ML Portfolio 4: Searching for Similarity - Clustering

Bushra Rahman

Overview of Clustering

Clustering is a type of **unsupervised algorithm** that works on **unlabeled data**. This means that any designated target column either doesn't exist or is ignored; instead, all columns are considered for the goal of learning more about the data. In clustering, data points are grouped into clusters based on **distance**. Two types of clustering algorithms are **k-means clustering** and **hierarchical clustering**. **K-means clustering** is an iterative algorithm that identifies *k* centroids, assigns each observation to the nearest centroid, then recalculates the centroids. These steps are repeated until the model converges to an optimal solution. **Hierarchical clustering** (also called *hierarchical agglomerative clustering*) organizes and combines clusters into a hierarchy. Clusters that are nearest to each other are combined, until all clusters merge into one cluster at the top of the hierarchy.

Exploring the Data

The CSV file **"vgsales"** is a dataset of video games with sales greater than 100,000 copies, scraped from the VGChartz Network and uploaded to Kaggle (URL https://www.kaggle.com/datasets/gregorut/videogamesales (https://www.kaggle.com/datasets/gregorut/videogamesales)). This dataset contains 16,598 rows of video game titles and 11 columns of variables. Of those 11 variables, 5 are sales in the millions in different regions. Four of the remaining 6 variables are factors for Platform, Genre, Publisher, and Year. The remaining 2 variables are Name and Rank, which are unique for each observation.

```
#import and explore data
setwd("C:/Users/Bushra/CS4375/Clustering")
vgsales <- read.csv("vgsales.csv")

vgsales$Platform <- as.factor(vgsales$Platform)
vgsales$Genre <- as.factor(vgsales$Genre)
vgsales$Publisher <- as.factor(vgsales$Publisher)
vgsales$Year <- as.factor(vgsales$Year)

set.seed(1234)

names(vgsales)
```

```
##  [1] "Rank"         "Name"         "Platform"     "Year"         "Genre"
##  [6] "Publisher"    "NA_Sales"     "EU_Sales"     "JP_Sales"     "Other_Sales"
## [11] "Global_Sales"
```

```
head(vgsales)
```

	R…	Name	Platform	Y…	Genre	Publisher	NA_Sa…	EU_Sa…
	<int>	<chr>	<fct>	<fct>	<fct>	<fct>	<dbl>	<dbl>
1	1	Wii Sports	Wii	2006	Sports	Nintendo	41.49	29.02
2	2	Super Mario Bros.	NES	1985	Platform	Nintendo	29.08	3.58
3	3	Mario Kart Wii	Wii	2008	Racing	Nintendo	15.85	12.88
4	4	Wii Sports Resort	Wii	2009	Sports	Nintendo	15.75	11.01
5	5	Pokemon Red/Pokemon Blue	GB	1996	Role-Playing	Nintendo	11.27	8.89
6	6	Tetris	GB	1989	Puzzle	Nintendo	23.20	2.26

6 rows | 1-10 of 12 columns

Of all the columns, the 5 sales columns are all in the same units (millions of dollars). So, they are the most suitable columns to use for clustering. Because they are in the same units, there is no need to scale the data using the **scale()** function.

```
df <- vgsales[-c(1:6)]
df <- na.omit(df)
head(df)
```

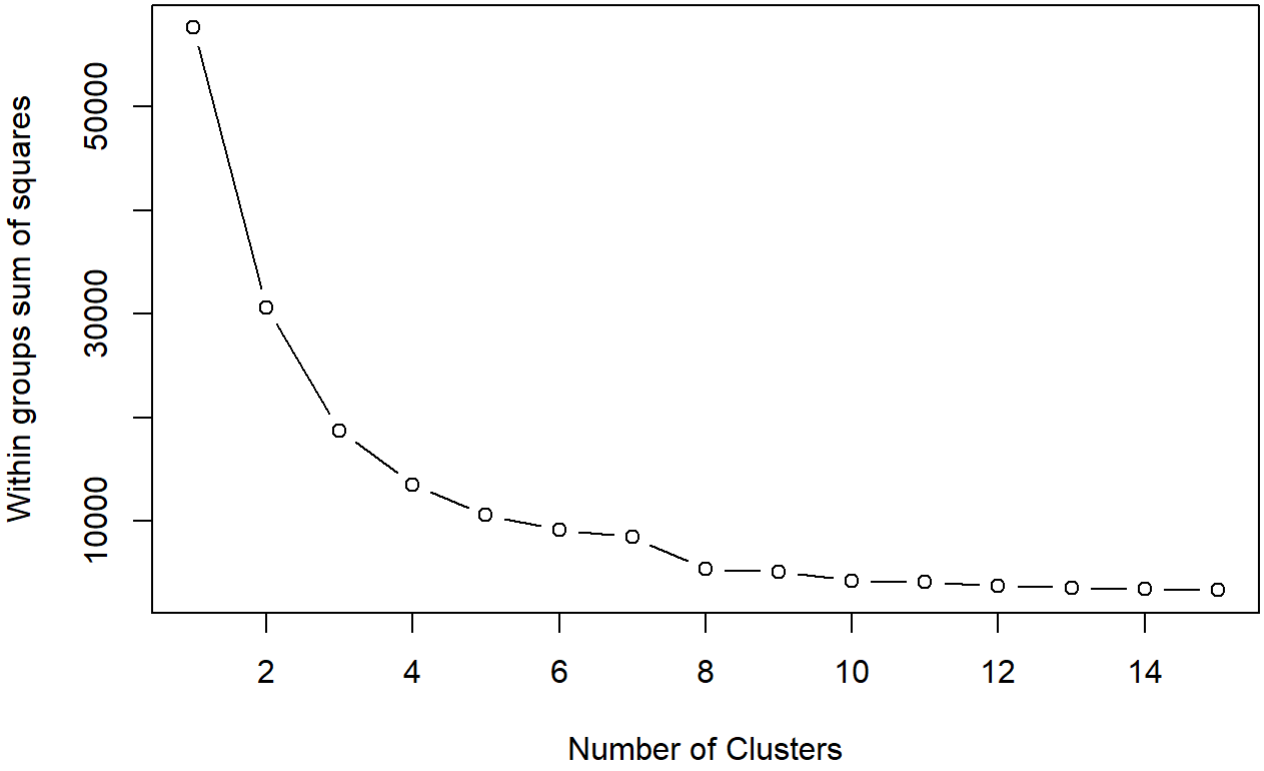
	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	41.49	29.02	3.77	8.46	82.74
2	29.08	3.58	6.81	0.77	40.24
3	15.85	12.88	3.79	3.31	35.82
4	15.75	11.01	3.28	2.96	33.00
5	11.27	8.89	10.22	1.00	31.37
6	23.20	2.26	4.22	0.58	30.26

6 rows

K-Means Clustering

The first step in k-means clustering is to determine the number of clusters. This can be achieved through the following function, which calculates ***wss*** (within groups sum of squares) per potential value of *k*. The *wss* value (also called *within-ss*) describes the compactness of the *k* clusters. The smaller the *wss* value, the more compact the clusters are, and the better the model is.

```
wsplot <- function(data, nc=15, seed=1234){
  wss <- (nrow(data)-1)*sum(apply(data,2,var))
  for (i in 2:nc){
    set.seed(seed)
    # set iter.max = 30 b/c kmeans() isn't converging at default=10 iterations
    wss[i] <- sum(kmeans(data,centers=i,iter.max=30)$withinss)
  }
  plot(1:nc, wss, type="b", xlab="Number of Clusters",
       ylab="Within groups sum of squares")
}
# call the function
wsplot(df)
```



The plot of *wss* against number of clusters shows that the *wss* value tapers off at about 3 clusters. Values greater than 3 do return lower *wss* values, but at the cost of increasing the cluster amount with only diminishing returns. A large number of clusters would obviously have a lower *wss* value, but it wouldn't really offer any valuable information on *how* the data is clustered. Therefore, **3** clusters is the most ideal value of *k* for this data.

```
set.seed(1234)
fit_km <- kmeans(df, 3, nstart=20)

fit_km$size
```

```
## [1]   577    25 15996
```

```
fit_km$centers
```

```
##     NA_Sales EU_Sales   JP_Sales Other_Sales Global_Sales
## 1  2.3012652 1.473449 0.57528596  0.45805893    4.8078510
## 2 13.7300000 7.368800 3.59200000  2.37080000   27.0624000
## 3  0.1701594 0.087505 0.05434359  0.02964366    0.3419449
```

```
fit_km$withinss
```

```
## [1] 6529.005 6887.063 5338.148
```

The k-means model for this data produced 3 clusters whose sizes are 577 observations, 25 observations, and 15996 observations. Clearly, the majority of the data is clustered together, with 2 other smaller clusters present as well. The centers of each cluster are given by the cluster mean, or the mean of points in the cluster. The compactness of each cluster is described by its *wss* value.

Cross-tabulating the clusters with the Genre column of vgsales shows how many observations of each genre were present in each of the 3 clusters. Predictably, most observations fall into the 3rd cluster, and few fall into the 2nd cluster. There appears to be little correlation between Genre and the clusters. A different possibility is that the clusters are more correlated with sales, such as Global Sales.

```
table_km <- table(vgsales$Genre, fit_km$cluster)
table_km
```

```
##
##                   1    2    3
##   Action        122    4 3190
##   Adventure      11    0 1275
##   Fighting       34    0  814
##   Misc           45    3 1691
##   Platform       59    6  821
##   Puzzle         13    1  568
##   Racing         40    2 1207
##   Role-Playing   59    3 1426
##   Shooter        91    1 1218
##   Simulation     21    1  845
##   Sports         73    4 2269
##   Strategy        9    0  672
```

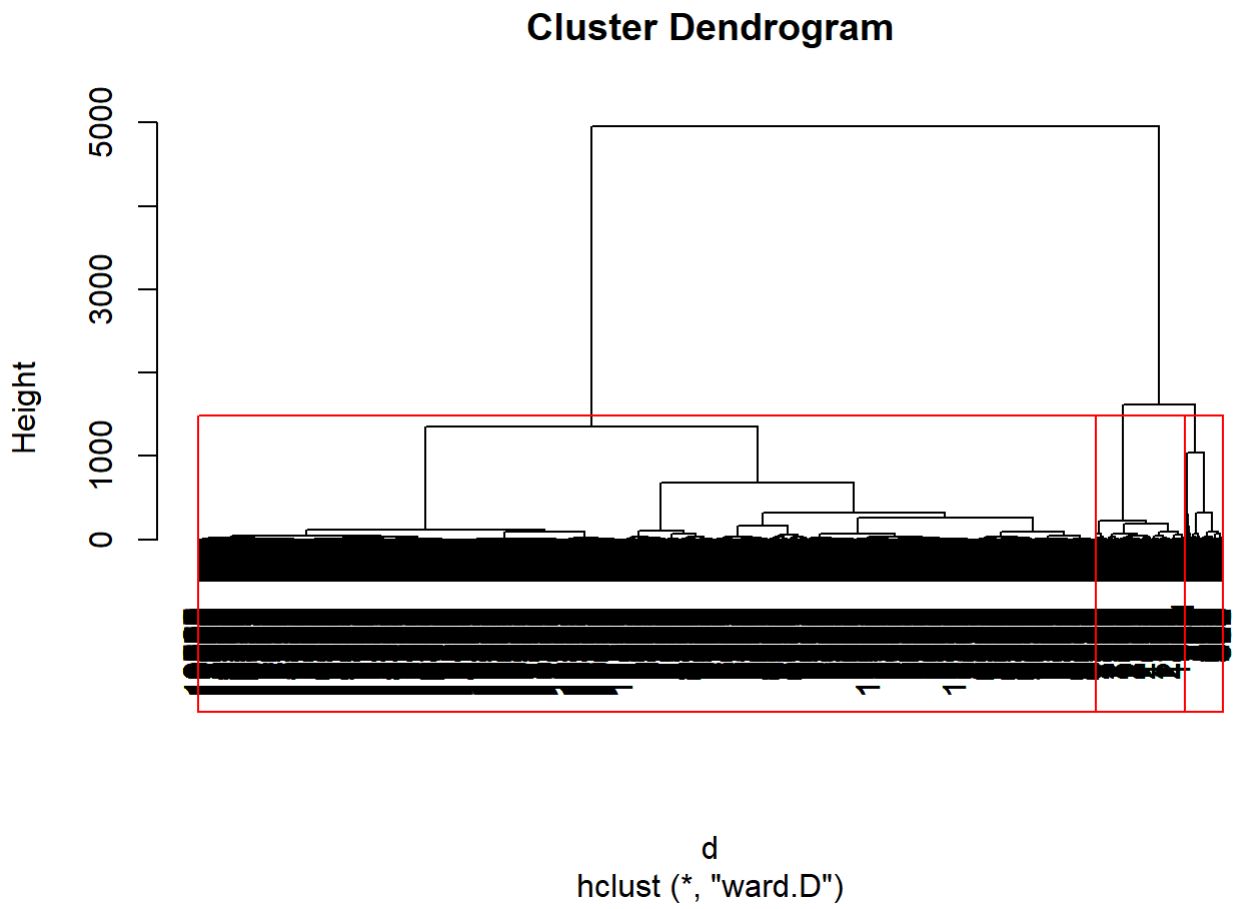
Hierarchical Clustering

Unlike k-means clustering, hierarchical clustering does not require specification of the number of clusters beforehand. In hierarchical clustering, observations are organized into a hierarchy that is displayed as a **dendrogram**. First, the hierarchy is modeled.

```
# Ward Hierarchical Clustering
d <- dist(df, method = "euclidean") # distance matrix
fit_h <- hclust(d, method="ward.D")
```

Now, the dendrogram of the model can be plotted. In hierarchical clustering, the number of clusters is visually determined from the dendrogram. Using the **cutree()** method, the dendrogram can be cut into different clusters. The following plot cuts the dendrogram into 3 clusters, since that was the ideal number of clusters for kmeans(). Note that hierarchical clustering tends not to work well with large amounts of data, which can clearly be seen with how crowded the dendrogram looks.

```
plot(fit_h) # display dendrogram
groups <- cutree(fit_h, k=3) # cut tree into 3 clusters
# draw dendrogram with red borders around the 3 clusters
rect.hclust(fit_h, k=3, border="red")
```



Again, cross-tabulating the clusters with the Genre column of vgsales shows how many observations of each genre were present in each cluster. At a glance, there does appear to be a lot of similarity between the k-means table and the hierarchical clustering table. Notably, both tables' **cluster 1** appears to be almost identical. **Cluster 3** is also roughly similar between both, but notably, **cluster 2** contains a far greater amount of observations in hierarchical clustering compared to k-means clustering.

```
table_cut <- table(vgsales$Genre, groups)
print(table_cut)
```

```
##                groups
##                    1    2    3
##   Action         125  298 2893
##   Adventure       12   31 1243
##   Fighting        34   85  729
##   Misc            54  117 1568
##   Platform        66  127  693
##   Puzzle          14   40  528
##   Racing          45  137 1067
##   Role-Playing    52  147 1289
##   Shooter         92  160 1058
##   Simulation      23   71  773
##   Sports          83  221 2042
##   Strategy         9   22  650
```

Model-Based Clustering

K-means and hierarchical clustering are *heuristic* algorithms, meaning they approximate clusters rather than using a formal model that can guarantee the best result. This is what **model-based clustering** is for: it is a formal approach to clustering that uses an expectation–maximization (**EM**) algorithm to iteratively find the most likely number of clusters [1]. In model-based clustering, it is assumed that the data follows an underlying probability distribution, and that distribution is calculated from the data [2]. Applying model-based clustering to our data produces a **Gaussian mixture model** of 3 clusters whose sizes are 6454, 8426, and 1718. This cluster distribution is much more even, since model-based clustering follows the normal Gaussian distribution.

```
# Model Based Clustering
library(mclust)
```

```
## Warning: package 'mclust' was built under R version 4.2.3
```

```
## Package 'mclust' version 6.0.0
## Type 'citation("mclust")' for citing this R package in publications.
```

```
fit_mb <- Mclust(df)
summary(fit_mb) # display the best model
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VVV (ellipsoidal, varying volume, shape, and orientation) model with 3
## components:
##
##  log-likelihood     n df      BIC      ICL
##        155059.6 16598 62 309516.8 308334.1
##
## Clustering table:
##    1    2    3
## 6454 8426 1718
```

```
# running multiple times in console gives diff clusters
```

Algorithm Comparison

Among the 3 algorithms, model-based clustering returns the most optimal result because it takes a formal approach. K-means clustering and hierarchical clustering returned fairly similar results, but k-means clustering consistently performs better on large amounts of data. K-means also has the advantage of trying several different random starting clustering assignments (the code above specified 20 starts), while hierarchical clustering is greedy and does not re-assign already-assigned clusters. However, hierarchical clustering has an advantage in not requiring us to know the ideal number of clusters beforehand. By producing a dendrogram, hierarchical clustering can help to visually validate the clustering result of k-means clustering.

Sources

[1] https://www.statmethods.net/advstats/cluster.html (https://www.statmethods.net/advstats/cluster.html)

[2] https://en.proft.me/2017/02/1/model-based-clustering-r/ (https://en.proft.me/2017/02/1/model-based-clustering-r/)