Lecture 14

Multidimensional Scaling

Reading: Izenman Chapter 13
The main reference for these slides is from Dr. Markus Kalisch's Lecture Notes at https://stat.ethz.ch/education/semesters/ss2012/ams/slides/v4.1.pdf

DSA 8070 Multivariate Analysis

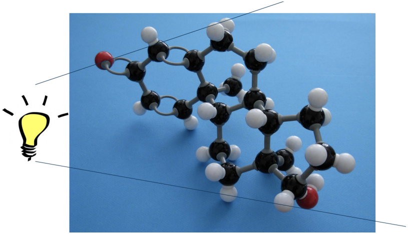
Whitney Huang
Clemson University

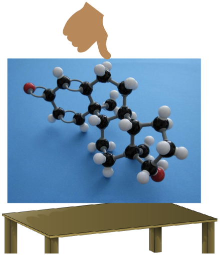
Agenda

1. **Main Idea**
2. **Classical Multidimensional Scaling**
3. **Non-metric Multidimensional Scaling**

Basic Idea of Multidimensional Scaling (MDS)

- Represent high-dimensional point cloud in low (usually 2) dimensional Euclidean space while *preserving as well as possible the inter-point distances*
- Classical/Metric MDS: Use a clever projection
- Non-metric MDS: Squeeze data on table





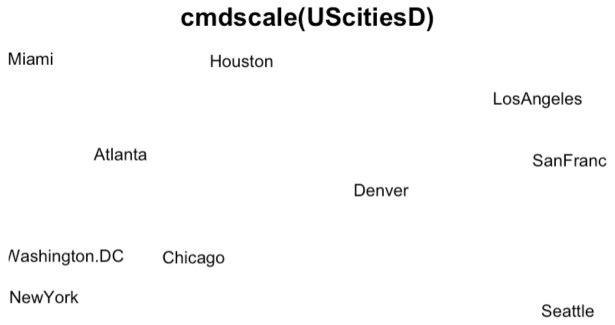
Source: Dr. Markus Kalisch's Lecture Notes on MDS

Classical MDS (cMDS)

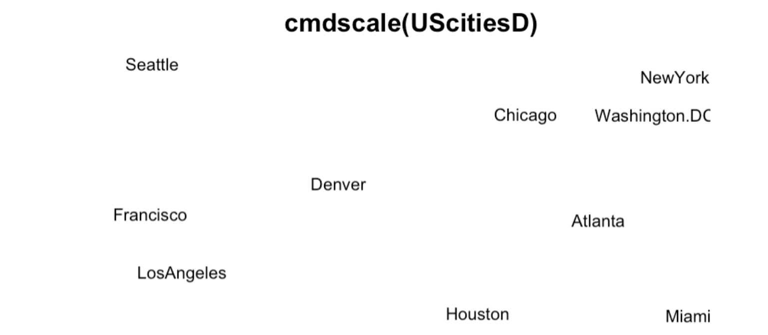
- **Goal**: Given pairwise distances among points, recover the position of the points!
- **Example**: Distance between 10 US major cities

```
> UScitiesD
              Atlanta Chicago Denver Houston LosAngeles Miami NewYork SanFrancisco Seattle
Chicago           587
Denver           1212     920
Houston           701     940    879
LosAngeles       1936    1745    831    1374
Miami             604    1188   1726     968       2339
NewYork           748     713   1631    1420       2451  1092
SanFrancisco     2139    1858    949    1645        347  2594    2571
Seattle          2182    1737   1021    1891        959  2734    2408          678
Washington.DC     543     597   1494    1220       2300   923     205         2442    2329
```

Classical MDS: First Try


```
loc <- cmdscale(UScitiesD)
x <- loc[, 1]; y <- loc[, 2]
plot(x, y, type = "n", xlab = "", ylab = "", asp = 1,
     axes = FALSE, main = "cmdscale(UScitiesD)")
text(x, y, rownames(loc), cex = 0.8)
```
```


```

Classical MDS: Flip Axes

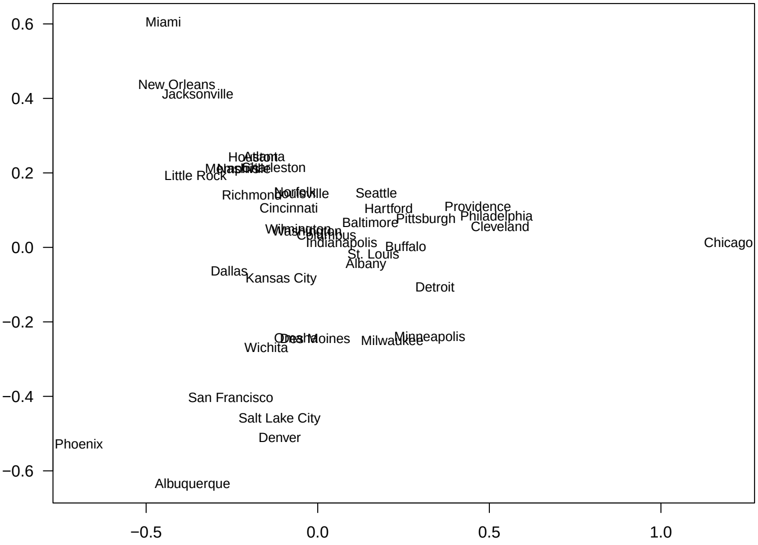
```
# Flip Axes
x1 <- -loc[, 1]; y1 <- -loc[, 2]
plot(x1, y1, type = "n", xlab = "", ylab = "", asp = 1,
     axes = FALSE, main = "cmdscale(UScitiesD)")
text(x1, y1, rownames(loc), cex = 0.8)
```
```


```

Another Example: Air Pollution in US Cities

```
> summary(dat)
      SO2              temp            manu            popul       
 Min.   :  8.00   Min.   :43.50   Min.   :  35.0   Min.   :  71.0  
 1st Qu.: 13.00   1st Qu.:50.60   1st Qu.: 181.0   1st Qu.: 299.0  
 Median : 26.00   Median :54.60   Median : 347.0   Median : 515.0  
 Mean   : 30.05   Mean   :55.76   Mean   : 463.1   Mean   : 608.6  
 3rd Qu.: 35.00   3rd Qu.:59.30   3rd Qu.: 462.0   3rd Qu.: 717.0  
 Max.   :110.00   Max.   :75.50   Max.   :3344.0   Max.   :3369.0  
      wind            precip         predays     
 Min.   : 6.000   Min.   : 7.05   Min.   : 36.0  
 1st Qu.: 8.700   1st Qu.:30.96   1st Qu.:103.0  
 Median : 9.300   Median :38.74   Median :115.0  
 Mean   : 9.444   Mean   :36.77   Mean   :113.9  
 3rd Qu.:10.600   3rd Qu.:43.11   3rd Qu.:128.0  
 Max.   :12.700   Max.   :59.80   Max.   :166.0  
```

- Range of `manu` and `popul` is much bigger than range of `wind`
- Need to standardize to give every variable equal weight

Air Pollution in US Cities Example



Classical MDS: Technical Details

- **Input**: $D = \{d_{ij}\}_{i,j=1}^{n}$, the Euclidean distances between n objects in p dimensions
- **Output**: $\{\boldsymbol{X}_i\}_{i=1}^{n}$, the "position" of points up to rotation, reflection, shift
- Two steps:
  - Compute inner products matrix $\boldsymbol{B} = \boldsymbol{X}\boldsymbol{X}^T$ from distance

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

  - Perform spectral decomposition to compute positions from $\boldsymbol{B}$ (see next slide)

Classical MDS: Technical Details

- Since $\boldsymbol{B} = \boldsymbol{X}\boldsymbol{X}^T$, we need the "square root" of $\boldsymbol{B}$
- Since $\boldsymbol{B}$ is a symmetric and positive definite $n \times n$ matrix $\Rightarrow \boldsymbol{B}$ can be diagonalized:

  $$\boldsymbol{B} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T$$

  Λ is a diagnoal matrix with $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ on diagonal

- Assuming the rank of $\boldsymbol{B} = p$, so that the last $n - p$ of its eigenvalues will be zero $\Rightarrow \boldsymbol{B}$ can be written as

  $$\boldsymbol{B} = \boldsymbol{V}_1 \Lambda_1 \boldsymbol{V}_1^T,$$

  where $\boldsymbol{V}_1$ contains the first p eigenvectors and Λ_1 the p non-zero eigenvalues. Take "square root": $\boldsymbol{X} = \boldsymbol{V}_1 \boldsymbol{\Lambda}_1^{-\frac{1}{2}}$

Classical MDS: Low-Dimensional Representation

- Keep only few (e.g. 2) largest eigenvalues and corresponding eigenvectors
- The resulting $\boldsymbol{X}$ will be the low-dimensional representation we were looking for
- "Goodness of fit" (GOF) if we reduce to m dimensions:

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

- Finds "optimal" low-dim representation: Minimizes

$$S = \sum_{i=1}^{n} \sum_{j=1}^{n} \left(d_{ij}^2 - (d_{ij}^m)^2 \right)^2$$

Classical MDS: Pros and Cons

- + Optimal for Euclidean input data
- + Still optimal, if B has non-negative eigenvalues
- + Very fast to compute
- - There is no guarantee it will be optimal if B has negative eigenvalues

Non-metric MDS: Idea

- Sometimes, there is no well-defined metric on original points
- Absolute values are not that meaningful, but the ranking is important
- Non-metric MDS finds a low-dimensional representation, which respects the ranking of distances

Non-metric MDS: Theory

- δ_{ij} is the true dissimilarity, d_{ij} is the distance of representation
- Minimize STRESS:

$$S = \frac{\sum_{i<j} (\theta(\delta_{ij}) - d_{ij})^2}{\sum_{i<j} d_{ij}^2},$$

where $\theta(\cdot)$ is an increasing function

- Optimize over both position of points and θ
- $\hat{d}_{ij} = \theta(\delta_{ij})$ is called "disparity"
- Solved numerically (isotonic regression); Classical MDS as starting value; very time consuming

Non-metric MDS: Pros andn Cons

- +: Fulfills a clear objective (minimize STRESS) without many assumptions
- +: Results don't change with rescaling or monotonic variable transformation
- +: Works even if you only have rank information
- -: computation can be slow in “large” problems
- -: Usually only local (not global) optimum found
- -: Only gets ranks of distances right

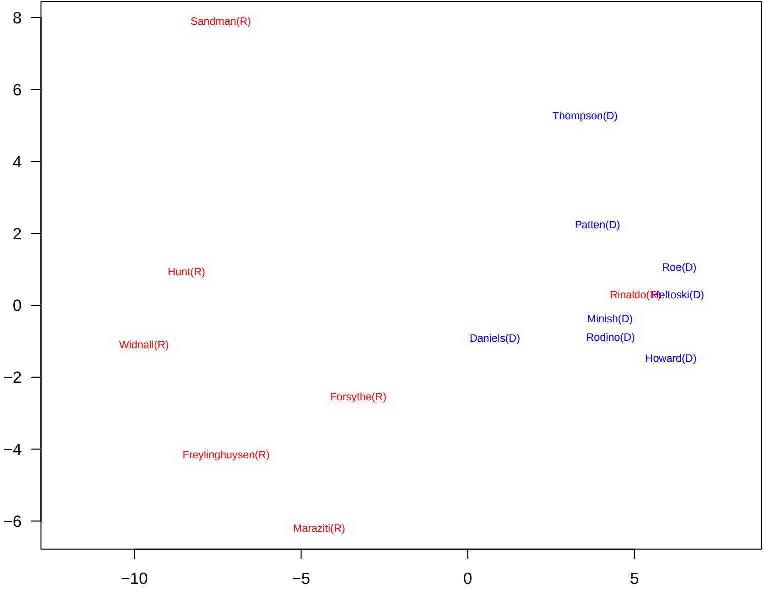
House of Representatives Voting Data

Romesburg (1984) gives a set of data that shows the number of times 15 congressmen from New Jersey voted differently in the House of Representatives on 19 environmental bills

```
> voting[1:6, 1:6]
                  Hunt(R) Sandman(R) Howard(D) Thompson(D) Freylinghuysen(R) Forsythe(R)
Hunt(R)                 0          8        15          15                10           9
Sandman(R)              8          0        17          12                13          13
Howard(D)              15         17         0           9                16          12
Thompson(D)            15         12         9           0                14          12
Freylinghuysen(R)      10         13        16          14                 0           8
Forsythe(R)             9         13        12          12                 8           0
```

Question: Do people in the same party vote alike?

Non-metric MDS: Voting Example



Summary

- Classical MDS:
  - Finds low-dim projection that respects distances
  - Optimal for euclidean distances
  - No clear guarantees for other distances
  - Fast to compute (can use `cmdscale` in `R`)
- Non-metric MDS:
  - Squeezes data points on table
  - Respects only rankings of distances
  - (Locally) solves clear objective
  - Computation can be slow (can use `isoMDS` from the `R` package "`MASS`")