HT2015: SC4
Statistical Data Mining and Machine Learning

Dino Sejdinovic
Department of Statistics
Oxford

http://www.stats.ox.ac.uk/~sejdinov/sdmml.html

Eigenvalue Decomposition (EVD)

Eigenvalue decomposition plays a significant role in PCA. PCs are eigenvectors of $S = \frac{1}{n-1}\mathbf{X}^\top\mathbf{X}$ and PCA properties are derived from those of eigenvectors and eigenvalues.

- For any $p \times p$ **symmetric** matrix S, there exists p eigenvectors $v_1, \ldots, v_p$ that are pairwise orthogonal and p associated eigenvalues $\lambda_1, \ldots, \lambda_p$ which satisfy the eigenvalue equation $Sv_i = \lambda_i v_i \; \forall i$.
- S can be written as $S = V\Lambda V^\top$ where
  - $V = [v_1, \ldots, v_p]$ is a $p \times p$ orthogonal matrix ($VV^\top = V^\top V = I_p$).
  - $\Lambda = diag\{\lambda_1, \ldots, \lambda_p\}$
  - If S is a **real-valued** matrix, then the eigenvalues are real-valued as well,
  $$\lambda_i \in \mathbb{R}, \; \forall i.$$
  - If S is **positive-semidefinite** matrix, then the eigenvalues are non-negative,
  $$\lambda_i \geq 0, \; \forall i.$$
- To compute the PCA of a dataset $\mathbf{X}$, we can:
  1. Estimate the covariance matrix using the sample covariance S.
  2. Compute the EVD of S using the R command `eigen`.

Singular Value Decomposition (SVD)

Unlike EVD, SVD **always** exists, even for non-square matrices.

- Real-valued $n \times p$ matrix $\mathbf{X}$ can be written as $X = UDV^\top$ where
  - U is an $n \times n$ orthogonal matrix: $UU^\top = U^\top U = I_n$
  - D is a $n \times p$ matrix with decreasing **non-negative** elements on the diagonal (the singular values) and zero off-diagonal elements.
  - V is a $p \times p$ orthogonal matrix: $VV^\top = V^\top V = I_p$
- SVD can be computed using very fast and numerically stable algorithms. The relevant R command is `svd`.

SVD and PCA

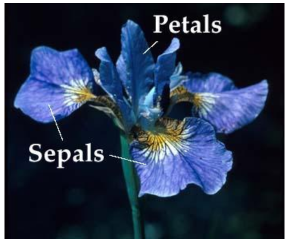
- Let $\mathbf{X} = UDV^\top$ be the SVD of the $n \times p$ data matrix $\mathbf{X}$.
- Note that
$$(n-1)S = \mathbf{X}^\top\mathbf{X} = (UDV^\top)^\top(UDV^\top) = VD^\top U^\top UDV^\top = VD^\top DV^\top,$$
using orthogonality ($U^\top U = I_n$) of U.
- The eigenvalues of S are thus the diagonal entries of $\frac{1}{n-1}D^2$ and the columns of V are the eigenvectors of S.
- We also have
$$\mathbf{X}\mathbf{X}^\top = (UDV^\top)(UDV^\top)^\top = UDV^\top VD^\top U^\top = UDD^\top U^\top,$$
using orthogonality ($V^\top V = I_p$) of V.
- SVD gives the optimal low-rank approximations of $\mathbf{X}$:
$$\min_{\tilde{\mathbf{X}}} \|\tilde{\mathbf{X}} - \mathbf{X}\|^2 \qquad \text{s.t. } \tilde{\mathbf{X}} \text{ has maximum rank } r < n, p.$$
Keep only the r largest singular values of $\mathbf{X}$, zeroing out the smaller singular values in the SVD.

Iris Data

50 samples from each of the 3 species of iris: `setosa`, `versicolor`, and `virginica`

Each measuring the length and widths of both sepal and petals

Collected by E. Anderson (1935) and analysed by R.A. Fisher (1936)

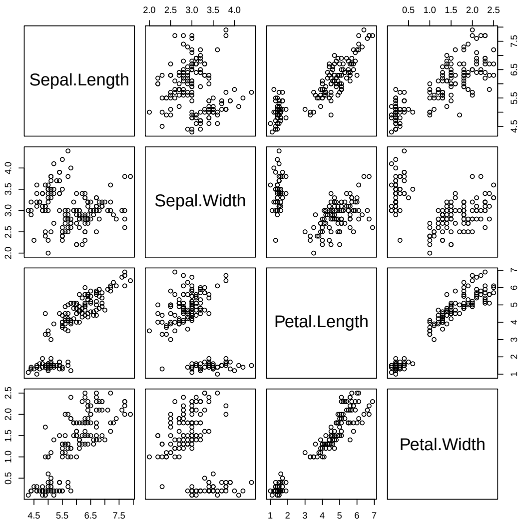


Iris Data

```
> data(iris)
> iris[sample(150,20),]
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
54           5.5         2.3          4.0         1.3 versicolor
33           5.2         4.1          1.5         0.1     setosa
30           4.7         3.2          1.6         0.2     setosa
73           6.3         2.5          4.9         1.5 versicolor
107          4.9         2.5          4.5         1.7  virginica
4            4.6         3.1          1.5         0.2     setosa
90           5.5         2.5          4.0         1.3 versicolor
83           5.8         2.7          3.9         1.2 versicolor
50           5.0         3.3          1.4         0.2     setosa
92           6.1         3.0          4.6         1.4 versicolor
128          6.1         3.0          4.9         1.8  virginica
57           6.3         3.3          4.7         1.6 versicolor
9            4.4         2.9          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
86           6.0         3.4          4.5         1.6 versicolor
66           6.7         3.1          4.4         1.4 versicolor
85           5.4         3.0          4.5         1.5 versicolor
147          6.3         2.5          5.0         1.9  virginica
8            5.0         3.4          1.5         0.2     setosa
41           5.0         3.5          1.3         0.3     setosa
```


Iris Data

```
> iris1 <- iris[,-5]
> pairs(iris1)
```



Biplots

- PCA plots show the data items (rows of $\mathbf{X}$) in the space spanned by PCs.
- **Biplots** allow us to visualize the **original variables** (columns of $\mathbf{X}$) in the same plot.
- As for PCA, we would like the geometry of the plot to preserve as much of the covariance structure as possible.

Biplots

Recall that $X = [X^{(1)}, \ldots, X^{(p)}]^\top$ and $\mathbf{X} = UDV^\top$ is the SVD of the data matrix.

- The 'full' PC projection of x_i is the i-th row of UD:

$$z_i = V^\top x_i = D^\top U_i^\top, \text{ equivalently: } \mathbf{X}V = UD.$$

- The j-th unit vector $\mathbf{e}_j \in \mathbb{R}^p$ points in the direction of the original variable $X^{(j)}$. Its PC projection η_j is:

$$\eta_j = V^\top \mathbf{e}_j = V_j^\top$$

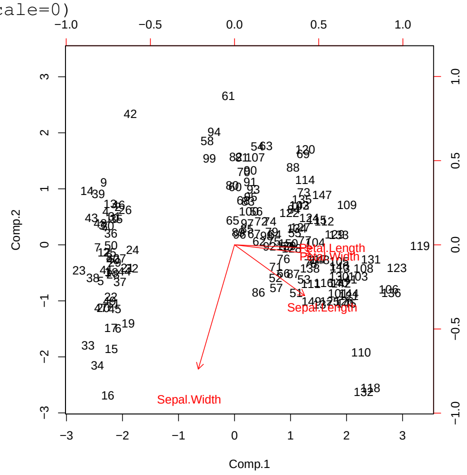
  (the j-th row of V)
- The projection of $\mathbf{e}_j$ indicates the weighting each PC gives to the original variables.
- Dot products between the projections gives entries of the data matrix:

$$x_{ij} = \sum_{k=1}^{p} U_{ik} D_{kk} V_{jk} = \langle D^\top U_i^\top, V_j^\top \rangle = \langle z_i, \eta_j \rangle.$$

- These relationships can be plotted in 2D by focussing on first two PCs.
- Quality depends on the **proportion of variance explained** by the first two PCs.

Iris data biplot

```
 > loadings(iris.pca)
             Comp.1 Comp.2 Comp.3 Comp.4
Sepal.Length  0.521 -0.377  0.720  0.261
Sepal.Width  -0.269 -0.923 -0.244 -0.124
Petal.Length  0.580        -0.142 -0.801
Petal.Width   0.565        -0.634  0.524
 > biplot(iris.pca,scale=0)
```



Biplots

- There are other projections we can consider for biplots:

$$x_{ij} = \sum_{k=1}^{p} U_{ik} D_{kk} V_{jk} = \langle D^\top U_i^\top, V_j^\top \rangle = \langle D^{1-\alpha}_{1:p,1:p} U^\top_{i,1:p}, D^{\alpha}_{1:p,1:p} V_j^\top \rangle.$$

  where $0 \le \alpha \le 1$, i.e., we change representation to

$$\tilde{z}_i = D^{1-\alpha}_{1:p,1:p} U^\top_{i,1:p}, \;\; \tilde{\eta}_j = D^{\alpha}_{1:p,1:p} V_j^\top$$

- case $\alpha = 1$:
  - Sample covariance of the projected points is:

$$\widehat{\text{Cov}}\,(\tilde{Z}) = \frac{1}{n-1} U^\top_{1:n,1:p} U_{1:n,1:p} = \frac{1}{n-1} I_p.$$

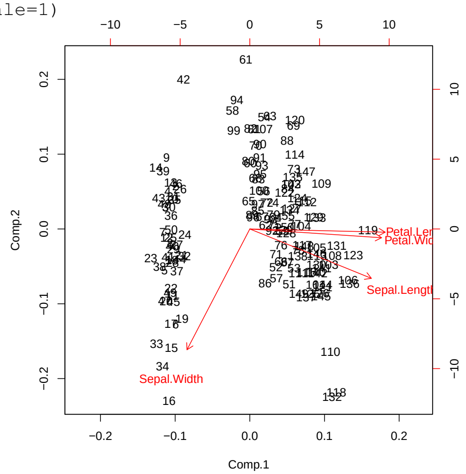
    Projected points are uncorrelated and dimensions are equi-variance.
  - Sample covariance between $X^{(i)}$ and $X^{(j)}$ is:

$$\hat{\mathbb{E}}(X^{(i)} X^{(j)}) = \frac{1}{n-1} \left(VD^\top DV^\top \right)_{i,j} = \frac{1}{n-1} \langle D_{1:p,1:p} V_i^\top, D_{1:p,1:p} V_j^\top \rangle$$

    The angle between the projected variables corresponds to the correlation.

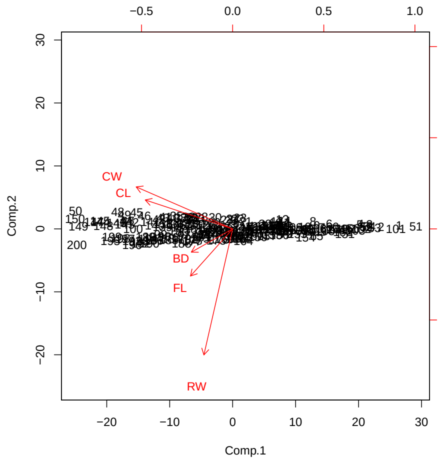
Iris Data biplot - scaled

```
 > ?biplot
...
scale: The variables are scaled by lambda ^ scale and the observations
are scaled by lambda ^ (1-scale) where lambda are the singular values
as computed by princomp. (default=1)
...
> biplot(iris.pca,scale=1)
```

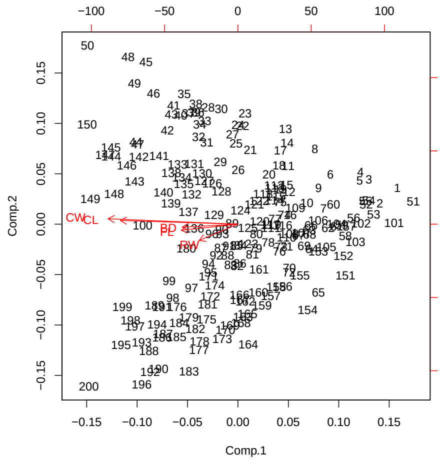


Crabs Data biplots

```
> biplot(Crabs.pca,scale=0)
```




```
> biplot(Crabs.pca,scale=1)
```



US Arrests Data

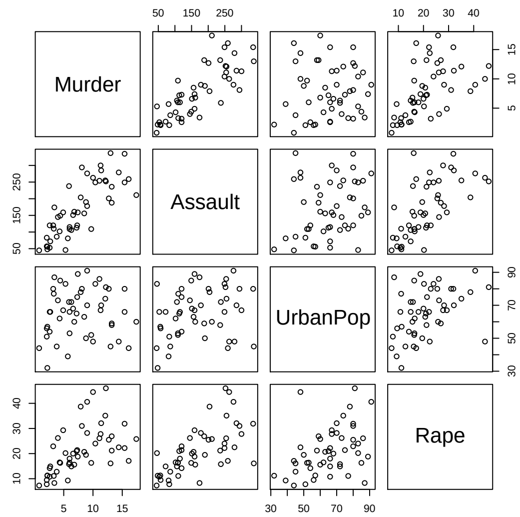
This data set contains statistics, in arrests per 100,000 residents for assault, murder, and rape in each of the 50 US states in 1973. Also given is the percent of the population living in urban areas.

```
pairs(USArrests)
usarrests.pca <- princomp(USArrests,cor=T)
plot(usarrests.pca)

pairs(predict(usarrests.pca))
biplot(usarrests.pca)
```


US Arrests Data Pairs Plot

```
> pairs(USArrests)
```



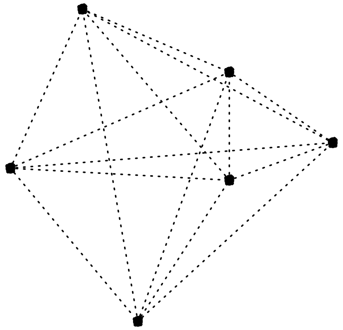
US Arrests Data Biplot

```
> biplot(usarrests.pca)
```

Multidimensional Scaling

Suppose there are n points $\mathbf{X}$ in $\mathbb{R}^p$, but we are only given the $n \times n$ matrix $\mathbf{D}$ of inter-point distances.

Can we reconstruct $\mathbf{X}$?



Multidimensional Scaling

Rigid transformations (translations, rotations and reflections) do not change inter-point distances so cannot recover $\mathbf{X}$ exactly. However $\mathbf{X}$ can be recovered up to these transformations!

- Let $d_{ij} = \|x_i - x_j\|_2$ be the distance between points x_i and x_j.

$$\begin{aligned} d_{ij}^2 &= \|x_i - x_j\|_2^2 \\ &= (x_i - x_j)^\top (x_i - x_j) \\ &= x_i^\top x_i + x_j^\top x_j - 2x_i^\top x_j \end{aligned}$$

- Let $\mathbf{B} = \mathbf{X}\mathbf{X}^\top$ be the $n \times n$ matrix of dot-products, $b_{ij} = x_i^\top x_j$. The above shows that $\mathbf{D}$ can be computed from $\mathbf{B}$.
- Some algebraic exercise shows that $\mathbf{B}$ can be recovered from $\mathbf{D}$ if we assume $\sum_{i=1}^n x_i = 0$.

Multidimensional Scaling

- If we knew $\mathbf{X}$, then SVD gives $\mathbf{X} = UDV^\top$. As $\mathbf{X}$ has rank $k = \min(n, p)$, we have at most k singular values in D and we can assume $U \in \mathbb{R}^{n \times k}$, $D \in \mathbb{R}^{k \times p}$ and $V \in \mathbb{R}^{p \times p}$.
- The eigendecomposition of $\mathbf{B}$ is then:

$$\mathbf{B} = \mathbf{X}\mathbf{X}^\top = UDD^\top U^\top = U\Lambda U^\top.$$

- This eigendecomposition can be obtained from $\mathbf{B}$ without knowledge of $\mathbf{X}$!
- Let $\tilde{x}_i^\top = U_i \Lambda^{\frac{1}{2}}$ be the ith row of $U\Lambda^{\frac{1}{2}}$. Pad $\tilde{x}_i$ with 0s so that it has length p.

$$\tilde{x}_i^\top \tilde{x}_j = U_i \Lambda U_j^\top = b_{ij} = x_i^\top x_j$$

  and we have found a set of vectors with dot-products given by $\mathbf{B}$.
- The vectors $\tilde{x}_i$ differs from x_i only via the orthogonal matrix V so are equivalent up to rotation and reflections.

US City Flight Distances

We present a table of flying mileages between 10 American cities, distances calculated from our 2-dimensional world. Using D as the starting point, metric MDS finds a configuration with the same distance matrix.

ATLA	CHIG	DENV	HOUS	LA	MIAM	NY	SF	SEAT	DC
0	587	1212	701	1936	604	748	2139	2182	543
587	0	920	940	1745	1188	713	1858	1737	597
1212	920	0	879	831	1726	1631	949	1021	1494
701	940	879	0	1374	968	1420	1645	1891	1220
1936	1745	831	1374	0	2339	2451	347	959	2300
604	1188	1726	968	2339	0	1092	2594	2734	923
748	713	1631	1420	2451	1092	0	2571	2408	205
2139	1858	949	1645	347	2594	2571	0	678	2442
2182	1737	1021	1891	959	2734	2408	678	0	2329
543	597	1494	1220	2300	923	205	2442	2329	0

US City Flight Distances

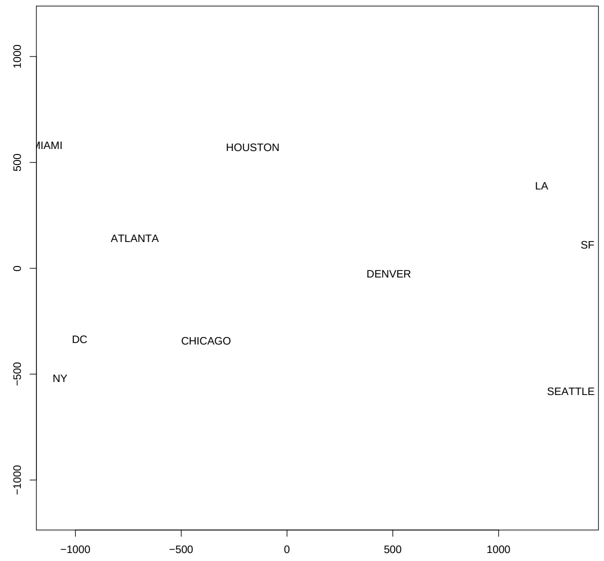
```
library(MASS)

us <- read.csv("http://www.stats.ox.ac.uk/~sejdinov/teaching/data/uscities.csv")

## use classical MDS to find lower dimensional views of the data
## recover X in 2 dimensions

us.classical <- cmdscale(d=us,k=2)
plot(us.classical)
text(us.classical,labels=names(us))
```


US City Flight Distances



Lower-dimensional Reconstructions

In classical MDS derivation, we used all eigenvalues in the eigendecomposition of $\mathbf{B}$ to reconstruct

$$\tilde{x}_i = U_i \Lambda^{\frac{1}{2}}.$$

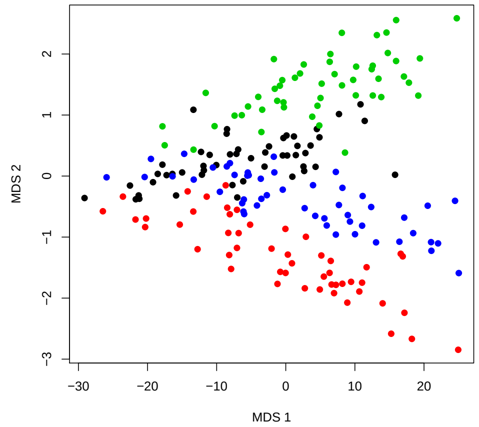
We can use only the largest $k < \min(n, p)$ eigenvalues and eigenvectors in the reconstruction, giving the 'best' k-dimensional view of the data.

This is analogous to PCA, where only the largest eigenvalues of $\mathbf{X}^\top \mathbf{X}$ are used, and the smallest ones effectively suppressed.

Indeed, PCA and classical MDS are duals and yield effectively the same result.

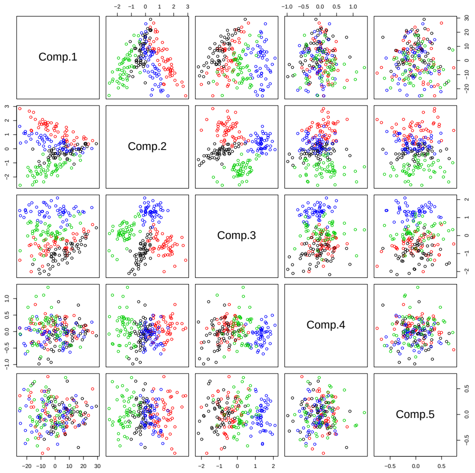
Crabs Data

```
library(MASS)
crabs$spsex=paste(crabs$sp,crabs$sex,sep="")
varnames<-c('FL','RW','CL','CW','BD')
Crabs <- crabs[,varnames]
Crabs.class <- factor(crabs$spsex)
crabsmds <- cmdscale(d= dist(Crabs),k=2)
plot(crabsmds, pch=20, cex=2,col=unclass(Crabs.class))
```



Crabs Data

Compare with previous PCA analysis.
Classical MDS solution corresponds to the first 2 PCs.



Varieties of MDS

Generally, MDS is a class of dimensionality reduction techniques which represents data points $x_1, \dots, x_n \in \mathbb{R}^p$ in a lower-dimensional space $z_1, \dots, z_n \in \mathbb{R}^k$ which tries to preserve inter-point (dis)similarities.

- It requires only the matrix $\mathbf{D}$ of pairwise dissimilarities $d_{ij} = d(x_i, x_j)$. For example, we can use Euclidean distance $d_{ij} = \|x_i - x_j\|_2$, but other dissimilarities are possible.
- MDS finds representations $z_1, \dots, z_n \in \mathbb{R}^k$ such that

$$\|z_i - z_j\|_2 \approx d(x_i, x_j) = d_{ij},$$

  and differences in dissimilarities are measured by the appropriate loss $\Delta(d_{ij}, \|z_i - z_j\|_2)$.
- Goal: Find $\mathbf{Z}$ which minimizes the **stress function**

$$S(\mathbf{Z}) = \sum_{i \neq j} \Delta(d_{ij}, \|z_i - z_j\|_2).$$

Varieties of MDS

- Choices of (dis)similarities and stress functions lead to different objective (**stress**) functions and different algorithms.
  - **Classical**: preserves similarities (`cmdscale`)

$$S(\mathbf{Z}) = \sum_{i \neq j} \left(b_{ij} - \langle z_i - \bar{z}, z_j - \bar{z} \rangle\right)^2$$

  - **Metric Shephard-Kruskal**: distances, rather than squared distances

$$S(\mathbf{Z}) = \sum_{i \neq j} (d_{ij} - \|z_i - z_j\|_2)^2$$

  - **Sammon**: preserves shorter distances more (`sammon`)

$$S(\mathbf{Z}) = \sum_{i \neq j} \frac{(d_{ij} - \|z_i - z_j\|_2)^2}{d_{ij}}$$

  - **Non-Metric Shephard-Kruskal**: ignores actual distance values, only preserves ranks (`isoMDS`)

$$S(\mathbf{Z}) = \min_{g \text{ increasing}} \frac{\sum_{i \neq j} (g(d_{ij}) - \|z_i - z_j\|_2)^2}{\sum_{i \neq j} \|z_i - z_j\|_2}$$

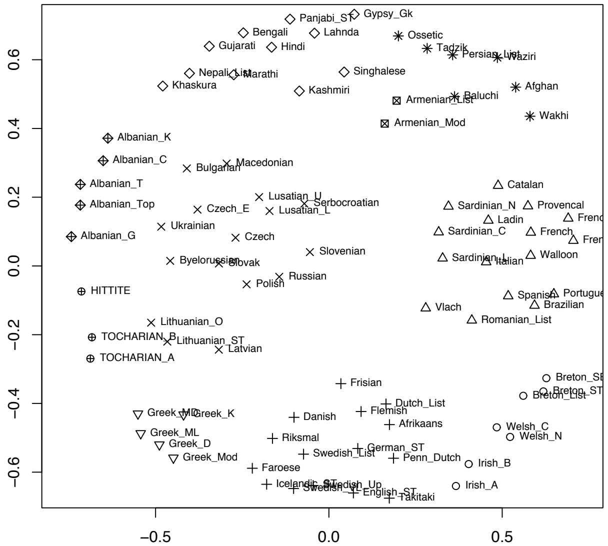
Example: Language data

Presence or absence of 2867 homologous traits in 87 Indo-European languages.

```
> X<-read.table("http://www.stats.ox.ac.uk/~sejdinov/teaching/data/cognate.txt")
> X[1:15,1:16]
               V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16
Irish_A         0  0  0  0  1  0  0  0  0   0   0   0   0   0   0   0
Irish_B         0  0  0  0  1  0  0  0  0   0   0   0   0   0   0   0
Welsh_N         0  0  0  1  0  0  0  0  0   0   0   0   0   0   0   0
Welsh_C         0  0  0  1  0  0  0  0  0   0   0   0   0   0   0   0
Breton_List     0  0  0  0  1  0  0  0  0   0   0   0   0   0   0   0
Breton_SE       0  0  0  0  1  0  0  0  0   0   0   0   0   0   0   0
Breton_ST       0  0  0  0  1  0  0  0  0   0   0   0   0   0   0   0
Romanian_List   0  1  0  0  0  0  0  0  0   0   0   0   0   0   0   0
Vlach           0  1  0  0  0  0  0  0  0   0   0   0   0   0   0   0
Italian         0  1  0  0  0  0  0  0  0   0   0   0   0   0   0   0
Ladin           0  1  0  0  0  0  0  0  0   0   0   0   0   0   0   0
Provencal       0  1  0  0  0  0  0  0  0   0   0   0   0   0   0   0
French          0  1  0  0  0  0  0  0  0   0   0   0   0   0   0   0
Walloon         0  1  0  0  0  0  0  0  0   0   0   0   0   0   0   0
French_Creole_C 0  1  0  0  0  0  0  0  0   0   0   0   0   0   0   0
```

Example: Language data

Using MDS with non-metric (Sammon) scaling.

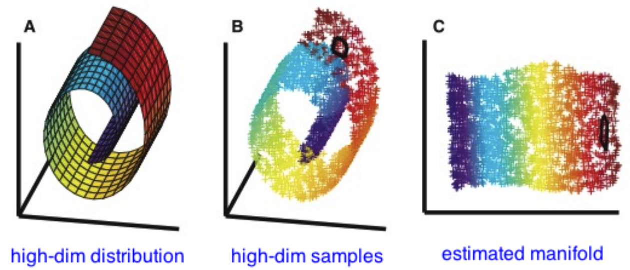


Nonlinear Dimensionality Reduction

Two aims of different varieties of MDS:

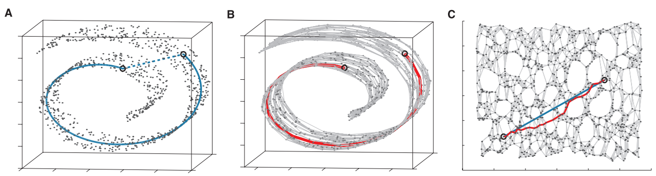
- To visualize the (dis)similarities among items in a dataset, where these (dis)disimilarities may not have Euclidean geometric interpretations.
- To perform **nonlinear** dimensionality reduction.

Many high-dimensional datasets exhibit low-dimensional structure ("live on a low-dimensional menifold").



Isomap

Isomap is a non-linear dimensional reduction technique based on classical MDS. Differs from other MDSs as it uses estimates of **geodesic distances** between the data points.

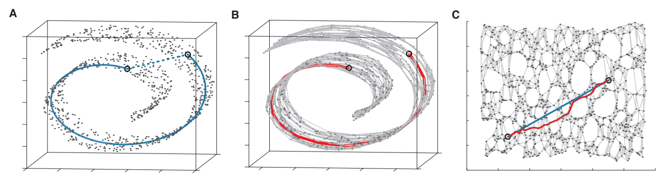


Tenenbaum et al. (2000)

Isomap

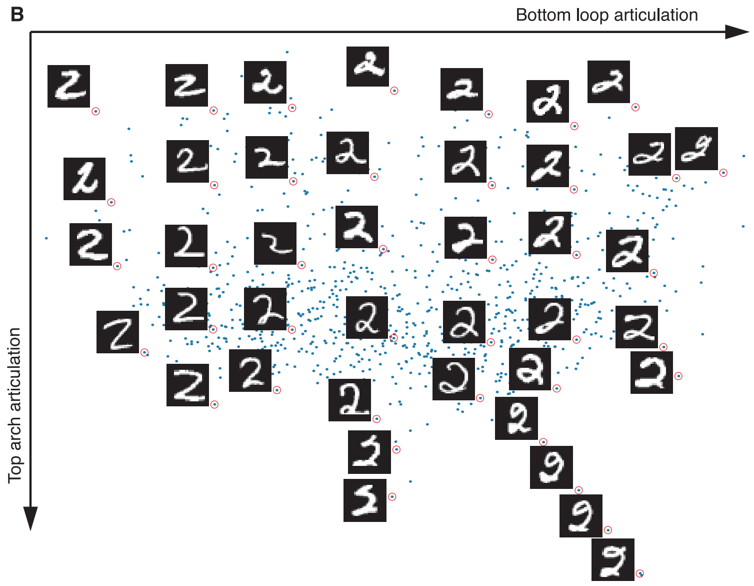
Isomap

- Calculate Euclidean distances d_{ij} for $i, j = 1, \ldots, n$ between all data points.
- Form a graph G with n samples as nodes, and edges between the respective K nearest neighbours (K-Isomap) or between i and j if $d_{ij} < \epsilon$ (ϵ-Isomap).
- For i, j linked by an edge, set $d^G_{ij} = d_{ij}$. Otherwise, set d^G_{ij} to the shortest-path distance between i and j in G.
- Run classical MDS using distances d^G_{ij}.

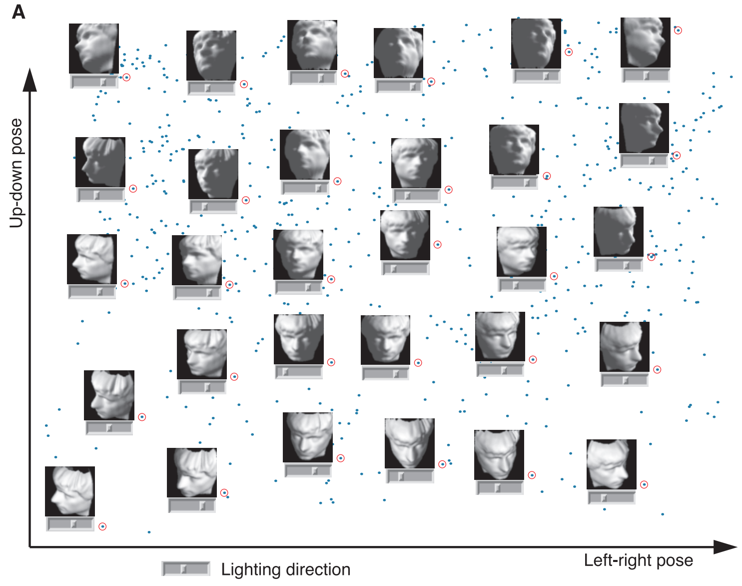


R function: `isomap{vegan}`.

Handwritten Characters



Faces



Nonlinear Dimensionality Reduction Techniques

- Kernel PCA
- Locally Linear Embedding
- Laplacian Eigenmaps
- Maximum Variance Unfolding