# Multidimensional Scaling

*Joseph Nathan Cohen*

*Fall 2019*

## Explanation

***Multidimensional scaling*** (MDS) is an exploratory technique that creates graphical depictions of the overall (dis)similarities among some set of subjects. The method is a useful way to visually represent multidimensional differences within small sets. In this tutorial, we will focus on basic, "classical" scaling. Know that more sophisticated and flexible techniques are available in R.

### Illustrative Example

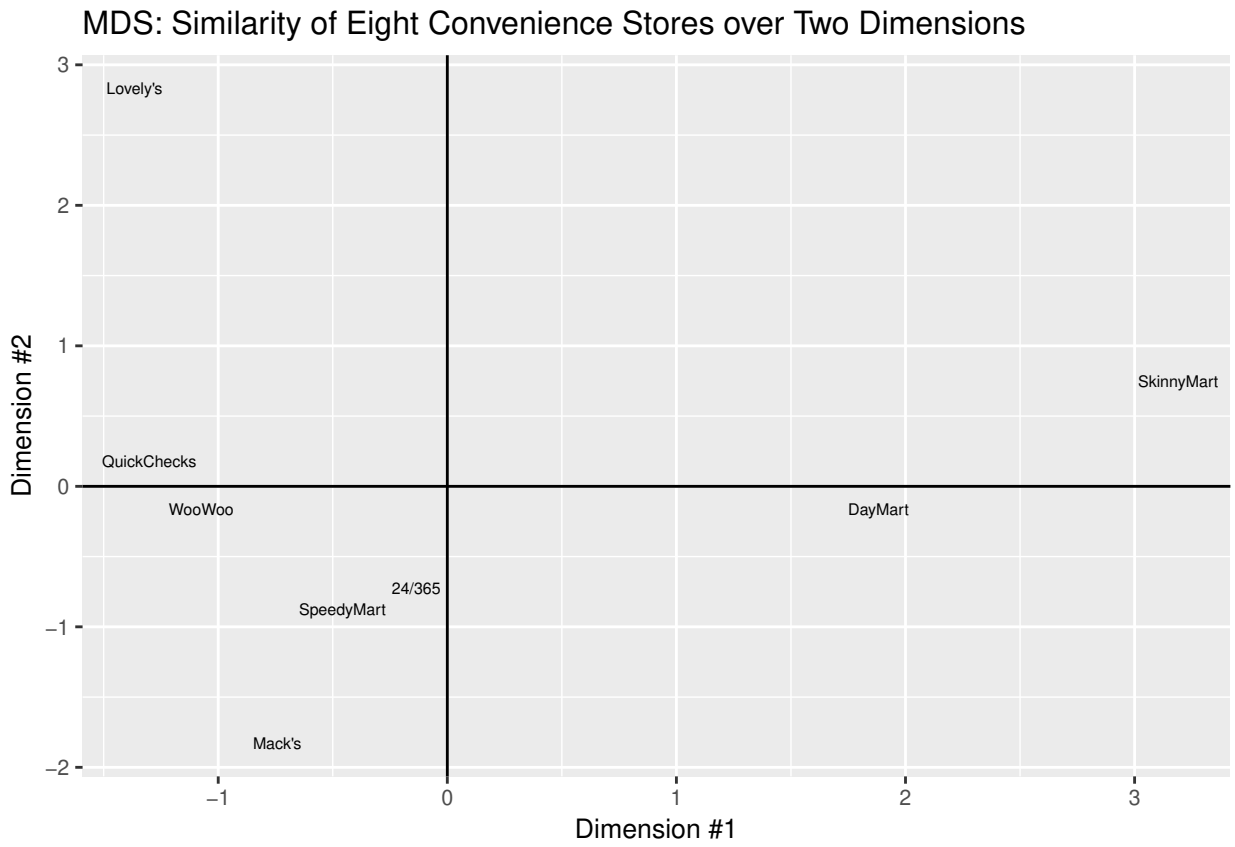
SpeedyMart is convenience store chain. The company is developing its marketing strategy. Part of this process involves identifying their competition. Companies are taken to be more direct competitors if they are more similar SpeedyGas.

The team assembles data on SpeedyMart and 7 other convenience store chains in Fake Town, USA. This data includes information on prices (average mark up), overall selection (number of products sold), hours opened per week, hot/prepared food selection (number of products available), average age of the internal decor (in months), and average walking distance from people's front doors to the closest chain location (in minutes).

The (fictitious) data are given below. Based on these data, which companies most directly compete with SpeedyMart? And how easy is it to tell?

| store | prices | selection | open | hot | decor | walk |
|---|---|---|---|---|---|---|
| SpeedyMart | 0.18 | 825 | 168 | 16 | 48 | 27 |
| QuickChecks | 0.12 | 900 | 168 | 60 | 35 | 48 |
| Mack's | 0.17 | 750 | 126 | 30 | 125 | 19 |
| 24/365 | 0.32 | 700 | 168 | 45 | 50 | 33 |
| Lovely's | 0.12 | 2200 | 126 | 80 | 28 | 120 |
| WooWoo | 0.18 | 500 | 168 | 75 | 45 | 50 |
| DayMart | 0.35 | 650 | 84 | 12 | 30 | 50 |
| SkinnyMart | 0.41 | 240 | 84 | 0 | 14 | 130 |

With MDS, you can generate a visual depiction of these units' overall similarity. Such an operation might render something like this:




We have reduced the differences among five variables to a graphically-depicted two-dimensional space. The method doesn't tell us *what* these dimensions mean. Such a determination involves interpretive rather than calculative work. However, it does tell us that the 24/365 store is most similar to SpeedyMart, and thus the store's most direct competitor. QuickChecks and WooWoo (who are themselves strong competitors) and Mack's appear to also operate closer to SpeedyMart's competitive space. The operation suggests that Lovely's and SkinnyMart are very different stores from SpeedyMart, and thus less of a competitive concern.

## The Model

Where as many of the analytical methods that we studied in this course are based on correlations and covariances, MDS uses *distance metrics*, which measure the magnitudes of differences between units of analysis. Perhaps the best-known distance measure is the *Euclidian distance* metric, which is measured as:

$$D_i = \sqrt{\sum_{k=1}^{q}(x_{ik} - x_{jk})^2}$$

Where:

- $x_{ij}$ is the value of variable $k$ on unit $i$
- $x_{jk}$ is the value of variable $k$ for unit $j$
- $q$ is the total number of variables being used to calculate the distances.

So the distance between units $i$ and $j$ are measured through the differences between them on $q$ metrics.

MDS takes these distance metrics and maps them onto lower-dimensional spaces. See Everitt and Hothorn (2011, pp. 106 - 110) for a fuller exposition of how these coordinates are calculated.

# Implementation

We implement the operation over the following steps:

1. Standardize the variables
2. Calculate a distance matrix
3. Create an MDS object
4. Diagnose the MDS's fit
5. Visualize the MDS

## Data

In this exercise, we will look at data comparing living standards in the United States' ten largest metro areas. Our variables include:

- Cost of living index (variable: *costliving*). Index by Expatisan
- Murder rate per 100,000 people (*murder*)
- per capita GDP in metro area (*gdp*)
- Transit ratings (*transit*). Index by AllTransit
- Metro area unemployment rate (*unemployment*)
- Poverty rate (*poverty*)

Here's the data:

```
##             city costliving murder   gdp transit unemployment poverty
## 1       New York        246   3.39 71084     9.6          3.3    17.5
## 2    Los Angeles        196   7.01 67763     7.7          3.7    20.0
## 3        Chicago        184  24.13 61170     9.1          3.8    16.4
## 4         Dallas        161  12.48 64824     6.9          2.8    17.3
## 5     Washington        205  16.72 74000     9.3          2.9    11.0
## 6        Houston        148  11.50 63311     5.9          3.2    19.6
## 7  San Francisco        238   6.35 89978     9.6          2.4    12.0
## 8   Philadelphia        172  20.06 63519     9.0          3.1    16.3
## 9         Boston        200   8.35 78463     9.4          2.4    12.3
## 10       Atlanta        169  16.41 56840     7.9          3.0    17.3
```

Our task to to identify whic movies have similar combinations of budgets, receipts, critic ratings, and popular ratings.

## Step 1: Standardize Variables

Our first step is to standardize variables, such that variables denominated in larger units do not exert undue influence over our distance estimates. we will use the **standardize()** command in the ***psycho*** package:

```
library(psycho)
STD.DATA <- standardize(DATA)

#A look at the top lines.
head(STD.DATA)[1:5]
```

```
##          city costliving      murder        gdp    transit
## 1    New York  1.6947769 -1.39715876  0.2047148  0.9056851
## 2 Los Angeles  0.1284397 -0.85037879 -0.1371285 -0.5777646
## 3     Chicago -0.2474813  1.73549775 -0.8157714  0.5153036
## 4      Dallas -0.9679964 -0.02416707 -0.4396511 -1.2023751
## 5  Washington  0.4103804  0.61626030  0.5048700  0.6714562
## 6     Houston -1.3752441 -0.17219038 -0.5953900 -1.9831381
```

### Step 2: Calculate Distance Matrix

Next, we calculate the distance matrix using the **dist()** command in the base package. Here, we use the Euclidean distance.

```
#Calculate Distance Matrix
DIST <- dist(STD.DATA[-1], method = "euclidean")
```

### Step 3: Create MDS Object

We use this distance matrix as input for an MDS object using the **cmdscale()** in the base package. Here is an example of an MDS operation that tries to reduce the differences in these six variables to three dimensions:

```
#Create Classical Scaling Matrix
#Try reduction to four dimensions
MDS <- cmdscale(DIST, k = 3 , eig = TRUE)
```

### Step 4: Diagnose Fit

As is always the case in data reduction, information is loss in the process of simplifying these relationships. The number of large eignvalues gives you a guide to how many dimensions you can use without too much information loss.

```
MDS$eig
```

```
##  [1]  3.108821e+01  1.123322e+01  9.502425e+00  1.525951e+00  4.302774e-01
##  [6]  2.199240e-01  1.538565e-15  1.499720e-15  1.177870e-15 -1.614922e-15
```

Eignvalues get small around the third value, suggesting that the information gains after using a third dimension get smaller. It suggests that a two dimensional MDS will encapsulate much of the difference between these units, but that there will still be some information loss.

Does the MDS recover the original distances fully?

```
max(abs(dist(STD.DATA[-1]) - dist(cmdscale(DIST, k = 3))))
```

```
## [1] 0.2908375
```

Were this zero, we could confidently say that there is no information loss. However, there will be some information loss here. Still three dimensions cover a lot of ground. Below, we find that three dimensions cover 96% of summed eignvalues.

```
cumsum(abs(MDS$eig))/sum(abs(MDS$eig))
```

```
##  [1] 0.5757075 0.7837300 0.9597009 0.9879592 0.9959273 1.0000000 1.0000000
##  [8] 1.0000000 1.0000000 1.0000000
```

By the third dimension, 96% of the summed eignvalues have been captured. A three dimensional solution should encapsulate these differences well. ‘

### Step 5: Visualize

Finally, we plot the visualizations. In this example, we will use a three-dimensional figure (so that you have an example in your notes).

Here, we are going to reduce to three dimensions. As shown above, it captures a lot of these differences. Plus, I want to be sure to include a model to visualize a three dimension solution as well as the two dimensional example above. For 3D plots, we use the **text3D()** from the package ***plot3D***.

```r
#Data table for plotting:
RESULTS <- cbind(STD.DATA[1], MDS$points)
names(RESULTS) <- paste(c("city", "X1", "X2", "X3"))

#Note: Creating a list of shortened city names to
#make figure more legible
RESULTS$city
```

```
##  [1] New York      Los Angeles   Chicago       Dallas        Washington
##  [6] Houston       San Francisco Philadelphia  Boston        Atlanta
## 10 Levels: Atlanta Boston Chicago Dallas Houston Los Angeles ... Washington
```

```r
citynames <- c("NY", "LA", "CHI", "DAL", "DC", "HOU", "SF", "PHI", "BOS", "ATL")

#Plotting 3D
#Note that color depict Z-axis
library(plot3D)
text3D(x = RESULTS$X1, y = RESULTS$X2, z = RESULTS$X3,
       colvar = RESULTS$X3, col = gg.col(100),
       labels = citynames, bty="b2", theta = 60, phi = 20)
```