

How can multidimensional scaling be performed in R, and could you provide an example?

Authored by
stats writer

June 26, 2024

RECOMMENDED CITATION

stats writer (2024). *How can multidimensional scaling be performed in R, and could you provide an example?*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=154456>

Multidimensional scaling is a statistical method used to visualize and analyze the similarity or dissimilarity between a set of objects based on their characteristics or attributes. This technique is commonly used in fields such as psychology, marketing, and ecology.

In R, multidimensional scaling can be performed using the "cmdscale" function from the "stats" package. This function takes in a matrix or data frame containing the distance or dissimilarity measures between the objects and outputs a set of coordinates representing their positions in a lower dimensional space.

For example, let's say we have a data frame containing the distances between 6 different cities based on their population, area, and average temperature. We can use the "cmdscale" function to obtain a 2-dimensional representation of these cities and plot them on a graph, allowing us to visually compare their similarities and differences.

In summary, multidimensional scaling in R is a useful tool for analyzing and visualizing the relationships between objects based on their characteristics, and the "cmdscale" function provides a simple and efficient way to perform this technique.

Perform Multidimensional Scaling in R (With Example)

In statistics, multidimensional scaling is a way to visualize the similarity of observations in a dataset in an abstract cartesian space (usually a 2-D space).

The easiest way to perform multidimensional scaling in R is by using the built-in cmdscale() function, which uses the following basic syntax:

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

where:

d: A distance matrix usually calculated by the dist()

function.eig: Whether or not to return eigenvalues.
k: The number of dimensions to visualize the data in.
Default is 2.

The following example shows how to use this function in practice.

Example: Multidimensional Scaling in R

Suppose we have the following data frame in R that contains information about various basketball players:

```
#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

We can use the following code to perform multidimensional scaling with the `cmdscale()` function and visualize the results in a 2-D space:

```
#calculate distance matrix
```

```
d <- dist(df)
```

```
#perform multidimensional scaling
```

```
fit <- cmdscale(d, eig=TRUE, k=2)
```

```
#extract (x, y) coordinates of multidimensional scaling
```

```
x <- fit$points
```

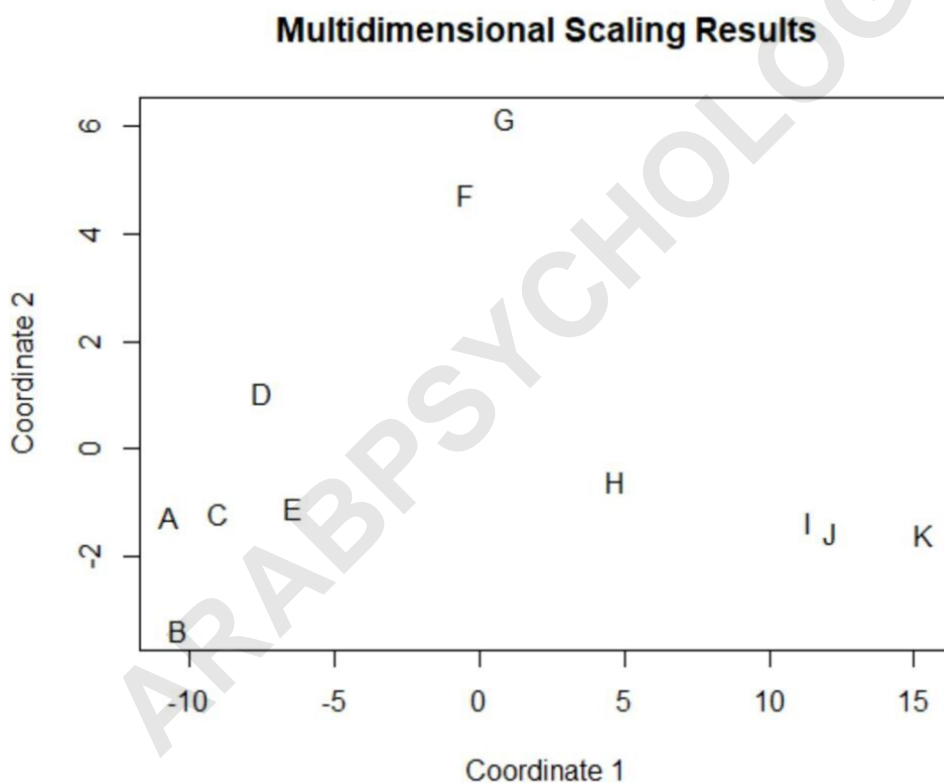
```
y <- fit$points
```

```
#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))
```



Players from the original data frame who have similar values across the original four columns (points, assists, blocks, and rebounds) are located close to each other

in the plot.

For example, players A and C are located close to each other. Here are their values from the original data frame:

```
#view data frame values for players A and C  
df
```

```
points assists blocks rebounds
```

```
A 4 3 7 4
```

```
C 6 2 6 5
```

Their values for points, assists, blocks, and rebounds are all quite similar, which explains why they're located so close together in the 2-D plot.

By contrast, consider players B and K who are located far apart in the plot.

```
#view data frame values for players B and K  
df
```

```
points assists blocks rebounds
```

```
B 4 2 3 5
```

```
K 28 11 1 2
```

Thus, the 2-D plot is a nice way to visualize how similar each players are across all of the variables in the data frame.

Players who have similar stats are grouped close together while players who have very different stats are located far apart from each other in the plot.

Note that you can also extract the exact (x, y) coordinates for each player in the plot by typing `fit`, which is the name of the variable that we stored the results of `cmdscale()` function in:

`#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

ARABPSYCHOLOGY.COM