MDS III

By Dr. Kenneth E. Homa
Georgetown University

Mengqian LU

Some Theory: Classical MDS

- Objective: $\boxed{\{d_{ij}\} \approx (rescaled)\{\delta_{ij}\}}$
- Input: Euclidean distances between n objects in p dimensions

所以要Centering

- Output: Coordinates of points invariant to rotation, shift and reflection

- Two steps:
  1. Compute inner product matrix, B, from distance $D = \left\{d_{ij}\right\}$
  2. Compute positions from B

$$d_{ij}^2 = (x_i - x_j)^T (x_i - x_j) = x_i^T x_i + x_j^T x_j - 2x_i^T x_j$$

Let coordinates be $x_i \;\; (i = 1, \ldots, n)$, where $x_i = (x_{i1}, \ldots, x_{ip})^T$

$$d_{ij}^2 = \sum_{k=1}^{p} (x_{ik} - x_{jk})^2, \text{assuming } \bar{x} = 0$$

$$B = \mathbf{XX}^T, \text{with } b_{ij} = \sum_{k=1}^{p} x_{ik} x_{jk} = x_i^T x_j$$

B: Inner product matrix

$$\boxed{d_{ij}^2 = b_{ii} + b_{jj} - 2b_{ij}}$$

Centering of coordinate matrix X:

$$\boxed{\sum_{i=1}^{n} b_{ij} = 0}$$

$$\sum\nolimits_{i=1}^{n} b_{ij} = 0 \text{ and } \boxed{d_{ij}^2 = b_{ii} + b_{jj} - 2b_{ij}}$$

$$\sum_{i} d_{ij}^2 = \sum_{i} (b_{ii} + b_{jj} - 2b_{ij}) \Leftrightarrow \frac{1}{n}\sum_{i=1}^{n} d_{ij}^2 = \frac{1}{n}\sum_{i=1}^{n} b_{ii} + b_{jj}$$

$$\sum_{i} d_{ij}^2 = \sum_{i} (b_{ii} + b_{jj} - 2b_{ij}) \Leftrightarrow \frac{1}{n}\sum_{j=1}^{n} d_{ij}^2 = \frac{1}{n}\sum_{j=1}^{n} b_{jj} + b_{ii}$$

$$\sum_{i,j} d_{ij}^2 = \sum_{i,j} (b_{ii} + b_{jj} - 2b_{ij}) \Leftrightarrow \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2 = \frac{2}{n}\sum_{i=1}^{n} b_{ii}$$

$$b_{ij} = -\frac{1}{2}(d_{ij}^2 - d_{i\bullet}^2 - d_{\bullet j}^2 + d_{\bullet\bullet}^2)$$

$B = \mathbf{X}\mathbf{X}^T$

B is a symmetric and positive definite n-by-n matrix

B can be diagonalized: $B = V \Lambda V^T$

$\Lambda = diag(\lambda_1, ..., \lambda_p)$, with $\lambda_1 \geq \lambda_2 ... \geq \lambda_p$ (eigenvalues)

Columns of V are eigenvectors

Some eigenvalues will be zero;

Drop them: $B = V_1 \Lambda_1 V_1^T$

对称正定矩阵所有特征值为正?
YES, 但是未必是欧氏范数

$\mathbf{X} = V_1 \Lambda_1^{\frac{1}{2}}$

"Take square root" of matrix B: $\boxed{\mathbf{X} = V_1 \Lambda_1^{\frac{1}{2}}}$

X contains the point configuration in R^p

Does this remind you of PCA?

MDS vs. PCA

	MDS	PCA
Data	N x N Distance Matrix	P x P Covariance Matrix
Approach	Spectral decomposition	
Eigenvector/Eigenvalue	Same nonzero eigenvalues with a constant factor	
Results	The same	
Interpretations	▪ Dimension reduction without coordinates ▪ Preserve distances between data points ▪ Normally target 2 or 3d representation	▪ Dimension reduction with coordinates ▪ Preserve covariance of data ▪ Feature extractions, reconstruction
Computation	▪ $O((n+d)D^2)$	▪ $O((D+d)n^2)$

MDS: Analysis and Visualization Using R

- Metric MDS in R

```
cmdscale() most popular function
wcmdscale() weighted classical MDS
```

Principle Cordinates

```
pco() in ecodist
pco() in labdsv
pcoa() in ape
```

Package: smacof

Example with **smacof**

$$d_{ij}(X) \approx \delta_{ij}$$

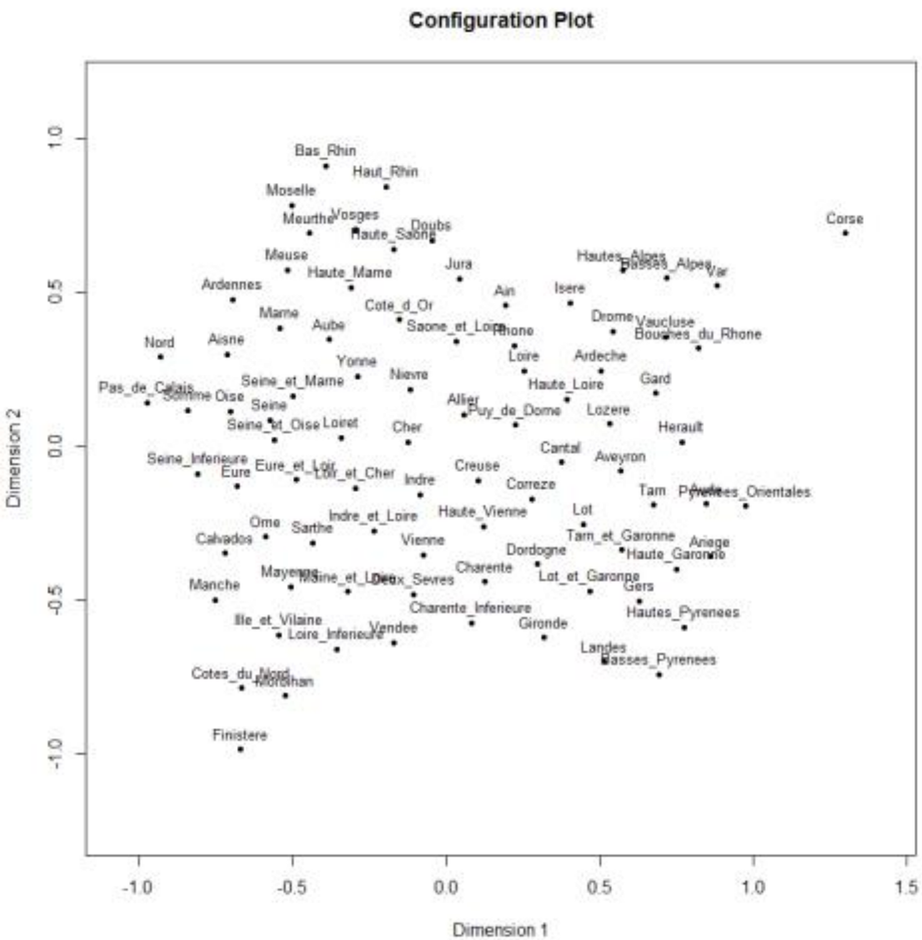
$$d_{ij}(X) = \sqrt{\sum_{l=1}^{P} (x_{il} - x_{jl})^2}$$

- The elements of X are *configurations* of the objects.
- The *configurations* represent the coordinates in the configuration plot.

- Distances between French department centroids in 1830

cran.r-project.org

- Distances between French department centroids in 1830



Packages used: `smacof` `RgoogleMaps`

data(Guerry)

cran.r-project.org

Classical MDS – Goodness of fit

- Goodness of fit (GOF) by reducing p-dimension to m-dimension

$$GOF = \frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{p} \lambda_i}$$ in cmdscale() $$GOF = \frac{\sum_{i=1}^{m} |\lambda_i|}{\sum_{i=1}^{p} |\lambda_i|}$$ is used

like Cumulative
like Cumulative

- Minimize the differences between $\{d_{ij}\}$ and $\{\delta_{ij}\}$

Notes: Classical MDS is good for Euclidean input data, quite fast in terms of computation

```
library(smacof)
library(RgoogleMaps)

data(Guerry)
fit.guerry = mds(Guerry)
plot(fit.guerry)

theta = 82*pi/180 ## degrees to radians
rot = matrix(c(cos(theta), sin(theta), -sin(theta),
cos(theta)), ncol = 2)
configs82 = fit.guerry$conf %*% rot ## rotated configurations
francemap1 = GetMap(destfile="mypic1.png", zoom = 6,
center = c(46.55, 3.05),maptype = "satellite")

PlotOnStaticMap(francemap1)
text(configs82*280, labels = rownames(configs82),
col = "white", cex = 0.7)
```

cran.r-project.org

Classical MDS – Goodness of fit

- Goodness of fit (GOF) by reducing p-dimension to m-dimension

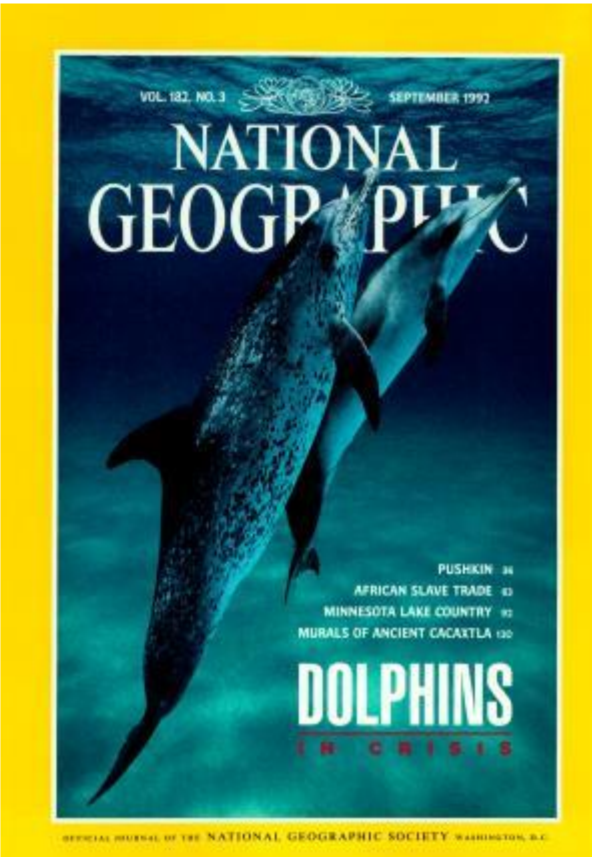
$$GOF = \frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{p} \lambda_i}$$

- What if negative eigenvalues? – cmdscale() uses $GOF = \frac{\sum_{i=1}^{m} |\lambda_i|}{\sum_{i=1}^{p} |\lambda_i|}$
- Goal: (1) Minimize the differences between $\{d_{ij}\}$ and $\{\delta_{ij}\}$;

  (2) good GOF, e.g. $\boxed{80\%}$ **Not a god-given Number**

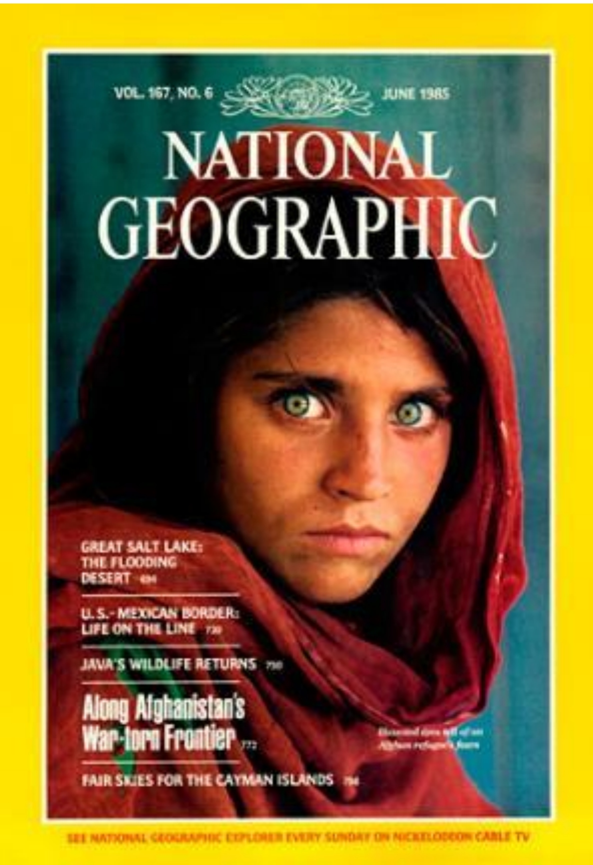
Notes: Classical MDS is good for Euclidean input data, quite fast in terms of computation

Non-metric (Ordinal) MDS



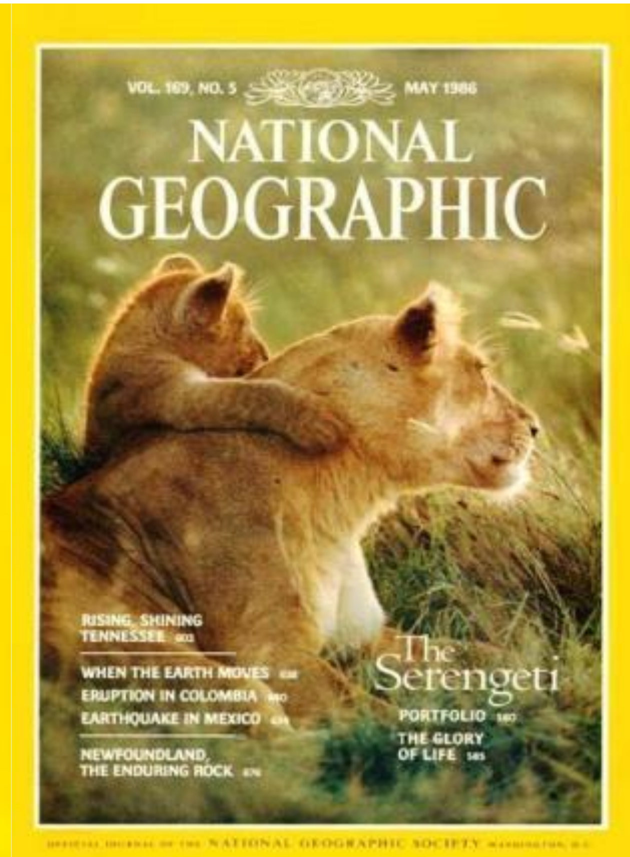
1

2



9

8



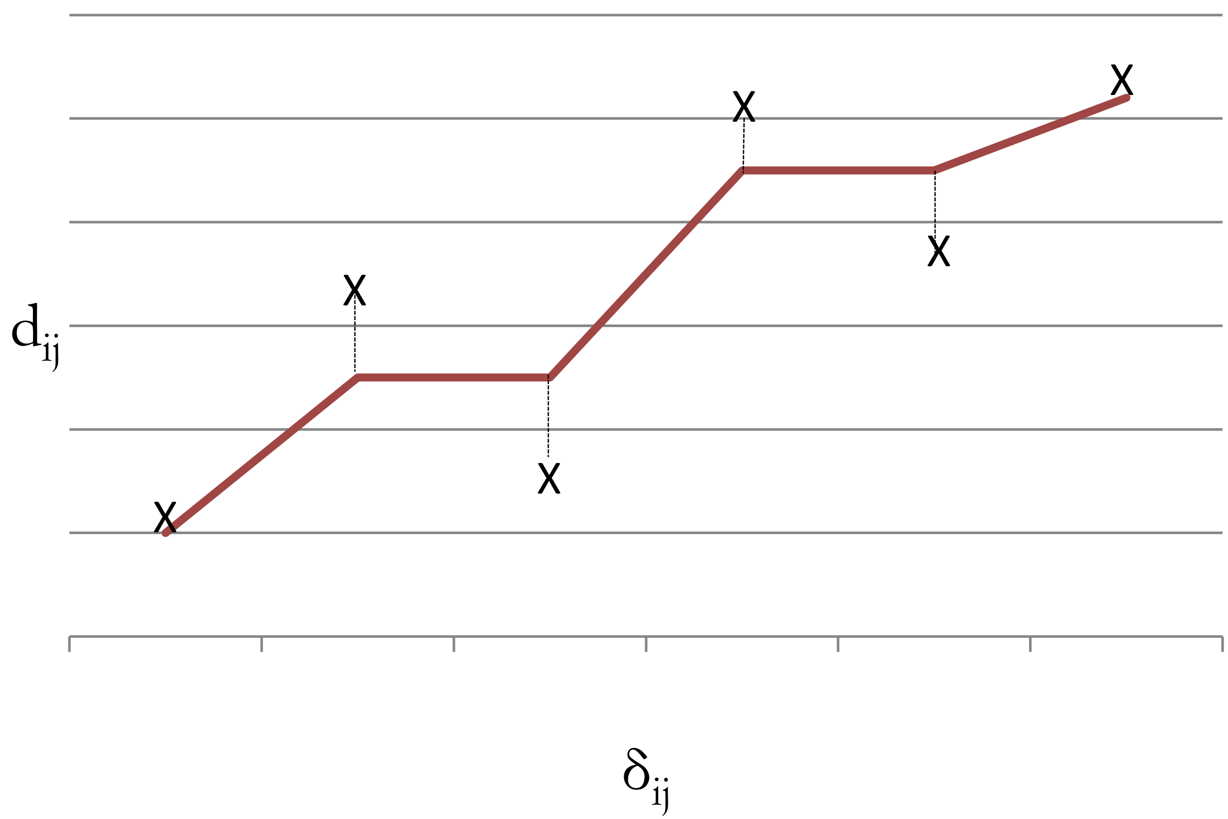
6

5

Non-metric (Ordinal) MDS: Theory

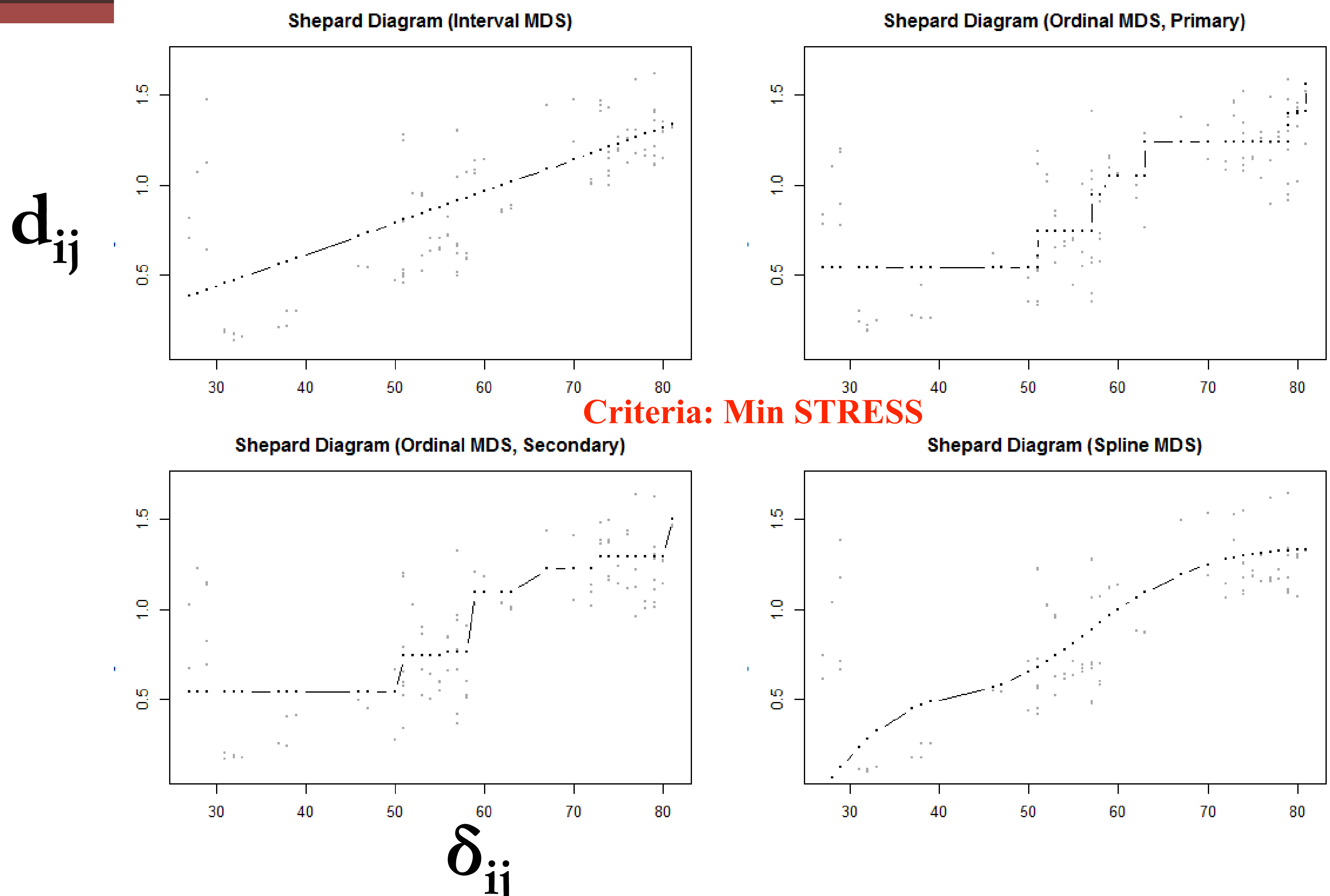
- Construct disparities $\{\hat{d}_{ij}\}$ from $\{d_{ij}\}$
- $\{\delta_{ij}\}$ are the real dissimilarities, $\{d_{ij}\}$ is the distance of representation
- $\hat{d}_{ij} = \theta(\delta_{ij})$ by least-square monotonic regression isoreg {stats}
- STRESS: $\sqrt{\dfrac{\sum_{i<j}(d_{ij} - \hat{d}_{ij})^2}{\sum_{i<j} d_{ij}^2}}$ (Kruskal's stress) isoMDS {MASS}
- Optimal solution by minimizing the STRESS

Monotonic regression



Vertical dotted line: $d_{ij} - \hat{d}_{ij}$

Transform Dissimilarities in R **smacof**



Example in R **smacof**: GOF

Normalized STREE: Kruskal's *stress*-1

$$\sigma(X) = \sqrt{\frac{\sum_{i<j} w_{ij}(d_{ij}(X) - \hat{d}_{ij})^2}{n(n-1)/2}}$$

Kruskal's *stress* $\sigma(X) = \sum_{i<j} w_{ij}(d_{ij}(X) - \delta_{ij})^2$

Define: $\sum_{i<j} w_{ij}\delta_{ij}^2 = n(n-1)/2$

δ_{ij} is the true dissimilarities

- "stress-per-point": **smacof** provides contribution as %, similar to the concept of influential points in regression

What is a "good" STRESS threshold?

Some stress-1 benchmarks for ordinal MDS

Poor	Fair	Good	Excellent	Perfect
0.2	0.1	0.05	0.025	0.00

Kruskal (1964). "Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis." *Psychometrika*, **29**, 1-27 就像心理学用*代表显著程度

However, many aspects need to be considered

Borg I, Groenen PJF, Mair P (2012). *Applied Multidimensional Scaling*. Springer, New York

Construct δ_{ij}

`sim2diss()` in **smacof**

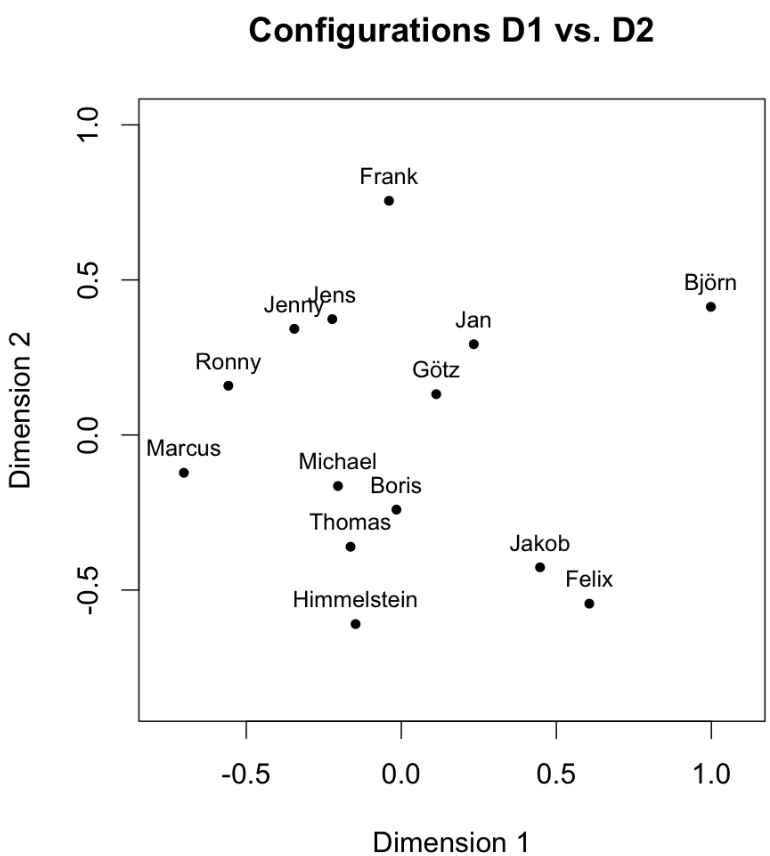
[NOTE] smacof always requires dissimilarities as input

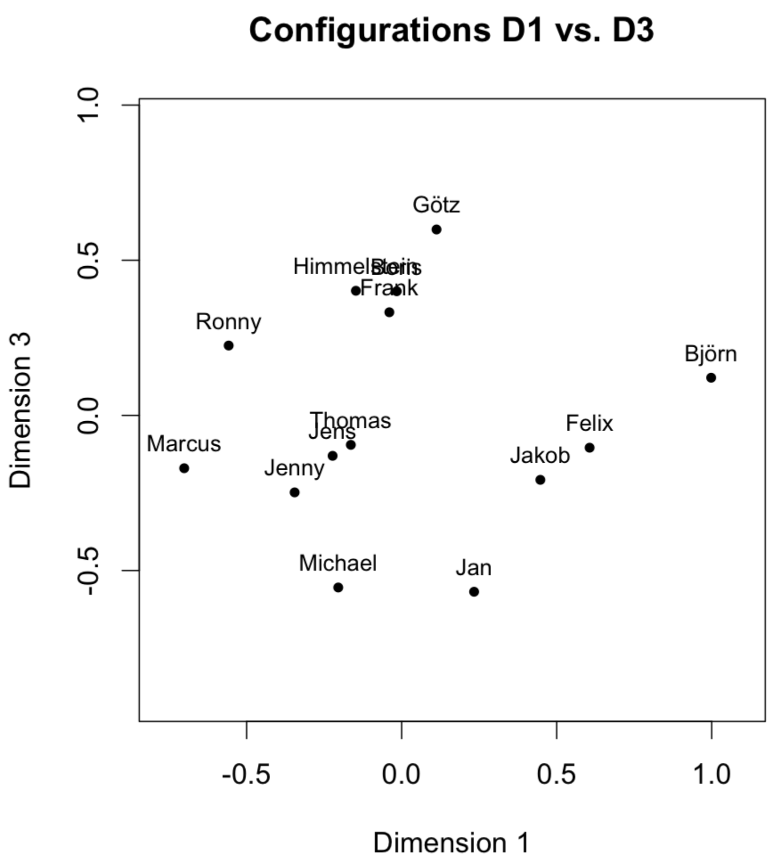
Example:

```
> head(RockHard)
  Year Month               Band                   Album Götz Thomas Frank Björn Jan Boris Himmelstein Michael Jens Ronny
1 2013     1              Attic          The Invocation  8.5    8.0   8.0   9.0 7.0   8.5         8.5     7.0   NA    NA
2 2013     1            Paradox       Tales Of The Weird  7.5    7.0   8.0   9.0 7.0   7.5         7.0     6.5   NA    NA
3 2013     1               Züül       To The Frontlines  8.0    7.0   7.5   8.5 7.0   8.5         8.0     6.5   NA    NA
4 2013     1 Chapel Of Disease     Summoning Black Gods  8.0    7.5   8.0   8.0 7.5   8.0         8.0     6.5   NA    NA
5 2013     1  Dropkick Murphys Signed And Sealed In Blood 7.0   7.5   7.5   6.0 6.5   7.0         8.5     8.5   NA    NA
6 2013     1          Saturnus      Saturn In Ascension  6.0    7.0   6.5   8.0 7.0   7.5         7.0     6.0   NA    NA
  Felix Jakob Marcus Jenny
1    NA   8.5    7.5   7.0
2    NA   7.5    8.0   7.5
3    NA   8.0    7.0   6.5
4    NA   8.0    6.5   6.0
5    NA   7.5    7.0   6.5
6    NA   8.5    7.5   7.0
```

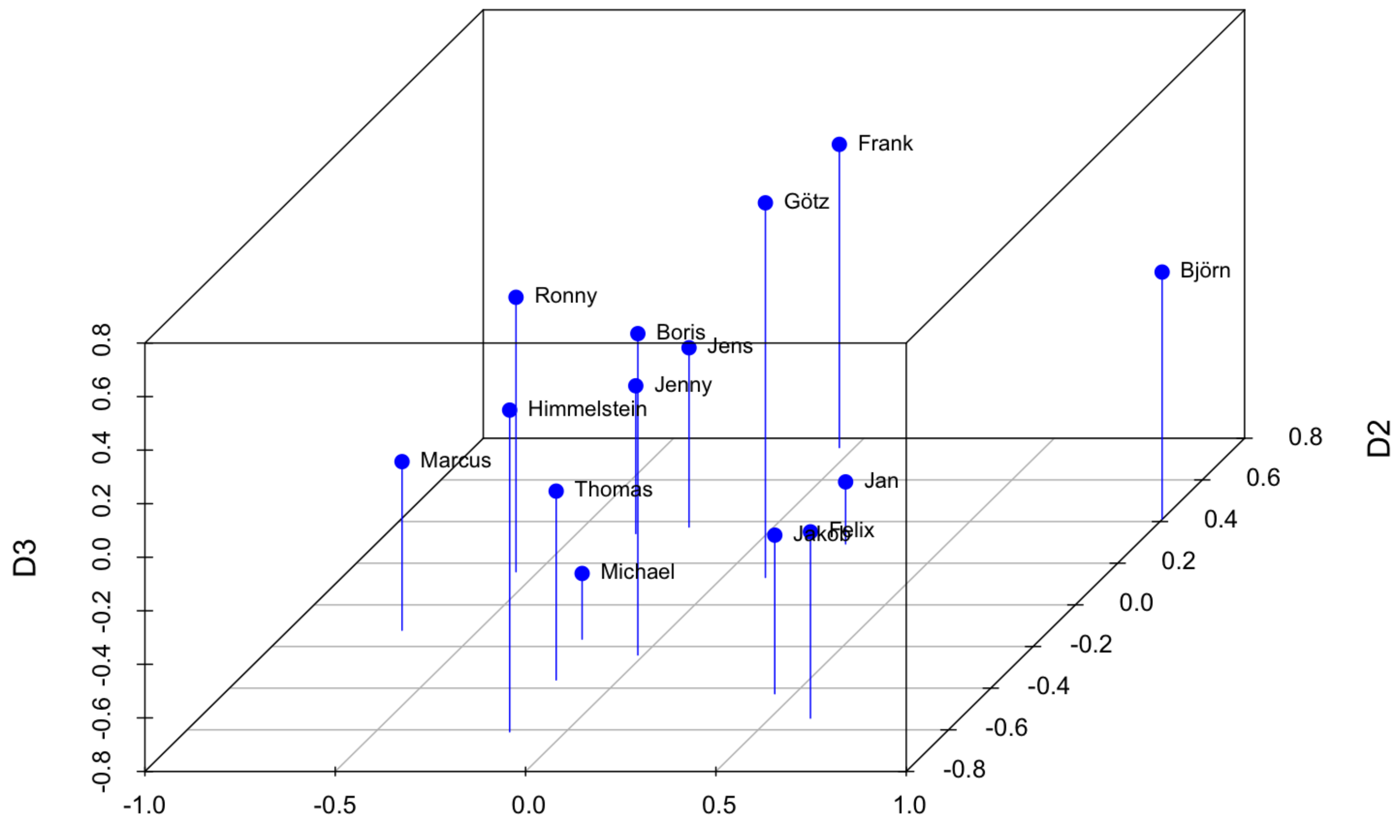
```
ratings = RockHard[,5:18]
rockdiss = dist(t(ratings))
```

```
fit.rock = mds(rockdiss, ndim = 3)
plot(fit.rock, plot.dim = c(1,2), main = "Configurations D1 vs. D2")
plot(fit.rock, plot.dim = c(1,3), main = "Configurations D1 vs. D3”)
```



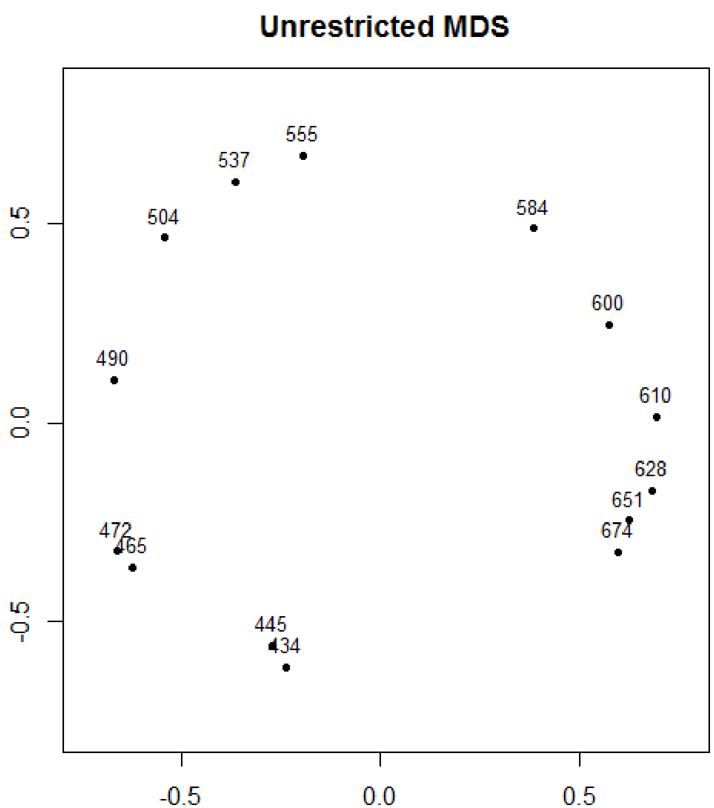


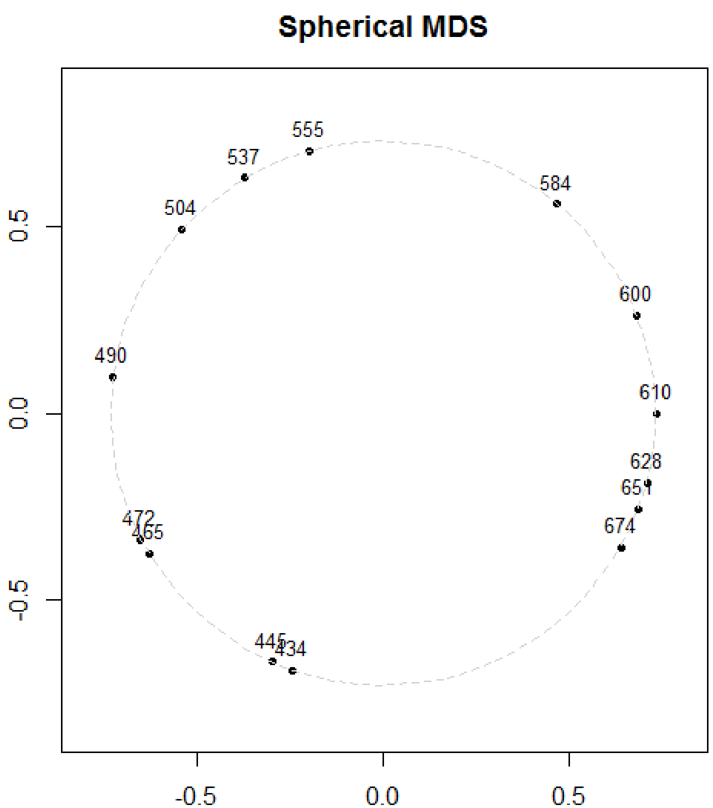
```
s3d = scatterplot3d(fit.rock$conf[,1],fit.rock$conf[,2],
fit.rock$conf[,3],color="blue", pch=19,angle = 70, type="h”,
xlab="D1”,ylab="D2”,zlab="D3”)
s3d.coords = s3d$xyz.convert(fit.rock$conf[,1], fit.rock$conf[,2],
fit.rock$conf[,3])
text(s3d.coords$x, s3d.coords$y,labels=dimnames(fit.rock$conf)[[1]],
cex=.7, pos=4)
```



Confirmatory MDS

```
ekmanD = sim2diss(ekman, method = 1)
fit.basic = mds(ekmanD, type = "ordinal")
fit.circ = smacofSphere(ekmanD, type = "ordinal", verbose = FALSE)
plot(fit.basic, main = "Unrestricted MDS")
plot(fit.circ, main = "Spherical MDS")
```





MDS issue: initial configuration

- MDS often ends up in a local minimum.
  - **smacof** uses classical MDS to determine starting configurations
  - Use several random initializations , choose the one with lowest STRESS (still no guarantee though)

```
> Lawler
     T1M1 T2M1 T3M1 T1M2 T2M2 T3M2 T1M3 T2M3
T2M1 0.53
T3M1 0.56 0.44
T1M2 0.65 0.38 0.40
T2M2 0.42 0.52 0.30 0.56
T3M2 0.40 0.31 0.53 0.56 0.40
T1M3 0.01 0.01 0.09 0.01 0.17 0.10
T2M3 0.03 0.13 0.03 0.04 0.09 0.02 0.43
T3M3 0.06 0.01 0.30 0.02 0.01 0.30 0.40 0.40
```

Lawler dataset (*Lawler 1967*) in R
The performance of managers

```
> Lawler
     T1M1 T2M1 T3M1 T1M2 T2M2 T3M2 T1M3 T2M3
T2M1 0.53
T3M1 0.56 0.44
T1M2 0.65 0.38 0.40
T2M2 0.42 0.52 0.30 0.56
T3M2 0.40 0.31 0.53 0.56 0.40
T1M3 0.01 0.01 0.09 0.01 0.17 0.10
T2M3 0.03 0.13 0.03 0.04 0.09 0.02 0.43
T3M3 0.06 0.01 0.30 0.02 0.01 0.30 0.40 0.40
```

About the matrix:

Criteria:

T1 = quality of output,

T2 = ability to generate output,

T3 = demonstrated effort to perform

Evaluation:

M1 = rating by superior

M2 = peer rating

M3 = self-rating

Steps:

1. Convert similarities to dissimilarities (**smacof** only works with dissimilarities)
2. (1) Classical MDS – Check STRESS

   (2) Random initialization – Check STRESS

3. Compare & Assess

```
LawlerD = sim2diss(Lawler) # Converts similarities to dissimilarities
fitclas = mds(LawlerD)          # Classical MDS
fitclas$stress                  # Check STRESS
## [1] 0.2414665
stressvec = NULL
set.seed(429)                   # In order to reproduce
for(i in 1:20) {
 fitran = mds(LawlerD, init = "random")
 stressvec[i] = fitran$stress
}
stressvec                       # Check STRESS
# [1] 0.2442704 0.2576444 0.2544550 0.2521948 0.2569839 0.2547181
# [7] 0.2515084 0.2689958 0.2423060 0.2573334 0.2563338 0.2617981
# [13] 0.2461281 0.2423095 0.2526809 0.2618253 0.2521941 0.2425599
# [19] 0.2465002 0.2430243
# The 9th random solution is the best in terms of STRESS.
```

cran.r-project.org

Find benchmark for STRESS

1. **Random dissimilarities**

```
set.seed(429)
stressvec = randomstress(n = 9, ndim = 2, nrep = 500)
mean(stressvec) # to get a benchmark
## [1] 0.3065868

fit = mds(LawlerD)
fit$stress
# [1] 0.2414665 # 0.24<0.3 but not a guarantee of a
                  good solution
```

2. Modern approaches focus on **permutations of dissimilarity matrix**

```
set.seed(429)
res.perm = permtest(fit, nrep = 1000, verbose =
FALSE)
```

```
> set.seed(429)
> res.perm = permtest(fit, nrep = 1000, verbose = FALSE)
> res.perm
Call: permtest.smacof(object = fit, nrep = 1000, verbose = FALSE)

SMACOF Permutation Test
Number of objects: 9
Number of replications (permutations): 1000

Observed stress value: 0.241
p-value: 0.325
```

0.325 is the new benchmark to judge whether our MDS is sound

- Permutation tests provide more useful null distribution (or critical criterion) than random dissimilarity approach.

Assess and present the results

"*STRESS-per-point*" contributions

```
> fit$spp # spp: Stress per point in percentages
 T1M1      T2M1      T3M1      T1M2      T2M2      T3M2
10.678739 14.836683 11.288724 10.329160  9.911150 10.470871
     T1M3      T2M3      T3M3
10.600777 12.332179  9.551717
# T2M1 is the ability to generate output, rated by supervisor
> plot(fit, plot.type="stressplot")
> plot(fit, plot.type="bubbleplot")
```

```
> plot(fit, plot.type="stressplot")
> plot(fit, plot.type="bubbleplot")
```