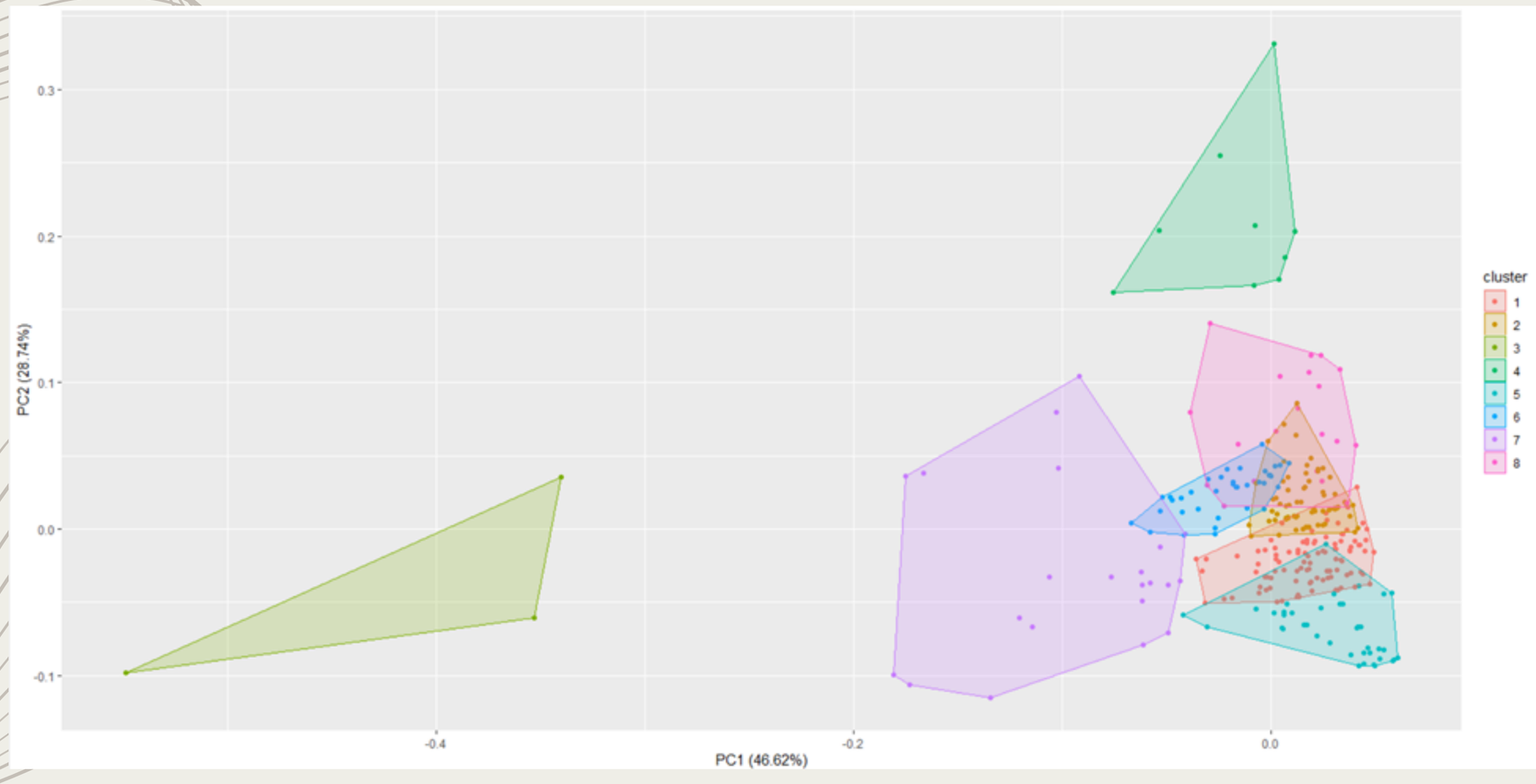
```
plot(standardized_data[, c(2, 4)],
     col = kmeans_model_manhattan$cluster,
     main = "K-means with 8 clusters (Manhattan)")
```



3

Clustering with Manhattan distance

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



3

Clustering with Manhattan distance

Hierarchical Clustering (Manhattan distance)

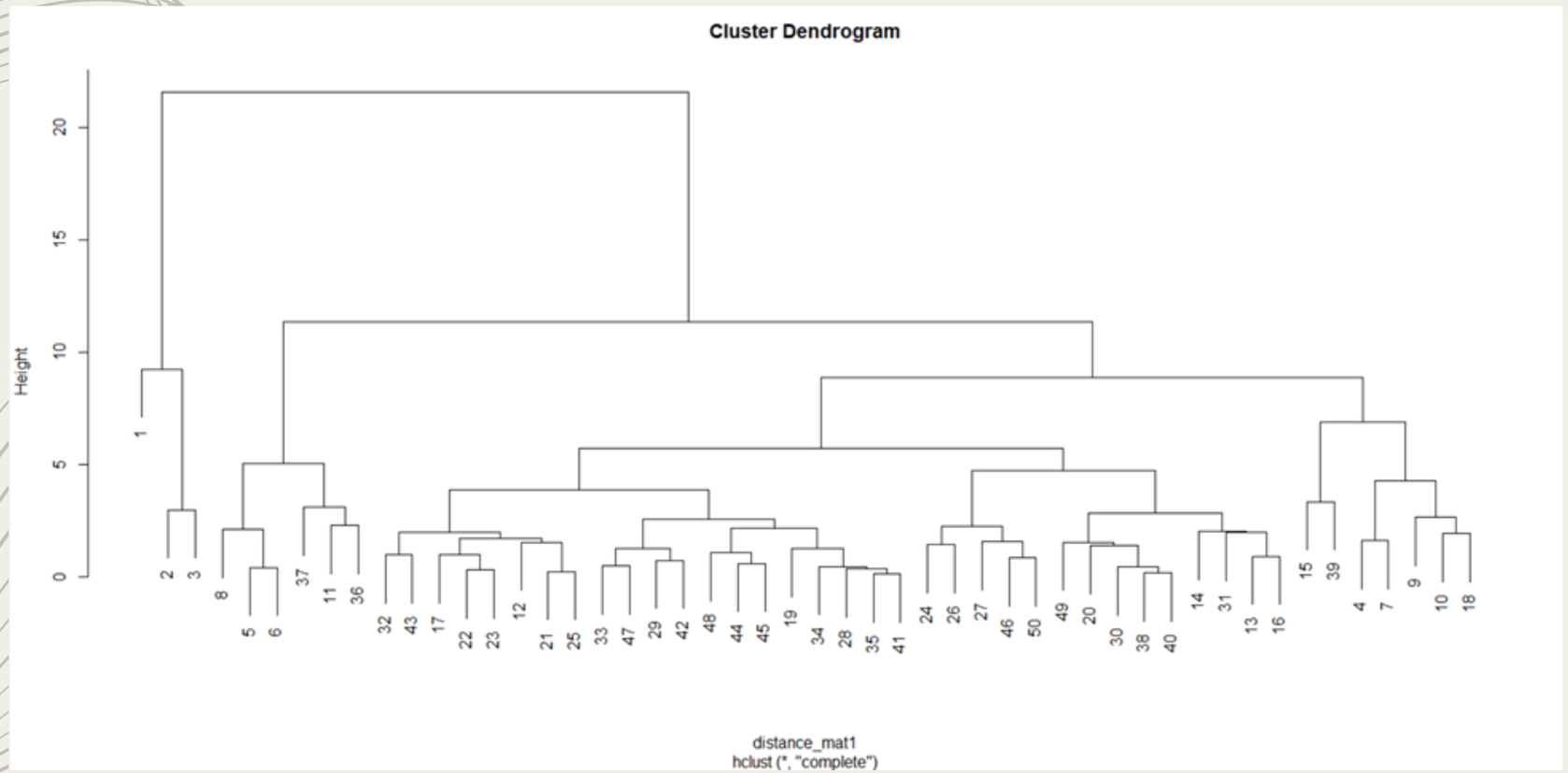
```
test2 = (standardized_data[1:50,1:4])
distance_mat1 <- dist(test2, method = 'manhattan')
distance_mat1
Hierar_cl1 <- hclust(distance_mat1)
Hierar_cl1
```

```
Call:
hclust(d = distance_mat1)

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

3

Clustering with Manhattan distance

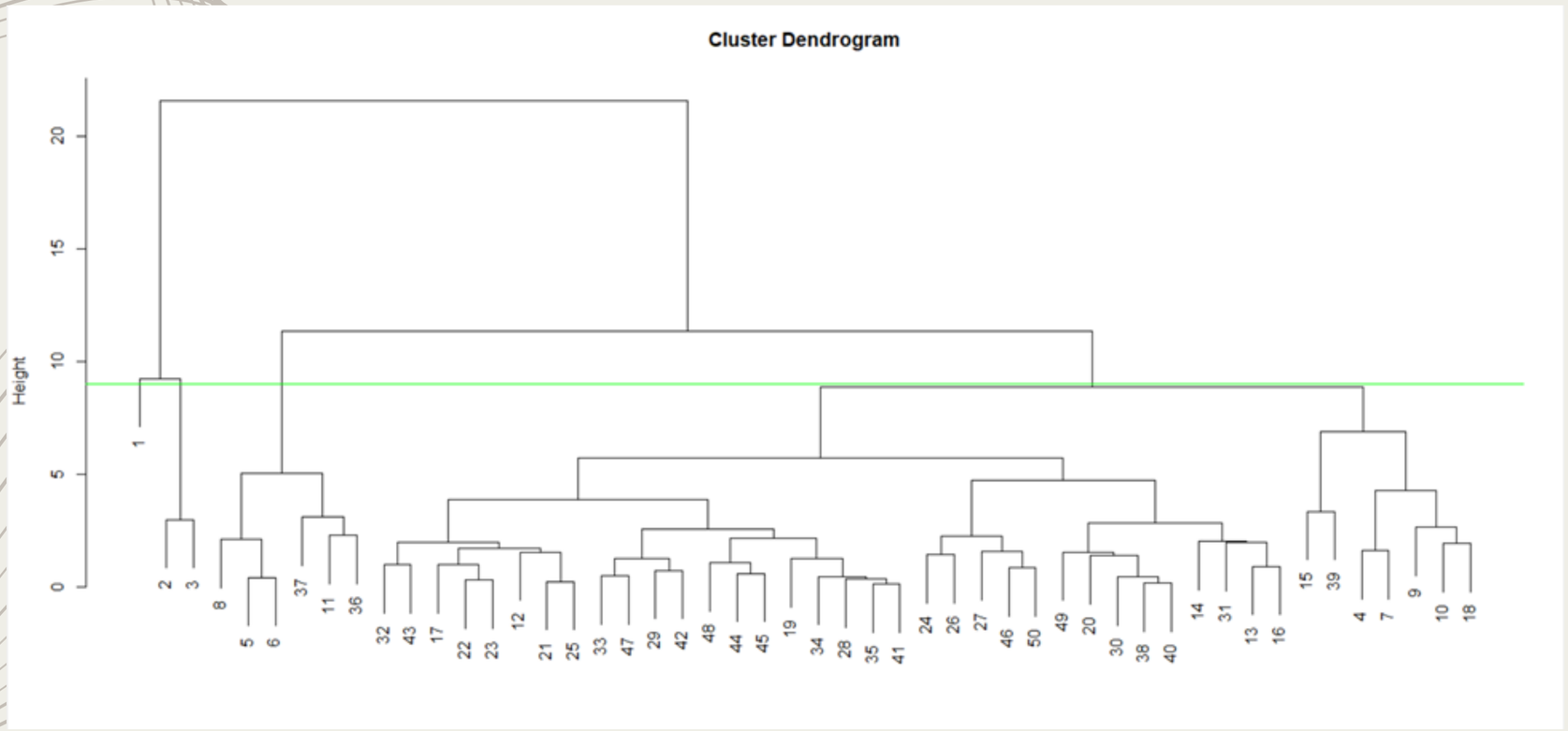
```
plot(Hierar_cl1)
```



3

Clustering with Manhattan distance

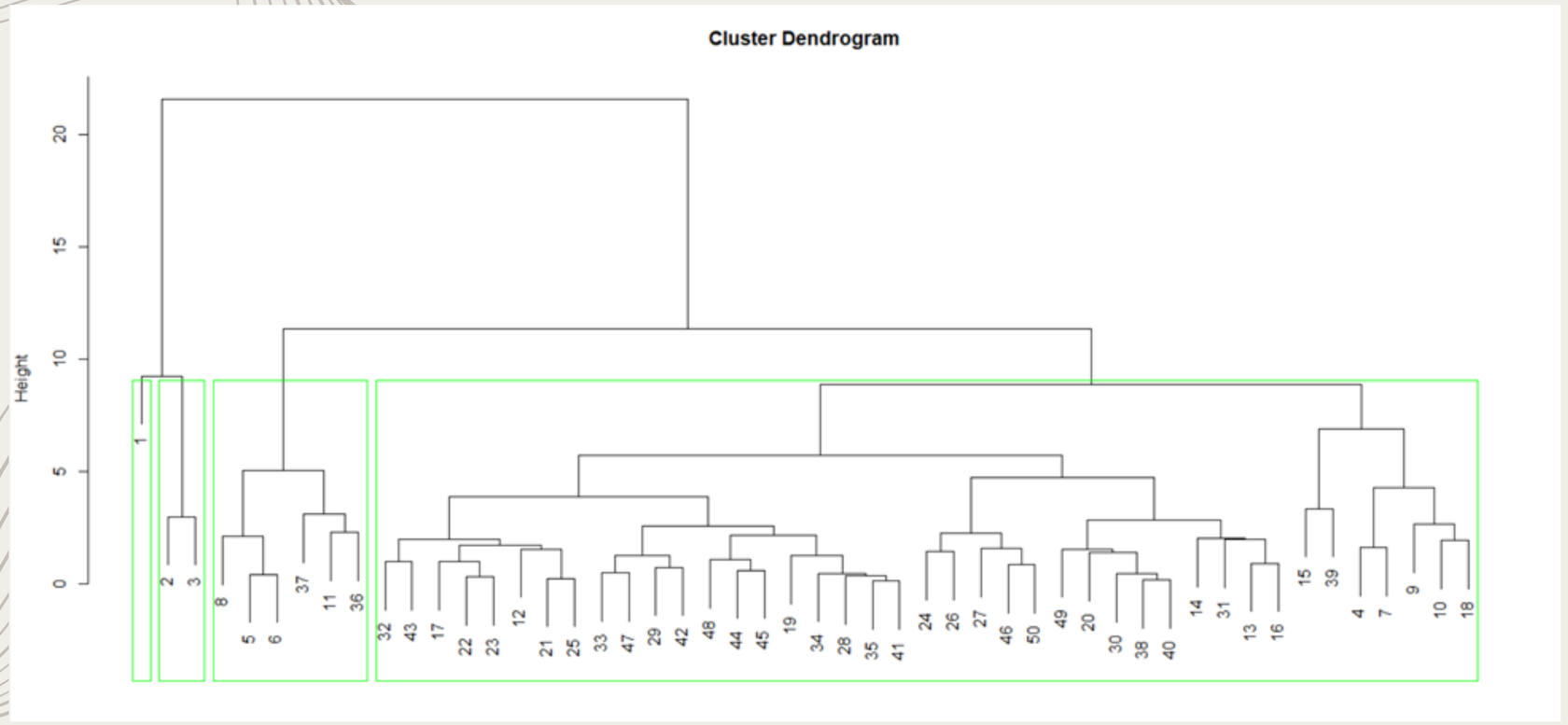
```
plot(Hierar_cl1)
abline(h = 9, col = "green")
```



3

Clustering with Manhattan distance

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



3

Clustering with Manhattan distance

DBSCAN (Manhattan distance)

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

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

```
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
```

3

Clustering with Manhattan distance

see all each cluster

```
Dbscan_cl1$cluster
```

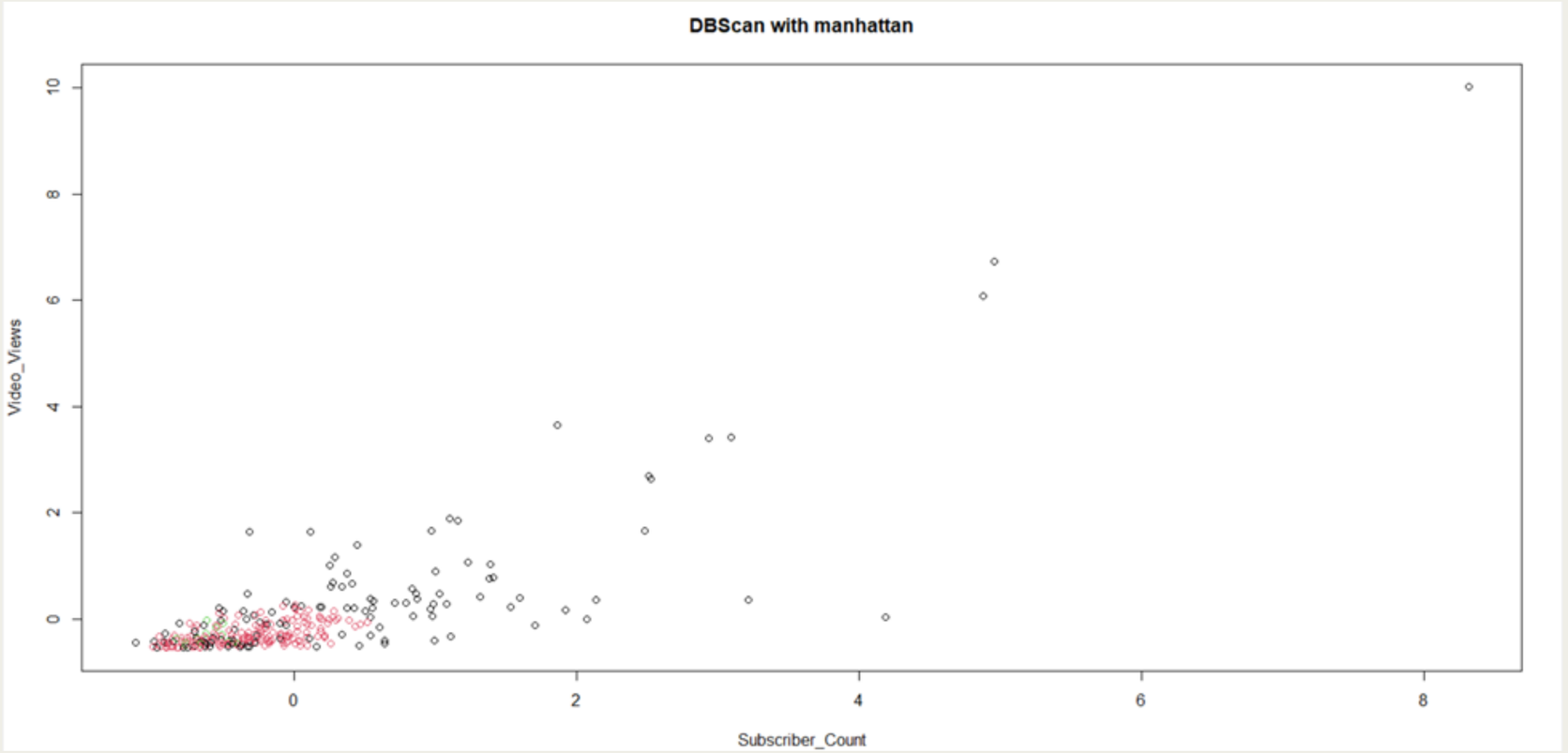
```
> Dbscan_cl1$cluster
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [44] 1 1 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 1 0 0 1 1 1 0 1 0 0 1 1 0 0 1 1 1 1 1 1 1 1 0 1
 [87] 0 1 1 1 0 1 2 1 1 1 0 1 1 0 0 1 2 1 1 1 1 1 1 0 0 1 0 1 1 0 1 1 1 1 0 1 2 1 1 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 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 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
```

3

Clustering with Manhattan distance

Subscriber_Count and Video_Views.

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

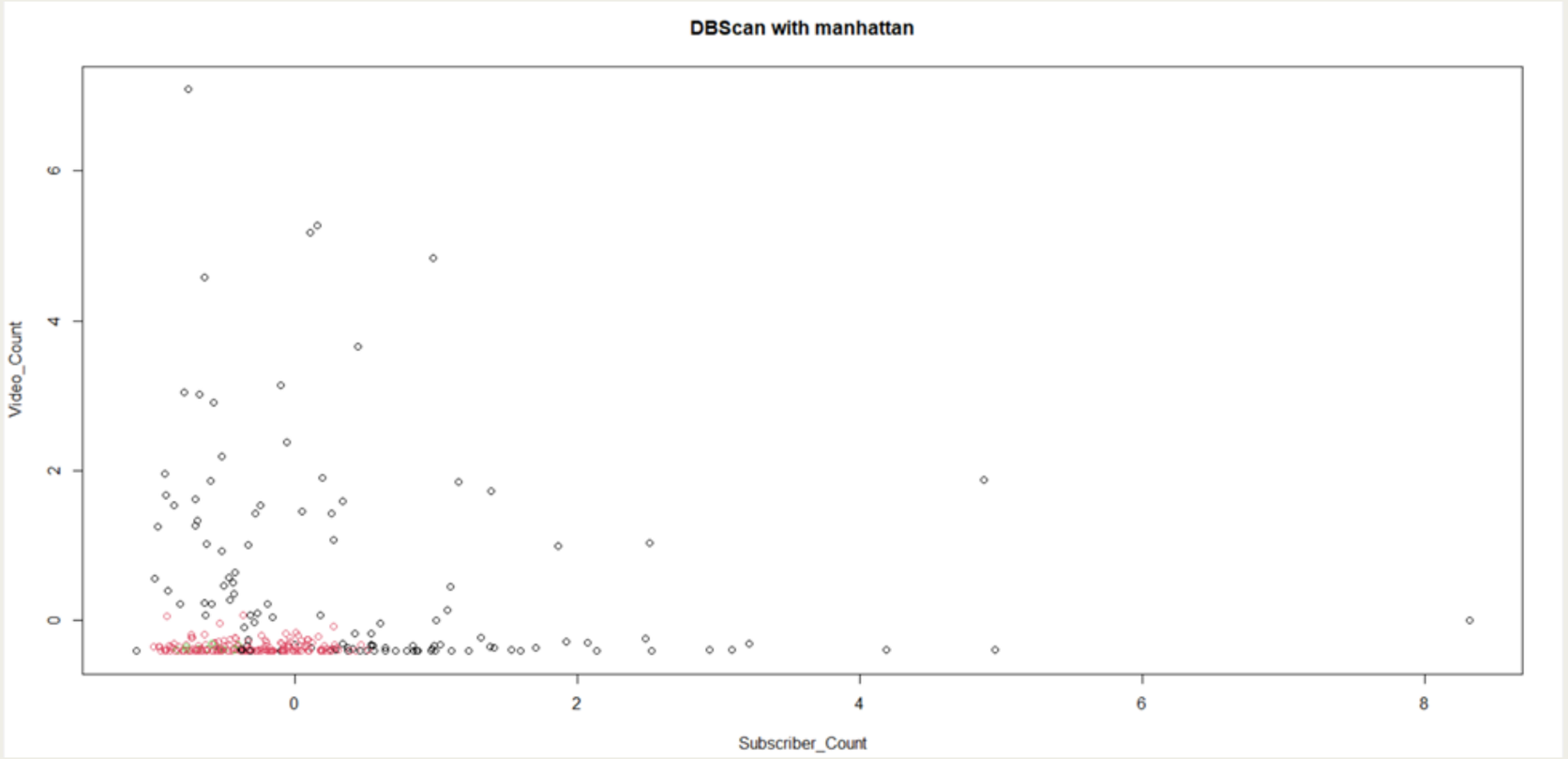


3

Clustering with Manhattan distance

Subscriber_Count and Video_Count.

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

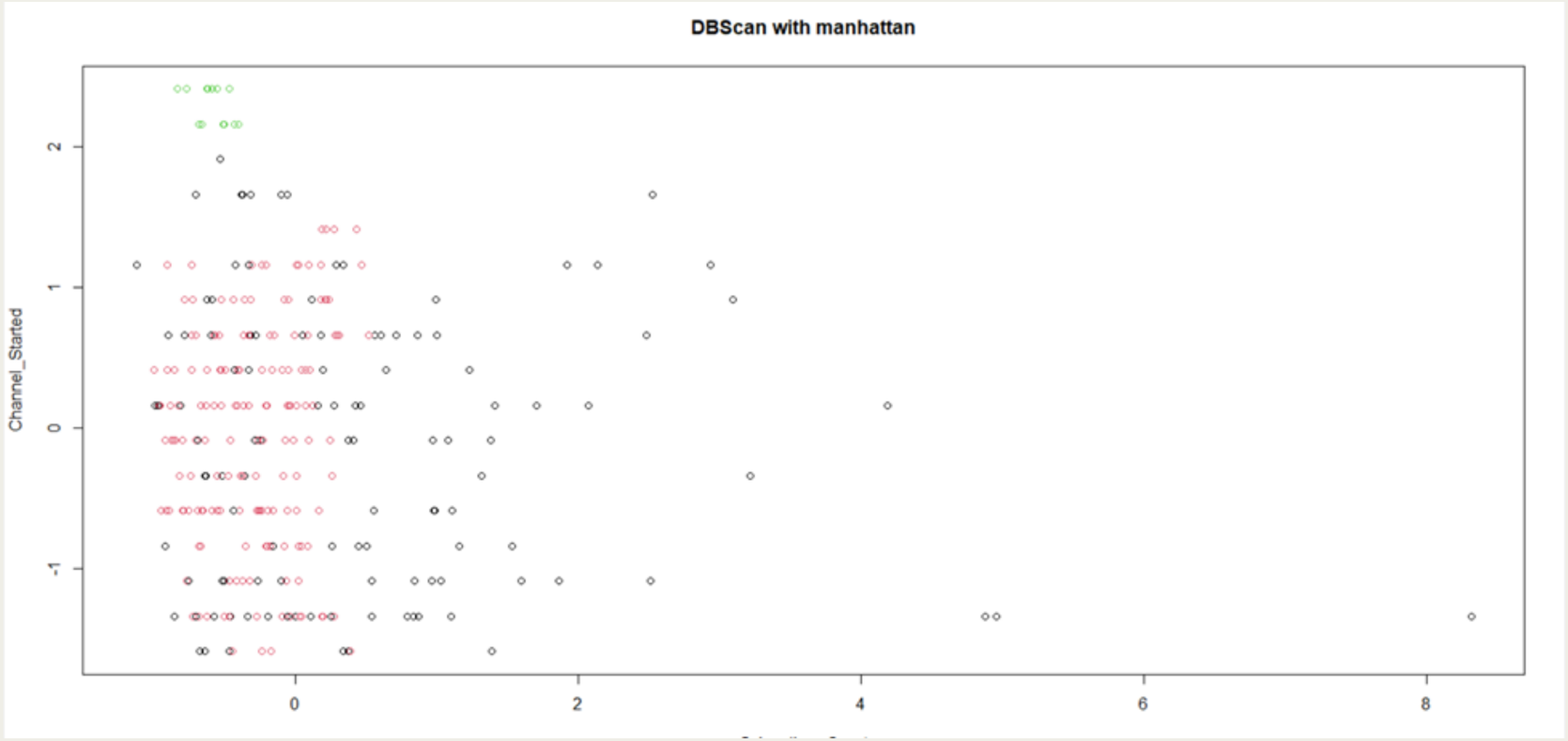


3

Clustering with Manhattan distance

Subscriber_Count and Channel_Started.

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

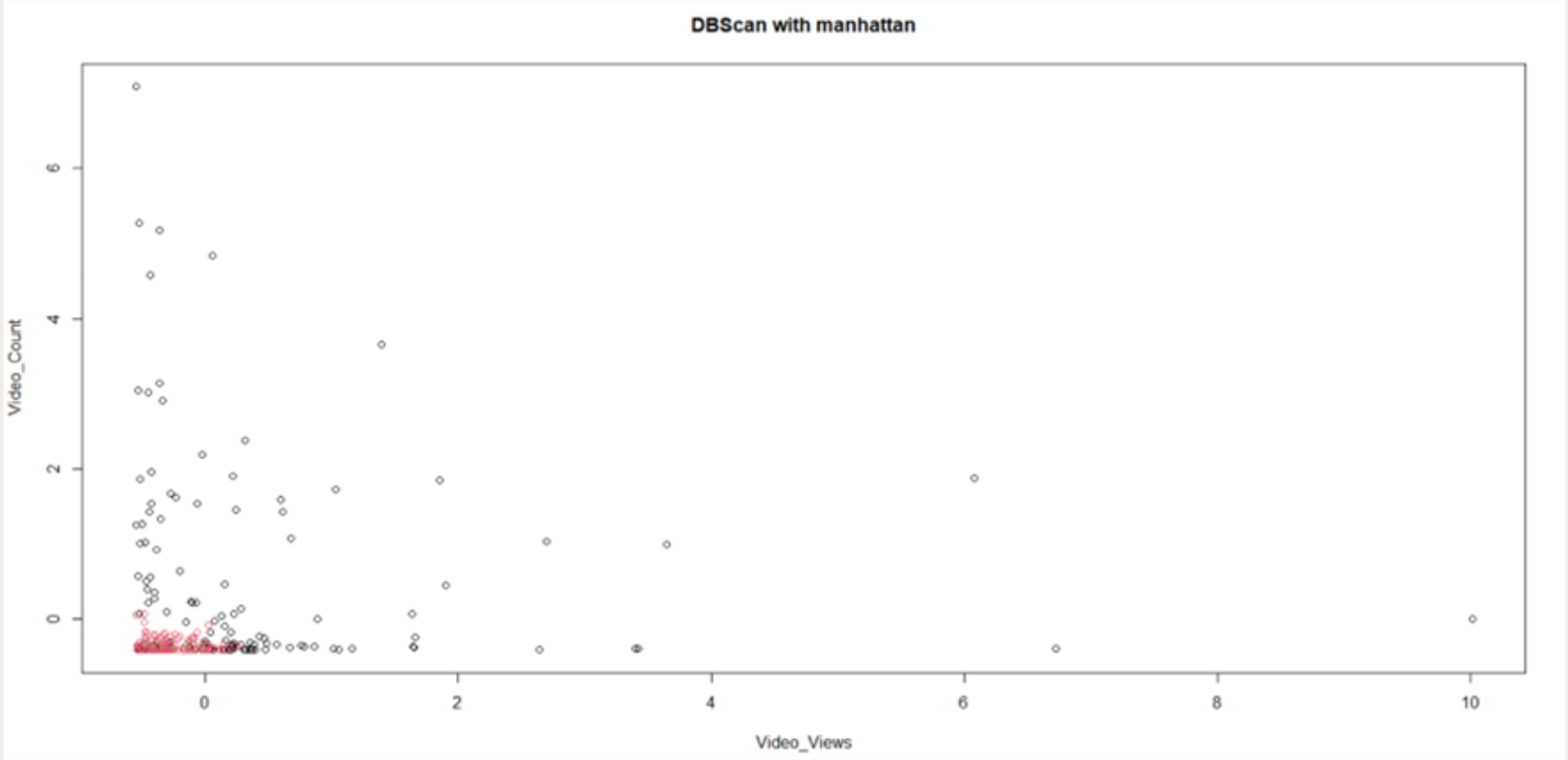


3

Clustering with Manhattan distance

Video_Views and Video_Count.

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

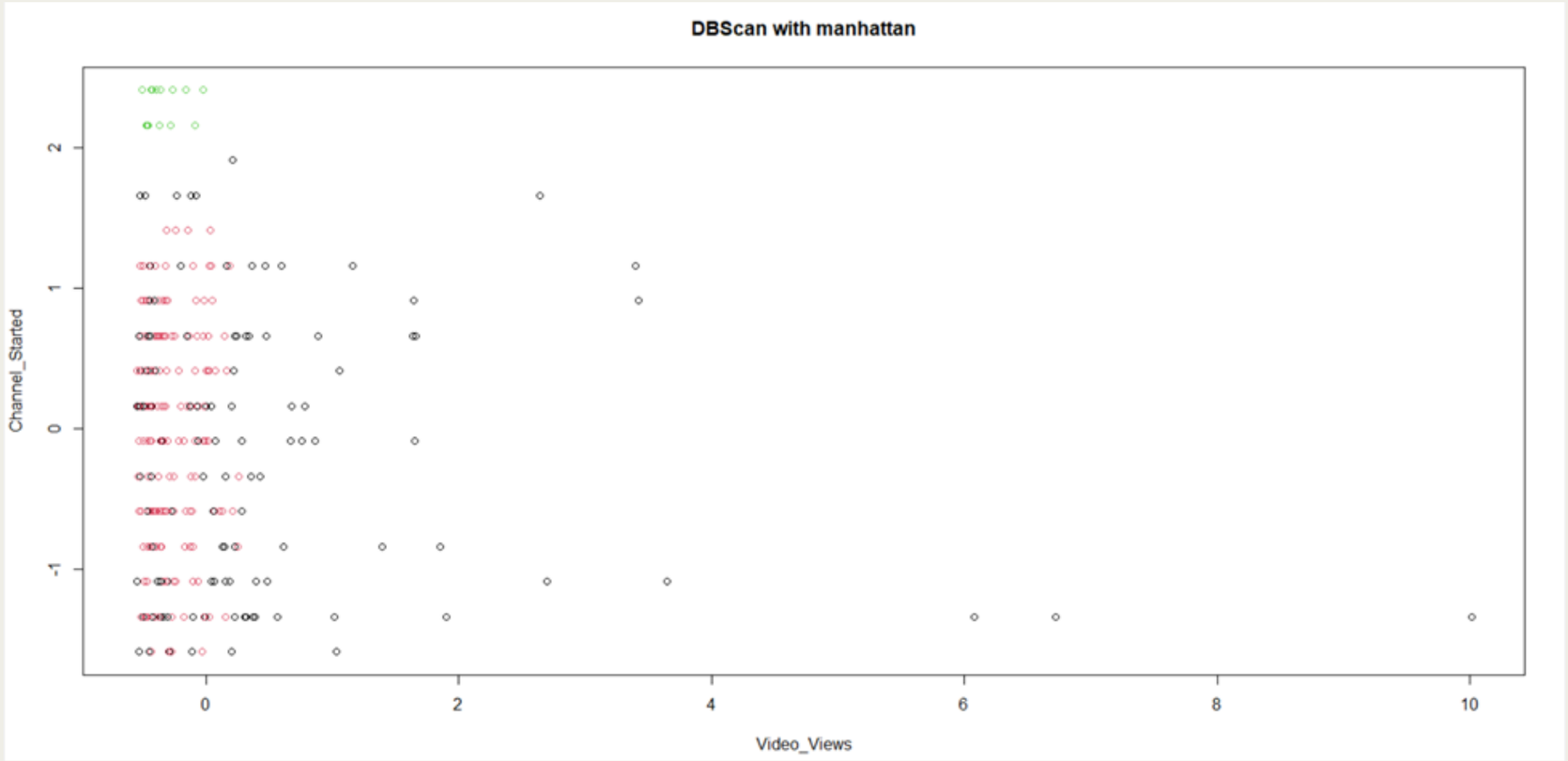


3

Clustering with Manhattan distance

Video_Views and Channel_Started.

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

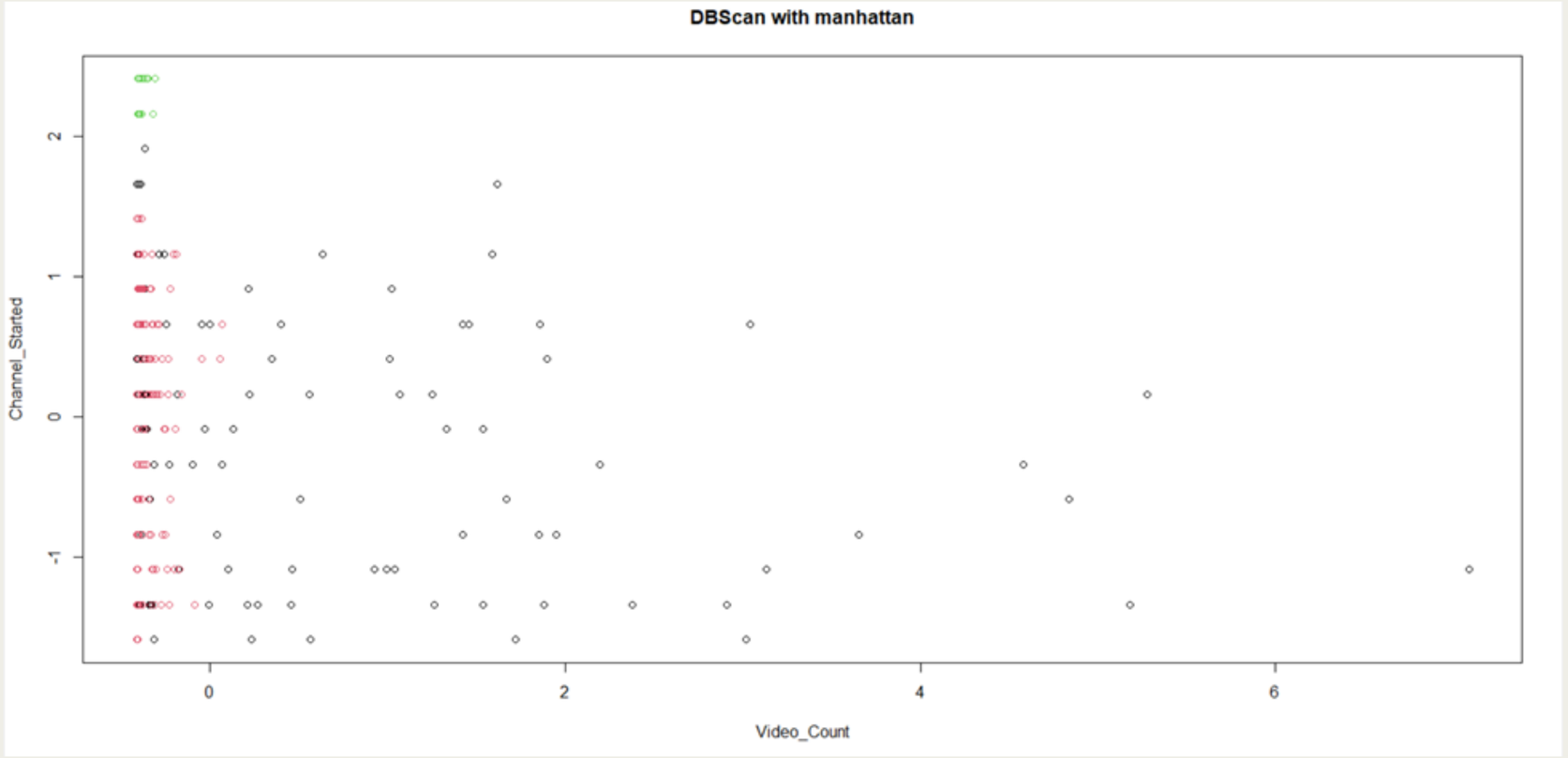


3

Clustering with Manhattan distance

Video_Count and Channel_Started.

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



Thank you