Multidimensional Scaling

Max Turgeon

STAT 4690–Applied Multivariate Analysis

Recap: PCA

- We discussed several interpretations of PCA.
  - **Pearson**: PCA gives the best linear approximation to the data (at a fixed dimension).
- We also used PCA to visualized multivariate data:
  - Fit PCA
  - Plot PC1 against PC2.

Multidimensional scaling

- **Multidimensional scaling** is a method that looks at these two goals explicitely.
  - It has PCA has a special case.
  - But it is much more general.
- The input of MDS is a **dissimilarity matrix** Δ, and it aims to represent the data in a lower-dimensional space such that the resulting dissimilarities $\tilde{\Delta}$ are as close as possible to the original dissimilarities.
  - $\Delta \approx \tilde{\Delta}$.

Example of dissimilarities

- Dissimilaries measure how *different* two observations are.
  - Larger disssimilarity, more different.
- Therefore, any distance measure can be used as a dissimilarity measure.
  - Euclidean distance in $\mathbb{R}^p$.
  - Mahalanobis distance.
  - Driving distance between cities.
  - Graph-based distance.
- Any *similarity* measure can be turned into a dissimilarity measure using a monotone decreasing transformation.
  - E.g. $r_{ij} \Longrightarrow 1 - r_{ij}^2$

Two types of MDS

- **Metric MDS**
  - The embedding in the lower dimensional space uses the same dissimilarity measure as in the original space.
- **Nonmetric MDS**
  - The embedding in the lower dimensional space only uses the rank information from the original space.

Metric MDS–Algorithm

- Input: An $n \times n$ matrix Δ of dissimilarities.
- Output: An $n \times r$ matrix $\tilde{X}$, with $r < p$.

Algorithm

1. Create the matrix D containing the square of the entries in Δ.
2. Create S by centering both the rows and the columns and multiplying by $-\frac{1}{2}$.
3. Compute the eigenvalue decomposition $S = U\Lambda U^T$.
4. Let $\tilde{X}$ be the matrix containing the first r columns of $\Lambda^{1/2}U^T$.

Example i

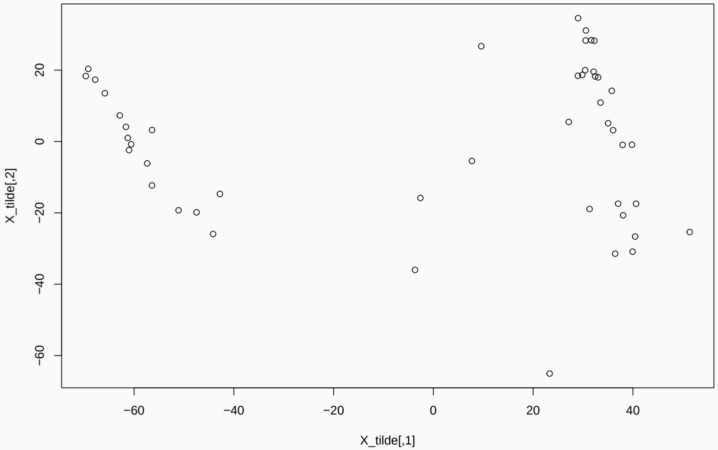
```r
Delta <- dist(swiss)
D <- Delta^2

# Center columns
B <- scale(D, center = TRUE, scale = FALSE)
# Center rows
B <- t(scale(t(B), center = TRUE, scale = FALSE))
B <- -0.5 * B
```

Example ii

```
decomp <- eigen(B)
Lambda <- diag(pmax(decomp$values, 0))
X_tilde <-  decomp$vectors %*% Lambda^0.5
```

```
plot(X_tilde)
```

Example iii



Example iv

```
mds <- cmdscale(Delta, k = 2)

all.equal(X_tilde[,1:2], mds,
          check.attributes = FALSE)
```

```
## [1] TRUE
```

Example v

```
library(tidyverse)
# Let's add annotations
dimnames(X_tilde) <- list(rownames(swiss),
                          paste0("MDS", seq_len(ncol(X
X_tilde <- as.data.frame(X_tilde) %>%
  rownames_to_column("District")
```

Example vi

```
X_tilde <- X_tilde %>%
  mutate(Canton = case_when(
    District %in% c("Courtelary", "Moutier",
                    "Neuveville") ~ "Bern",
    District %in% c("Broye", "Glane", "Gruyere",
                    "Sarine", "Veveyse") ~ "Fribourg",
    District %in% c("Conthey", "Entremont", "Herens",
                    "Martigwy", "Monthey",
                    "St Maurice", "Sierre",
                    "Sion") ~ "Valais"))
```

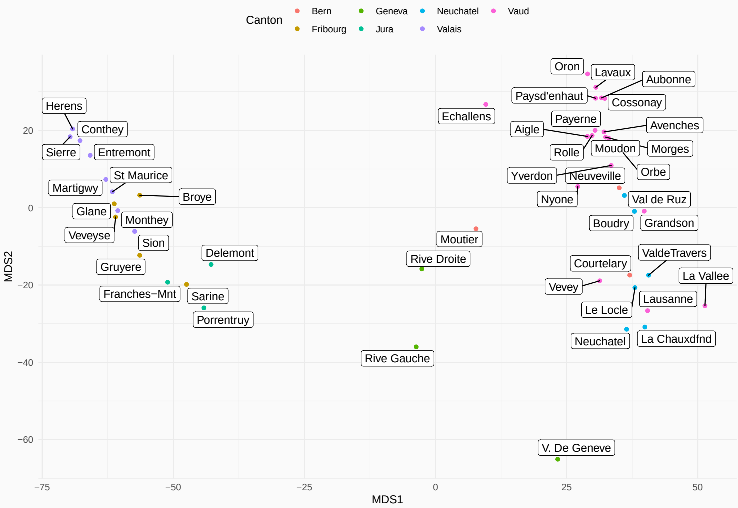
Example vii

```
X_tilde <- X_tilde %>%
  mutate(Canton = case_when(!is.na(Canton) ~ Canton,
    District %in% c("Boudry", "La Chauxdfnd",
                    "Le Locle", "Neuchatel",
                    "ValdeTravers",
                    "Val de Ruz") ~ "Neuchatel",
    District %in% c("V. De Geneve", "Rive Droite",
                    "Rive Gauche") ~ "Geneva",
    District %in% c("Delemont", "Franches-Mnt",
                    "Porrentruy") ~ "Jura",
    TRUE ~ "Vaud"))
```

Example viii

```r
library(ggrepel)
X_tilde %>%
  ggplot(aes(MDS1, MDS2)) +
  geom_point(aes(colour = Canton)) +
  geom_label_repel(aes(label = District)) +
  theme_minimal() +
  theme(legend.position = "top")
```

Example ix



Canton
Bern
Fribourg
Geneva
Jura
Neuchatel
Valais
Vaud
MDS2
MDS1
20
0
−20
−40
−60
−75
−50
−25
0
25
50
Herens
Conthey
Sierre
Entremont
St Maurice
Martigwy
Broye
Glane
Monthey
Veveyse
Sion
Delemont
Gruyere
Franches−Mnt
Sarine
Porrentruy
Echallens
Moutier
Rive Droite
Rive Gauche
Oron
Lavaux
Aubonne
Paysd'enhaut
Cossonay
Payerne
Avenches
Aigle
Rolle
Moudon
Morges
Orbe
Yverdon
Neuveville
Nyone
Val de Ruz
Boudry
Grandson
ValdeTravers
Courtelary
La Vallee
Vevey
Lausanne
Le Locle
Neuchatel
La Chauxdfnd
V. De Geneve

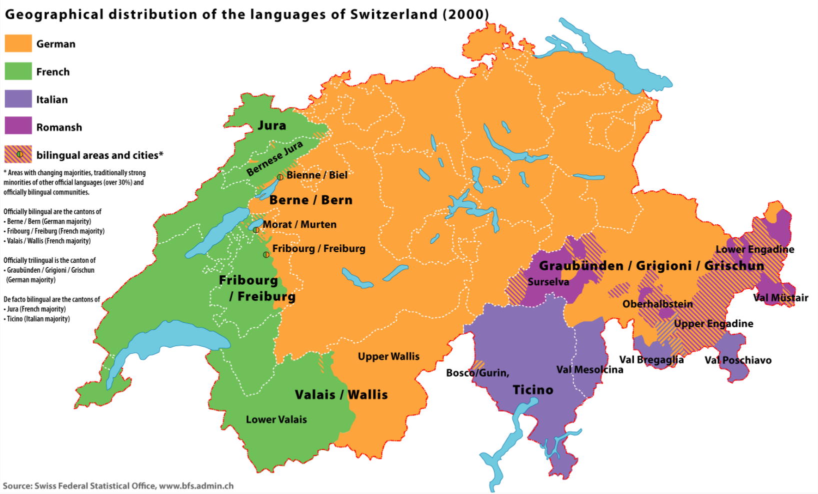

Geographical distribution of the languages of Switzerland (2000)
German
French
Italian
Romansh
bilingual areas and cities*
* Areas with changing majorities, traditionally strong minorities of other official languages (over 30%) and officially bilingual communities.
Officially bilingual are the cantons of
• Berne / Bern (German majority)
• Fribourg / Freiburg (French majority)
• Valais / Wallis (French majority)
Officially trilingual is the canton of
• Graubünden / Grigioni / Grischun (German majority)
De facto bilingual are the cantons of
• Jura (French majority)
• Ticino (Italian majority)
Jura
Bernese Jura
Bienne / Biel
Berne / Bern
Morat / Murten
Fribourg / Freiburg
Fribourg / Freiburg
Upper Wallis
Valais / Wallis
Lower Valais
Bosco/Gurin,
Ticino
Val Mesolcina
Val Bregaglia
Val Poschiavo
Graubünden / Grigioni / Grischun
Surselva
Oberhalbstein
Upper Engadine
Lower Engadine
Val Müstair
Source: Swiss Federal Statistical Office, www.bfs.admin.ch

Figure 1

Another example i

```r
library(psych)
cities[1:5, 1:5]
```

```
##      ATL  BOS ORD  DCA  DEN
## ATL    0  934 585  542 1209
## BOS  934    0 853  392 1769
## ORD  585  853   0  598  918
## DCA  542  392 598    0 1493
## DEN 1209 1769 918 1493    0
```

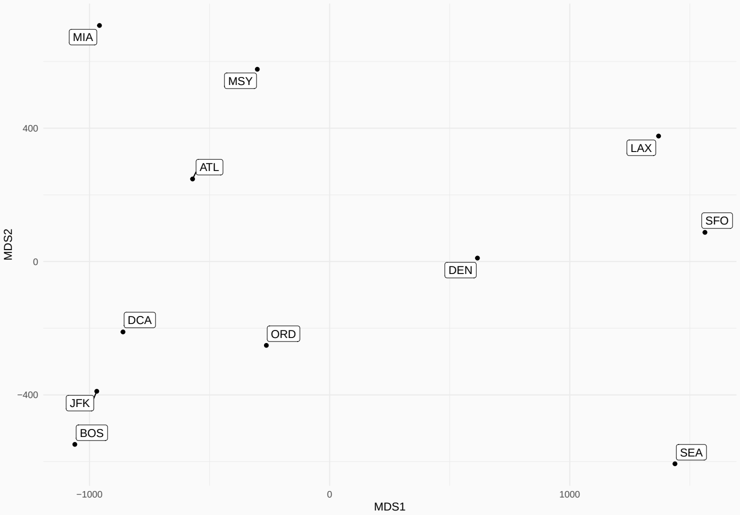
Another example ii

```r
mds <- cmdscale(cities, k = 2)
colnames(mds) <- c("MDS1", "MDS2")

mds <- mds %>%
  as.data.frame %>%
  rownames_to_column("Cities")
```

Another example iii

```r
mds %>%
  ggplot(aes(MDS1, MDS2)) +
  geom_point() +
  geom_label_repel(aes(label = Cities)) +
  theme_minimal()
```

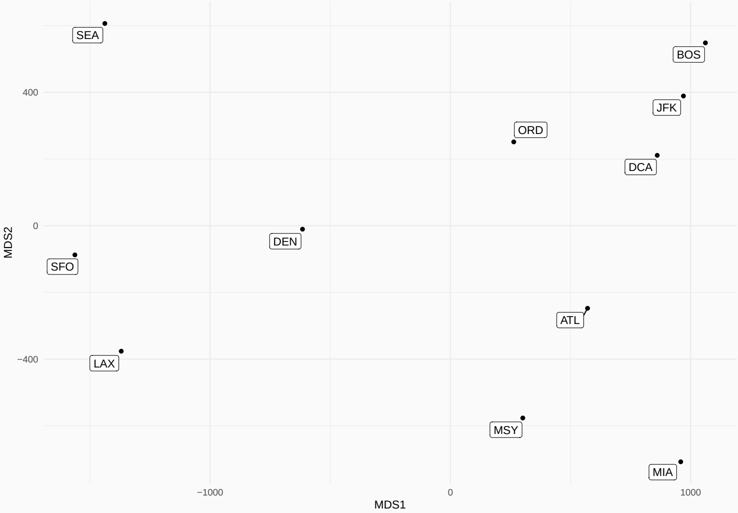
Another example iv



Another example v

```r
mds %>%
  mutate(MDS1 = -MDS1, MDS2 = -MDS2) %>%
  ggplot(aes(MDS1, MDS2)) +
  geom_point() +
  geom_label_repel(aes(label = Cities)) +
  theme_minimal()
```

Another example vi



Why does it work? i

- The algorithm may seem like black magic...
  - Double centering?
  - Eigenvectors of distances?
- Let's try to justify it.
- Let $\mathbf{Y_1}, \ldots, \mathbf{Y}_n$ be a set of points in $\mathbb{R}^p$.
- Recall that in $\mathbb{R}^p$, the Euclidean distance and the scalar product are related as follows:

Why does it work? ii

$$\begin{aligned} d(\mathbf{Y}_i, \mathbf{Y}_j)^2 &= \langle \mathbf{Y}_i - \mathbf{Y}_j, \mathbf{Y}_i - \mathbf{Y}_j \rangle \\ &= (\mathbf{Y}_i - \mathbf{Y}_j)^T (\mathbf{Y}_i - \mathbf{Y}_j) \\ &= \mathbf{Y}_i^T \mathbf{Y}_i - 2\mathbf{Y}_i^T \mathbf{Y}_j + \mathbf{Y}_j^T \mathbf{Y}_j. \end{aligned}$$

- In other words, the scalar product between $\mathbf{Y}_i$ and $\mathbf{Y}_j$ is given by

$$\mathbf{Y}_i^T \mathbf{Y}_j = -\frac{1}{2}\left(d(\mathbf{Y}_i, \mathbf{Y}_j)^2 - \mathbf{Y}_i^T \mathbf{Y}_i - \mathbf{Y}_j^T \mathbf{Y}_j \right).$$

Why does it work? iii

- Let S be the matrix whose (i, j)-th entry is $\mathbf{Y}_i^T \mathbf{Y}_j$, and note that D is the matrix whose (i, j)-th entry is $d(\mathbf{Y}_i, \mathbf{Y}_j)^2$.
- Now, assume that the dataset $\mathbf{Y_1}, \ldots, \mathbf{Y}_n$ has sample mean $\bar{\mathbf{Y}} = 0$ (i.e. it is centred). The average of the i-th row of D is

Why does it work? iv

$$\begin{aligned}
\frac{1}{n}\sum_{j=1}^{n} d(\mathbf{Y}_i, \mathbf{Y}_j)^2 &= \frac{1}{n}\sum_{j=1}^{n}\left(\mathbf{Y}_i^T\mathbf{Y}_i - 2\mathbf{Y}_i^T\mathbf{Y}_j + \mathbf{Y}_j^T\mathbf{Y}_j\right)\\
&= \mathbf{Y}_i^T\mathbf{Y}_i - \frac{2}{n}\sum_{j=1}^{n}\mathbf{Y}_i^T\mathbf{Y}_j + \frac{1}{n}\sum_{j=1}^{n}\mathbf{Y}_j^T\mathbf{Y}_j\\
&= \mathbf{Y}_i^T\mathbf{Y}_i - 2\mathbf{Y}_i^T\bar{\mathbf{Y}} + \frac{1}{n}\sum_{j=1}^{n}\mathbf{Y}_j^T\mathbf{Y}_j\\
&= S_{ii} + \frac{1}{n}\sum_{j=1}^{n} S_{jj}.
\end{aligned}$$

Why does it work? v

- Similarly, the average of the j-th column of D is given by

$$\frac{1}{n}\sum_{i=1}^{n} d(\mathbf{Y}_i, \mathbf{Y}_j)^2 = \frac{1}{n}\sum_{i=1}^{n} S_{ii} + S_{jj}.$$

- We can then deduce that the mean of **all** the entries of D is given by

$$\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} d(\mathbf{Y}_i, \mathbf{Y}_j)^2 = \frac{1}{n}\sum_{i=1}^{n} S_{ii} + \frac{1}{n}\sum_{j=1}^{n} S_{jj}.$$

Why does it work? vi

- Putting all of this together, we now have that

$$\begin{aligned}\mathbf{Y}_i^T\mathbf{Y}_i + \mathbf{Y}_j^T\mathbf{Y}_j &= \frac{1}{n}\sum_{j=1}^n d(\mathbf{Y}_i, \mathbf{Y}_j)^2 \\ &\quad + \frac{1}{n}\sum_{i=1}^n d(\mathbf{Y}_i, \mathbf{Y}_j)^2 \\ &\quad - \frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n d(\mathbf{Y}_i, \mathbf{Y}_j)^2.\end{aligned}$$

Why does it work? vii

- **In other words**, we can recover the scalar products from the square distances through double centering and scaling.
- Moreover, since we assumed the data was centred, the SVD of the matrix S is related to the SVD of the sample covariance matrix.
  - In this context, up to a constant, MDS and PCA give the same results.
- **Note**: This idea that double centering allows us to go from dissimilaries to scalar products will come back again in the next lecture on kernel methods.

Further comments

- In PCA, we performed an eigendecomposition of the sample covariance matrix.
    - This is a $p \times p$ matrix.
- In MDS, we performed an eigendecomposition of the doubly centred and scaled matrix of squared distances.
    - This is an $n \times n$ matrix.
- If our dissimilarities are computed using the Euclidean distance, both methods will give the same answer.
    - **BUT**: the smallest matrix will be faster to compute and faster to decompose.
    - $n > p \Rightarrow \mathrm{PCA}; n < p \Rightarrow \mathrm{MDS}$

Stress function i

- Nonmetric MDS approaches the problem a bit differently.
- We still have the same output Δ of dissimilarities, but we also have an objective function called the **stress function**.
- Recall that our goal is to represent the data in a lower-dimensional space such that the resulting dissimilarities $\tilde{\Delta}$ are as close as possible to the original dissimilarities.
  - $\Delta_{ij} \approx \tilde{\Delta}_{ij}$, for all i, j.

Stress function ii

- The stress function is defined as

$$\text{Stress}(\tilde{\Delta}; r) = \sqrt{\frac{\sum_{i,j=1}^{n} w_{ij}(\Delta_{ij} - \tilde{\Delta}_{ij})^2}{c}},$$

  where
  - w_{ij} are nonnegative weights;
  - c is a normalising constant.
- Note that the stress function depends on both the dimension r of the lower space and the distances $\tilde{\Delta}$.
- **Goal**: Find points in $\mathbb{R}^r$ such that their similarities minimise the stress function.

Sammon's Nonlinear Mapping

- The stress function is

$$\text{Stress}(\tilde{\Delta}; r) = \frac{1}{c} \sum_{i=1, i<j}^{n} \frac{(\Delta_{ij} - \tilde{\Delta}_{ij})^2}{\Delta_{ij}},$$

  where

$$c = \sum_{i=1, i<j}^{n} \Delta_{ij}.$$

- We don't make any assumption on the dissimilarities Δ, but we assume that $\tilde{\Delta}$ arises from the Euclidean distance in $\mathbb{R}^r$.
  - This makes the minimisation problem easier and amenable to Newton's method.

Example i

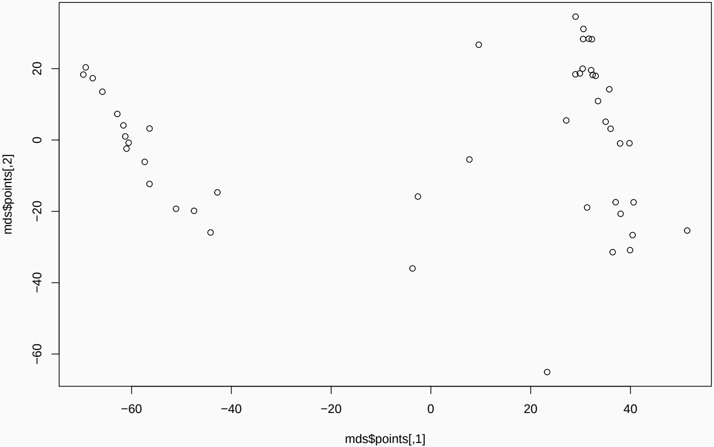
```r
library(MASS)

Delta <- dist(swiss)
mds <- sammon(Delta, k = 2)
```

```
## Initial stress        : 0.01959
## stress after   0 iters: 0.01959
```

Example ii

```
plot(mds$points)
```

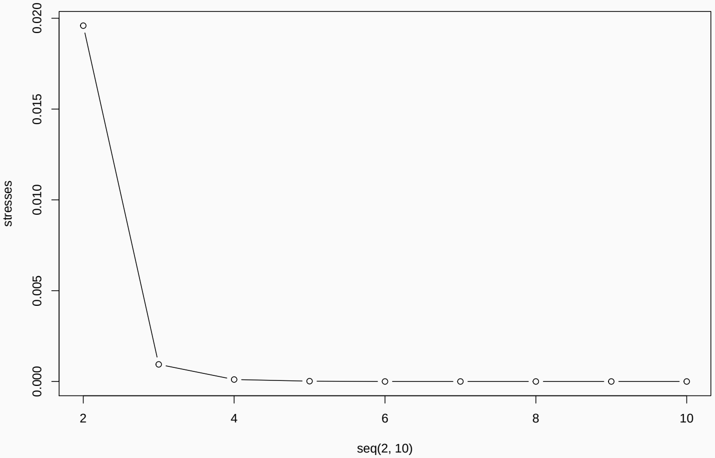
Example iii



Example iv

```
# Fit for different values of k
stresses <- sapply(seq(2, 10),
                   function(k) {
                     sammon(Delta, k = k,
                            trace = FALSE)$stress
                     })
plot(seq(2, 10), stresses, type = 'b')
```

Example   v



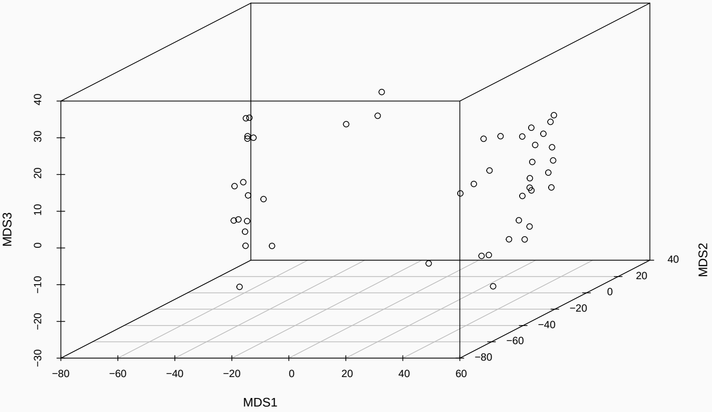
Example vi

```r
library(scatterplot3d)
mds <- sammon(Delta, k = 3)
```

```
## Initial stress        : 0.00243
## stress after  10 iters: 0.00095, magic = 0.500
## stress after  20 iters: 0.00094, magic = 0.500
```

```r
scatterplot3d(mds$points,
              xlab = "MDS1", ylab = "MDS2",
              zlab = "MDS3")
```

Example vii


MDS3
40
30
20
10
0
−10
−20
−30
−80
−60
−40
−20
0
20
40
60
MDS1
40
20
0
−20
−40
−60
−80
MDS2

Kruskal's Nonmetric MDS

- Kruskal's approach is based on **ranks**.
- In other words: instead of finding points in $\mathbf{R}^r$ with similar distances, his method tries to preserve the relative ordering of the dissimilarities.
    - The most dissimilar points in $\mathbb{R}^p$ should be represented by the most dissimilar points in $\mathbb{R}^r$, but the actual magnitude is irrelevant.
- This is achieved by allowing a monotone transformation f of the dissimilarities. We thus get

$$\text{Stress}(\tilde{\Delta}; r) = \sqrt{\frac{\sum_{i=1, i<j}^{n} (f(\Delta_{ij}) - \tilde{\Delta}_{ij})^2}{\sum_{i=1, i<j}^{n} \tilde{\Delta}_{ij}}}.$$

Example (cont'd) i

```
mds_s <- sammon(Delta, k = 2)
```

```
## Initial stress        : 0.01959
## stress after   0 iters: 0.01959
```

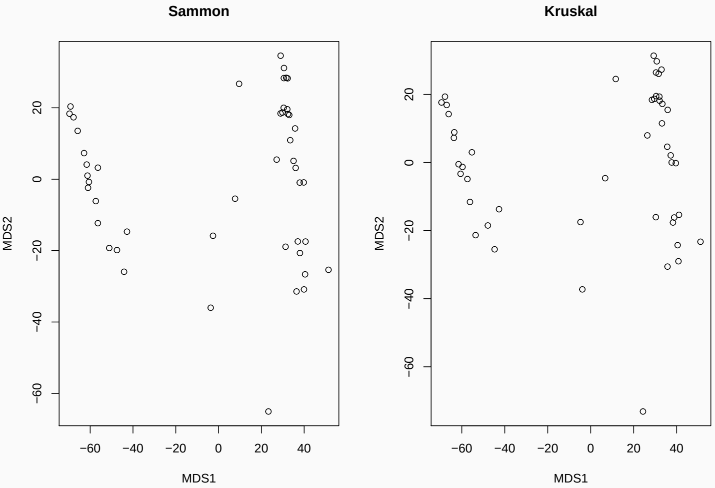
```
mds_k <- isoMDS(Delta, k = 2)
```

Example (cont'd) ii

```
## initial  value 5.463800
## iter   5 value 4.499103
## iter   5 value 4.495335
## iter   5 value 4.492669
## final  value 4.492669
## converged
```

```
par(mfrow = c(1, 2))
plot(mds_s$points, main = "Sammon",
     xlab = "MDS1", ylab = "MDS2")
plot(mds_k$points, main = "Kruskal",
     xlab = "MDS1", ylab = "MDS2")
```

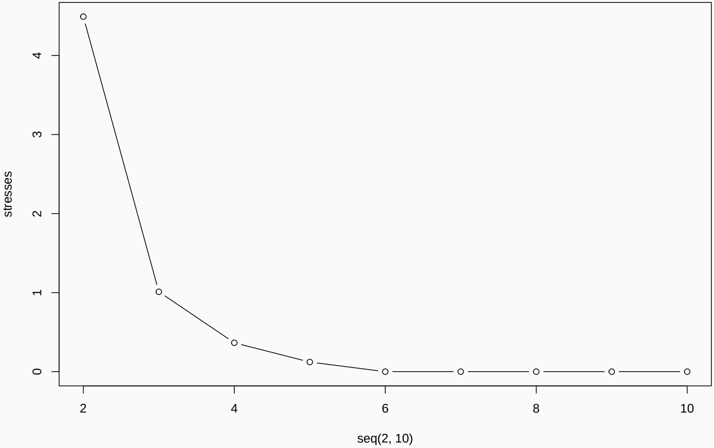
Example (cont'd) iii



Example (cont'd) iv

```r
# Sammon and Kruskal have different
# optimal k
stresses <- sapply(seq(2, 10),
                   function(k) {
                     isoMDS(Delta, k = k,
                            trace = FALSE)$stress
                   })
plot(seq(2, 10), stresses, type = 'b')
```

Example (cont'd) v



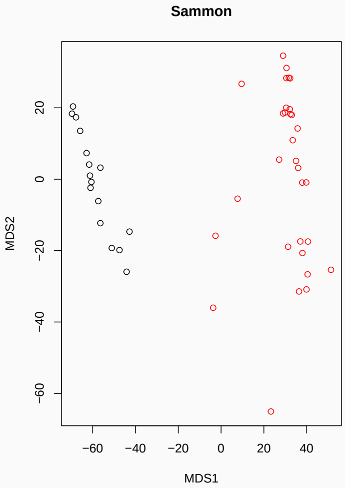
Example (cont'd) vi

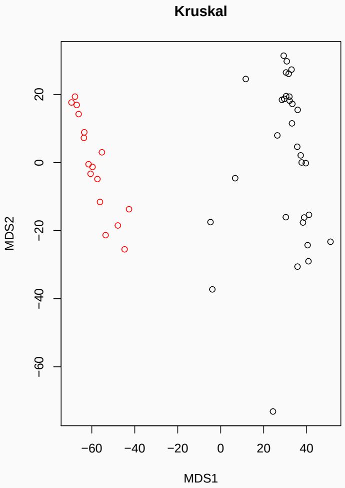
```r
mds_opt_s <- sammon(Delta, k = 3,
                    trace = FALSE)
mds_opt_k <- isoMDS(Delta, k = 6,
                    trace = FALSE)

# Let's cluster in the MDS space
cluster_s <- kmeans(mds_opt_s$points, centers = 2)
cluster_k <- kmeans(mds_opt_k$points, centers = 2)
```

Example (cont'd) vii

```
par(mfrow = c(1, 2))
plot(mds_s$points, main = "Sammon",
     xlab = "MDS1", ylab = "MDS2",
     col = cluster_s$cluster)
plot(mds_k$points, main = "Kruskal",
     xlab = "MDS1", ylab = "MDS2",
     col = cluster_k$cluster)
```

Example (cont'd) viii

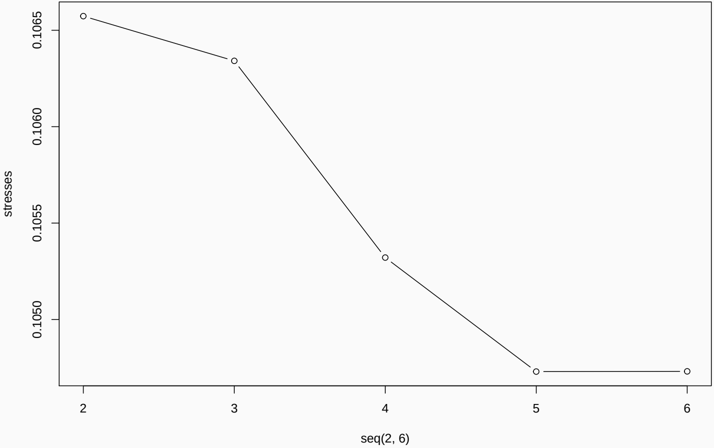




Example (cont'd) ix

```r
# More interestingly, you can use MDS to
# cluster data where you only have distances
stresses <- sapply(seq(2, 6),
                   function(k) {
                     isoMDS(as.matrix(cities),
                            k = k,
                            trace = FALSE)$stress
                   })
plot(seq(2, 6), stresses, type = 'b')
```

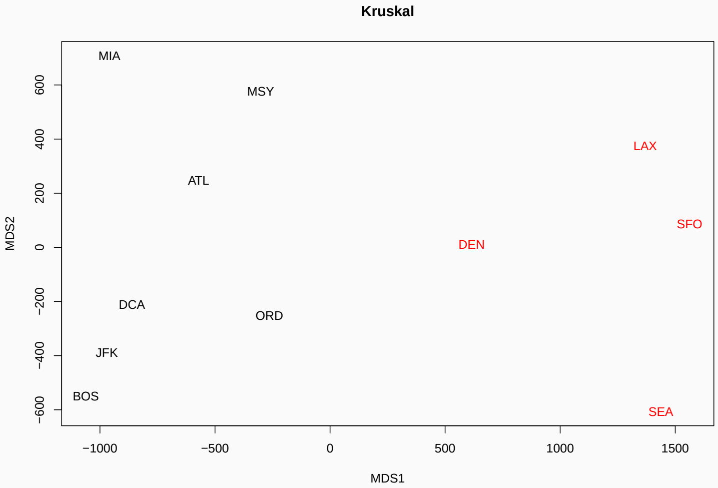
Example (cont'd) x



Example (cont'd) xi

```
mds_cities <- isoMDS(as.matrix(cities), k = 6,
                     trace = FALSE)
cluster_cities <- kmeans(mds_cities$points,
                         centers = 2)

plot(mds_cities$points, main = "Kruskal",
     xlab = "MDS1", ylab = "MDS2",
     type = 'n')
text(mds_cities$points, colnames(cities),
     col = cluster_cities$cluster)
```

Example (cont'd) xii



Summary

- Multidimensional scaling is mainly a method for visualising multivariate data.
- It works by finding points in a lower dimensional space with similar dissimilarities than the one on the original space.
- It only requires a matrix of dissimilarities
    - Therefore, it allows us to visualise data with limited information.
- MDS is an example of a **nonlinear dimension reduction** method.