Multidimensional Scaling (MDS)

Maths and Statistics Help Centre

Usage

- A multivariate technique.
- Aims to reveal the data structure by plotting points in 1 or 2 dimensions.
- Displays the structure of distances.
- Similar to representing data as a geometrical picture.
- Data represents the amount of dissimilarity between each pair of variables.
- Data can be on similarities, dissimilarities, distances or proximities.
- Also called Principal Coordinate Analysis.
- E.g. reconstruct a map of Europe based on distances from a road map.

How does MDS work?

- It has no assumptions about statistical distributions.
- Uses the distances of the raw data to define components.

Choosing the right number of dimensions

- If only 2 or 3 of the eigenvalues are large, then 2 or 3 dimension are sufficient.
- We can look at the proportion of variation explained (R^2) to see at what point the extra dimensions begin to plateau. An R^2 value of 0.6 is generally considered the minimum acceptable level.
- Other tests that can be used to test the validity and reliability of the results; these include Kruskal's Stress test, split data tests (e.g. leave one out cross validation), data stability tests (i.e. eliminating one brand), and test-retest reliability.

Implementation in R

- cmdscale (.)

Example

Using the built in R data set, `eurodist`. The data gives the road distances, in km, between 21 cities in Europe.

`> eurodist` This shows the data. Below is a subset of the distance matrix between the 21 cities in the dataset.

```
                Athens Barcelona Brussels Calais Cherbourg Cologne Copenhagen Geneva Gibraltar Hamburg Hook of Holland
Barcelona         3313
Brussels          2963      1318
Calais            3175      1326      204
Cherbourg         3339      1294      583    460
Cologne           2762      1498      206    409       785
Copenhagen        3276
......
```

We now perform classical MDS for the European distances using the `cmdscale` function. The default number of dimensions is k = 2.

`euro.mds.2 <- cmdscale(eurodist, eig=T)` Define `euro.mds.2` to store all the information about the MDS with 2 dimensions.

`> euro.mds.2$points` Outputs the new points based on the new dimensions.

```
                        [,1]        [,2]
Athens           2290.274680  1798.80293
Barcelona        -825.382790   546.81148
```

```
Brussels           59.183341  -367.08135
Calais            -82.845973  -429.91466
Cherbourg        -352.499435  -290.90843
Cologne           293.689633  -405.31194
Copenhagen        681.931545 -1108.64478
Geneva             -9.423364   240.40600
Gibraltar       -2048.449113   642.45854
Hamburg           561.108970  -773.36929
Hook of Holland   164.921799  -549.36704
Lisbon          -1935.040811    49.12514
Lyons            -226.423236   187.08779
Madrid          -1423.353697   305.87513
Marseilles       -299.498710   388.80726
Milan             260.878046   416.67381
Munich            587.675679    81.18224
Paris            -156.836257  -211.13911
Rome              709.413282  1109.36665
Stockholm         839.445911 -1836.79055
Vienna            911.230500   205.93020
```

`> euro.mds.2$eig` Prints the two eigenvalues

```
[1] 19538377 11856555
```

`> euro.mds.2$GOF` Prints the proportion of variance explained by each component in the scaling.

```
[1] 0.7968344 0.8679134
```

The interpretation of the goodness of fit of the configuration is similar to R^2 – higher the better.

Changing the number of dimensions k can sometimes give a better representation of the data.
Increase the number of dimensions and run the commands as before.

```
euro.mds.4 <- cmdscale(eurodist, k=4, eig=T)
```

We need to find which maximum value of dimension –k is necessary.
If we are considering values of k from 1 to 10 we can use a loop to carry out MDS for all 10 possible numbers of dimensions.

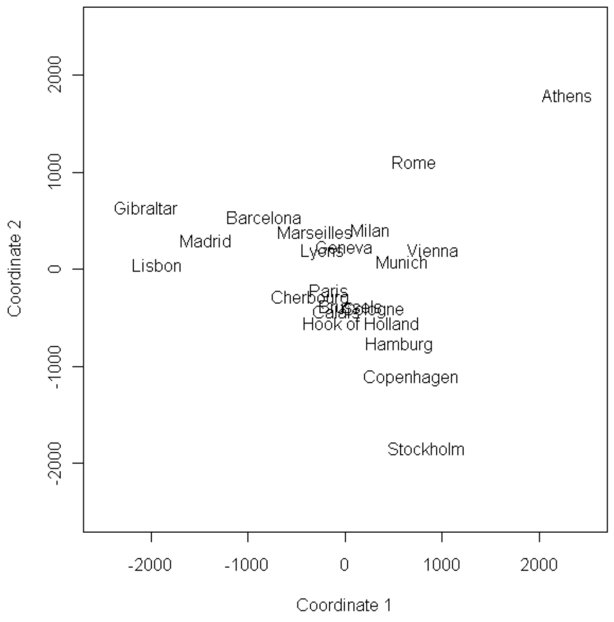
We write a loop to return a matrix with the first column the number of dimensions and the second representing the proportion of variation in the data explained by the eigenvalues or dimensions.

```
               P.k
 [1,]  1 0.4959033
 [2,]  2 0.7968344
 [3,]  3 0.8356380
 [4,]  4 0.8640328
 [5,]  5 0.8840672
 [6,]  6 0.8988302
 [7,]  7 0.9054881
 [8,]  8 0.9103764
 [9,]  9 0.9140588
[10,] 10 0.9167991
```

Looking at the values above we see that 2 or 3 dimensions seem reasonable as there is a plateau in the amount of extra information added for the extra dimensions.

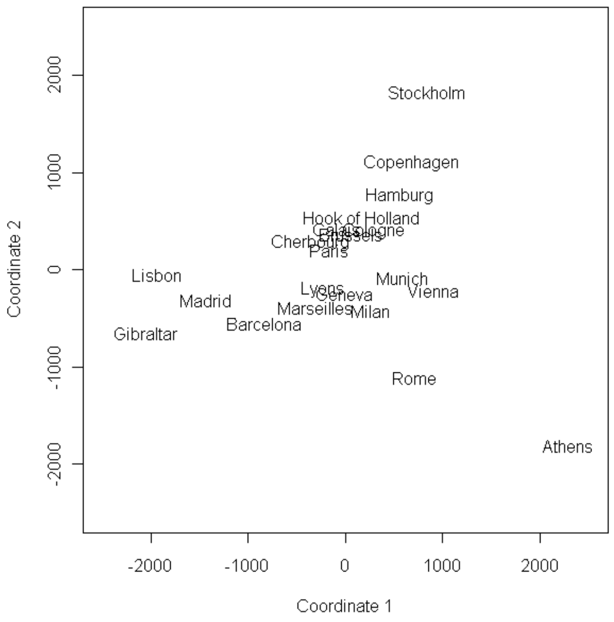
With 2 dimensions, we can generate a 2-D representation of the solution for k=2.

```
plot(euro.mds.2$points[,1], euro.mds.2$points[,2], type='n', xlab="Coordinate 1", ylab="Coordinate 2",
xlim=c(-2500,2500), ylim=c(-2500,2500) )
text(euro.mds.2$points[,1], euro.mds.2$points[,2], labels=labels(eurodist) )
```



We notice that the map is incorrectly orientated with respect to North and South. We use a rotation of 180^o to correctly orientate the map.

```
plot(euro.mds.2$points[,1], -euro.mds.2$points[,2], type='n', xlab="Coordinate 1", ylab="Coordinate 2",
xlim=c(-2500,2500), ylim=c(-2500,2500) )
text(euro.mds.2$points[,1], -euro.mds.2$points[,2], labels=labels(eurodist))
```


Stockholm
Copenhagen
Hamburg
Hook of Holland
Calais
Cologne
Brussels
Cherbourg
Paris
Lisbon
Munich
Lyons
Geneva
Vienna
Madrid
Marseilles
Milan
Barcelona
Gibraltar
Rome
Athens
Coordinate 1
Coordinate 2