Multidimensional scaling (MDS)

Visualising dissimilarity data

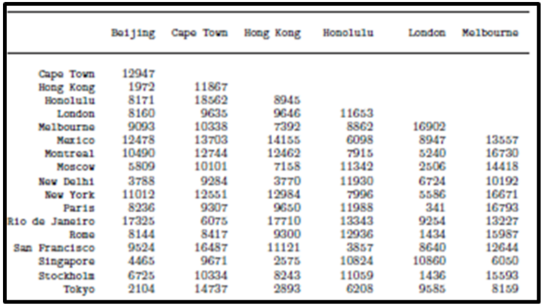
Francesca Little (reformatted and edited by Miguel Rodo)

2024-02-12

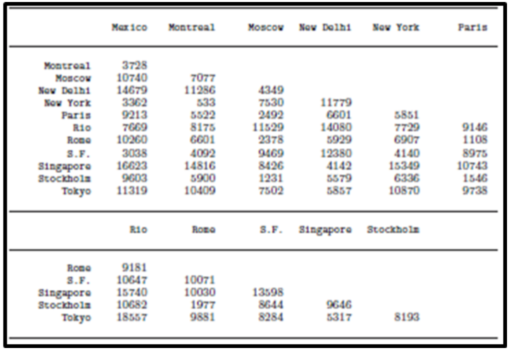
Key references

- AJ Izenman, "Modern Multivariate Statistical Techniques", Ch 13, Springer, 2013.
  - Much content without citation is based on this book, so give them full credit for uncited (correct!) content.
- Hastie, Trevor, Robert Tibshirani, and J. H. (Jerome H.) Friedman. "The Elements of Statistical Learning : Data Mining, Inference, and Prediction. 2nd ed. Springer Series in Statistics", Section 14.8. New York: Springer, 2009.

Example: representing cities' relative locations by airline distance



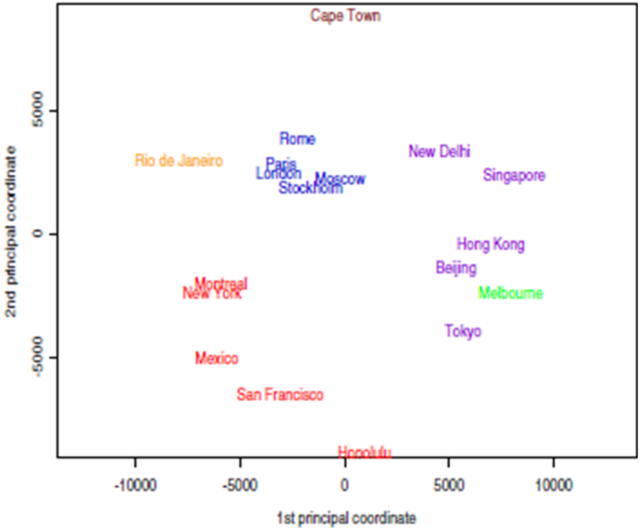
	Beijing	Cape Town	Hong Kong	Honolulu	London	Melbourne
Cape Town	12947					
Hong Kong	1972	11867				
Honolulu	8171	18562	8945			
London	8160	9635	9646	11653		
Melbourne	9093	10338	7392	8862	16902	
Mexico	12478	13703	14155	6098	8947	13557
Montreal	10490	12744	12462	7915	5240	16730
Moscow	5809	10101	7158	11342	2506	14418
New Delhi	3788	9284	3770	11930	6724	10192
New York	11012	12551	12984	7996	5586	16671
Paris	8236	9307	9650	11988	341	16793
Rio de Janeiro	17325	6075	17710	13343	9254	13227
Rome	8144	8417	9300	12936	1434	15987
San Francisco	9524	16487	11121	3857	8640	12644
Singapore	4465	9671	2575	10824	10860	6050
Stockholm	6725	10334	8243	11059	1436	15593
Tokyo	2104	14737	2893	6208	9585	8159



	Mexico	Montreal	Moscow	New Delhi	New York	Paris
Montreal	3728					
Moscow	10740	7077				
New Delhi	14679	11286	4349			
New York	3362	533	7530	11779		
Paris	9213	5522	2492	6601	5851	
Rio	7669	8175	11529	14080	7729	9146
Rome	10260	6601	2378	5929	6907	1108
S.F.	3038	4092	9469	12380	4140	8975
Singapore	16623	14816	8426	4142	15349	10743
Stockholm	9603	5900	1231	5579	6336	1546
Tokyo	11319	10409	7502	5857	10870	9738

	Rio	Rome	S.F.	Singapore	Stockholm
Rome	9181				
S.F.	10647	10071			
Singapore	15740	10030	13598		
Stockholm	10682	1977	8644	9646	
Tokyo	18557	9881	8284	5317	8193

Two-dimensional MDS map



What is MDS?

- Goal:
  - Represent dissimilarity-data in a low-dimensional space, typically for visualization.

What is MDS?

- Goal:
  - Represent dissimilarity-data in a low-dimensional space, typically for visualization.
- Contrast with typical dimensionality reduction techniques:
  - Whilst other such techniques, e.g. PCA, begin with the original data points (i.e. $\mathbf{X}$), MDS begins with the dissimilarities or similarities between points.

What is MDS?

- Goal:
  - Represent dissimilarity-data in a low-dimensional space, typically for visualization.
- Contrast with typical dimensionality reduction techniques:
  - Whilst other such techniques, e.g. PCA, begin with the original data points (i.e. $\mathbf{X}$), MDS begins with the dissimilarities or similarities between points.
- MDS is a family of techniques, differing by:
  - Valid interpretations of dissimilarities between points in generated map (ratio, interval or rank)
  - Objective function
    - Difference measure
    - Weights of per-dissimilarity errors
  - Method of optimisation (eigen-decomposition vs numerical)

Scales of measurement

- Stevens (1946) proposed four scales of measurement:
  - Ratio: a natural zero exists, meaning ratios can be meaningfully defined.
    - Example: height.
    - Non-example: temperature.
  - Interval: no natural exists, but the differences between points is directly comparable.
    - Example: temperature.
    - Non-example: rank-based judgements.
  - Ordinal: points have a greater than/less than relationship to one another, but the differences between points are not necessarily directly comparable.
    - Example: rank-based judgements.
    - Non-example: someone's name.
  - Nominal: no greater than/less than relationship between points.
    - Example: people's names.

Categories of MDS techniques

- Metric:
  - Aim to represent actual dissimilarities.
  - Typically, dissimilarities may be interpreted on at least the interval scale.
  - Sub-techniques:
    - Classical scaling
    - Least-squares scaling
- Non-metric:
  - Aim to preserve ranks.
  - Dissimilarities only interpretable in an ordinal sense.

Input data

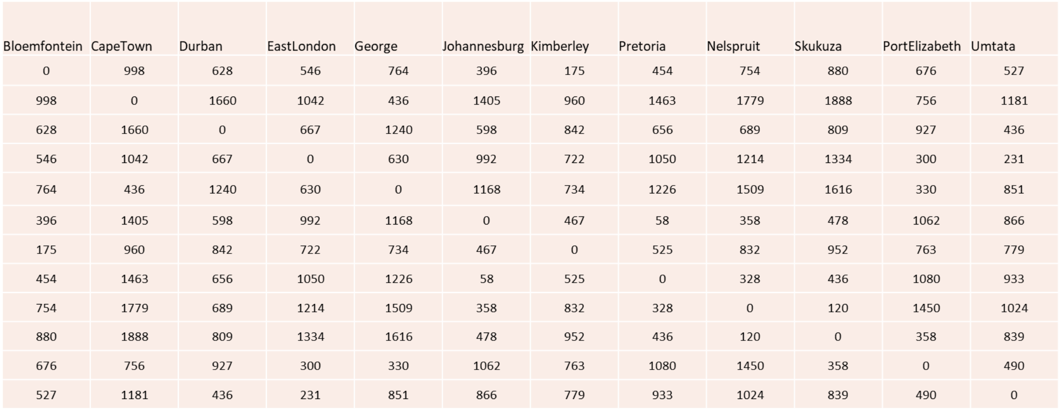
Similarity/dissimilarity measure

- *Purpose*:
  - Captures differences between two observations
- *Properties*:
  - Symmetric: Dissimilarity from object A to B the same as dissimilarity from object B to A
  - Each object has zero dissimilarity from itself
  - Typically: range does not cross zero
- *Notation*:
  - Dissimilarity from i-th object to j-th object: δ_{ij}

Proximity matrix

- *Nature*:
  - Matrix whose i-th, j-th element is δ_{ij}
- *Implied properties*:
  - Symmetric
  - Hollow (0s on diagonal)
  - Square ($n \times n$)
  - Typically, either non-negative or non-positive
- *Notation*:
  - Proximity matrix: $\Delta = (\delta_{ij})$
  - Number of lower off-diagonal elements: m, where $m = \binom{n}{2} = \frac{1}{2}n(n-1)$
    - Excludes duplicate and diagonal elements

Example proximity matrix: dissimilarities between SA cities



Bloemfontein	CapeTown	Durban	EastLondon	George	Johannesburg	Kimberley	Pretoria	Nelspruit	Skukuza	PortElizabeth	Umtata
0	998	628	546	764	396	175	454	754	880	676	527
998	0	1660	1042	436	1405	960	1463	1779	1888	756	1181
628	1660	0	667	1240	598	842	656	689	809	927	436
546	1042	667	0	630	992	722	1050	1214	1334	300	231
764	436	1240	630	0	1168	734	1226	1509	1616	330	851
396	1405	598	992	1168	0	467	58	358	478	1062	866
175	960	842	722	734	467	0	525	832	952	763	779
454	1463	656	1050	1226	58	525	0	328	436	1080	933
754	1779	689	1214	1509	358	832	328	0	120	1450	1024
880	1888	809	1334	1616	478	952	436	120	0	358	839
676	756	927	300	330	1062	763	1080	1450	358	0	490
527	1181	436	231	851	866	779	933	1024	839	490	0

Common (dis)similarity measures

Dissimilarity measures

- **Minkowski distance**:
  - $\delta_{ij} = \left(\sum_{k=1}^{r} |X_{ik} - X_{jk}|^p\right)^{1/p}$
  - $p = 1$: city-block/Manhattan distance
  - $p = 2$: Euclidean distance
  - $p = \infty$: Chebychev distance
- **Comparing two sequences**:
  - Hamming distance: $\#$ of indices with different values in two equal-length sequences
- **Real-world**:
  - Travel time between destinations
  - Difference in time to failure
  - Subjective:
    - Difference in rated quality

Similarity measures

- **Continuous**:
  - Centred dot product: $(\mathbf{x}_i - \bar{\mathbf{x}})'(\mathbf{x}_j - \bar{\mathbf{x}})$
  - Correlation coefficient: Pearson, Spearman, etc.
- **Binary** (comparing two sets):
  - Jaccard (JACCARD 1901): $|A \cap B|/|A \cup B|$
  - See Choi, Cha, and Tappert (n.d.) for many (many, many) more
- **Real-world**:
  - Frequency of signal confusion

Classical scaling

- **Classical scaling** (Torgerson 1952, 1958) (references inaccessible via UCT) is a variant of metric MDS that finds the optimal low-rank configuration of points such that their centred inner products match those in the original space as closely possible.
  - It essentially calculates the answer in this direction:
    - dissimilarities -> centred inner products -> low-rank configuration.
  - The motivation for this is that the optimisation problem has a known solution based on the eigendecomposition, thus avoiding iterative optimisation (apart from calculating eigen decomposition).
  - If the dissimilarities are Euclidean ($\delta_{ij} = |\mathbf{x}_i - \mathbf{x}_j| = \sqrt{\sum (x_{ik} - x_{jk})^2}$), then the solution matches the principal component solution.

Procedure for classical scaling: major steps

1. We begin with the dissimilarities Δ, and do not have access to the original data matrix X.
2. From Δ, we calculate the matrix B of centred dot products, i.e. where $b_{ij} = \mathbf{x}_i'\mathbf{x}_j$, for $\mathbf{x}_i$ and $\mathbf{x}_j$ the i-th and j-th rows of X.
3. We find the rank t matrix B^* whose L_2 norm from B is smallest.
4. The minimsing configuration is a function of the SVD of $\mathbf{B}$.

Obtaining centred inner products from dissimilarities

Given n points $x_1, \dots, x_n \epsilon \mathbb{R}^r$ that are variable-centred (i.e. $\sum_k x_{ik} = 0 \; \forall \; k \in 1,2,\dots,r$) compute an $n \times n$ matrix $\Delta = (\delta_{ij})$ of dissimilarities, where

$$\delta_{ij} = \left|\left|x_i - x_j\right|\right| = \left\{\sum_{k=1}^{r}(x_{ik} - x_{jk})^2\right\}^{1/2}.$$

Then $\delta_{ij}^2 = ||x_i||^2 + ||x_j||^2 - 2x_i^T x_j$.

Let $b_{ij} = x_i^T x_j$. Then,

$$\delta_{ij}^2 = b_{ii} + b_{jj} - 2b_{ij}.$$

Summing over i and/or j gives the following relationships:

$$\left.\begin{array}{l} \displaystyle\sum_{i=1}^{n} \delta_{ij}^2 = \sum_i b_{ii} + nb_{jj} \rightarrow nb_{jj} = \sum_i \delta_{ij}^2 - T \\ \displaystyle\sum_{j=1}^{n} \delta_{ij}^2 = \sum b_{jj} + nb_{ii} \rightarrow nb_{ii} = \sum \delta_{ij}^2 - T \\ \displaystyle\sum_{j=1}^{n}\sum_{i=1}^{n} \delta_{ij}^2 = 2n\sum_i b_{ii} = 2nT \end{array}\right\} *$$

Substituting * into the expression for δ_{ij}^2 gives $\delta_{ij}^2 =$

$$\begin{aligned} & b_{ii} + b_{jj} - 2b_{ij} \\ &= \frac{1}{n}\sum_i \delta_{ij}^2 - \frac{1}{n}T + \frac{1}{n}\sum_j \delta_{ij}^2 - \frac{1}{n}T - 2b_{ij} \\ &= \frac{1}{n}\delta_{.j}^2 + \frac{1}{n}\delta_{i.}^2 - \frac{2}{n}\mathrm{T} - 2b_{ij} \end{aligned}$$

$$\rightarrow b_{ij} = -{}^1\!/_2\left(\delta_{ij}^2 - \frac{1}{n}\delta_{.j}^2 - \frac{1}{n}\delta_{i.}^2 + \frac{1}{n}\delta_{..}\right)$$

And letting $a_{ij} = -{}^1\!/_2\,\delta_{ij}^2$,

$$b_{ij} = a_{ij} - a_{i.} - a_{.j} + a_{..}$$

Where $a_{i.} = \frac{1}{n}\sum_j a_{ij}$ etc.

So, if $A = (a_{ij})$ = a matrix of squared dissimilarities, and $B = (b_{ij})$, then

$$B = HAH$$

Where $H = I_n - \frac{1}{n}J_n$ is a centering matrix with J_n an $n \times n$ matrix of 1's.

→ B is a double-centered version of A.

Objective function for classical scaling

We wish to find n t-dimensional points, $\mathbf{Y}_1, \ldots, \mathbf{Y}_n \in \mathbb{R}^t$ such that

$$tr\{(B - B^*)^2\} = \sum_i \sum_j (b_{ij} - b^*_{ij})^2,$$

where B^* is the rank t matrix of centred inner products of the points.

- When we use Euclidean dissimilarities for proximity matrix, this is equivalent to PCA.
- From $\mathbf{B}$, the points are given by

$$\mathbf{Y} = \mathbf{V}_t \mathbf{\Lambda}_t^{1/2},$$

where $\mathbf{V}_t$ is the matrix of the first t eigenvectors of $\mathbf{B}$, and $\mathbf{\Lambda}_t$ is the diagonal matrix of the first t eigenvalues of $\mathbf{B}$.

Example of classical scaling from first principles I

- Assume that we have measured dissimilarities between four cities, saved as :

```
Delta_mat <- structure(c(0, 93, 82, 13                                 33,
60, 111, 0), dim = c(4L, 4L))
Delta_mat |> signif(2)
```

```
     [,1] [,2] [,3] [,4]
[1,]    0   93   82  130
[2,]   93    0   52   60
[3,]   82   52    0  110
[4,]  130   60  110    0
```

- Calculate $\mathbf{A}$, where $\mathbf{A}_{ij} = -\frac{1}{2}\delta_{ij}^2$:

```
# remember, R by default performs
# element-wise multiplication
A_mat <- -0.5 * Delta_mat^2
A_mat |> signif(3)
```

```
      [,1]  [,2]  [,3]  [,4]
[1,]     0 -4320 -3360 -8840
[2,] -4320     0 -1350 -1800
[3,] -3360 -1350     0 -6160
[4,] -8840 -1800 -6160     0
```

Example of classical scaling from first principles II

- Calculate $\mathbf{B} = \mathbf{HAH}$, where $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}'$:

```
H <- diag(4) - 1/4 * matrix(1, 4, 4)
B_mat <- H %*% A_mat %*% H
B_mat |> signif(3)
```

```
      [,1]     [,2]     [,3]  [,4]
[1,]  5040 -1550.00   259.00 -3740
[2,] -1550   508.00     5.31  1040
[3,]   259     5.31  2210.00 -2470
[4,] -3740  1040.00 -2470.00  5170
```

- Calculate the eigenvalues and eigenvectors of $\mathbf{B}$:

```
eig_obj <- eigen(B_mat)
```

- Calculate the principal coordinates:

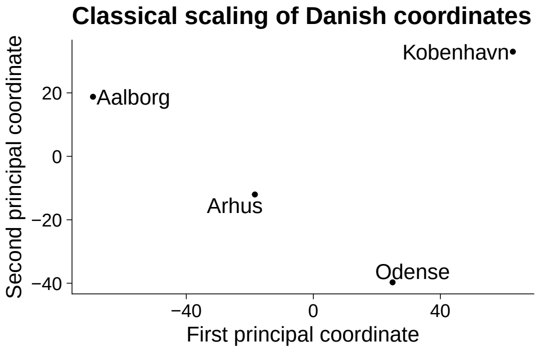
```
Y_mat <- eig_obj$vectors %*% diag(sqrt
Y_mat |> signif(2)
```

```
     [,1] [,2]   [,3]   [,4]
[1,]   63   33  0.042 -4e-07
[2,]  -18  -12 -4.900 -4e-07
[3,]   25  -40  2.600 -4e-07
[4,]  -69   19  2.300 -4e-07
```

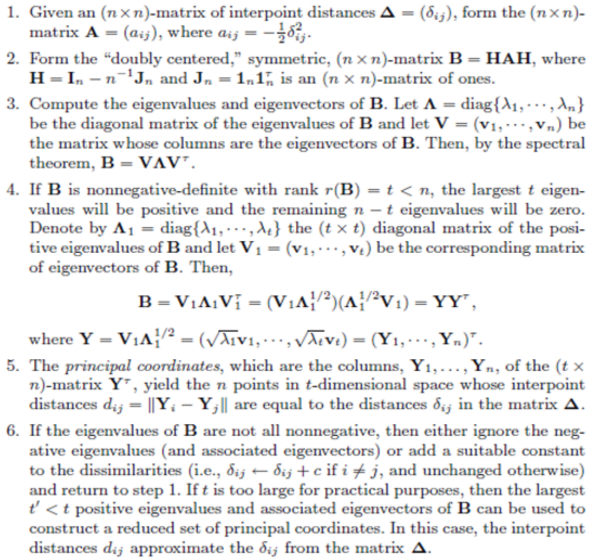
Example of classical scaling from first principles III

- Plot the first two principal coordinates:

```
plot_tbl <- Y_mat |>
 tibble::as_tibble() |>
  dplyr::mutate(
    city = c(
      "Kobenhavn", "Arhus",
      "Odense", "Aalborg"
      )
  )
p <- ggplot(plot_tbl, aes(V1, V2)) +
  geom_point(size = 3) +
  ggrepel::geom_text_repel(
    aes(label = city), size = 10
    ) +
  coord_equal()
```



Classical scaling algorithm



1. Given an $(n \times n)$-matrix of interpoint distances $\mathbf{\Delta} = (\delta_{ij})$, form the $(n \times n)$-matrix $\mathbf{A} = (a_{ij})$, where $a_{ij} = -\frac{1}{2}\delta_{ij}^2$.
2. Form the "doubly centered," symmetric, $(n \times n)$-matrix $\mathbf{B} = \mathbf{HAH}$, where $\mathbf{H} = \mathbf{I}_n - n^{-1}\mathbf{J}_n$ and $\mathbf{J}_n = \mathbf{1}_n\mathbf{1}_n^\tau$ is an $(n \times n)$-matrix of ones.
3. Compute the eigenvalues and eigenvectors of $\mathbf{B}$. Let $\mathbf{\Lambda} = \text{diag}\{\lambda_1, \cdots, \lambda_n\}$ be the diagonal matrix of the eigenvalues of $\mathbf{B}$ and let $\mathbf{V} = (\mathbf{v}_1, \cdots, \mathbf{v}_n)$ be the matrix whose columns are the eigenvectors of $\mathbf{B}$. Then, by the spectral theorem, $\mathbf{B} = \mathbf{V\Lambda V}^\tau$.
4. If $\mathbf{B}$ is nonnegative-definite with rank $r(\mathbf{B}) = t < n$, the largest t eigenvalues will be positive and the remaining $n - t$ eigenvalues will be zero. Denote by $\mathbf{\Lambda}_1 = \text{diag}\{\lambda_1, \cdots, \lambda_t\}$ the $(t \times t)$ diagonal matrix of the positive eigenvalues of $\mathbf{B}$ and let $\mathbf{V}_1 = (\mathbf{v}_1, \cdots, \mathbf{v}_t)$ be the corresponding matrix of eigenvectors of $\mathbf{B}$. Then,
$$\mathbf{B} = \mathbf{V}_1\mathbf{\Lambda}_1\mathbf{V}_1^\tau = (\mathbf{V}_1\mathbf{\Lambda}_1^{1/2})(\mathbf{\Lambda}_1^{1/2}\mathbf{V}_1) = \mathbf{YY}^\tau,$$
where $\mathbf{Y} = \mathbf{V}_1\mathbf{\Lambda}_1^{1/2} = (\sqrt{\lambda_1}\mathbf{v}_1, \cdots, \sqrt{\lambda_t}\mathbf{v}_t) = (\mathbf{Y}_1, \cdots, \mathbf{Y}_n)^\tau$.
5. The *principal coordinates*, which are the columns, $\mathbf{Y}_1, \ldots, \mathbf{Y}_n$, of the $(t \times n)$-matrix $\mathbf{Y}^\tau$, yield the n points in t-dimensional space whose interpoint distances $d_{ij} = \|\mathbf{Y}_i - \mathbf{Y}_j\|$ are equal to the distances δ_{ij} in the matrix $\mathbf{\Delta}$.
6. If the eigenvalues of $\mathbf{B}$ are not all nonnegative, then either ignore the negative eigenvalues (and associated eigenvectors) or add a suitable constant to the dissimilarities (i.e., $\delta_{ij} \leftarrow \delta_{ij} + c$ if $i \neq j$, and unchanged otherwise) and return to step 1. If t is too large for practical purposes, then the largest $t' < t$ positive eigenvalues and associated eigenvectors of $\mathbf{B}$ can be used to construct a reduced set of principal coordinates. In this case, the interpoint distances d_{ij} approximate the δ_{ij} from the matrix $\mathbf{\Delta}$.

Another classical scaling example: dissimilarities between SA cities I

First we load our data from the `DataTidy24STA5069Z` package, installing it if it's not available:

```r
if (!requireNamespace("remotes", quietly = TRUE)) {
  install.packages("remotes")
}
if (!requireNamespace("DataTidy24STA5069Z", quietly = TRUE)) {
  remotes::install_github("MiguelRodo/DataTidy24STA5069Z")
}
data("data_tidy_sa_distance", package = "DataTidy24STA5069Z")
```

Another example of classical scaling from first principles I

- Save proximity data as the matrix :

```
Delta_mat <- as.matrix(
  data_tidy_sa_distance
  ) / 1e3
Delta_mat[1:3, 1:3] |> signif(2)
```

```
     Bloemfontein CapeTown Durban
[1,]         0.00      1.0   0.63
[2,]         1.00      0.0   1.70
[3,]         0.63      1.7   0.00
```

- Calculate $\mathbf{A}$, where $\mathbf{A}_{ij} = -\frac{1}{2}\delta_{ij}^2$:

```
# remember, R by default performs
# element-wise multiplication
A_mat <- -0.5 * Delta_mat^2
A_mat[1:3, 1:3] |> signif(3)
```

```
     Bloemfontein CapeTown Durban
[1,]        0.000   -0.498 -0.197
[2,]       -0.498    0.000 -1.380
[3,]       -0.197   -1.380  0.000
```

Another example of classical scaling from first principles II

- Calculate $\mathbf{B} = \mathbf{HAH}$, where $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}'$:

```
H <- diag(nrow(A_mat)) -
  1/nrow(A_mat) *
  matrix(1, nrow(A_mat), nrow(A_mat))
B_mat <- H %*% A_mat %*% H
B_mat[1:3, 1:3] |> signif(3)
```

```
         [,1]    [,2]    [,3]
[1,]  0.00789  0.0939 -0.0199
[2,]  0.09390  1.1800 -0.6170
[3,] -0.01990 -0.6170  0.3470
```

- Calculate the eigenvalues and eigenvectors of $\mathbf{B}$:

```
eig_obj <- eigen(B_mat)
```

- Calculate the principal coordinates:

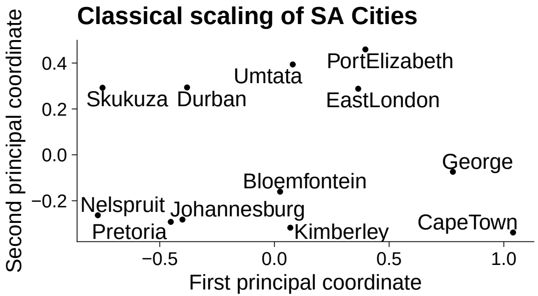
```
Y_mat <- eig_obj$vectors %*%
  diag(sqrt(eig_obj$values))
Y_mat[1:3, 1:2] |> signif(2)
```

```
       [,1]  [,2]
[1,]  0.025 -0.16
[2,]  1.000 -0.34
[3,] -0.380  0.29
```

Another example of classical scaling from first principles III

- Plot the first two principal coordinates:

```r
plot_tbl <- Y_mat |>
 tibble::as_tibble() |>
  dplyr::mutate(
    city = data_tidy_sa_distance |>
      colnames()
  )
p <- ggplot(plot_tbl, aes(V1, V2)) +
  geom_point(size = 3) +
  ggrepel::geom_text_repel(
    aes(label = city), size = 10) +
  coord_equal()
```



Least-squares scaling

For a matrix of dissimilarities $\mathbf{\Delta} = (\delta_{ij})$, a matrix of weights $\mathbf{W} = (w_{ij})$ and a monotonic function f, the least-squares scaling algorithm minimises the objective function

$$\mathcal{L}_f(\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_n; \mathbf{W}; f) = \sum_{i<j}^{n} w_{ij}(d_{ij} - f(\delta_{ij}))^2.$$

with respect to the n t-dimensional points $\mathbf{y}_1, \ldots, \mathbf{y}_n$, where $d_{ij} = \|\mathbf{y}_i - \mathbf{y}_j\|$ for $\|\cdot\|$ the Euclidean norm.

- The initial version of the objective function was proposed by Kruskal (1964) and lacked weights, which was extended to include weights by (de Leeuw 2005) (originally published in 1977). Full history in Patrick J. F. Groenen and Borg (n.d.).
- The square root of $\mathcal{L}_f$ is typically referred to as the stress function. If f does not merely preserve ranks, then it is the metric stress function.

Classical scaling versus least-squares scaling

- *Values approximated*:
  - In classical scaling, we approximate centred inner products.
  - In least-squares scaling, we approximate dissimilarities directly.
  - Even when the dissimilarities are Euclidean distances, the two methods will not give the same answer (Hastie, Tibshirani, and Friedman 2009).
    - For one thing, note that the centred inner products (as used by classical scaling) are a function of the squared dissimilarities.
    - For another, the mapping in least-squares scaling need not be linear.

Classical scaling versus least-squares scaling

- *Values approximated*:
  - In classical scaling, we approximate centred inner products.
  - In least-squares scaling, we approximate dissimilarities directly.
  - Even when the dissimilarities are Euclidean distances, the two methods will not give the same answer (Hastie, Tibshirani, and Friedman 2009).
    - For one thing, note that the centred inner products (as used by classical scaling) are a function of the squared dissimilarities.
    - For another, the mapping in least-squares scaling need not be linear.

- *Flexibility*:
  - Least-squares scaling is more flexible, in that:
    - It can handle non-Euclidean distance dissimilarities (without "breaking" an assumption).
    - It can handle transformed dissimilarities (including merely using ranks).
    - We can weight errors in dissimilarity approximations differently (e.g. downeight errors of large original dissimilarities).

Classical scaling versus least-squares scaling

- *Values approximated*:
  - In classical scaling, we approximate centred inner products.
  - In least-squares scaling, we approximate dissimilarities directly.
  - Even when the dissimilarities are Euclidean distances, the two methods will not give the same answer (Hastie, Tibshirani, and Friedman 2009).
    - For one thing, note that the centred inner products (as used by classical scaling) are a function of the squared dissimilarities.
    - For another, the mapping in least-squares scaling need not be linear.
- *Flexibility*:
  - Least-squares scaling is more flexible, in that:
    - It can handle non-Euclidean distance dissimilarities (without "breaking" an assumption).
    - It can handle transformed dissimilarities (including merely using ranks).
    - We can weight errors in dissimilarity approximations differently (e.g. downeight errors of large original dissimilarities).
- *Optimisation approach*:
  - Least-squares scaling an algebratic solution and so requires numerication optimisation.

Transformations of original dissimilarites

- In least-squares scaling, we may approximate not merely the dissimilarities themselves, but monotonic transformations of them, i.e. approximate $f(\delta_{ij})$ rather than δ_{ij}.
- Choices for f:
  - Linear transformation: $f(\delta_{ij}) = \alpha + \beta\delta_{ij}$.
    - For $\alpha = 0$, we merely rescale the dissimilarities and preserve the ratio scale.
    - For $\alpha > 0$, we effectively push apart the dissimilarities. This may help avoid "squashing" nearby points together when another point is very far away. However, we lose the ratio scale and then merely have the interval scale.
  - Others to pull in large values, e.g. log, square root, etc.
  - Monotonic rank-preserving transformations. Here, we lose even the interval scale, but are less sensitive to outliers.
    - When we use this transformation, we are effectively performing non-metric MDS. It is non-metric in the sense that you are not preserving the actual dissimilarities, but only the ranks of the dissimilarities.

Choices of weights

- Two common choices are:
  - Constant weights: $w_{ij} = 1/\sum_{i<j} \delta_{ij}^2$.
    - May help avoid computational precision errors.
    - If estimating f during the optimisation procedure (e.g. $\hat{f} = \hat{\alpha} + \hat{\beta}\delta_{ij}$), then one should rather substitute d_{ij} or $\hat{f}(\delta_{ij})$ for δ_{ij} as otherwise each configuration point $\mathbf{y}_i$ may be the same point.
  - Down-weight larger original distances: $w_{ij} = 1/(\delta_{ij} \sum_{i<j} \delta_{ij})$.
    - This may be useful when we have a few very large dissimilarities that we wish to down-weight.
    - This produces a Sammon mapping (Sammon 1969).
- Other variatons are possible as well.

Metric least-squares optimisation algorithm

A configuration of points minimising the metric stress function may be obtained as follows:

1. Assign points to initial coordinates (may be arbitrary, or that produced by classical scaling).
2. Repeat the following until convergence:

- Compute d_{ij} for all i and j (i.e. the Euclidean distance between all pairs of points).
- Move points in the direction that minimises stress across several random restarts.

Example of metric MDS: kinship grouping

- First we load our data, installing the required packages if not available:

```
if (!requireNamespace(
  "smacof", quietly = TRUE
  )) {
  install.packages("smacof")
}
data(
  "kinshipscales", package = "smacof"
  )
data(
  "kinshipdelta", package = "smacof"
  )
```

- The kinshipscales data show three aspects contributing to genetic distance of family members from an individual:
  - **Gender**: 1 = male, 2 = female.
  - **Generation**: -2 = two back, -1 = one back, 0 = same generation, 1 = one ahead, 2 = two ahead.
  - **Degree**: 1 = first, 2 = second, 3 = third, 4 = fourth.

```
kinshipscales[1:4, ]
```

```
         Gender Generation Degree
Aunt          2         -1      3
Brother       1          0      2
Cousin       NA          0      4
Daughter      2          1      1
```

Example of metric MDS: kinship grouping

The `kinshipdelta` data shows percentages of how often each of fifteen kinship terms were not grouped together by college students:

```
kinshipdelta[1:6, 1:6]
```

```
              Aunt Brother Cousin Daughter Father Granddaughter
Aunt             0      79     53       59     73            57
Brother         79       0     67       62     38            75
Cousin          53      67      0       74     77            74
Daughter        59      62     74        0     57            46
Father          73      38     77       57      0            79
Granddaughter   57      75     74       46     79             0
```

Metric MDS from first principles I

- First, we write the stress function:

```r
calc_stress <- function(y, delta_mat, dim = 2) {
  Y_mat <- matrix(y, ncol = dim)
  D_mat <- as.matrix(dist(Y_mat))
  error_mat <- ((D_mat - delta_mat)^2)[lower.tri(D_mat)]
  weight <- 1 / sum(D_mat[lower.tri(D_mat)]^2)
  sum(error_mat) * weight
}
```

- We generate an initial configuration using classical scaling:

```r
y_vec <- cmdscale(kinshipdelta) |> as.vector()
```

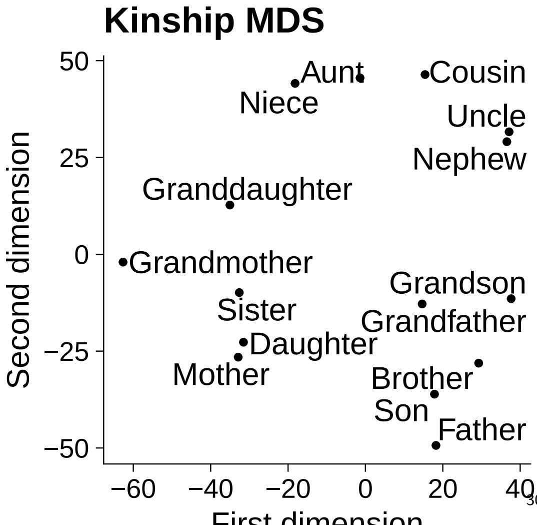
- We optimise:

```r
optim_obj <- optim(y_vec, calc_stress, delta_mat = kinshipdelta)
Y_mat <- matrix(optim_obj$par, ncol = 2)
```

Metric MDS from first principles II

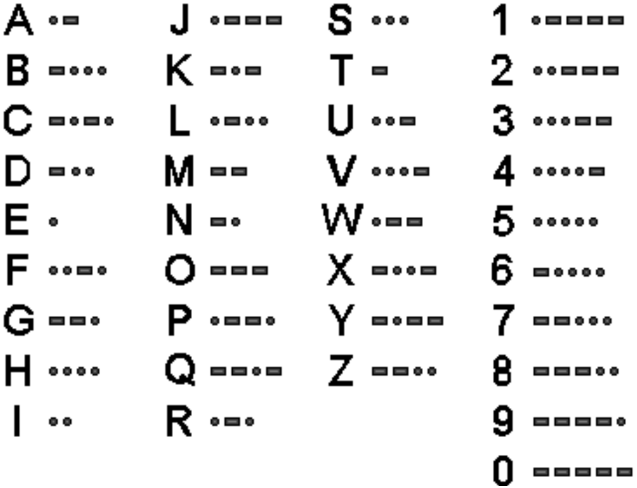
- Plot the results:

```
plot_tbl <- Y_mat |>
  tibble::as_tibble() |>
  dplyr::mutate(
    familial_term = kinshipdelta |>
      colnames()
  )
p <- ggplot(plot_tbl, aes(V1, V2)) +
  geom_point(size = 3) +
  ggrepel::geom_text_repel(
    aes(label = familial_term),
    size = 10
  ) +
  coord_equal()
```



Morse code example I

- Morse code consists of 36 sequences of "dots" and "dashes", representing the 26 letters of the alphabet and the digits 0 – 9:



A	·−	J	·−−−	S	···	1	·−−−−
B	−···	K	−·−	T	−	2	··−−−
C	−·−·	L	·−··	U	··−	3	···−−
D	−··	M	−−	V	···−	4	····−
E	·	N	−·	W	·−−	5	·····
F	··−·	O	−−−	X	−··−	6	−····
G	−−·	P	·−−·	Y	−·−−	7	−−···
H	····	Q	−−·−	Z	−−··	8	−−−··
I	··	R	·−·			9	−−−−·
						0	−−−−−

Morse code example II

- Participants with no knowledge of Morse code were asked to state whether two subsequent Morse code signals were the same or different.
    - Each pair of signals was presented in both orders possible.
    - When they were rated as different, the distance increased.
    - For each ordering of each pair, the average number of “different” calls across all participants was calculated.
    - Thus, 1260 dissimilarities were generated, where $\delta_{ij} \neq \delta_{ji}$ and $\delta_{ii} \neq 0$.

Morse code example III

Here is the original dissimilarity matrix:

```
data("data_tidy_morse", package = "DataTidy24STA5069Z")
data_tidy_morse[1:8, 1:8]
```

```
# A tibble: 8 x 8
      A     B     C     D     E     F     G     H
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  0.92  0.04  0.06  0.13  0.03  0.14  0.1   0.13
2  0.05  0.84  0.37  0.31  0.05  0.28  0.17  0.21
3  0.04  0.38  0.87  0.17  0.04  0.29  0.13  0.07
4  0.08  0.62  0.17  0.88  0.07  0.23  0.4   0.36
5  0.06  0.13  0.14  0.06  0.97  0.02  0.04  0.04
6  0.04  0.51  0.33  0.19  0.02  0.9   0.1   0.29
7  0.09  0.18  0.27  0.38  0.01  0.14  0.9   0.06
8  0.03  0.45  0.23  0.25  0.09  0.32  0.08  0.87
```

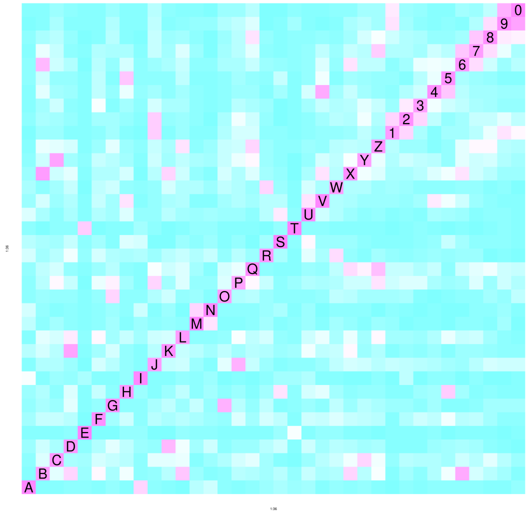
Morse code example IV

Here is code to plot of the original dissimilarity matrix:

```r
morse_mat <- as.matrix(data_tidy_morse)
image(
  1:36, 1:36, morse_mat,
  main = "Morsecodes raw confusion rates", col = cm.colors(36, 1)
)
cn_vec <- colnames(data_tidy_morse)
cn_vec <- substr(cn_vec, nchar(cn_vec), nchar(cn_vec))
text(1:36, 1:36, cn_vec)
```

Morse code example V

Here is the actual plot of the original dissimilarity matrix:



Generating symmetric dissimilarities

- To create symmetric, “hollow” proximities, the following transformation was used:

$$\tilde{\delta}_{ij} = \delta_{ii} + \delta_{jj} - \delta_{ji} - \delta_{ij}$$

- Here is the code to do so:

```
row_mat <- matrix(
  rep(diag(morse_mat), each = ncol(morse_mat)), byrow = TRUE, nrow = nrow(r
  )
morse_mat_tilde <-   row_mat + t(row_mat) - morse_mat - t(morse_mat)
morse_mat_tilde[1:3, 1:3]
```

```
        A    B    C
[1,] 0.00 1.67 1.69
[2,] 1.67 0.00 0.96
[3,] 1.69 0.96 0.00
```

Applying smacof using the mds function

- We can apply the mds function from the smacof package to the Morse code dissimilarities:

```
mds_obj <- smacof::mds(
  delta = morse_mat_tilde, # dissimilarities
  ndim = 2, # desired dimension of configuration
  type = "ratio", # type of dissimilarity
  init = "torgerson" # initialise with classical scaling
)
```

Extracting results from the mds fit

- Extract the coordinates:

```
mds_obj[["conf"]][1:5, ]
```

```
            D1          D2
1  0.749122600  0.43153708
2  0.032926713 -0.43546664
3 -0.187375174 -0.15147460
4  0.443400707 -0.01790054
5 -0.002802397  0.97857494
```

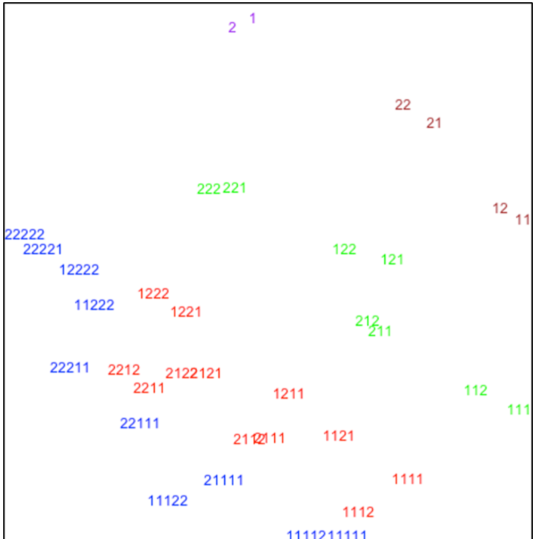
- Per-point contribution to total stress:

```
mds_obj[["spp"]][1:5]
```

```
       1        2        3        4        5
3.313617 1.686036 2.377347 2.620053 4.700401
```

Plot of Morse code MDS

- With 1s and 2s denoting dots and dashes respectively, and colour signal length:



Randomness in results

- MDS will be sensitive to the initial configuration.
  - Even if the plots are similar in terms of inter-point proximity, their orientation may be different - especially if using a random initialisation.
- At a minimum, set the seed. One can also choose the configuration that minimises stress.

Sammon mapping

- In Sammon mapping, we downweight larger original distances ($w_{ij} = 1/(\delta_{ij} \sum_{i<j} \delta_{ij})$) and use the identity function.
- One can use the `MASS::sammon` function to perform this:

```
MASS::sammon(
  d, y = cmdscale(d, k), k = 2, niter = 1e2,
  trace = TRUE, magic = 0.2, tol = 1e-4)
```

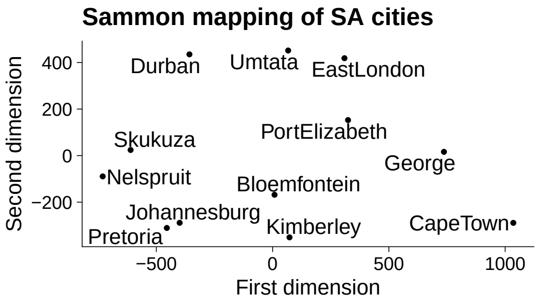
- For example, we can apply it to the SA city distances:

```
set.seed(12394)
data("data_tidy_sa_distance", package = "DataTidy24STA5069Z")
sammon_obj <- MASS::sammon(data_tidy_sa_distance |> as.matrix())
```

Sammon mapping of SA cities

- We plot the results:

```r
plot_tbl <- sammon_obj$points |>
  tibble::as_tibble() |>
  dplyr::mutate(
    city = data_tidy_sa_distance |>
      colnames()
  )
p <- ggplot(plot_tbl, aes(V1, V2)) +
  geom_point(size = 3) +
  ggrepel::geom_text_repel(
    aes(label = city), size = 10
  ) +
  coord_equal()
```



Non-metric MDS

- In non-metric MDS, we do not preserve the actual dissimilarities, but only the ranks of the dissimilarities.

- Dissimilarities can be strictly ordered from smallest to largest

$$\delta_{i_1,j_1} < \delta_{i_2,j_2} < \ldots < \delta_{i_m,j_m}$$

Where $(i_1, j_1), (i_m, j_m)$ indicates the pair of entities having the smallest and largest dissimilarities respectively.

- Nonmetric scaling finds a lower-dimensional space such that the distances

$$d_{i_1,j_1} < d_{i_2,j_2} < \ldots < d_{i_m,j_m}$$

matches exactly the ordering of the dissimilarities.

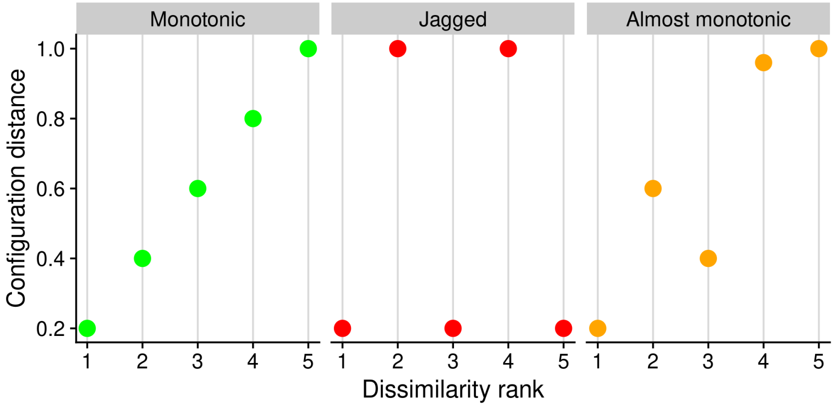
- Since a plot of the configuration distances d_{ij} against their rank order does not necessarily produce a monotonically looking scatterplot, thereby violating the monotonic condition, we approximate the d_{ij} by $\hat{d}_{ij}$ such that

Non-metric MDS

- Non-metric MDS aims not to approximate the actual dissimilarities, but merely preserve the ranks of the dissimilarities.
- The configuration is not optimised so that the configuration distances (d_{ij}) approximate the actual dissimilarities (δ_{ij}) as closely as possible, but rather so that the configuration distances increase montonically with the actual dissimilarities.
- To do this, we essentially do the following:
  - Generate an initial configuration.
  - Until convergence:
    - Fit a monotonically-increasing function of configuration distances against the ranks of the dissimilarities.
    - Adjust the configuration distances to more closely match the fitted values.

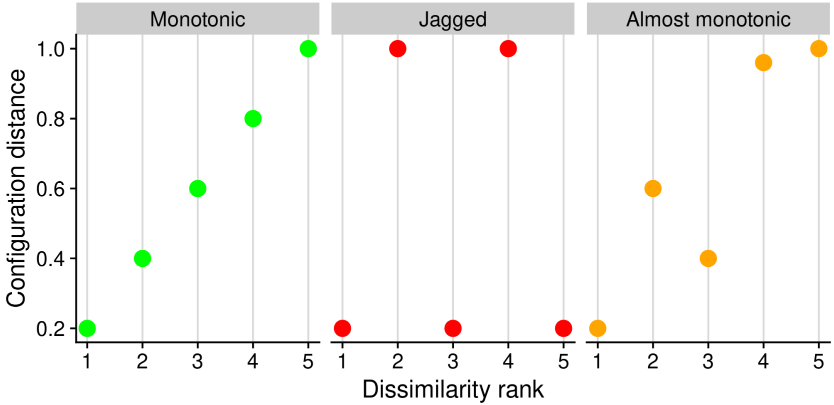
Shepard's diagram I

- We plot the configuration distances for each pair of points against the rank of their dissimilaritie:



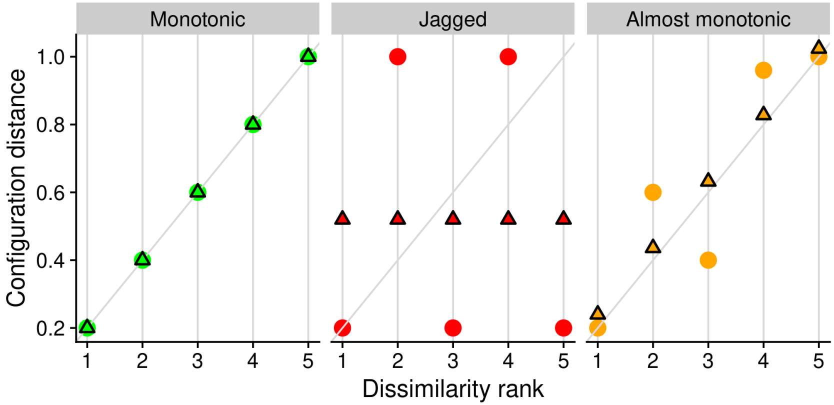
Shepard's diagram II

- We fit a monotonically increasing function to the configuration distances against the ranks of the dissimilarities (in this case, simply linear):



Shepard's diagram III

- For each rank k, the fitted value is denoted $\hat{d}_{i_k j_k}$ and termed the *disparity*:



Computing goodness of fit

- For a given configuration, the goodness of fit is given by the stress function:

$$S = \left[\sum_{i<j} w_{ij} (d_{ij} - \hat{d_{ij}})^2 \right]^{1/2}$$

- The stress function is a measure of the monotonicity (in the original ranks) of the configuration distances.

Methods for calculating disparities

- The basis of the stress function is the disparity.
- For a given configuration, there exist two common approaches for calculating a monotinically-increasing function of the configuration distances:
  - Isotonic regression
  - Monotonic splines
- Isotonic regression is simpler conceptually, but is not smooth.

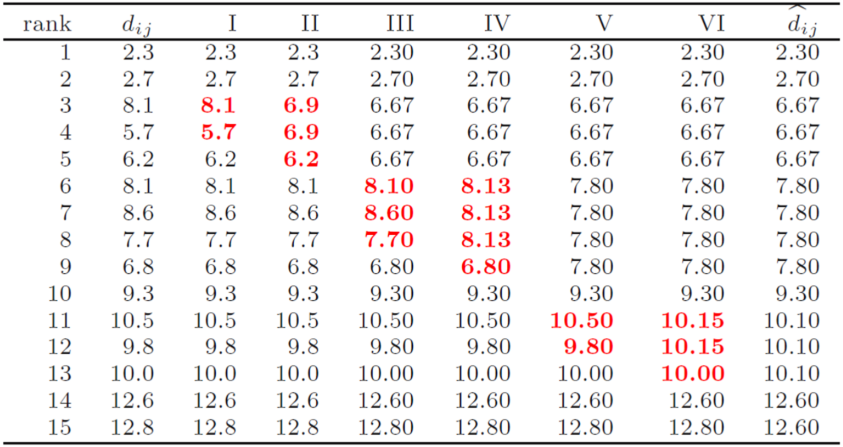
Isotonic regression algorithm

- Set $\hat{d}_{i_1 j_1} = d_{i_1 j_1}$.
    - Initially, set the lowest-rank disparity equal to the lowest-rank configuration distance.
- Sequentially from rank $k = 2$ to rank $k = m$:
    - Set $\hat{d}_{i_k j_k} = d_{i_k j_k}$
    - If $\hat{d}_{i_l j_l} \leq d_{i_k j_k}$ $\forall\ 1 \leq l < k$, then move to the next rank.
        - If the k-th configuration distance is not less than any lower-rank disparities, then the k-th disparity is equal to the k-th configuration distance.
    - If not, then:
        - Let v be the smallest index such that $\hat{d}_{i_v j_v} > \hat{d}_{i_k j_k}$.
        - Calculate the average disparity from the v-th to the k-th disparities, $\bar{d}_{i_k j_k} = \frac{1}{k-v+1} \sum_{l=v}^{k} \hat{d}_{i_l j_l}$.
        - Set $\hat{d}_{i_l j_l} = \bar{d}_{i_k j_k}$ for $l \in \{v, v+1, \dots, k\}$.
        - In other words, we force the disparities to not increase from v to k by setting them all to the average of the disparities from v to k.

Isotonic regression example

- Suppose that the ordered configuration distances are $\{2, 1, 1, 4, 2\}$.
- The first disparity is simply 2.
- The second disparity is 1, which is less than 2. So we set both the first and second disparities to $\frac{1+2}{2} = 1.5$.
- The third disparity is also 1, which is less than 1.5. So we set the first, second and third disparities to $\frac{1.5+1.5+1}{3} = 1.33$.
- The fourth disparity is 4, which is greater than 1.33. So we set the fourth disparity to 4.
- The fifth disparity is 2, which is less than 4. So we set the fourth and fifth disparities to $\frac{4+2}{2} = 3$.
- The final disparitiies are then $\{1.33, 1.33, 1.33, 3, 3\}$.

Another isotonic regression example (Izenmann, 2008)



rank	d_{ij}	I	II	III	IV	V	VI	$\widehat{d_{ij}}$
1	2.3	2.3	2.3	2.30	2.30	2.30	2.30	2.30
2	2.7	2.7	2.7	2.70	2.70	2.70	2.70	2.70
3	8.1	**8.1**	**6.9**	6.67	6.67	6.67	6.67	6.67
4	5.7	**5.7**	**6.9**	6.67	6.67	6.67	6.67	6.67
5	6.2	6.2	**6.2**	6.67	6.67	6.67	6.67	6.67
6	8.1	8.1	8.1	**8.10**	**8.13**	7.80	7.80	7.80
7	8.6	8.6	8.6	**8.60**	**8.13**	7.80	7.80	7.80
8	7.7	7.7	7.7	**7.70**	**8.13**	7.80	7.80	7.80
9	6.8	6.8	6.8	6.80	**6.80**	7.80	7.80	7.80
10	9.3	9.3	9.3	9.30	9.30	9.30	9.30	9.30
11	10.5	10.5	10.5	10.50	10.50	**10.50**	**10.15**	10.10
12	9.8	9.8	9.8	9.80	9.80	**9.80**	**10.15**	10.10
13	10.0	10.0	10.0	10.00	10.00	10.00	**10.00**	10.10
14	12.6	12.6	12.6	12.60	12.60	12.60	12.60	12.60
15	12.8	12.8	12.8	12.80	12.80	12.80	12.80	12.60

Splines

- A spline is a piece-wise polynomial function, which are made to be "smooth" at the points where the pieces join (termed "knots").
- For example, here is a linear spline with interior knots at 1, 3 and 5:

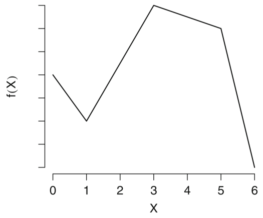


Figure 1: Harrell (2015)

- The function is made to be equal at the knots, i.e. the 0-th derivatives must be equal. However, the first derivatives are not.
- For a quadratic spline, the polynomials are of degree 2 $(ax^2 + bx + c)$ and the first and 0-th derivatives are equal at the knots. And so on.
- The cubic spline is the most common, and the polynomials are of degree 3. The natural cubic spline has the additional restriction that the function is linear beyond the first and last knots.

Monotonic splines

- Monotonic splines are splines that are made to be monotonically increasing.
- This is achieved by using a truncated power basis (see Harrell (2015) for details) and fitting non-linear weights (see uploaded spline section of of Izenman (2008) for details).
- By comparison with isotonic regression regression (left), monotonic splines may be far smoother (right) (Izenman 2008):

Variations on the stress function

- Raw stress: unweighted $\rightarrow S = \sum_{i<j}(d_{ij} - \hat{d_{ij}})^2$.
- Kruskal's stress formula (Stress-1):

  $w_{ij} = (\sum_{i<j} d_{ij}^2)^{-1} \rightarrow S = \left[\frac{\sum_{i<j}(d_{ij}-\hat{d}_{ij})^2}{\sum_{i<j}(d_{ij})^2}\right]^{1/2}$.
- Stress-2: $w_{ij} = (\sum_{i<j}(d_{ij} - \bar{d})^2)^{-1} \rightarrow S = \left[\frac{\sum_{i<j}(d_{ij}-\hat{d}_{ij})^2}{\sum_{i<j}(d_{ij}-\bar{d})^2}\right]^{1/2}$.
- Sammon's stress: $w_{ij} = 1/(\hat{d}_{ij}\sum_{i<j}\hat{d}_{ij}) \rightarrow S = \left[\frac{\sum_{i<j}(d_{ij}-\hat{d}_{ij})^2/\hat{d}_{ij}}{\sum_{i<j}\hat{d}_{ij}}\right]^{1/2}$.

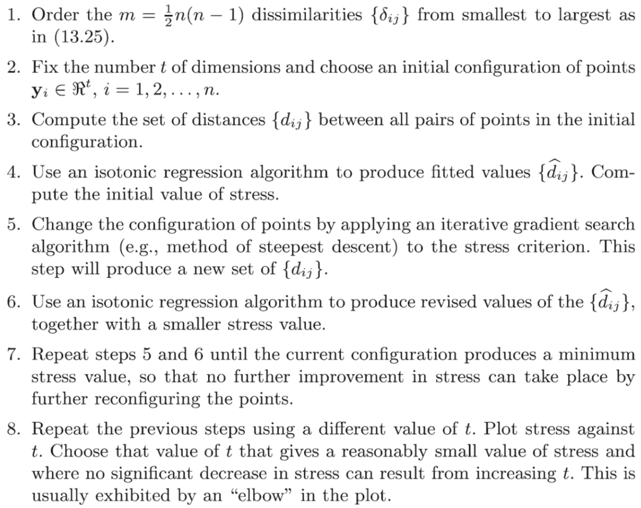
Minimising the stress function

- Kruskal (1964) proposed a gradient descent procedure, but other numerical optimsation functions, such as those in the `optim` function in `R`, may be used.
- A popular, alternative method is minimisation by majorisation, as first applied to MDS by de Leeuw (2005). It is apparently faster (Izenman 2008), and necessarily converges to the global minimum (Patrick J. F. Groenen and Van De Velden 2016).
  - This is termed the SMACOF algorithm, and is implemented in the `smacof` `R` package Mair, Groenen, and De Leeuw (2022).
  - Essentially:
    - A proxy function is defined that is equal to the stress function at a given point and (weakly) greater than the stress function at all other points.
    - Th proxy function is minimised, and the minimising point improves on the previous point in terms of the actual stress function.
    - The advantage is that the proxy function is chosen to be much simpler (linear/quadratic) than the stress function, and so easier to minimise.
  - See chapter 3 of Patrick J. F. Groenen and Van De Velden (2016) for details.

Example implementation of non-metric MDS

- See Francesca's slides from last year, under `MDS/Slides` in Vula Resources.

Algorithm for non-metric MDS



1. Order the $m = \frac{1}{2}n(n-1)$ dissimilarities $\{\delta_{ij}\}$ from smallest to largest as in (13.25).
2. Fix the number t of dimensions and choose an initial configuration of points $\mathbf{y}_i \in \Re^t$, $i = 1, 2, \ldots, n$.
3. Compute the set of distances $\{d_{ij}\}$ between all pairs of points in the initial configuration.
4. Use an isotonic regression algorithm to produce fitted values $\{\widehat{d}_{ij}\}$. Compute the initial value of stress.
5. Change the configuration of points by applying an iterative gradient search algorithm (e.g., method of steepest descent) to the stress criterion. This step will produce a new set of $\{d_{ij}\}$.
6. Use an isotonic regression algorithm to produce revised values of the $\{\widehat{d}_{ij}\}$, together with a smaller stress value.
7. Repeat steps 5 and 6 until the current configuration produces a minimum stress value, so that no further improvement in stress can take place by further reconfiguring the points.
8. Repeat the previous steps using a different value of t. Plot stress against t. Choose that value of t that gives a reasonably small value of stress and where no significant decrease in stress can result from increasing t. This is usually exhibited by an "elbow" in the plot.

Complete references I

Choi, Seung-Seok, Sung-Hyuk Cha, and Charles C Tappert. n.d. "A Survey of Binary Similarity and Distance Measures."

Groenen, Patrick J F, and Ingwer Borg. n.d. "The Past, Present, and Future of Multidimensional Scaling."

Groenen, Patrick J. F., and Michel Van De Velden. 2016. "Multidimensional Scaling by Majorization: A Review." *Journal of Statistical Software* 73 (8). https://doi.org/10.18637/jss.v073.i08.

Harrell, Frank E. 2015. *Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis*. Springer Series in Statistics. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-19425-7.

Hastie, Trevor., Robert. Tibshirani, and J. H. (Jerome H.) Friedman. 2009. *The Elements of Statistical Learning : Data Mining, Inference, and Prediction*. 2nd ed. Springer Series in Statistics. New York: Springer.

Complete references II

Izenman, Alan J. 2008. *Modern Multivariate Statistical Techniques*. Springer Texts in Statistics. New York, NY: Springer. https://doi.org/10.1007/978-0-387-78189-1.

JACCARD, P. 1901. "Etude Comparative de La Distribution Florale Dans Une Portion Des Alpes Et Des Jura." *Bull Soc Vaudoise Sci Nat* 37: 547–79. https://cir.nii.ac.jp/crid/1573387450552842240.

Kruskal, J. B. 1964. "Nonmetric Multidimensional Scaling: A Numerical Method." *Psychometrika* 29 (2): 115–29. https://doi.org/10.1007/BF02289694.

Leeuw, Jan de. 2005. "Applications of Convex Analysis to Multidimensional Scaling." https://escholarship.org/uc/item/7wg0k7xq.

Mair, Patrick. n.d. "Multidimensional Scaling in R: SMACOF."

Mair, Patrick, Patrick J. F. Groenen, and Jan De Leeuw. 2022. "More on Multidimensional Scaling and Unfolding in *R* : **Smacof** Version 2." *Journal of Statistical Software* 102 (10). https://doi.org/10.18637/jss.v102.i10.

Complete references III

Sammon, J. W. 1969. "A Nonlinear Mapping for Data Structure Analysis." *IEEE Transactions on Computers* C-18 (5): 401–9. https://doi.org/10.1109/T-C.1969.222678.

Stevens, S. S. 1946. "On the Theory of Scales of Measurement." *Science (New York, N.Y.)* 103 (2684): 677–80. https://doi.org/10.1126/science.103.2684.677.

Torgerson, Warren S. 1952. "Multidimensional Scaling: I. Theory and Method." *Psychometrika* 17 (4): 401–19.

———. 1958. "Theory and Methods of Scaling."