Learning Multidimensional Scaling (MDS) with R: A Step-by-Step Guide

Authored by

Mohammed looti

October 27, 2025

Introduction to Multidimensional Scaling (MDS)

In the expansive realm of multivariate statistics, **Multidimensional Scaling** (MDS) serves as an essential technique for visualizing complex similarity or dissimilarity structures within a dataset. Its fundamental purpose is to take high-dimensional data--where the relationships between observations are difficult to grasp--and project them into a lower-dimensional space, typically a two-dimensional (2-D) Cartesian space. This geometric transformation allows researchers and analysts to gain immediate, intuitive insights into the underlying structure, facilitating the identification of clusters, outliers, or gradual transitions that remain obscured in the original format.

The core methodology of MDS revolves around preserving relationships. It requires an input matrix that quantifies the distances or dissimilarities between every pair of items. The algorithm then attempts to assign spatial coordinates to each item in the reduced space such that the Euclidean distances between these new points closely reflect the original input dissimilarities. Essentially, if two items were very similar according to the initial metrics, they must appear close together in the resulting scatter plot. Conversely, items that were highly dissimilar are mapped far apart. This fidelity to the original distance structure is what makes MDS a powerful tool for visual data exploration.

The utility of MDS extends across diverse disciplines, including psychology (mapping perceptions), marketing (analyzing brand preferences), biology (comparing genetic similarity), and ecology (understanding species distribution). By simplifying complex inter-item relationships into an easily digestible visual format, MDS provides a crucial bridge between raw data complexity and human pattern recognition, serving as a vital step in hypothesis generation and explanatory data analysis. MDS is often the first step when the goal is to visualize proximity data without losing the fidelity of the original metric relationships.

Classical MDS in R: The `cmdscale()` Function

For statistical practitioners utilizing the **R statistical environment**, executing classical Multidimensional Scaling (also known as Principal Coordinates Analysis, or PCoA) is streamlined by the base function, **`cmdscale()`**. This powerful, built-in tool is specifically designed to handle situations where a direct distance matrix is available or can be calculated from the raw data. The goal of `cmdscale()` is mathematical: to find a configuration of points whose Euclidean distances in the chosen dimensional space best reproduce the input dissimilarities, minimizing the distortion introduced by the dimension reduction.

Understanding the necessary inputs for `cmdscale()` is paramount to successful implementation. The function offers flexibility through several key arguments, allowing the user to control the output dimensions and retrieve diagnostic metrics. The basic structure of the function call is concise:

cmdscale(d, eig = FALSE, k = 2, ...)

The following parameters are essential for governing the behavior and output of the `cmdscale()` routine:

d: This mandatory argument must be a **distance matrix** object, quantifying the dissimilarities between all pairs of observations. In R, this is most commonly generated using the **`dist()`** function, which supports various metrics (such as Euclidean, Manhattan, or maximum distance) applied to the rows of a numeric matrix or data frame. The chosen metric fundamentally determines the quality and meaningfulness of the subsequent MDS results.
eig: This logical parameter controls whether the function should return the **eigenvalues** of the scaled matrix. When set to `TRUE`, the eigenvalues provide a measure of the variance explained by each dimension. Larger positive eigenvalues indicate dimensions that account for more variance in the original dissimilarities, allowing users to assess the goodness-of-fit of the solution. The default setting is `FALSE`.
k: An integer that specifies the desired number of dimensions for the output space. While a value of **2** is most common for visualization purposes, yielding a 2-D scatter plot, higher values can be selected for analyzing the data structure in three or more dimensions, although interpretation becomes increasingly difficult beyond three. The default value is **2**.

Mastering these parameters is key to leveraging `cmdscale()` efficiently, ensuring that the generated low-dimensional representation accurately reflects the underlying structure of the data. The subsequent sections illustrate this implementation with a practical example.

Setting Up the Data: A Basketball Player Example

To provide a tangible illustration of Multidimensional Scaling in R, we will utilize a simulated dataset containing performance statistics for a group of basketball players. Our goal is to analyze the statistical profiles of these players--labeled A through K--and visualize how similar or dissimilar they are based on their metrics. This type of analysis is invaluable for coaches or analysts seeking to quickly identify player archetypes or potential skill redundancies within a team.

We begin by constructing a data frame in R. This structure captures four key performance indicators (KPIs) for each player: points scored, assists, blocks, and rebounds. These four variables define the initial high-dimensional space in which our players reside. The following R code snippet demonstrates the creation and display of this dataset, which must be inspected prior to initiating any statistical transformation.

#create data frame
df <- data.frame(points=c(4, 4, 6, 7, 8, 14, 16, 19, 25, 25, 28),

assists=c(3, 2, 2, 5, 4, 8, 7, 6, 8, 10, 11),
blocks=c(7, 3, 6, 7, 5, 8, 8, 4, 2, 2, 1),
rebounds=c(4, 5, 5, 6, 5, 8, 10, 4, 3, 2, 2))

#add row names
row.names(df) <- LETTERS

#view data frame
df

points assists blocks rebounds
A 4 3 7 4
B 4 2 3 5
C 6 2 6 5
D 7 5 7 6
E 8 4 5 5
F 14 8 8 8
G 16 7 8 10
H 19 6 4 4
I 25 8 2 3
J 25 10 2 2
K 28 11 1 2

The resulting data frame clearly maps the statistical profiles, with each row representing a distinct observation (player) and the columns detailing their performance attributes. This organized structure provides the necessary input for calculating the dissimilarities that will drive the subsequent Multidimensional Scaling analysis, projecting these multi-faceted profiles into an easily navigable visual space.

Calculating Distances and Running the MDS Model

Once the basketball player data is structured, the transformation process begins. The first critical step is the calculation of the **distance matrix**, which quantifies the numerical separation between every pair of players based on their four statistical metrics. We employ the **`dist()`** function for this purpose, utilizing the default Euclidean distance--a standard measure that treats each statistical difference equally. The output of this function, stored in the variable `d`, serves as the primary input for the MDS procedure.

With the distances established, we execute the `cmdscale()` function. In this specific application, we set the parameters `eig = TRUE` to obtain the eigenvalues (useful for assessing the variance

captured by the solution) and `k = 2`, ensuring the output is a two-dimensional configuration suitable for plotting. The results are captured in the `fit` object, which contains the new coordinates for each player in the reduced space.

The final stage is the visualization. Using standard plotting commands, we generate a scatter plot. The `plot()` function initiates the graph, defining "Coordinate 1" and "Coordinate 2" as the axes derived from the MDS output. Crucially, the `text()` function overlays the names of the players (A through K) onto their corresponding points. This label assignment transforms the abstract coordinates into an immediately interpretable map, allowing us to visually assess player groupings based on their statistical similarity.

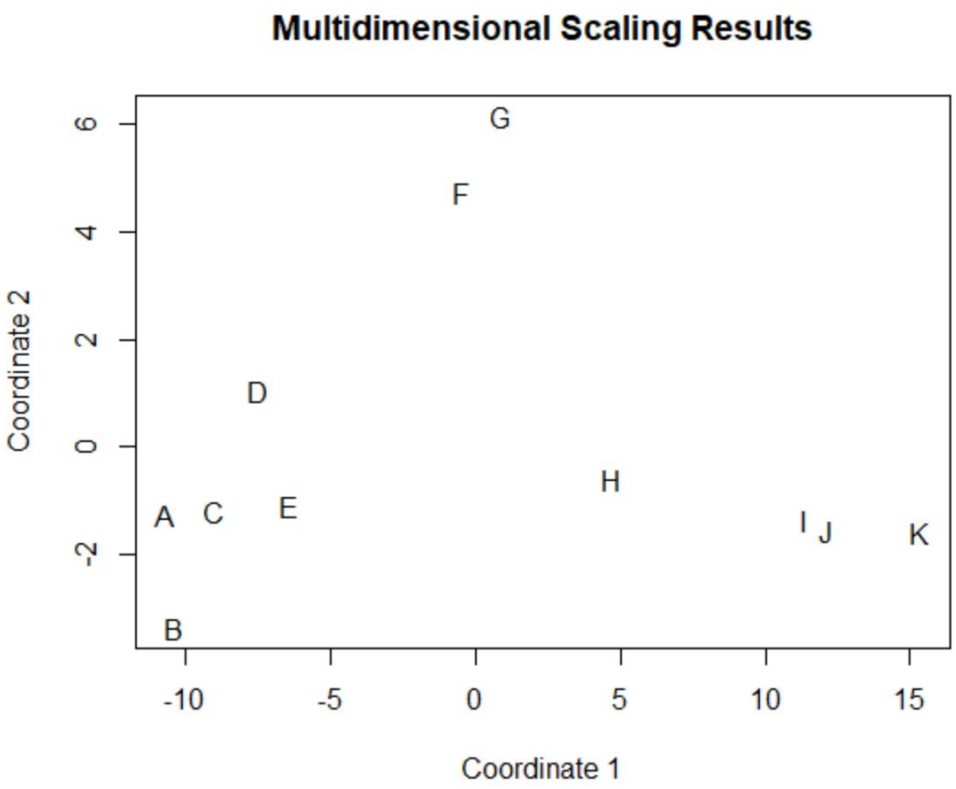
```
#calculate distance matrix
d <- dist(df)

#perform multidimensional scaling
fit <- cmdscale(d, eig=TRUE, k=2)

#extract (x, y) coordinates of multidimensional scaleing
x <- fit$points
y <- fit$points

#create scatter plot
plot(x, y, xlab="Coordinate 1", ylab="Coordinate 2",
main="Multidimensional Scaling Results", type="n")

#add row names of data frame as labels
text(x, y, labels=row.names(df))
```



The resulting image is the definitive output of the MDS analysis. It spatially arranges the basketball players, translating their complex four-dimensional statistical profiles into a simple 2-D map where proximity equals similarity. This visual arrangement allows for immediate pattern recognition that would be impossible to achieve simply by inspecting the raw table of numbers.

Visualizing and Interpreting the MDS Plot

The true analytic power of Multidimensional Scaling is realized through the careful interpretation of the generated scatter plot. The fundamental rule for interpreting an MDS plot is straightforward: the spatial distance between any two points is a proxy for the statistical dissimilarity between the observations they represent. Therefore, players who share similar performance metrics across points, assists, blocks, and rebounds will naturally gravitate toward each other in the 2-D space, forming observable clusters.

To illustrate this principle, let us analyze the cluster formed by players **A** and **C**, who are positioned closely together on the plot. This spatial proximity immediately suggests that their overall statistical contributions are highly comparable. We can verify this claim by recalling their raw statistics from the original data frame: Player A recorded (4 points, 3 assists, 7 blocks, 4 rebounds), while Player C recorded (6 points, 2 assists, 6 blocks, 5 rebounds). While they are not numerically identical, the values are close across all metrics, justifying their tight grouping in the reduced dimensional space.

#view data frame values for players A and C
df

points assists blocks rebounds
A 4 3 7 4
C 6 2 6 5

In contrast, examining the relationship between players **B** and **K** reveals the opposite effect. They are placed at the far extremes of the MDS plot, a significant spatial separation that signals profound differences in their underlying statistical profiles. Player B is located toward the bottom left, while Player K is positioned on the far right. A closer look at their raw data confirms this stark contrast in their performance attributes, highlighting how effectively the MDS procedure magnifies statistical dissimilarities into visible spatial gaps, which is precisely the distance we calculated in the distance matrix.

#view data frame values for players B and K
df

points assists blocks rebounds
B 4 2 3 5
K 28 11 1 2

Player K is clearly an offensive powerhouse (high points/assists) with low defensive stats (blocks/rebounds), whereas Player B is a low-scorer with a more balanced profile. This example demonstrates that the MDS plot is not merely a picture but a quantitative map: players with similar roles form distinct groupings, while unique or specialized players (outliers) are spatially isolated, offering critical insights for tactical analysis.

Decoding the Numerical Output: Principal Coordinates

While visual inspection of the scatter plot is highly informative, quantitative analysis often requires direct access to the numerical results. The precise coordinates that determine each player's position in the reduced space are known as the principal coordinates, and they represent the core output of the `cmdscale()` function. These coordinates serve as the new, concise "scores" for each observation along the derived dimensions.

The results of our MDS analysis were stored in the object named `fit`. By simply executing this variable name in the R console, we can retrieve the exact (x, y) coordinates used for plotting. This numerical matrix provides the quantitative foundation for the visual map, detailing exactly where each player sits along the newly created dimensions (Coordinate 1 and Coordinate 2). These

principal coordinates are crucial if the analyst wishes to incorporate the reduced-dimensional scores into further statistical modeling or clustering techniques, especially when interpreting the contribution of positive and negative eigenvalues.

#view (x, y) coordinates of points in the plot
fit

A -10.6617577 -1.2511291
B -10.3858237 -3.3450473
C -9.0330408 -1.1968116
D -7.4905743 1.0578445
E -6.4021114 -1.0743669
F -0.4618426 4.7392534
G 0.8850934 6.1460850
H 4.7352436 -0.6004609
I 11.3793381 -1.3563398
J 12.0844168 -1.5494108
K 15.3510585 -1.5696166

In this output, the columns and define the position of players A through K in the two derived MDS dimensions. These numerical values confirm the visual patterns observed in the plot; for instance, players I, J, and K have high positive values on Coordinate 1, indicating they share a common trait (likely high offensive output), while players A, B, and C have high negative values on Coordinate 1, indicating a different profile. By examining these principal coordinates, we gain a deeper, more quantitative insight into the structure revealed by the Multidimensional Scaling process.

Additional Resources for R Analysis

This tutorial successfully navigated the process of performing Multidimensional Scaling in R, covering the conceptual framework, practical implementation using the `dist()` function, and the subsequent visualization and interpretation of the results. MDS is just one component of a vast array of powerful multivariate techniques available for data exploration in the R environment. For instance, techniques like Principal Component Analysis (PCA) offer alternative methods for dimensionality reduction, focusing on variance maximization rather than distance preservation.

To further enhance your proficiency in data analysis and statistical modeling within R, we recommend exploring tutorials on related dimensionality reduction and clustering methods. These techniques often complement MDS, providing alternative perspectives on complex datasets:

How to Perform Principal Component Analysis in R

A Guide to Cluster Analysis in R
Visualizing Data with ggplot2 in R

Continuing to explore these resources will undoubtedly enhance your proficiency in leveraging R for various statistical and data science applications.