Class 8: Customer Segmentation Using Unsupervised Learning for M&S

Dr. Wei Miao

UCL School of Management

October 21, 2026

Section 1

Customer Segmentation

Customer Segmentation

Segmentation is a key step in the marketing strategy (STP) process, where customers are divided into meaningful groups based on characteristics relevant to designing and executing your marketing strategy.

It assumes that different customer groups provide varying levels of value to the company and/or require distinct marketing programmes to succeed (e.g., based on differing goals and needs).

Conventional Segmentation

- **Customer value segmentation** is for targeting decisions based on customers' potential long-term financial and strategic value to your company.
- **Demographic segmentation** uses variables such as age, gender, income, family life cycle, educational qualification, socio-economic status, religion, company size and income, etc. These serve as proxies for goals, preferences or psychographics, as well as to characterize segments for marketing mix decisions.
- **Psychographic segmentation** is for positioning and marketing mix design based on the psychology of the customer and consumer, including attitudes, identity, lifestyle, personality, etc.

Conventional segmentation methods often require **subjective** judgements. A more objective approach is to 'let the data speak' by utilising data analytics tools.

Section 2

K-Means in R

Syntax of `kmeans()`

```
kmeans(x, centers, iter.max = 10, nstart = 1,
       algorithm = c("Hartigan-Wong", "Lloyd", "Forgy",
                     "MacQueen"), trace=FALSE)
```

- `x`: data with selected variables to apply K-means
- `centers`: an integer `k` = number of clusters
- `iter.max` (integer, default = 10): maximum number of iterations for a single run. Increase if you see non-convergence or very slow improvement.
- `nstart` (integer, default = 1): number of random initializations when `centers` is an integer. The best solution (smallest total within-cluster sum of squares) is returned. Use 10–50+ for stability in practice.
- `algorithm` (character): one of "Hartigan-Wong" [default], "Lloyd", "Forgy", or "MacQueen". Hartigan-Wong is typically fast/accurate; Lloyd/MacQueen can be preferable if you encounter empty clusters.
- `trace` (logical or integer, default = FALSE): prints progress of the algorithm if `TRUE` or a positive integer, which can help debugging but produces verbose output.

Section 3

Data Cleaning

Data Loading

- Let's first try customer segmentation based on *total spending* and *Income*.
- **Exercise**: load data_full, create `total_spending`, and select `total_spending` and `Income` as the clustering variables into a new data frame `data_kmeans`.

Data Pre-processing

- To perform a cluster analysis in R, generally, the data should be prepared as follows:
  - Rows are observations (individuals) and columns are variables of interest for clustering.
  - Any missing value in the data must be removed or imputed.
  - The data must be standardised (i.e., scaled) to make variables comparable. Standardisation consists of transforming the variables such that they have mean zero and standard deviation one.[1]

[1] Another common method is to normalise the data, which consists of transforming the variables such that they have a minimum of zero and a maximum of one.

Data Pre-processing: Missing Values

- Check whether there are any missing values in the data.
- Use mean imputation to fill in missing values.

Data Pre-processing: Standardisation

- We need to re-scale the clustering variables using `scale()`, because the variables can be on very different scales.
  - **Exercise**: Scale the variables and create a new data frame `data_kmeans_scaled`.
  - **This is extremely important!**

```
data_kmeans_scaled <- data_kmeans %>%
    select(total_spending, Income) %>%
    mutate(
        total_spending = scale(total_spending),
        Income = scale(Income)
    )
```

Visualisation of the Data

- Let's visualise the data to see whether there are any natural clusters.
- **Exercise**: Create a scatter plot of `total_spending` and `Income` using `ggplot2`.
- Refer to the ggplot2 cheat sheet for more information on data visualisation in R.

ggplot2

ggplot2 is a system for declaratively creating graphics, based on The Grammar of Graphics. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details. Go to docs...

Section 4

Apply K-Means to M&S Case Study

Apply K-Means Clustering with 2 Clusters

- `set.seed()` is to allow replication of results.
- `kmeans()` is the function to perform K-means clustering.
- `centers` is the number of clusters to form.
- `nstart` is the number of sets to be chosen.

```
set.seed(888)
result_kmeans <- kmeans(data_kmeans_scaled,
    centers = 2,
    nstart = 10
)
```

More About Seed and Random Number in R

- In R, random number generation is controlled via a "seed". The random numbers generated are not truly random but **pseudo-random**, meaning they are generated by a deterministic algorithm that produces a sequence of numbers that appear random. Setting the seed ensures that you get the same sequence of pseudo-random numbers each time you run your code, making your results reproducible.
- Use `set.seed()` function to set the seed before generating random numbers. The argument to `set.seed()` is an integer value that initializes the random number generator.
  - For example, `set.seed(888)` sets the seed to 888. You can choose any integer value as the seed.
- For any models that involve random processes (e.g., K-means clustering; random forest), setting the seed is important for reproducibility, especially when your analysis involves random sampling or random processes.

Examine the returned object, `result_kmeans`

```
tidy(result_kmeans)
```

```
# A tibble: 2 x 5
  total_spending Income  size withinss cluster
           <dbl>  <dbl> <int>    <dbl> <fct>
1          1.02   0.950   822     726. 1
2         -0.713 -0.663  1178     553. 2
```

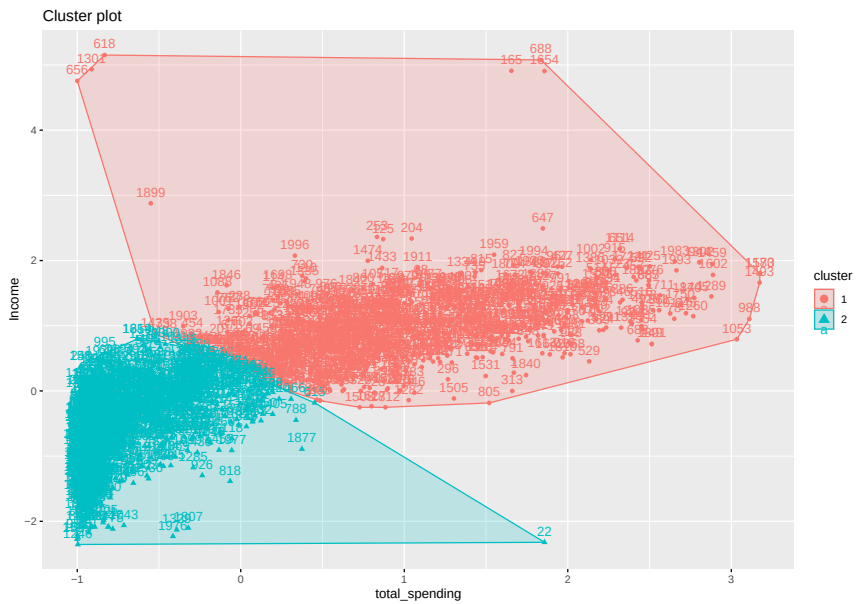
- `size`: The number of points in each cluster.
- `cluster`: **A vector of integers (from 1:k) indicating the cluster to which each point is allocated.**
- `withinss`: Vector of within-cluster sum of squares, one component per cluster.

Visualise the clusters

- We can definitely use `ggplot2` for visualisation, but `cluster` and `factoextra` already have built-in functions for visualising clusters.
- Use the function `fviz_cluster()` to generate visualisations

```
pacman::p_load(cluster, factoextra)

fviz_cluster(result_kmeans,
     data = data_kmeans_scaled
)
```



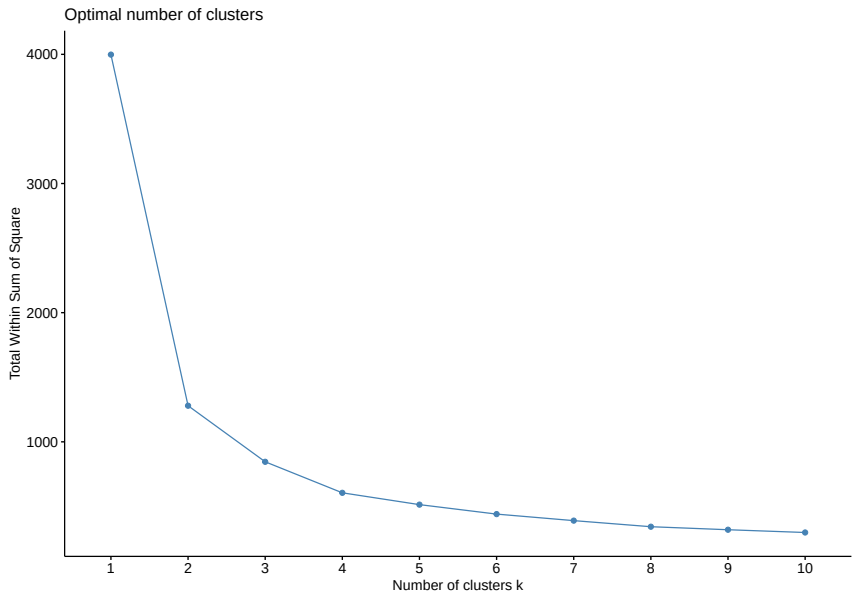
Section 5

Determine Optimal K

Determine the optimal number of clusters: Elbow Method

- The elbow method consists of plotting the explained variation as a function of the number of clusters, and picking the elbow of the curve as the number of clusters to use.

```
set.seed(888)
pacman::p_load(factoextra)
data_kmeans_scaled %>%
    fviz_nbclust(kmeans, method = "wss")
```



- There are alternative methods such as the silhouette method and the gap statistic method, but the elbow method is the most commonly used one.

Next Steps After Segmentation

- Compare the CLV in different segments, and decide which segments to serve.
- Develop marketing strategies for each segment. For example, for the high-value segment, you may want to increase the frequency of purchase by offering discounts or promotions.
- Develop a customer journey map for each segment.

After-Class Readings

- Useful source: K-means Cluster Analysis
- K-means is the most commonly used clustering algorithm, but there are many other clustering algorithms available, such as hierarchical clustering, DBSCAN, Gaussian mixture models, etc. You can refer to this link to explore these algorithms for more advanced clustering tasks.