CS-E4840
Information Visualization
Lecture 9: Dimensionality reduction

Tassu Takala <tapio.takala@aalto.fi>
28 March 2019

Go to http://www.iki.fi/kaip/p/dr2.nb.html

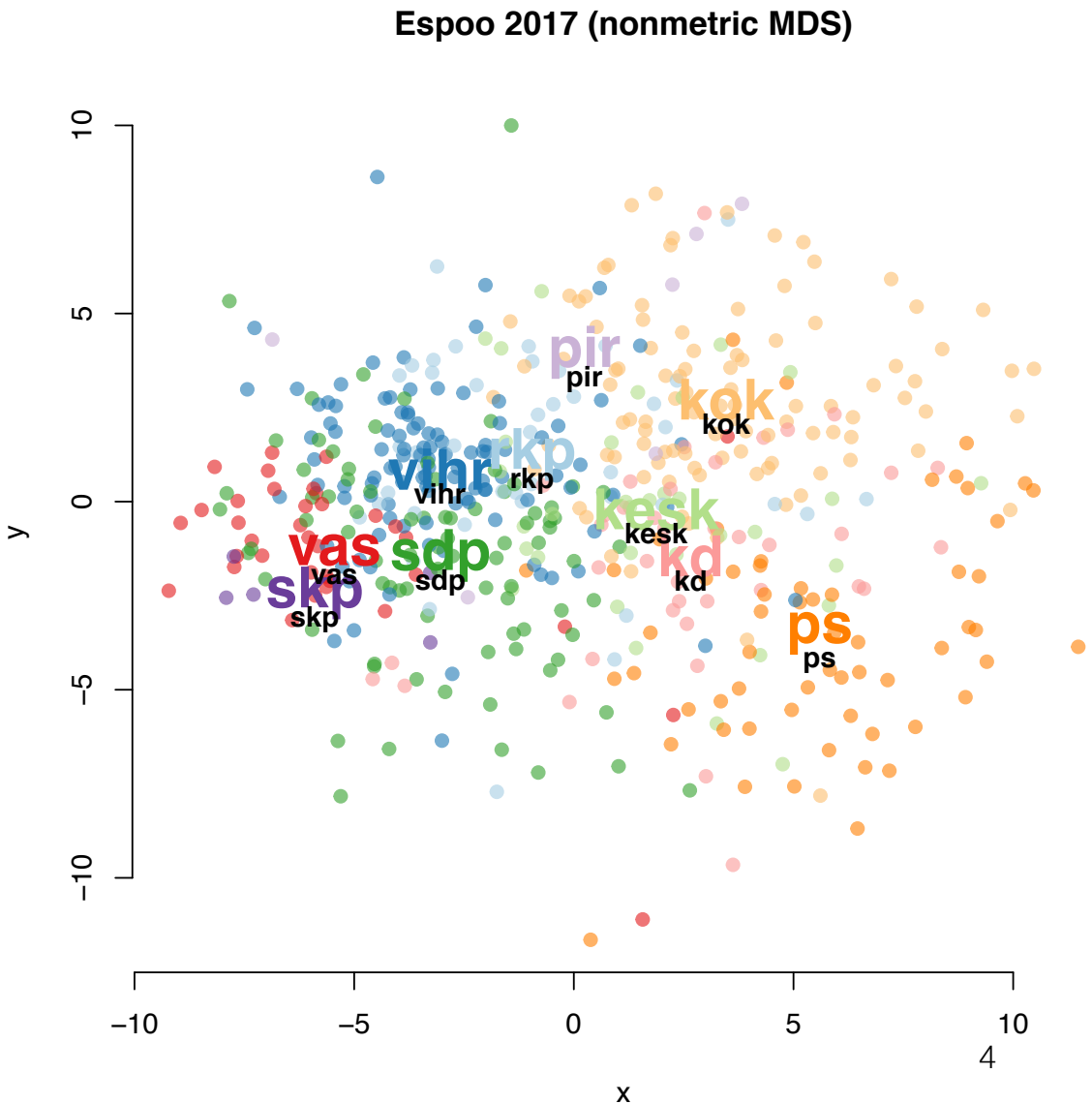
Literature on dimensionality reduction for visualisation

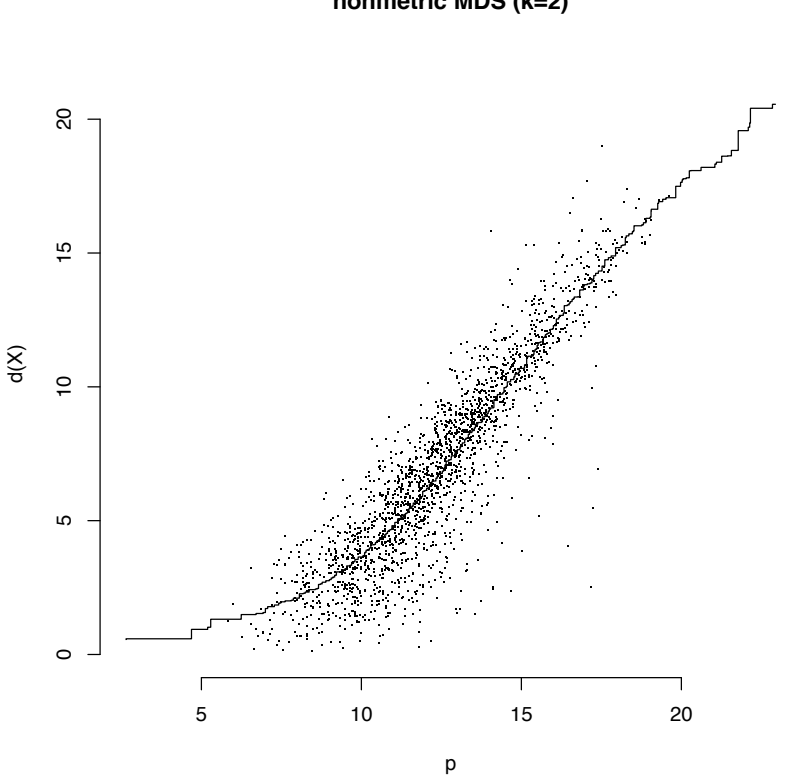
- MDS: Borg, Kroenen, Modern multidimensional scaling: theory and applications. Springer, 1997.
- PCA: any book on matrix algebra.
- Jarkko Venna 2007, Academic Dissertation, http://lib.tkk.fi/Diss/2007/isbn9789512287529/
- Lee & Verleysen, 2007. Nonlinear dimensionality reduction. Springer.
- For a reasonably recent brief review see Verleysen & Lee, 2013 (recommended reading before exam!). https://doi.org/10.1007/978-3-642-42054-2_77
- See the references in the slides! Notice that most doi.org links can be accessed from within Aalto network (but usually not from home).
- (Not to be confused with dimensionality reduction for machine learning where target dimensionality is often higher!)
- Go to http://www.iki.fi/kaip/p/dr2.nb.html

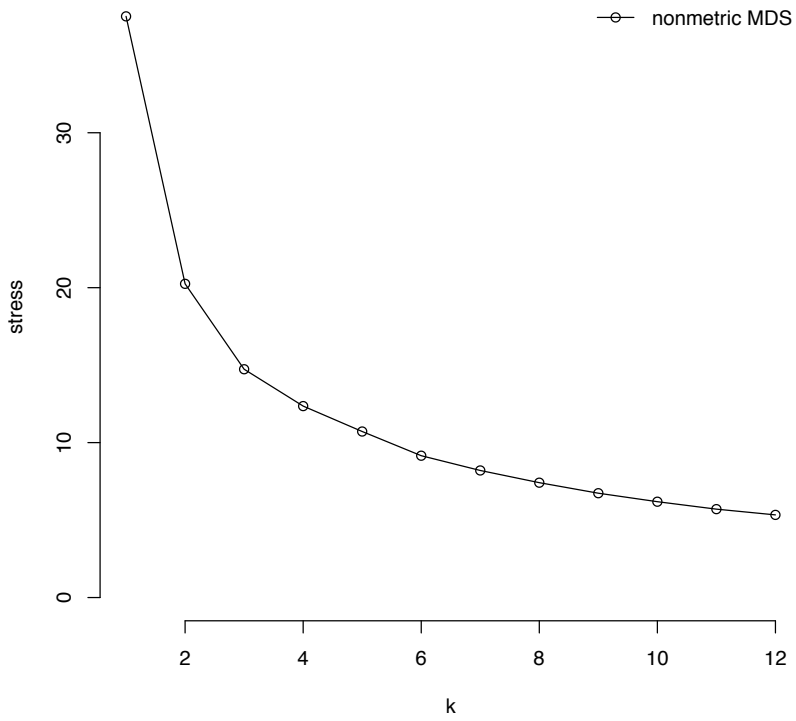
Municipal elections in Espoo in 2017

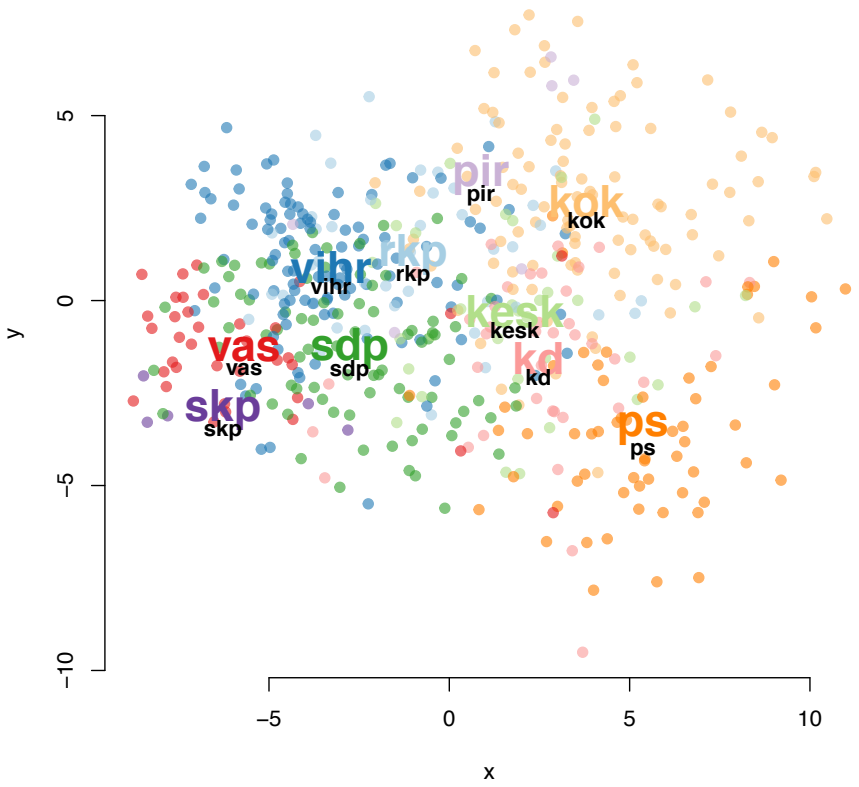
- Survey of candidates done by Helsingin Sanomat
- Here included only 10 parties with largest number of candidates nationally.
- Each candidate rated each of the **m=49** statements on a scale from 1 to 5, where 1=disagree and 5=agree:
  1. *Euthanasia should be allowed.*
  2. *I prefer public instead of the private sector to produce my local health services.*
  3. *Same gender couples should have the same marital and adoptation rights than the different genre couples.*
  4. *Good brother networks influence municipal decision-making.*
  5. *...*
- **n=515** candidates in total, i.e., we have a data 515x49 matrix.
- Distance p_{ij} between candidates i and j is the Euclidean distance of their respective 49-dimensional rating vectors. What is a 2-dimensional representation that preserves these distances faithfully?

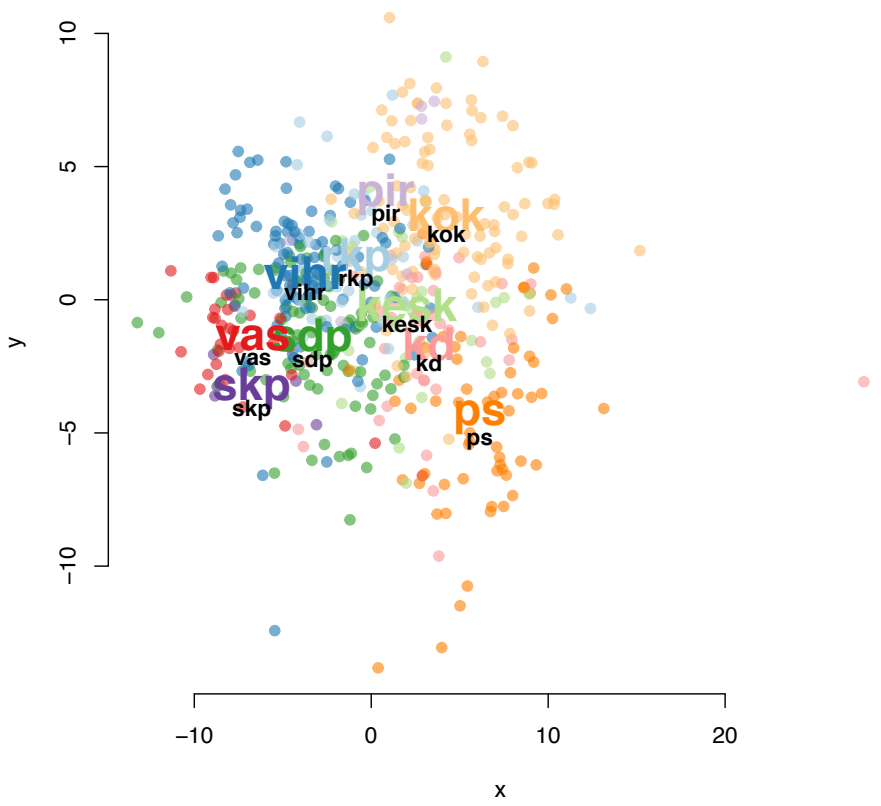
Municipal elections in Espoo in 2017

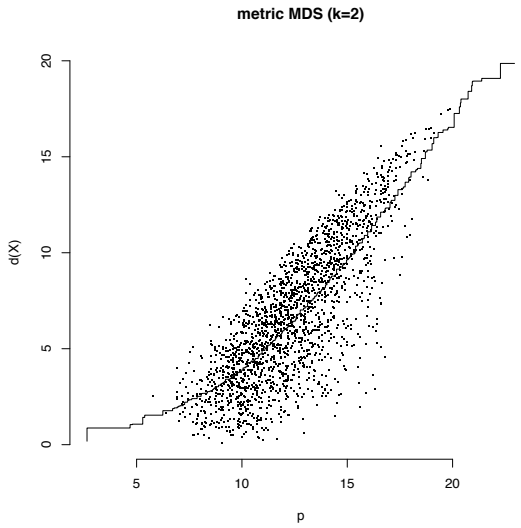


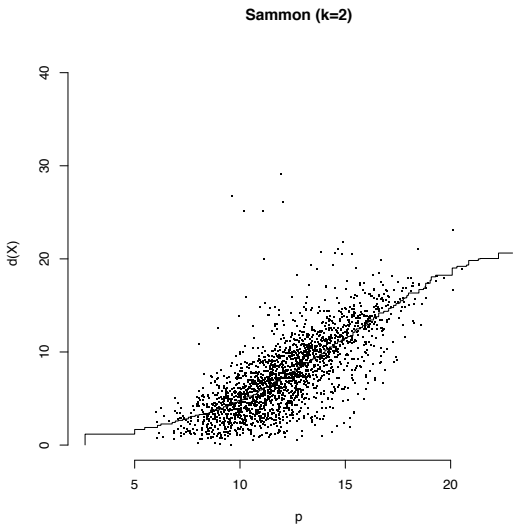












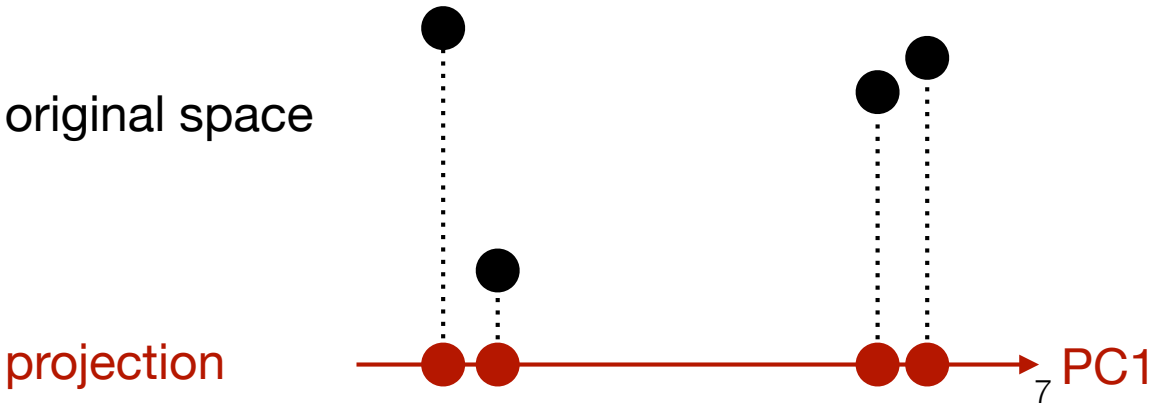
See http://www.iki.fi/kaip/p/dimensionality_reduction_1.nb.html

Projection pursuit methods

- MDS (and variants) are based on distance matrix between points.
- If data is composed of vectors (such as Espoo municipal elections 2017 data) we can use projection pursuit methods.
- Projection pursuit methods try to find a linear subspace u that maximise some quantity
- E.g., for the election data let *X* be the 515x49 data matrix and *f* a function. Problem: find 49-dimensional unit vector *u* such that *f(A u)* is maximised.
  - if f is variance we have principal component analysis (PCA)
  - if f is a measure of non-gaussianity we have independent component analysis (ICA)
- We can find several directions u (possibly with orthogonality conditions).

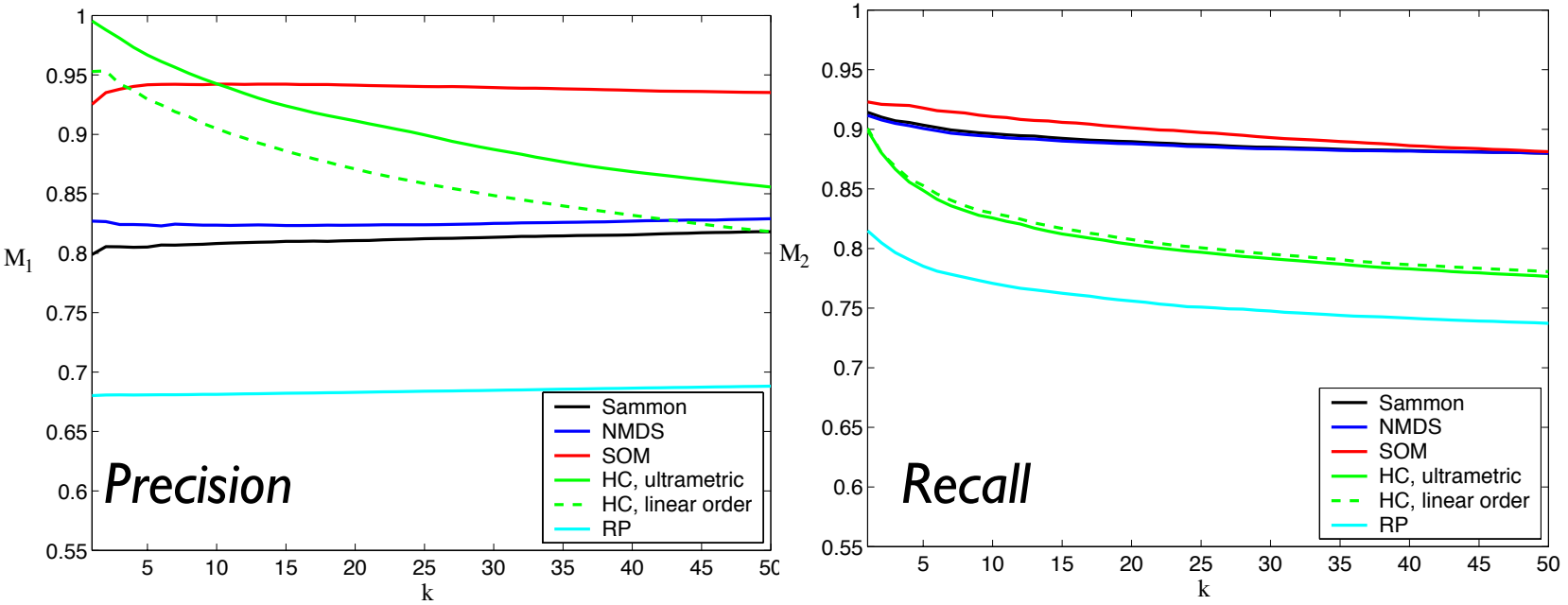
Precision and recall

- **Precision:** if the points are nearby in embedding they are nearby in the original space
- **Recall:** if the points are nearby in the original space they are nearby in the embedding
- Projection pursuit methods such as **PCA**:
  - the distance between the points in projection is at most the distance in the original space
  - if the points are nearby in the original space they are nearby in the embedding (**good recall**)
  - if the points are nearby in the embedding they may be distant in the projection (possibly **bad precision**)



Performance of MDS

- MDS is tries to preserve the large distances at the expense of small ones, hence, it can "collapse" some small distances on the expense of preserving large distances
- A projection is *trustworthy (precision)* if *k* closest neighbours of a sample on the projection are also close by in the original space. A projection *preserves the original neighbourhoods* (*recall*) if all *k* closest neighbours of a sample in the original space are also close by in the projection.



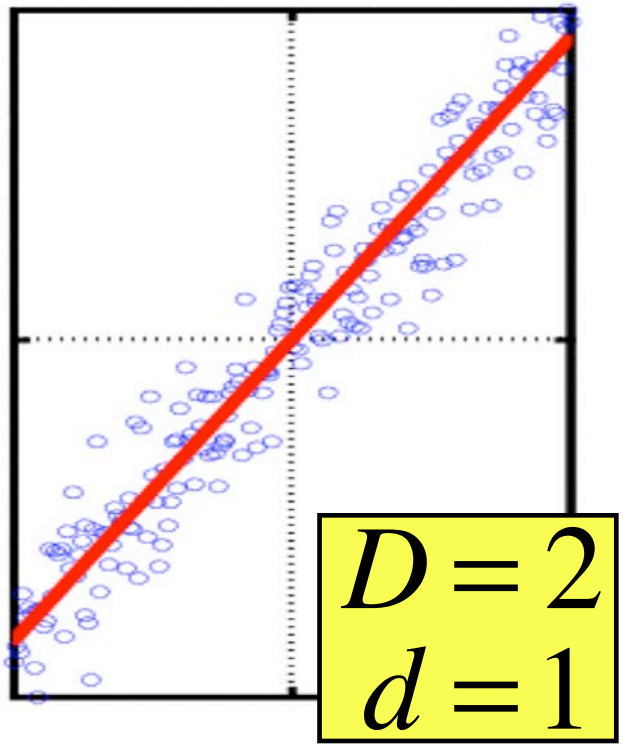
Figures from Kaski, et al. 2003, https://doi.org/10.1186/1471-2105-4-48

Precision and recall as a function of the neighbourhood size *k* for a yeast data set. Non-metric (ordinal) MDS (NMDS) is shown in blue. Larger precision and recall is better.

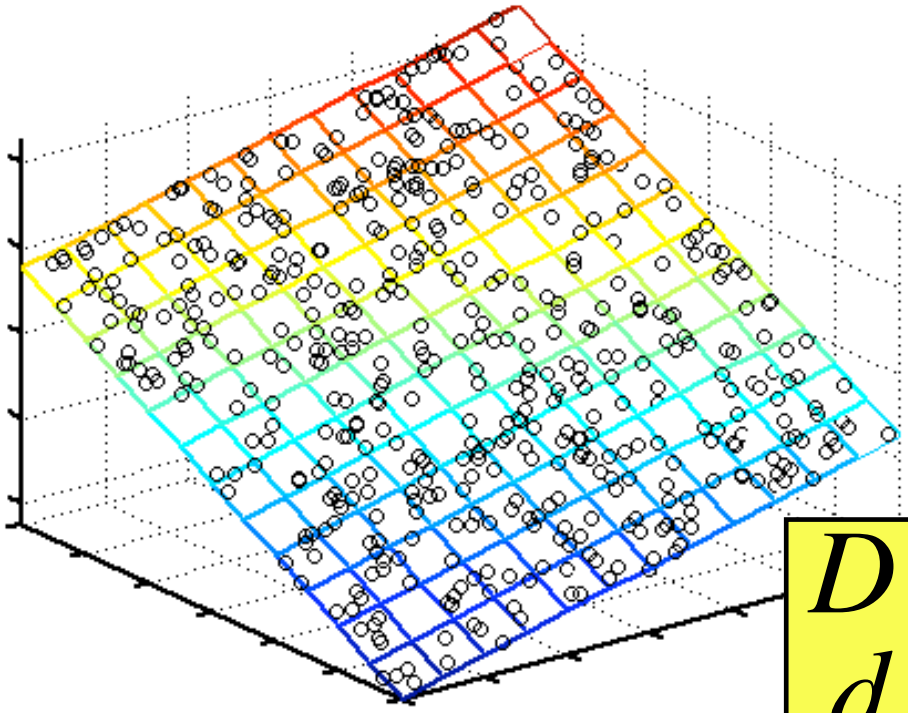
Performance of MDS

- Relatively better recall, worse precision
- MDS algorithms typically have running times of the order $O(N^2)$, where N is the number of data items.
- This is not very good: N=1,000 data items are ok, but N=1,000,000 is getting slow.
- Some solutions: use landmark points (i.e., use MDS only on a subset of data points and place the remaining points according to those, use MDS on cluster centroids etc.), use some other algorithm or modification of MDS.
- MDS is not guaranteed to find the global optimum of the stress (cost) function, nor it is guaranteed to converge to the same solution at each run (many of the MDS algorithms are quite good and reliable, though)

Principal component analysis



$D = 2$
$d = 1$



$D = 3$
$d = 2$

Principal component analysis (PCA)

- The **principal component analysis** (PCA) finds the eigenvalues and -vectors of a matrix
- PCA is an example of the projection pursuit methods. It tries to find a linear subspaces that have **maximal variance**.
- Thus, the interesting quality in PCA is variance (distance). I.e., you could think PCA as a linearised version of MDS (actually PCA is equivalent to one modification of MDS).
- PCA (unlike MDS) assumes that the data points are vectors in a high-dimensional Euclidean space,
- The data points are projected to d-dimensional Euclidean subspace ($d \ll D$) of the original space.
- The projection to d-dimensional subspace is linear, $A = \sum_{\alpha=1}^{d} e_\alpha e_\alpha^T$

  $y_i = Ax_i$, and where e_α are orthogonal unit vectors.
- Goal: nearby points remain nearby, distant points remain distant.

Principal component analysis (PCA)

- Goal, more formally: *find such projection to d dimensional subspace that the average error in the squared Euclidean distances between data points is minimised.*

$$\sum_{i,j=1}^{N} \left| \| x_i - x_j \|^2 - \| y_i - y_j \|^2 \right|$$

  where || || is the Euclidean distance and $y_i = A\, x_i$.

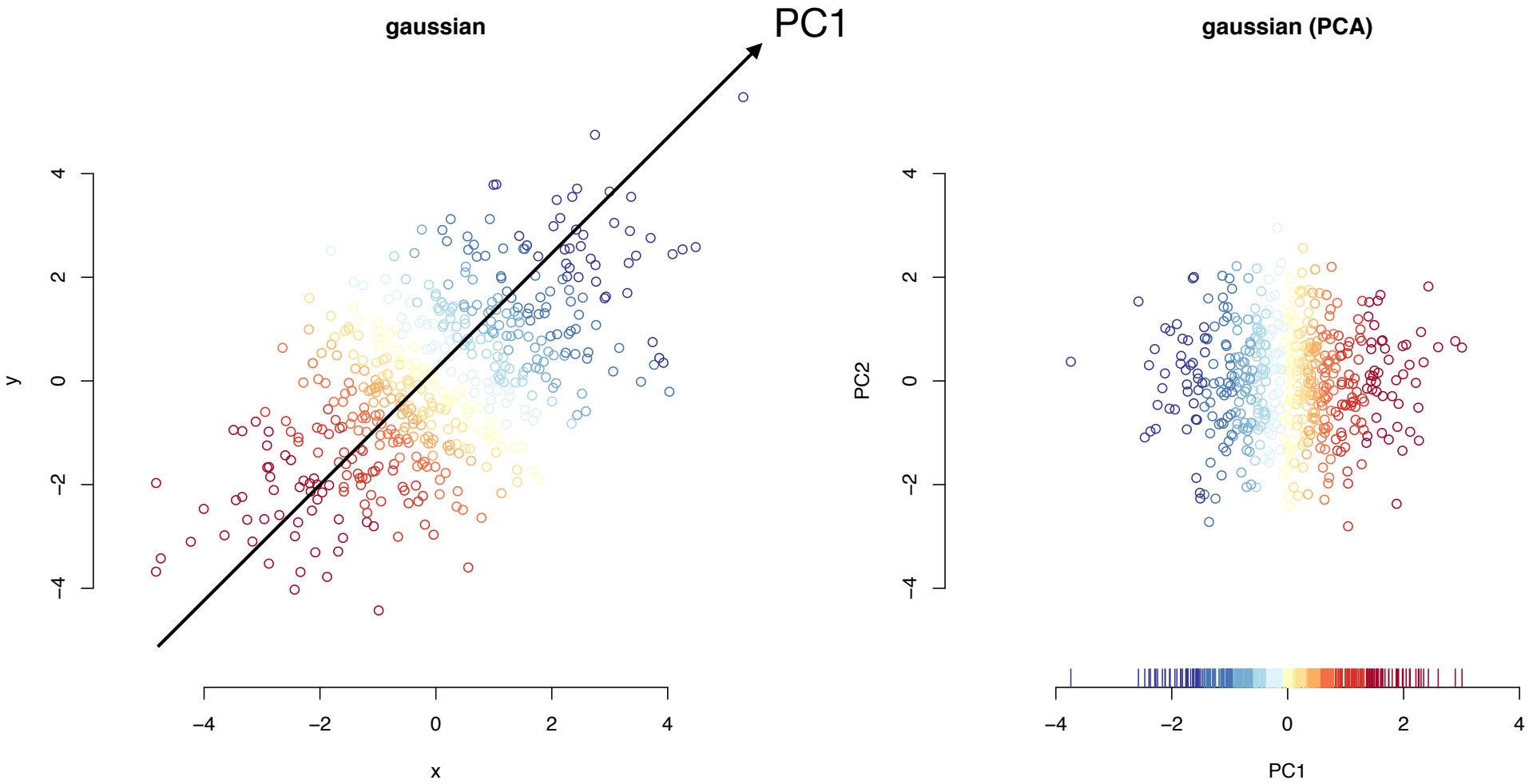
- Denote the mean vector by, $\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$
- The *covariance matrix* reads then, $C = \frac{1}{N} \sum_{i=1}^{N} (x_i - \overline{x})(x_i - \overline{x})^T.$
- The covariance matrix can be decomposed (*spectral decomposition*) as

$$C = \sum_{\alpha=1}^{D} \lambda_a e_\alpha e_\alpha^T$$

  where λ_α are the *eigenvalues* ($\lambda_1 \geq \lambda_2 \geq ... \geq 0$) and e_α are the corresponding orthogonal unit *eigenvectors*.

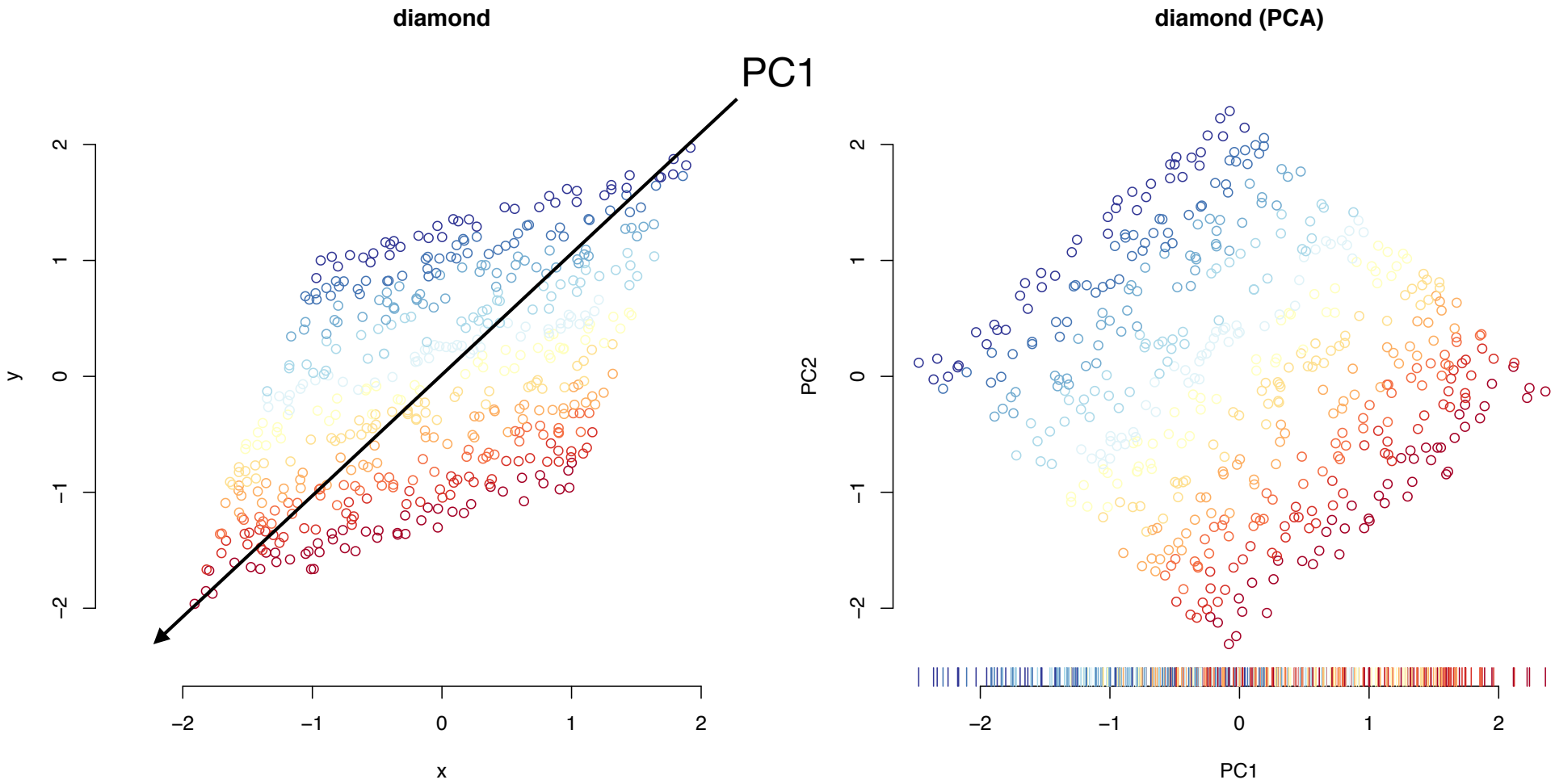
- The maximum variance projection is then given by $A = \sum_{\alpha=1}^{d} e_\alpha e_\alpha^T$

Gaussian data



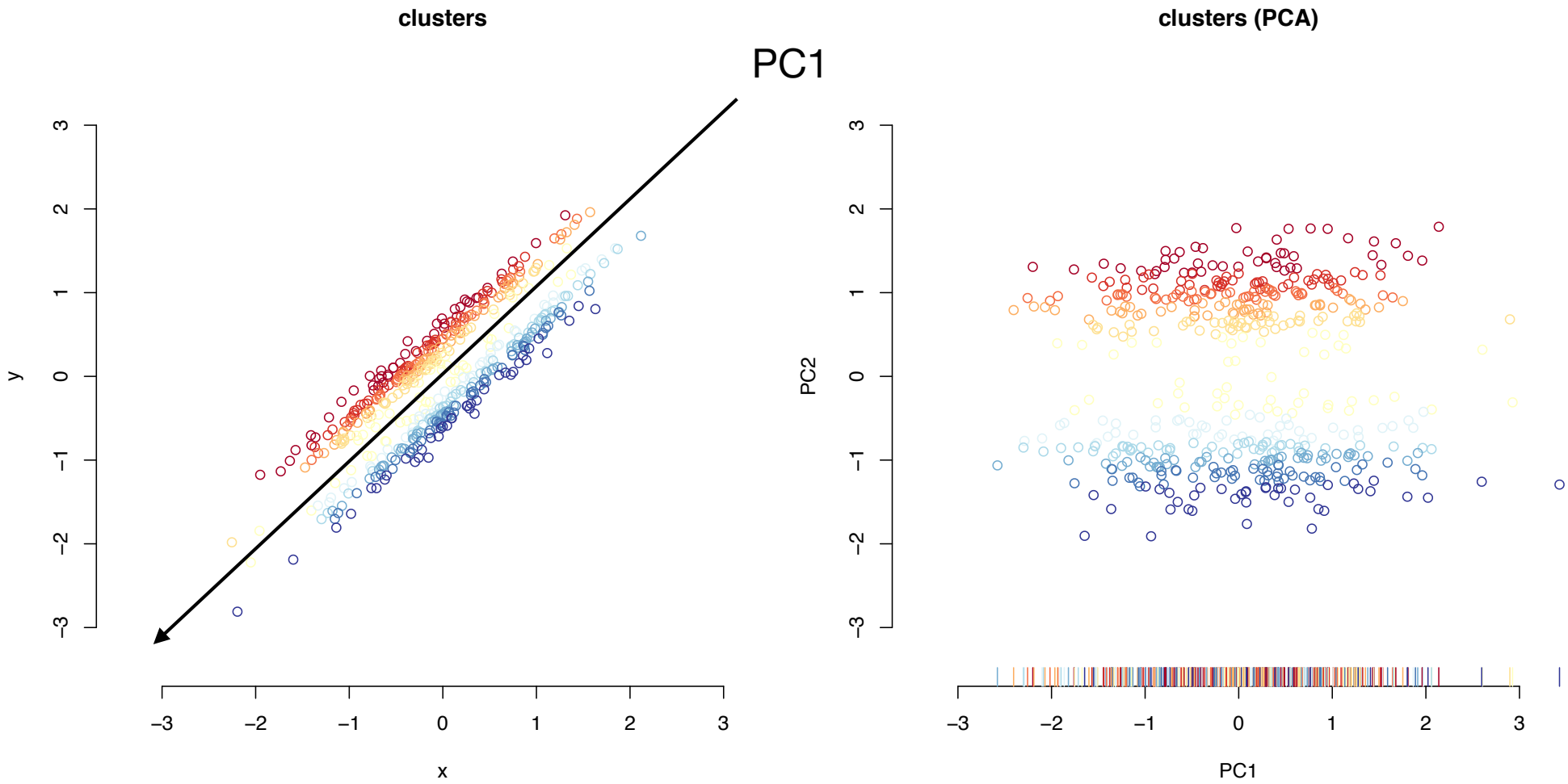
PC1 finds the direction of largest variance.

Diamond shaped data



PC1 misses the square structure.

Two clusters



PC1 misses the cluster structure.

Principal component analysis (PCA)

- PCA can be computed easily with (almost) any software that is capable of doing linear algebra.
- PCA is stable, there are no additional parameters, and it is guaranteed always to converge to the same optima.
- Hence, PCA is usually the first dimension reduction method to try (if it doesn't work, then try something more fancy)

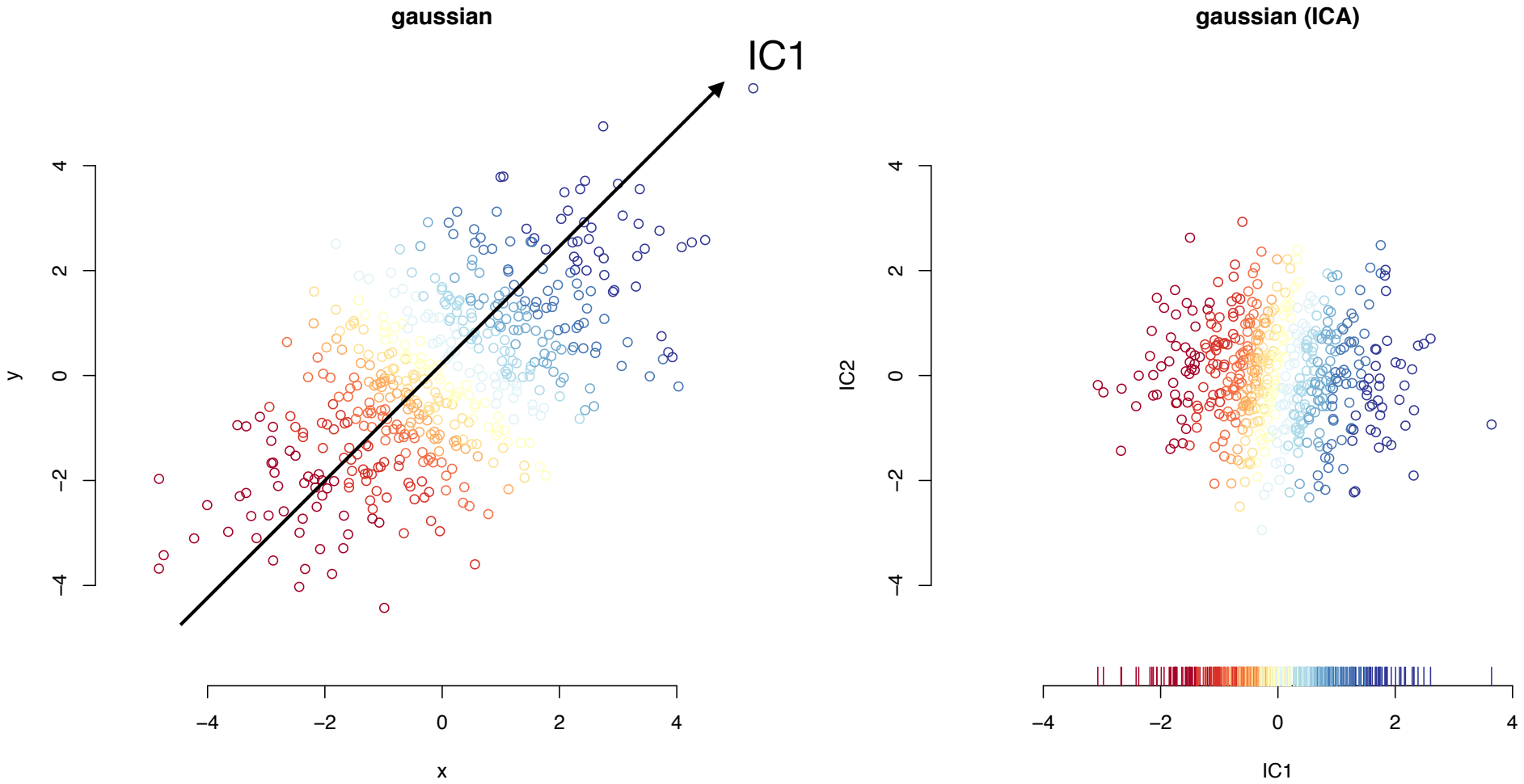
```
d <- 2
X <- scale(X,center=TRUE,scale=FALSE)
X %*% svd(t(X) %*% X)$u[,1:d]
```

- If you find PCA difficult, this may help :-)
  https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues

Independent component analysis (ICA)

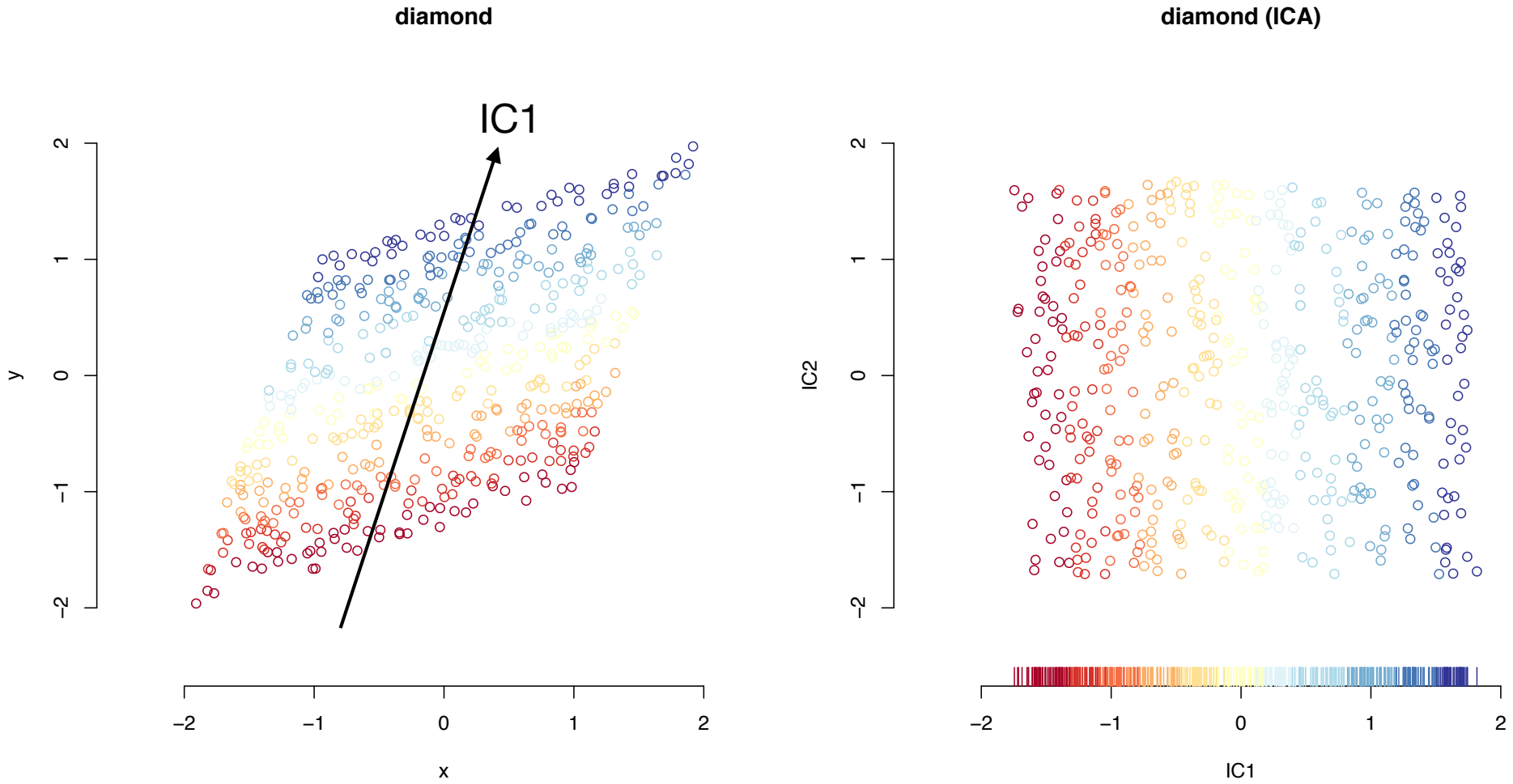
- Goal: function *f* is a measure of non-Gaussianity. Non-Gaussian directions are usually most independent.
- Hence, ICA finds separate processes. Directions are not necessarily orthogonal.
- ICA is unstable and may end up to a local minimum.
- There are robust libraries to compute ICA: use the libraries!

Gaussian data



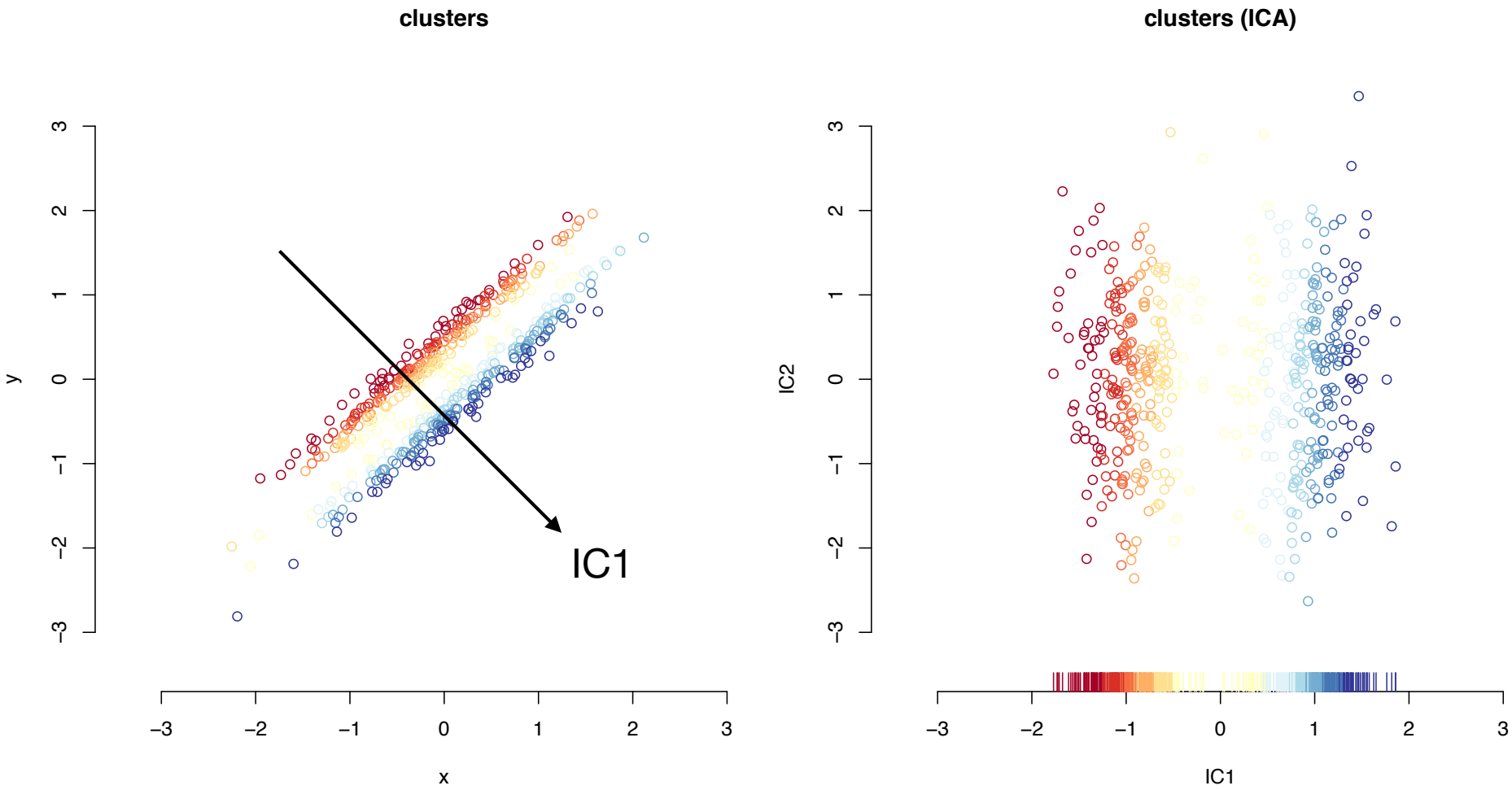
ICA ignores total variance (but finds the maximal variance direction here by co-incidence).

Diamond shaped data



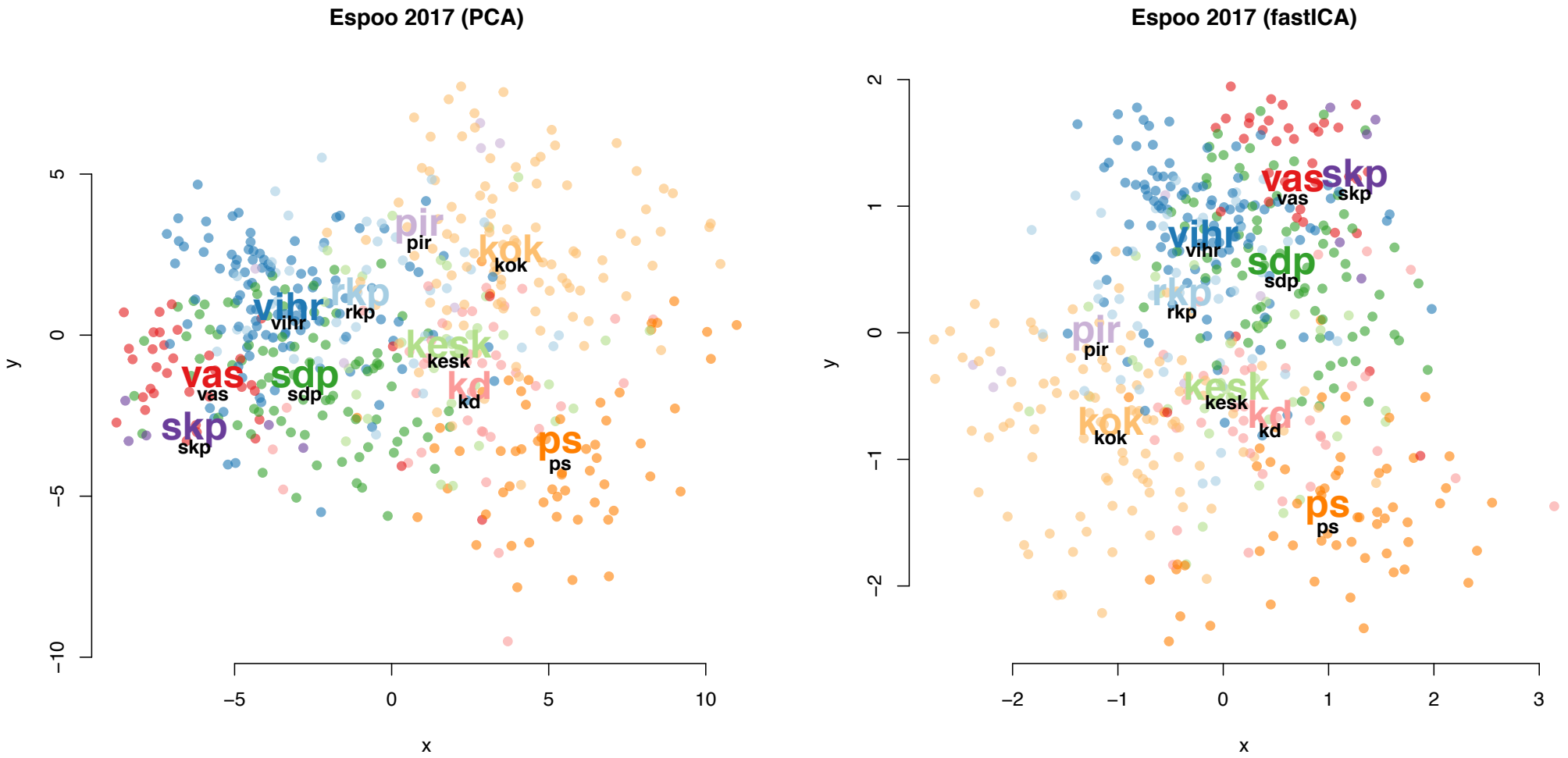
IC1 finds the box in the diamond.

Two clusters



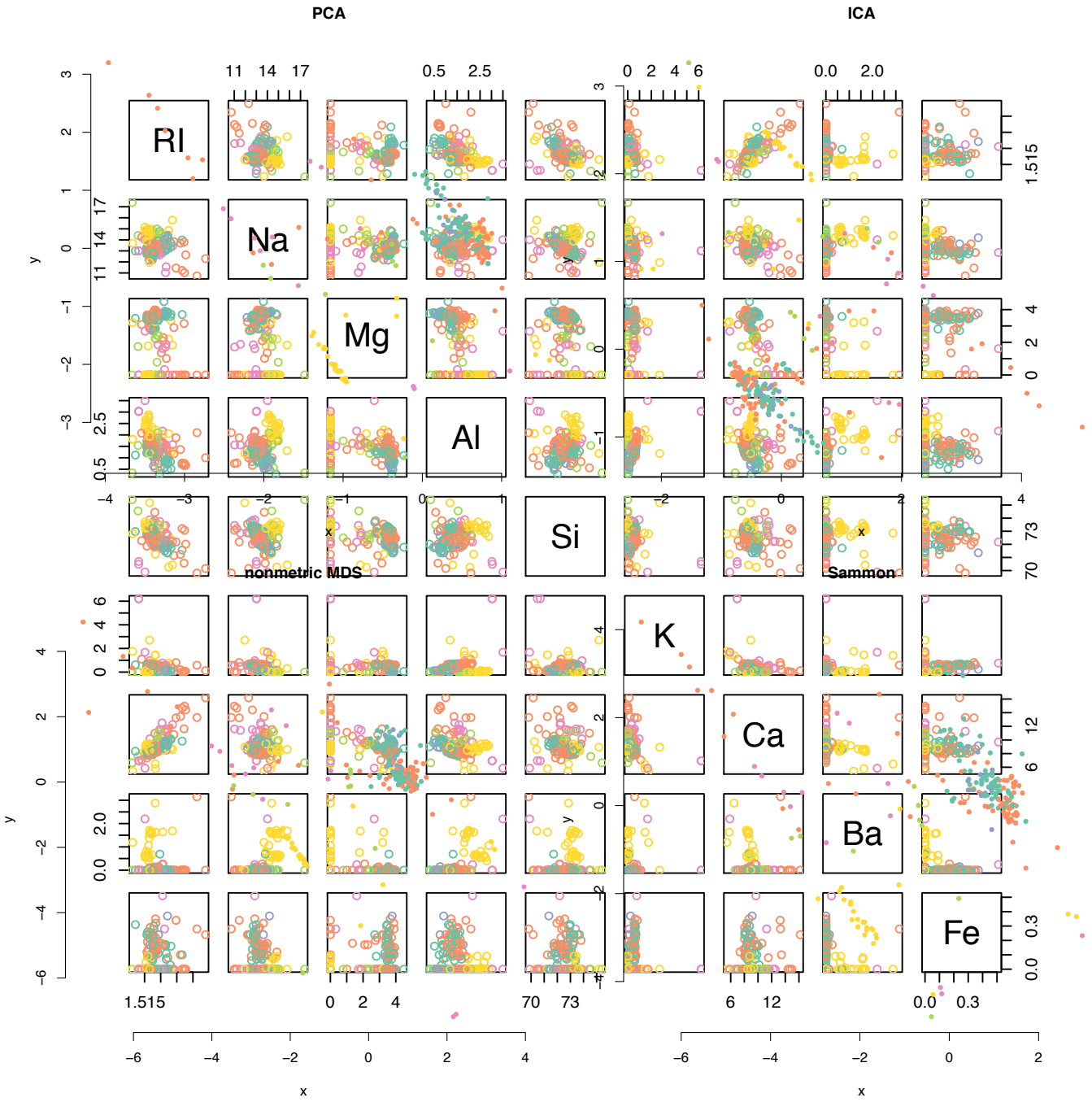
IC1 finds the two clusters.

Municipal elections in Espoo in 2017

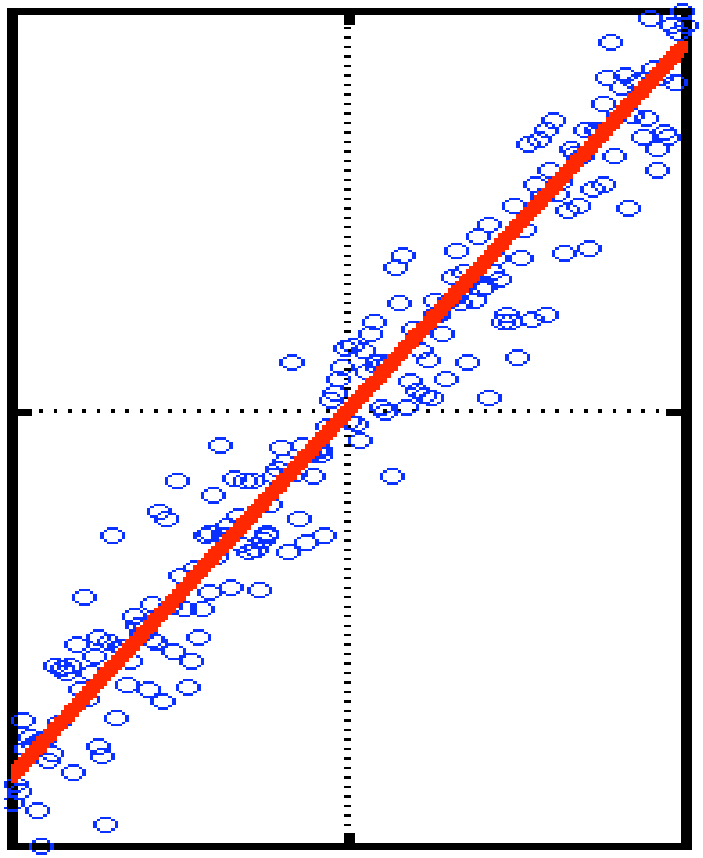


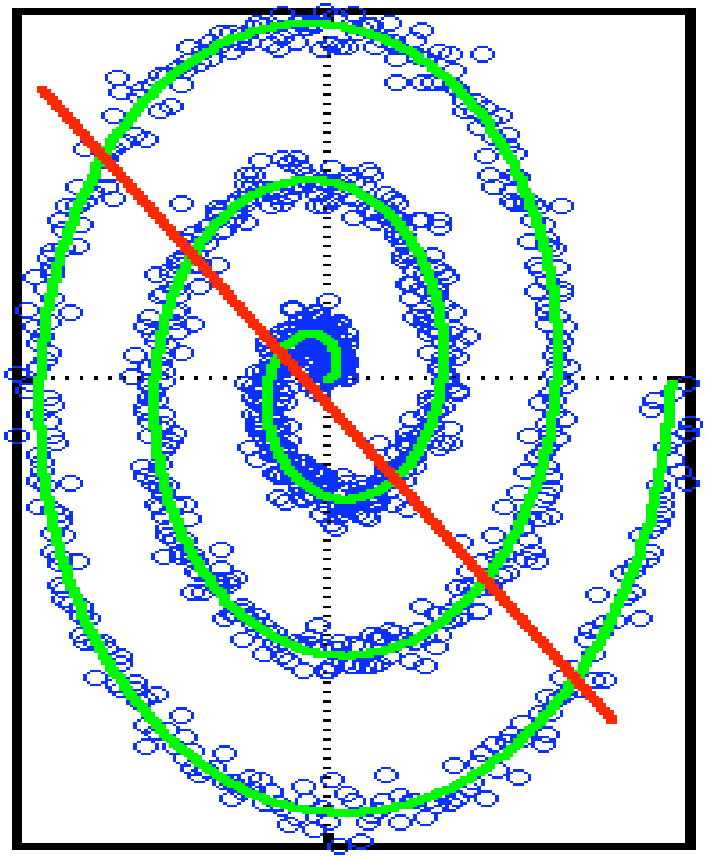
Glass data

- 9D glass identification database
- PCA always projects close-by points close to each other, resulting to reasonable recall
- However, PCA (and MDS) may also “collapse” far away data points into the same location (unless the data lies within low-dimensional linear subspace of the original space), this may lead to not so good precision

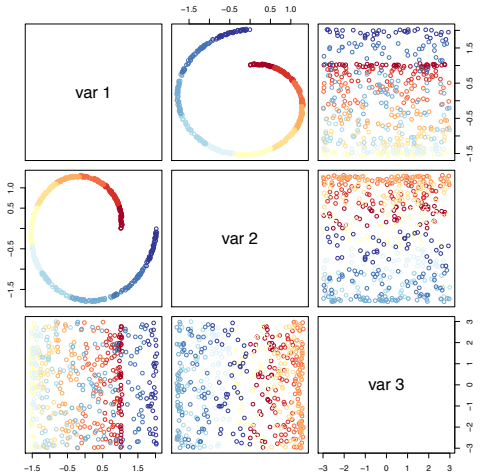


Next: visualising manifolds



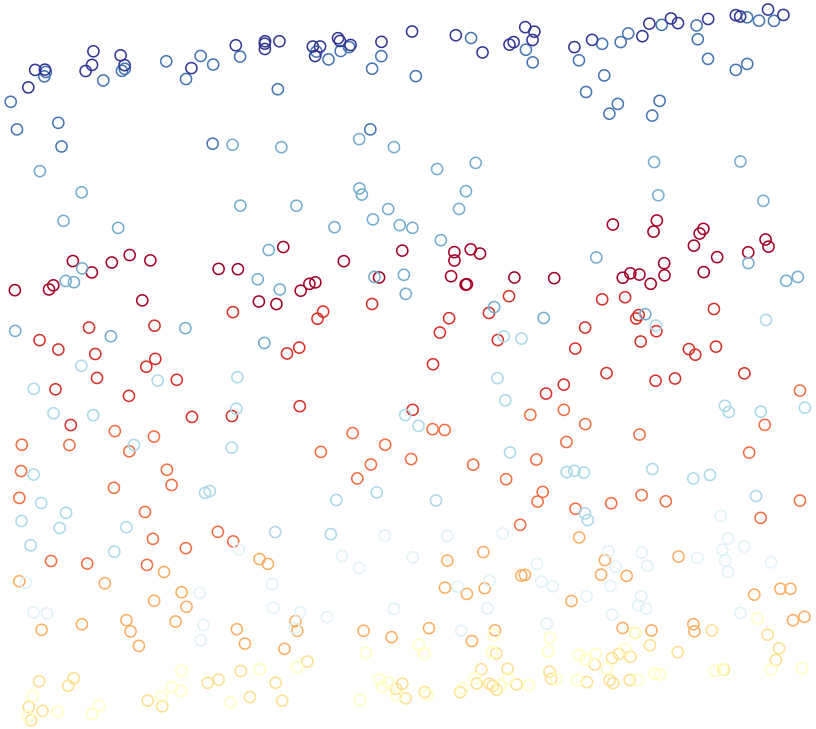


- The first principal component is given by the red line. The green line on the right gives the “correct” non-linear dimension (which PCA is of course unable to find).

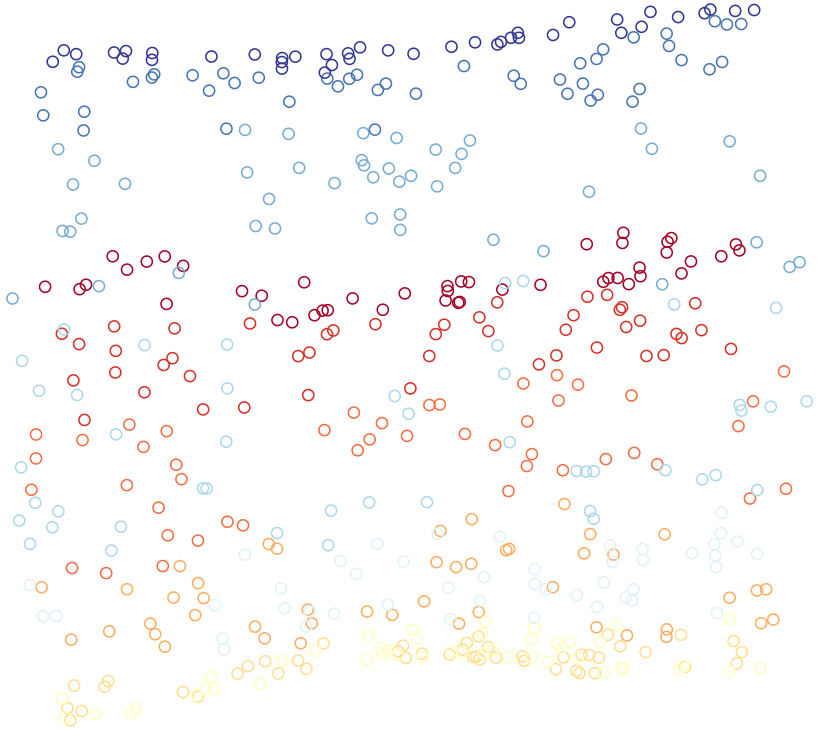


Swiss roll

PCA



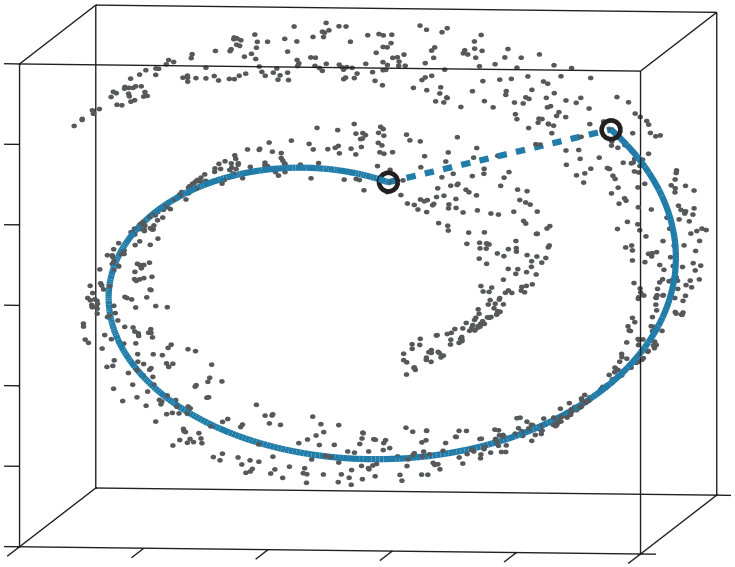
nonmetric MDS



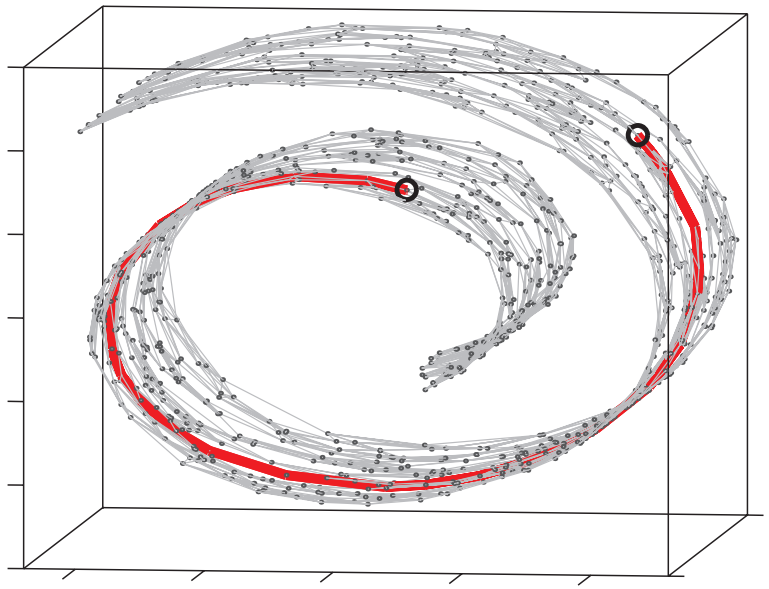
Isometric mapping of data manifolds (ISOMAP)

- Tenenbaum et al. 2000, https://doi.org/10.1126/science.290.5500.2319 See http://web.mit.edu/cocosci/isomap/datasets.html (fig)
- ISOMAP is an example of graph-based methods.
- ISOMAP is a variant of MDS. The difference to MDS is in how the distances (or proximities) are defined.
- ISOMAP first finds k nearest neighbours for each data point and constructs a k-nearest-neighbours graph. The distance between two data points (that are not nearest neighbours) is defined as the topological a.k.a. **graph-theoretical distance** (shortest path, i.e. minimum number of links) between the points.
- The resulting distances are fed to the standard linear (metric, because triangle inequality is satisfied) MDS, which finds the actual embedding.

Original data. The graph-distance between two items is shown by solid line, a shortcut is shown by the dotted line.

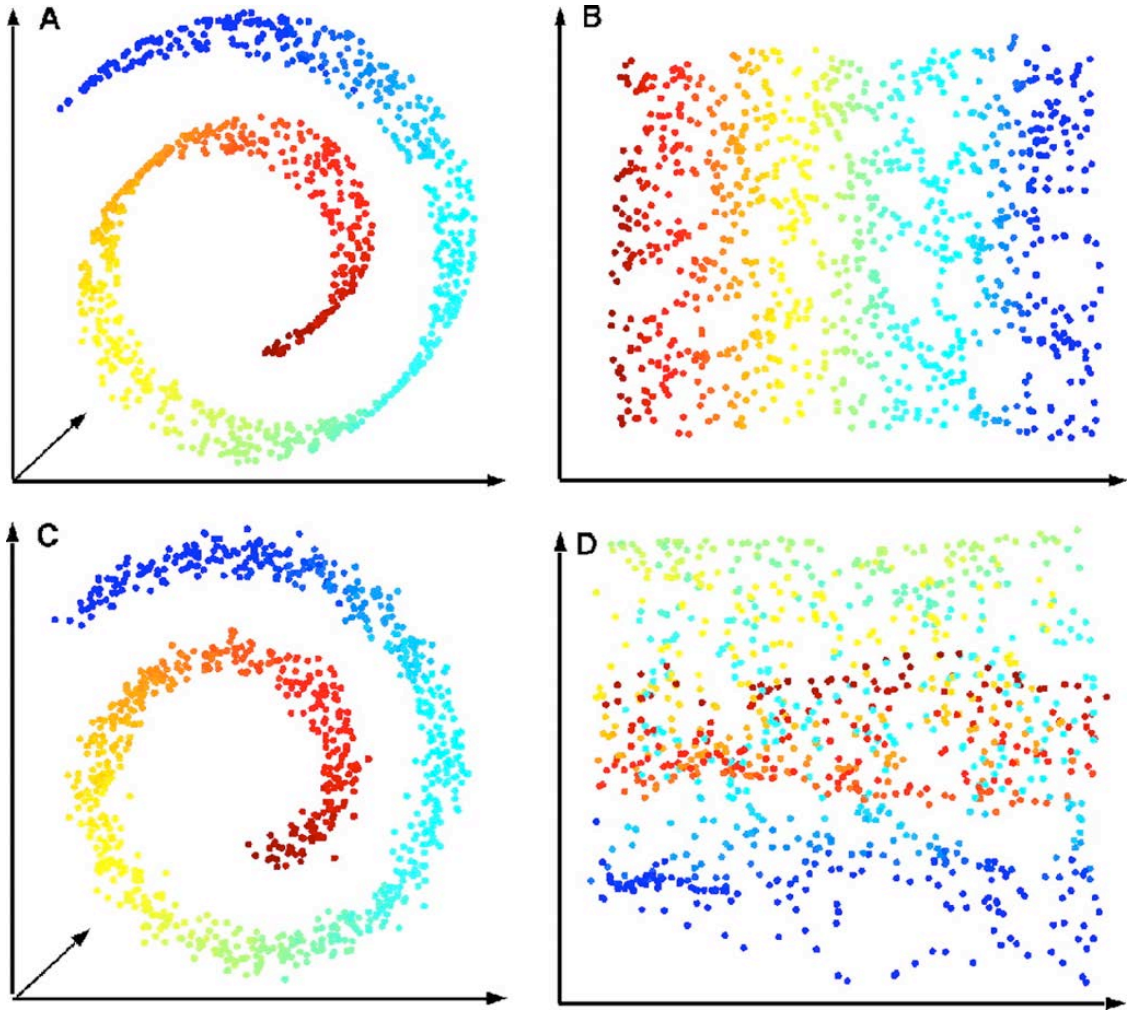


k-nearest neighbours graph used to find the graph-theoretical distances.



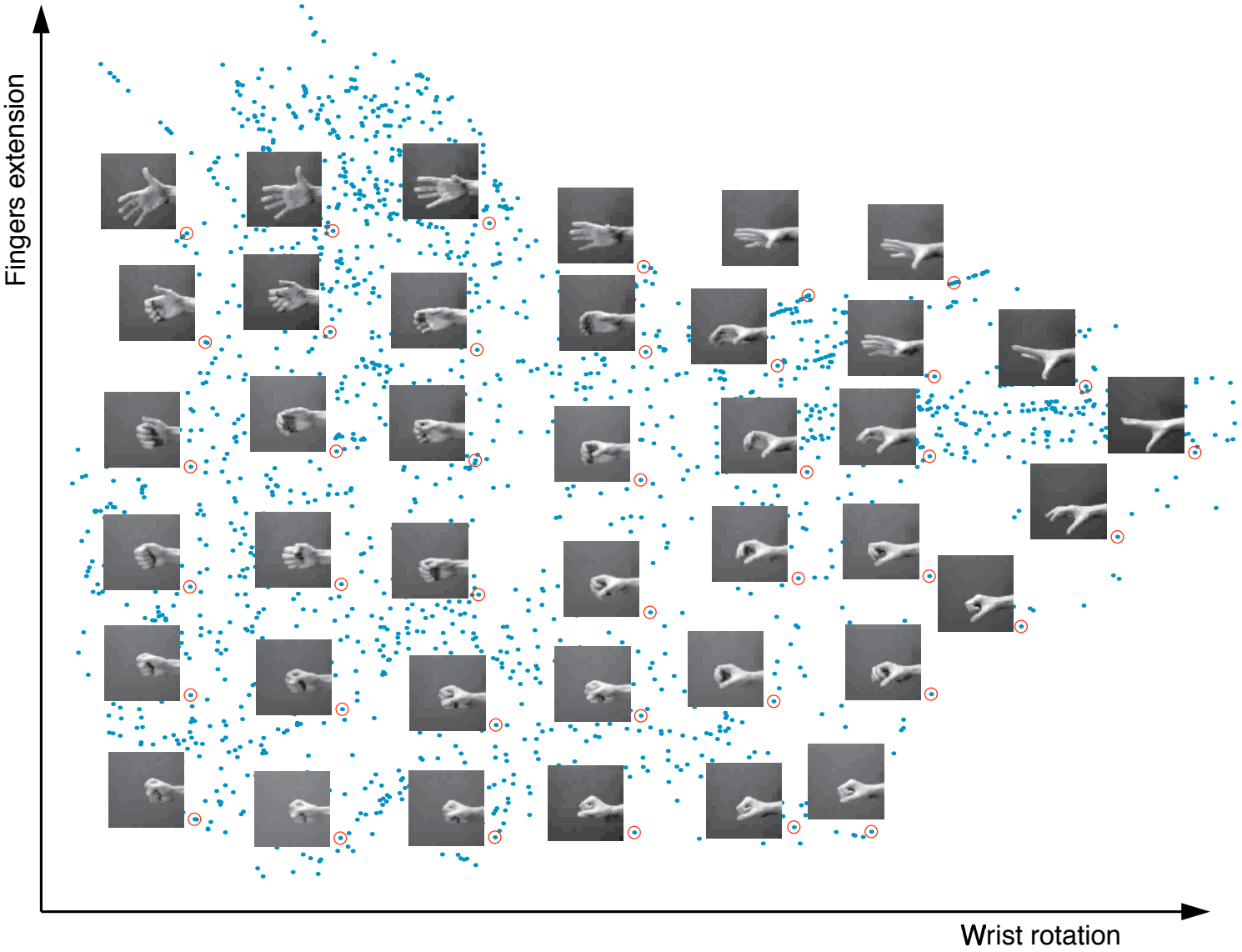
Isometric mapping of data manifolds (ISOMAP)

- Assumptions:
  - graph is connected
  - neighbourhood on graph reflects neighbourhoods on manifolds (no “shortcuts”)
- Weakness (Balasubramian et al. 2002, https://doi.org/10.1126/science.295.5552.7a, fig): sensitive to shortcuts (making the algorithm topologically unstable, see the figure right)
- Time complexity ~$O(N^2)$
- Extension: landmark ISOMAP (identify subsets of inputs as landmarks, makes the algorithm faster)



(A) The “Swiss roll” data used by Tenenbaum et al. (1) to illustrate their algorithm (n = 1000). (B) The two-dimensional (2D) representation computed by the ε−Isomap variant of the Isomap algorithm, with ε = 5. Nearby points in the 2D embedding are also nearby points in the 3D manifold, as desired. (C) Data shown in A, with zero-mean normally distributed noise added to the coordinates of each point, where the standard deviation of the noise was chosen to be 2% of smallest dimension of the bounding box enclosing the data. (D) The Isomap (ε = 5) solution for the noisy data. There are gross “folds” in the embedding, and neither the metric nor the topological structure of the solution in (B) is preserved.

Isometric mapping of data manifolds (ISOMAP)



ISOMAP (k=6) applied to 2,000 images of a hand in different configurations. The images were generated by making a series of opening and closing movements of the hand at different wrist orientations, designed to give rise to a two-dimensional manifold. The images were treated as 4,096-dimensional (= 64x64 pixels) vectors, with input-space distances defined in the Euclidean metric.

Locally linear embedding (LLE)

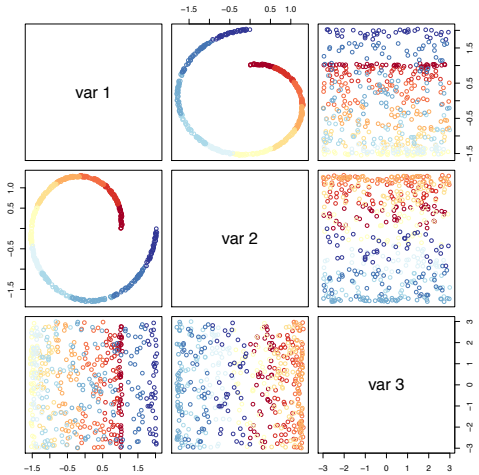
- LLE tries to maintain the relationships of nearby points
- Roweis et al. 2000, https://doi.org/10.1126/science.290.5500.2323

- Recipe:
  1. find the set *N(i)*, *k* closest data points to *i*th data point x_i
  2. try to express x_i as a linear combination of its neighbours: find weights minimising

$$\sum_i \left(x_i - \sum_{j \in N(i)} w_{ij} x_j \right)^2 \quad \text{s.t.} \quad \sum_{j \in N(i)} w_{ij} = 1$$

  3. fix the weights, and find points in plane minimising (y_i are the coordinates in embedding)

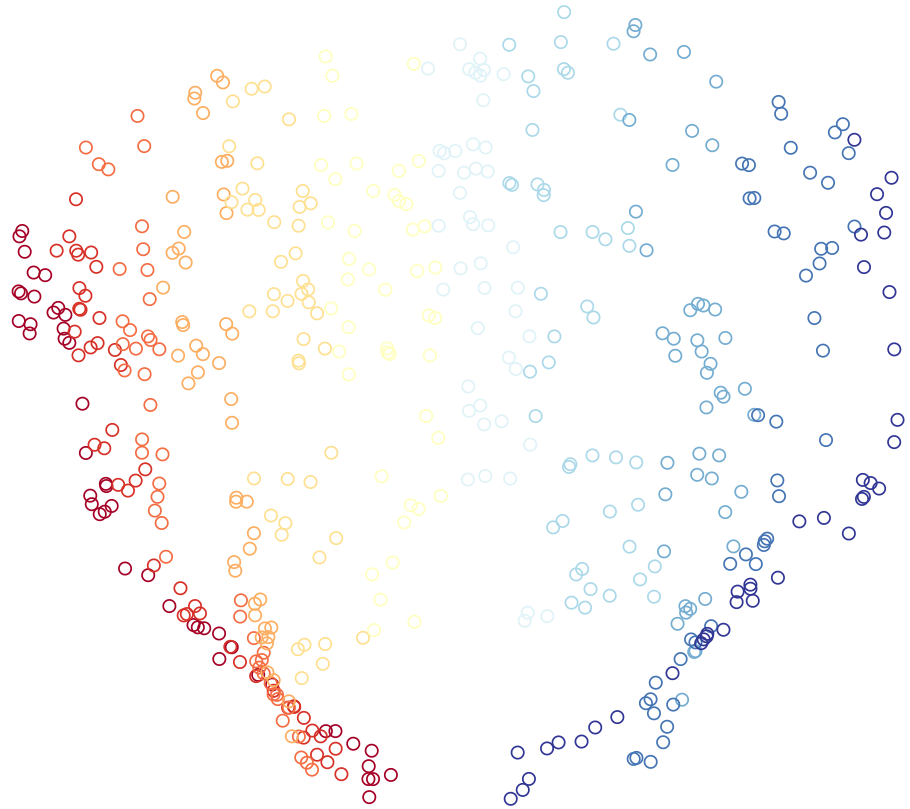
$$\sum_i \left(y_i - \sum_{j \in N(i)} w_{ij} y_j \right)^2$$



Swiss roll

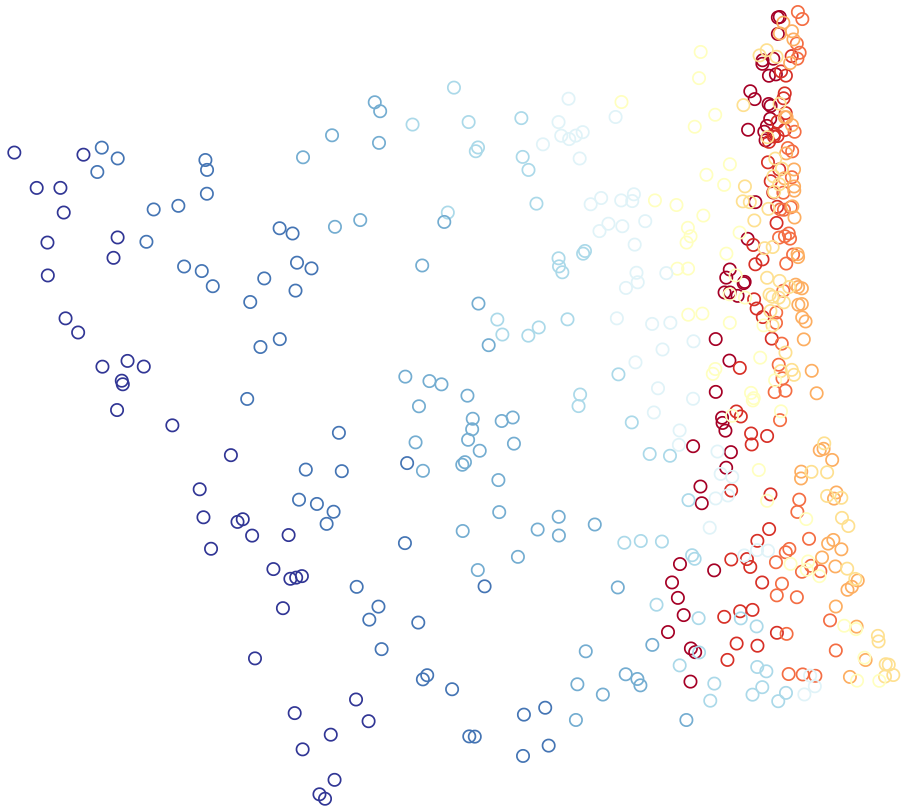
ISOMAP

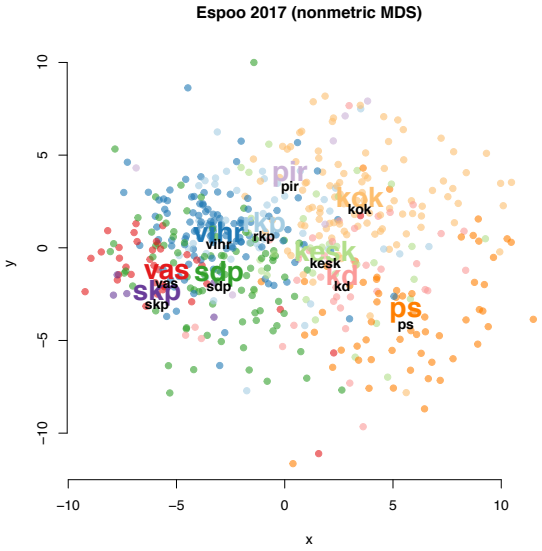
topological distances



LLE

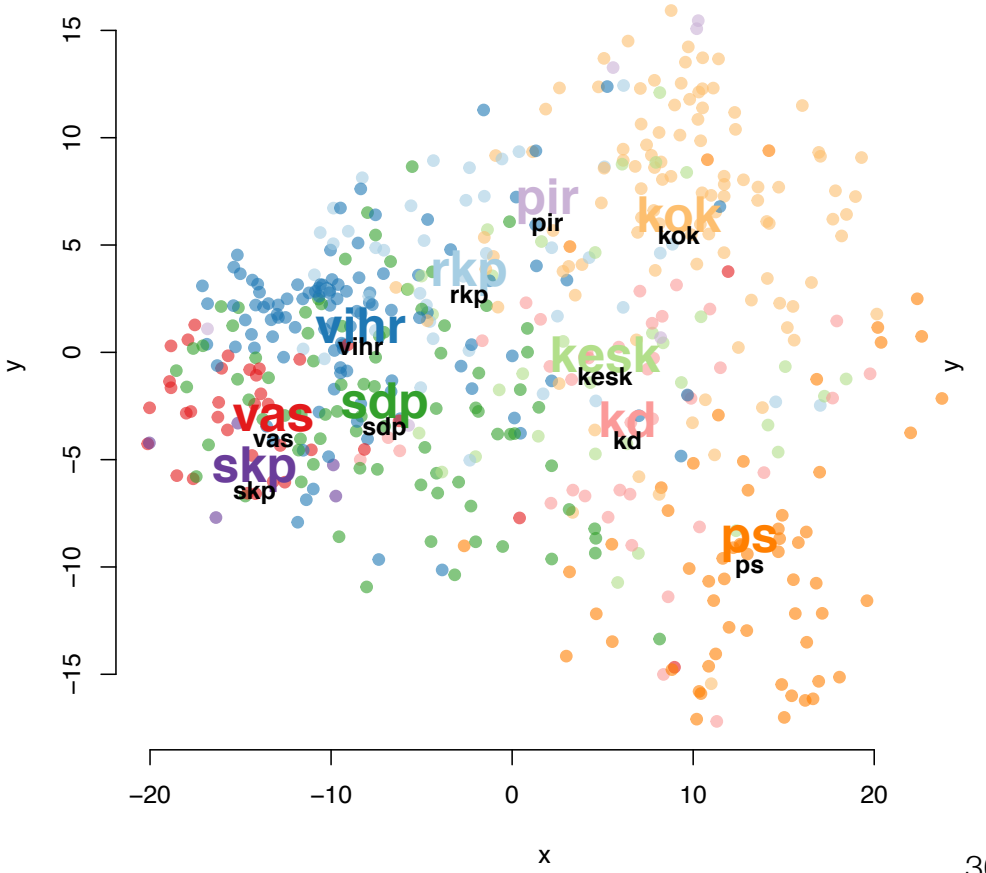
local metric distances



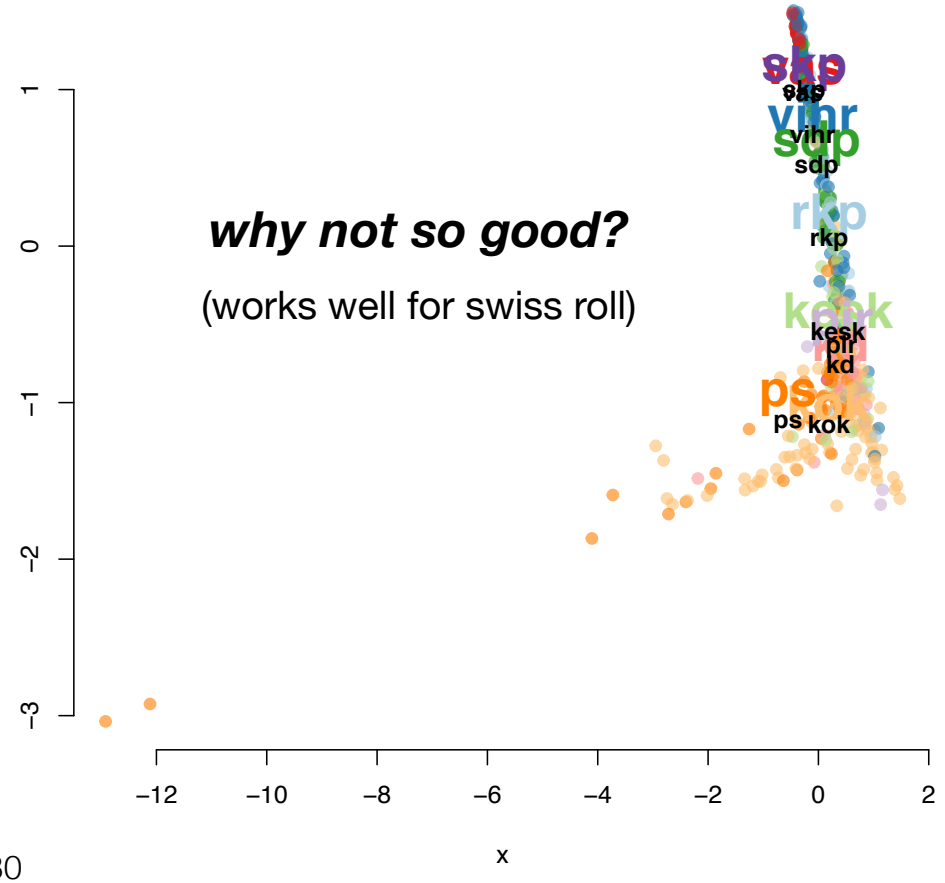


Municipal elections in Espoo in 2017

Espoo 2017 (ISOMAP)



Espoo 2017 (LLE)



why not so good?

(works well for swiss roll)

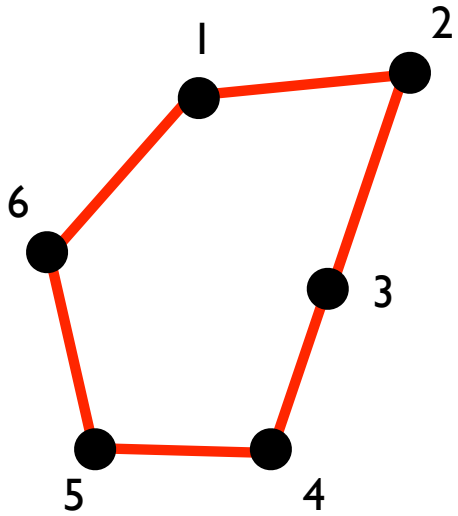
Laplacian eigenmap

- Eigenmap is a spectral method, like PCA.
- As in ISOMAP, construct *k*-nearest neighbors graph. Assign $W_{ij}=1$, if *i* and *j* are neighbours, otherwise assign $W_{ij}=0$. Define diagonal matrix *D*, $D_{ii}=\sum_j W_{ij}$, and graph Laplacian, $L=D-W$.
- The embedding of data points is given by the eigenvectors of *L*, corresponding to the *d* smallest non-zero eigenvalues.
- Physical intuition: find lowest frequency vibrational modes of a mass-spring system (mass=nodes, springs=links of the graph).
- Very straightforward to implement, e.g., with R

Laplacian eigenmap

N=6

k=2




```
> L
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   -2    1    0    0    0    1
[2,]    1   -2    1    0    0    0
[3,]    0    1   -2    1    0    0
[4,]    0    0    1   -2    1    0
[5,]    0    0    0    1   -2    1
[6,]    1    0    0    0    1   -2
> s <- svd(L)
> s$u
```

eigenvectors

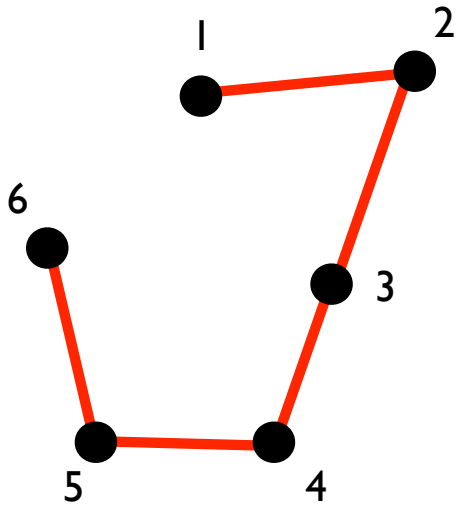
```
           [,1]          [,2]       [,3]       [,4]          [,5]       [,6]
[1,] -0.4082483 -1.934666e-16 -0.5773503 -0.5773503 -3.951502e-16  0.4082483
[2,]  0.4082483  5.000000e-01  0.2886751 -0.2886751 -5.000000e-01  0.4082483
[3,] -0.4082483 -5.000000e-01  0.2886751  0.2886751 -5.000000e-01  0.4082483
[4,]  0.4082483  4.163336e-16 -0.5773503  0.5773503  2.081668e-16  0.4082483
[5,] -0.4082483  5.000000e-01  0.2886751  0.2886751  5.000000e-01  0.4082483
[6,]  0.4082483 -5.000000e-01  0.2886751 -0.2886751  5.000000e-01  0.4082483
> s$d
[1] 4.000000e+00 3.000000e+00 3.000000e+00 1.000000e+00 1.000000e+00 1.155603e-16
```

zero eigenvalue

Laplacian eigenmap

N=6

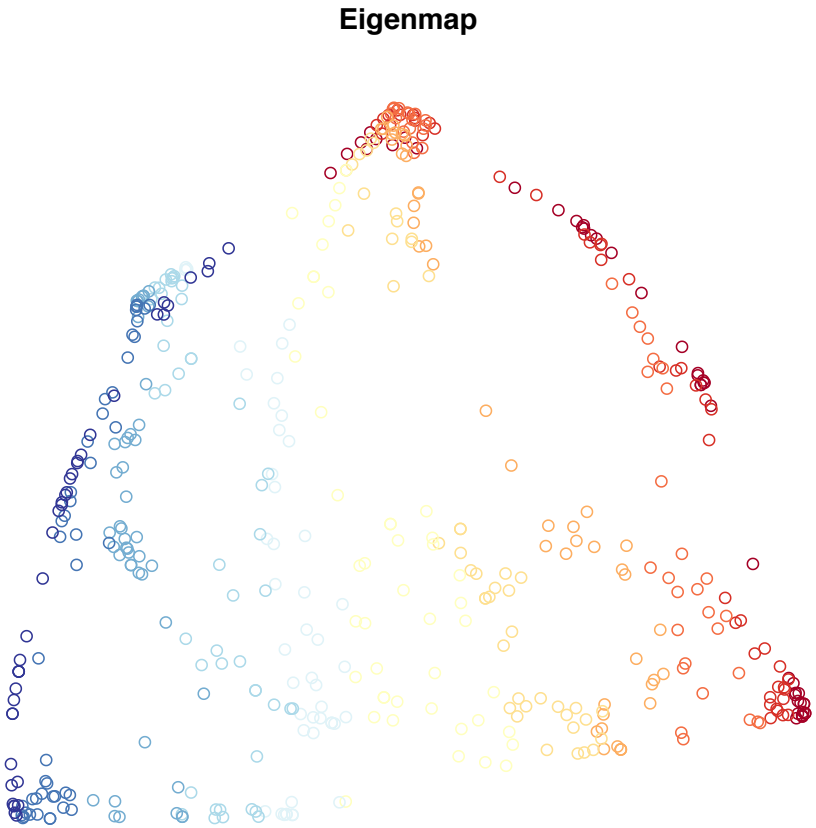
k=1

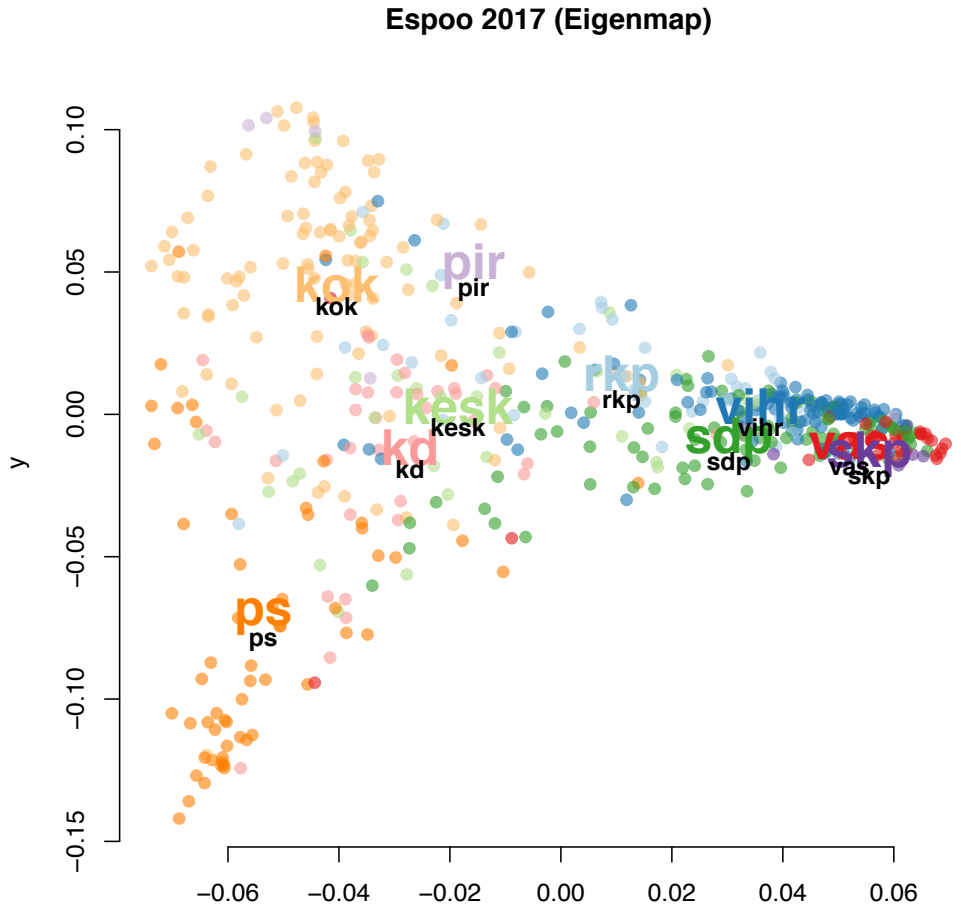



```
> L
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   -1    1    0    0    0    0
[2,]    1   -2    1    0    0    0
[3,]    0    1   -2    1    0    0
[4,]    0    0    1   -2    1    0
[5,]    0    0    0    1   -2    1
[6,]    0    0    0    0    1   -1
> s <- svd(L)
> s$u
           [,1]       [,2]       [,3]          [,4]       [,5]      [,6]
[1,] -0.1494292 -0.2886751 -0.4082483  5.000000e-01  0.5576775 0.4082483
[2,]  0.4082483  0.5773503  0.4082483  2.775558e-16  0.4082483 0.4082483
[3,] -0.5576775 -0.2886751  0.4082483 -5.000000e-01  0.1494292 0.4082483
[4,]  0.5576775 -0.2886751 -0.4082483 -5.000000e-01 -0.1494292 0.4082483
[5,] -0.4082483  0.5773503 -0.4082483  8.326673e-17 -0.4082483 0.4082483
[6,]  0.1494292 -0.2886751  0.4082483  5.000000e-01 -0.5576775 0.4082483
> s$d
[1] 3.732051e+00 3.000000e+00 2.000000e+00 1.000000e+00 2.679492e-01 7.510881e-17
```

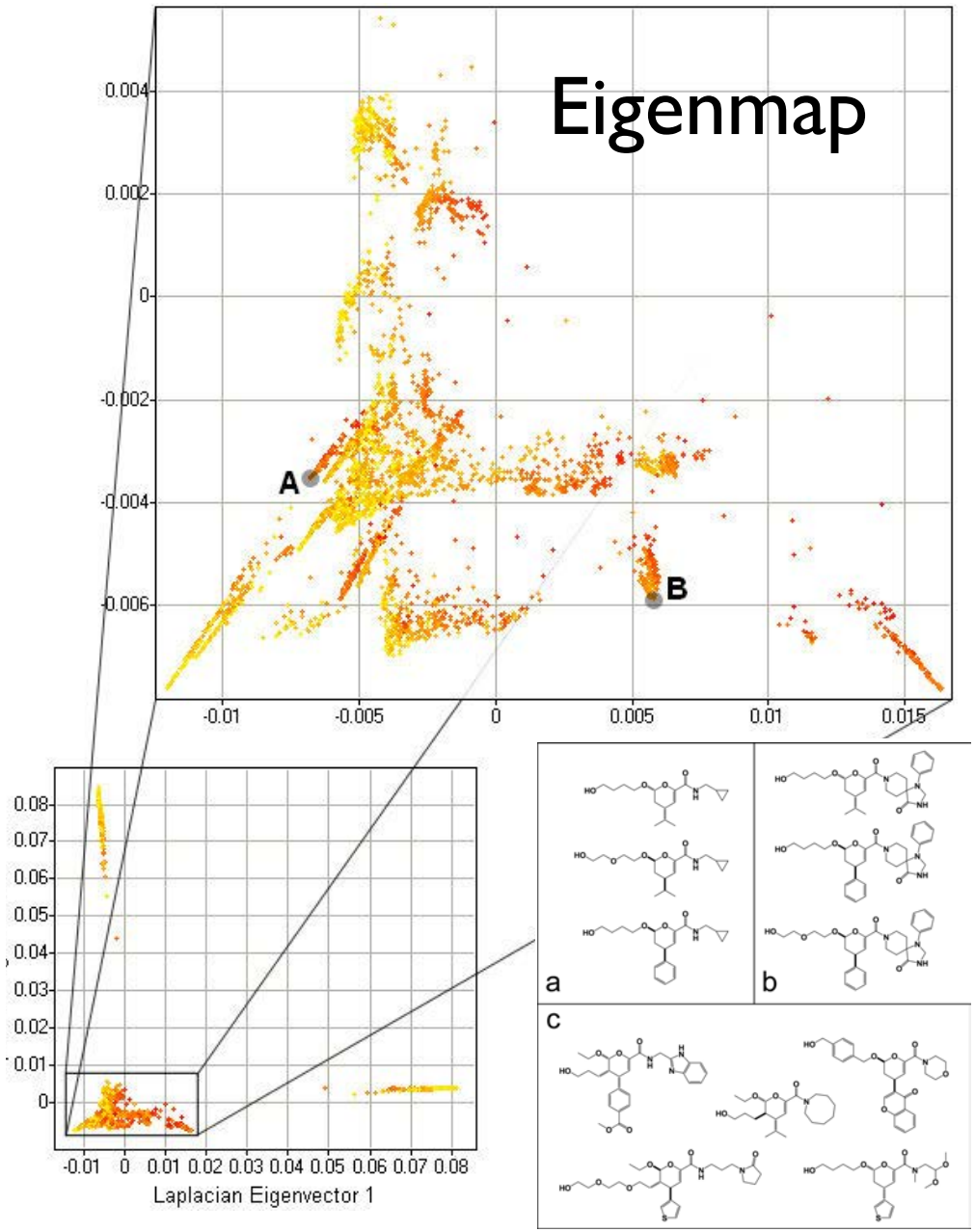
Laplacian eigenmap

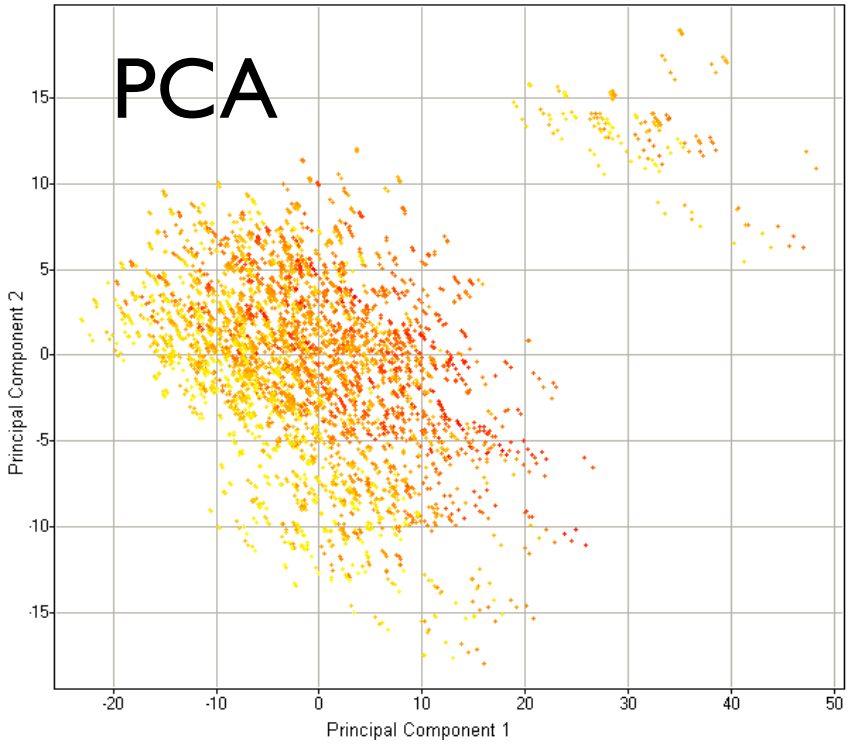
- Eigenmap can be viewed as trying to preserve the expected time a random walk on the neighbourhood graph takes to travel from one point to the other and back. This leads to tendency to magnify some distances (and shrink others), leading to relatively bad precision.





Laplacian eigenmap





Eigenmap (unlike PCA) shows clusters of similar chemical compounds (A&B). The input data is a network of small molecules encoded as molecular descriptors and connected by similarity.

Forman et al. 2005, https://doi.org/10.1186/1471-2105-6-260

Curvilinear component analysis (CCA)

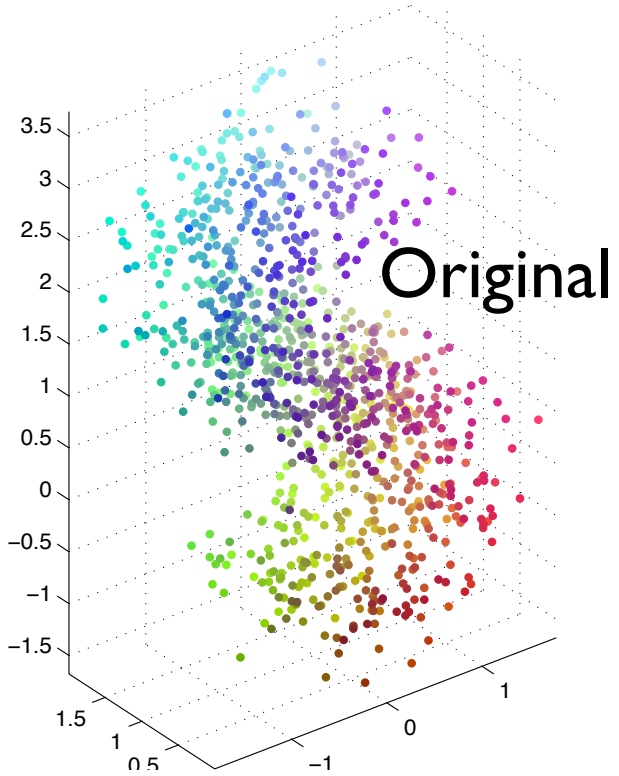
- Demartines et al. 1997, https://doi.org/10.1109/72.554199
- *Curvilinear component analysis* (CCA) is like (absolute) MDS, except that only short distances are taken into account.
- More formally, the cost function reads

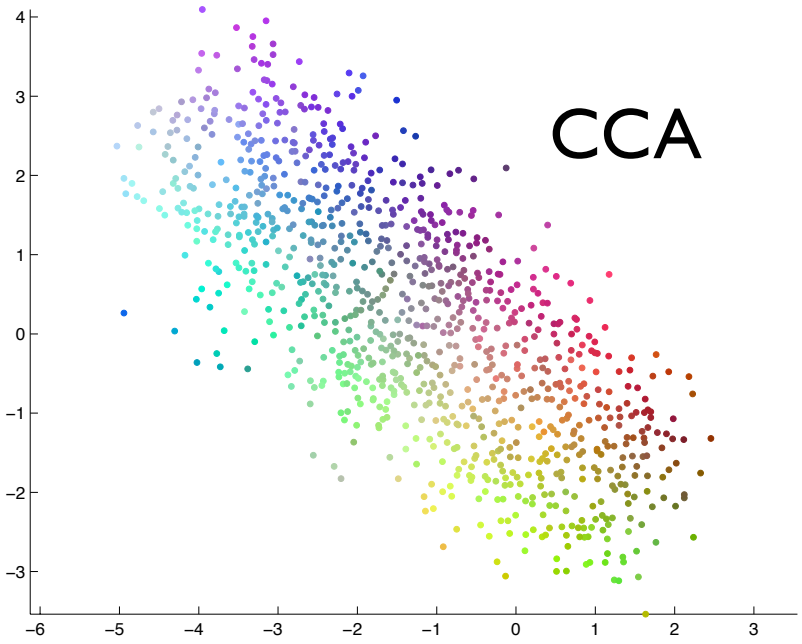
$$\sigma_r = \sum_{i<j} \left(d(x_i, x_j) - d(y_i, y_j)\right)^2 F(d(y_i, y_j), \lambda_y)$$

where $F(d,\lambda_y)$ equals unity, if $d<\lambda_y$, and zero otherwise; and d denotes the Euclidean distance of points in the original space (x) and in the projection (y), respectively. ($F(d,\lambda_y)$, could be any monotonically decreasing function in d.)

Curvilinear component analysis (CCA)

- CCA performs generally well in terms of precision; it appears to be quite robust.
- Notice outliers at right: they are result of small neighbourhood.





Figures by Jarkko Venna

Local multidimensional scaling

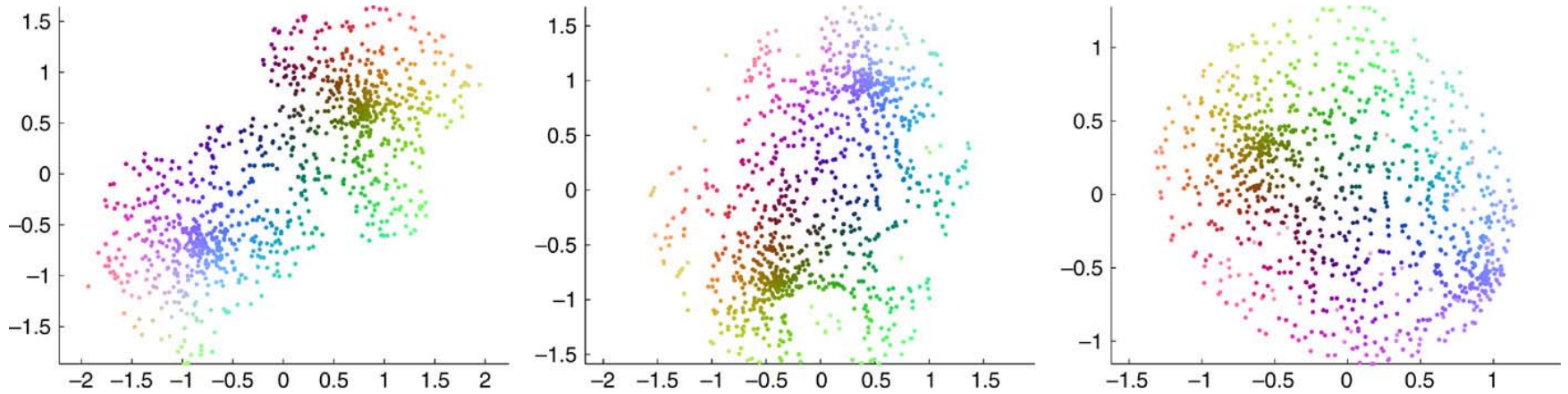


Fig. 6. Three projections of a three-dimensional spherical cell with local MDS. On the left, trustworthiness of the projection is maximized by selecting $\lambda = 0$. In the middle and right, discontinuity of the projection is penalized as well, by setting $\lambda = 0.1$ and $\lambda = 0.9$, respectively.

- Extension of curvilinear component analysis (CCA obtained when λ=0)
- Parameter λ controls the tradeoff between precision and recall
- Venna et al. 2006, https://doi.org/10.1016/j.neunet.2006.05.014

$$
\begin{aligned}
E = & \frac{1}{2}\sum_i \sum_{j \neq i} [(1-\lambda)(d(\mathbf{x}_i, \mathbf{x}_j) \\
& - d(\mathbf{y}_i, \mathbf{y}_j))^2 F(d(\mathbf{y}_i, \mathbf{y}_j), \sigma_i) \\
& + \lambda (d(\mathbf{x}_i, \mathbf{x}_j) - d(\mathbf{y}_i, \mathbf{y}_j))^2 F(d(\mathbf{x}_i, \mathbf{x}_j), \sigma_i)] \\
= & \frac{1}{2}\sum_i \sum_{j \neq i} (d(\mathbf{x}_i, \mathbf{x}_j) - d(\mathbf{y}_i, \mathbf{y}_j))^2 \\
& \times [(1-\lambda) F(d(\mathbf{y}_i, \mathbf{y}_j), \sigma_i) + \lambda F(d(\mathbf{x}_i, \mathbf{x}_j), \sigma_i)].
\end{aligned}
$$

trustworthiness = precision, continuity = recall

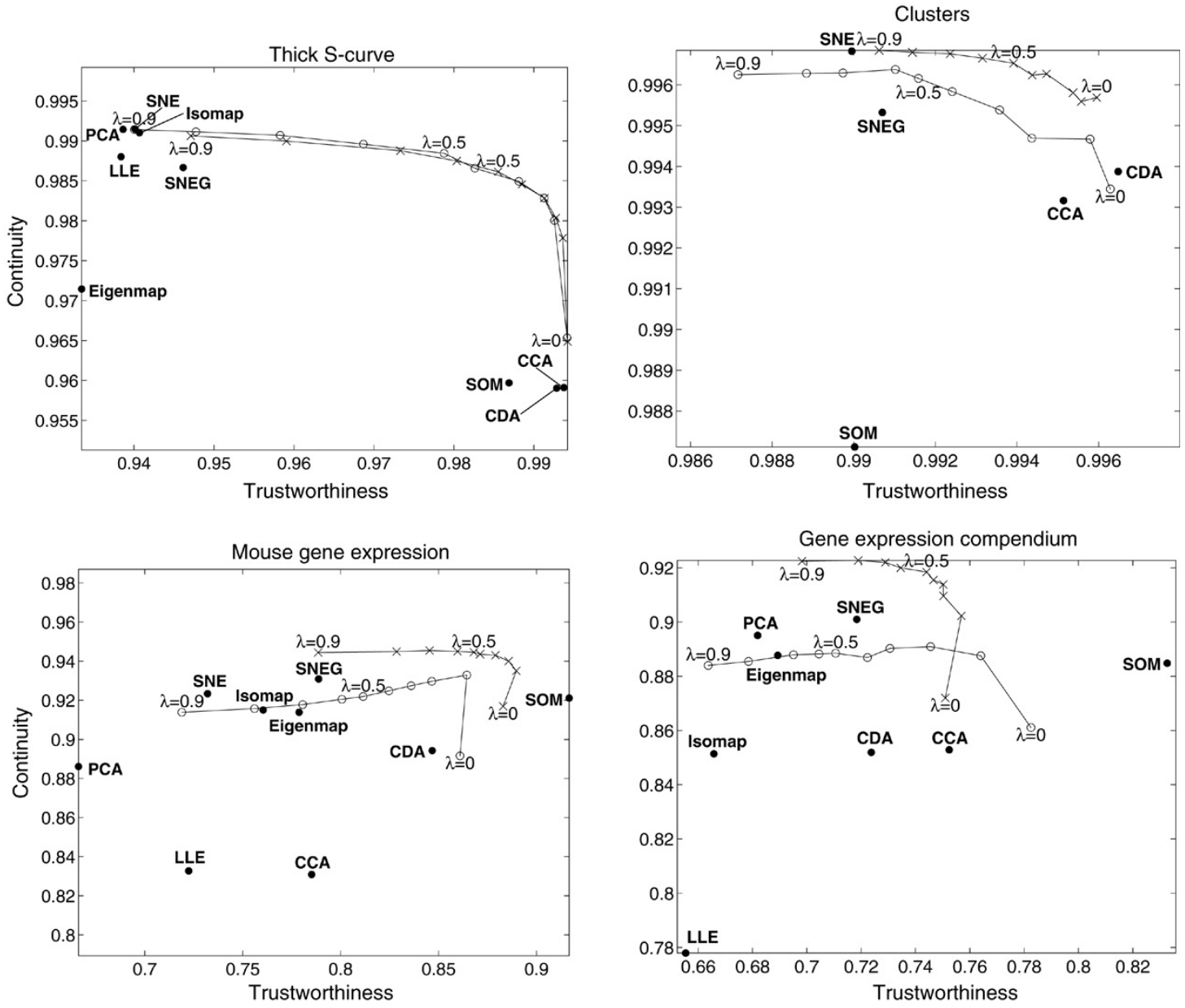
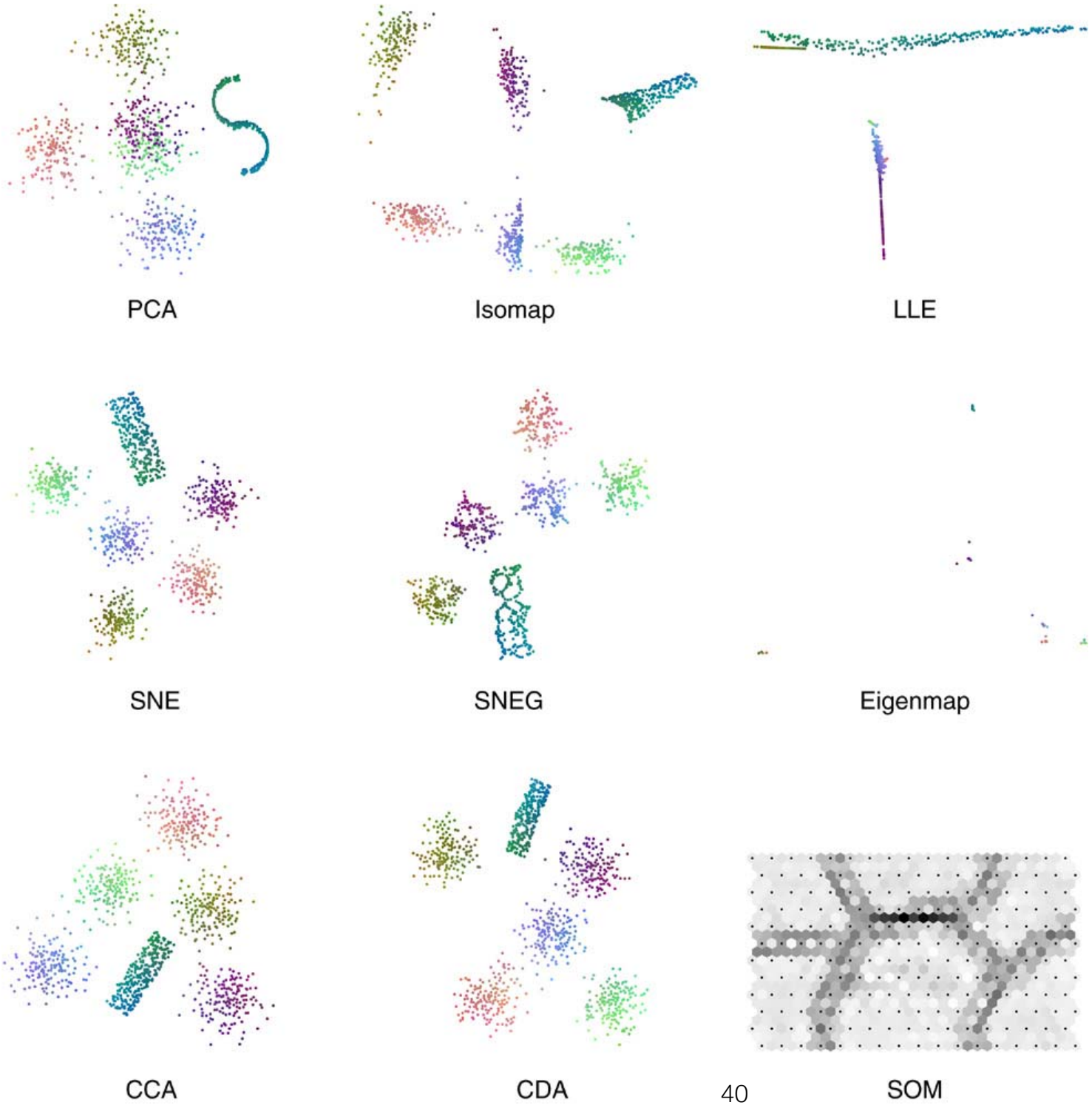


Fig. 4. The relationship between trustworthiness and continuity of the mapping as a function of λ, for a neighborhood of size 10. *Line with open circles*: local MDS, *line with* ×s: local MDS with geodesic distances. Other methods are included (black dots with a name attached) for reference. Methods not shown are too far down or to the left to fit in the image.

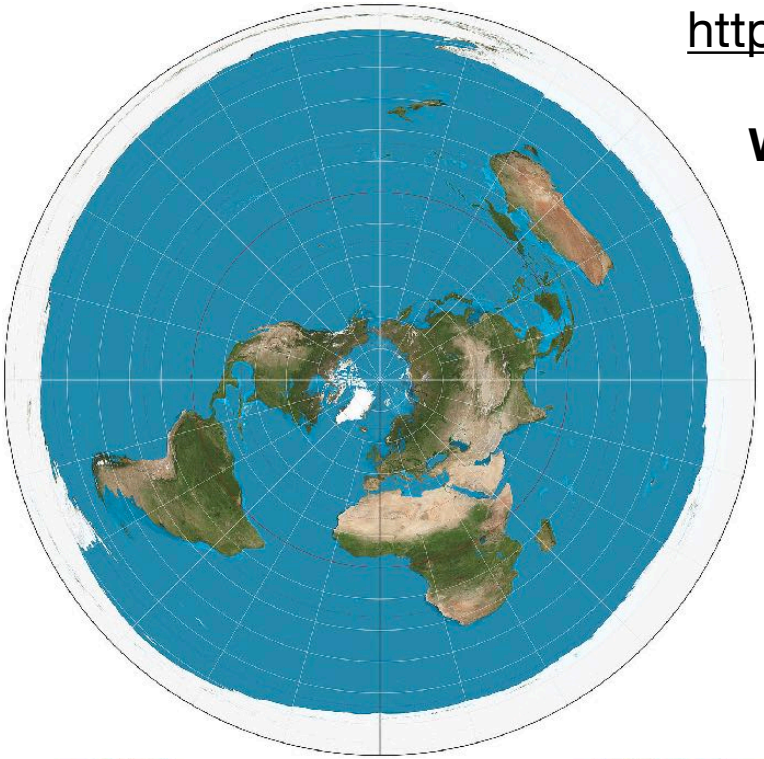
trustworthiness = precision, continuity = recall



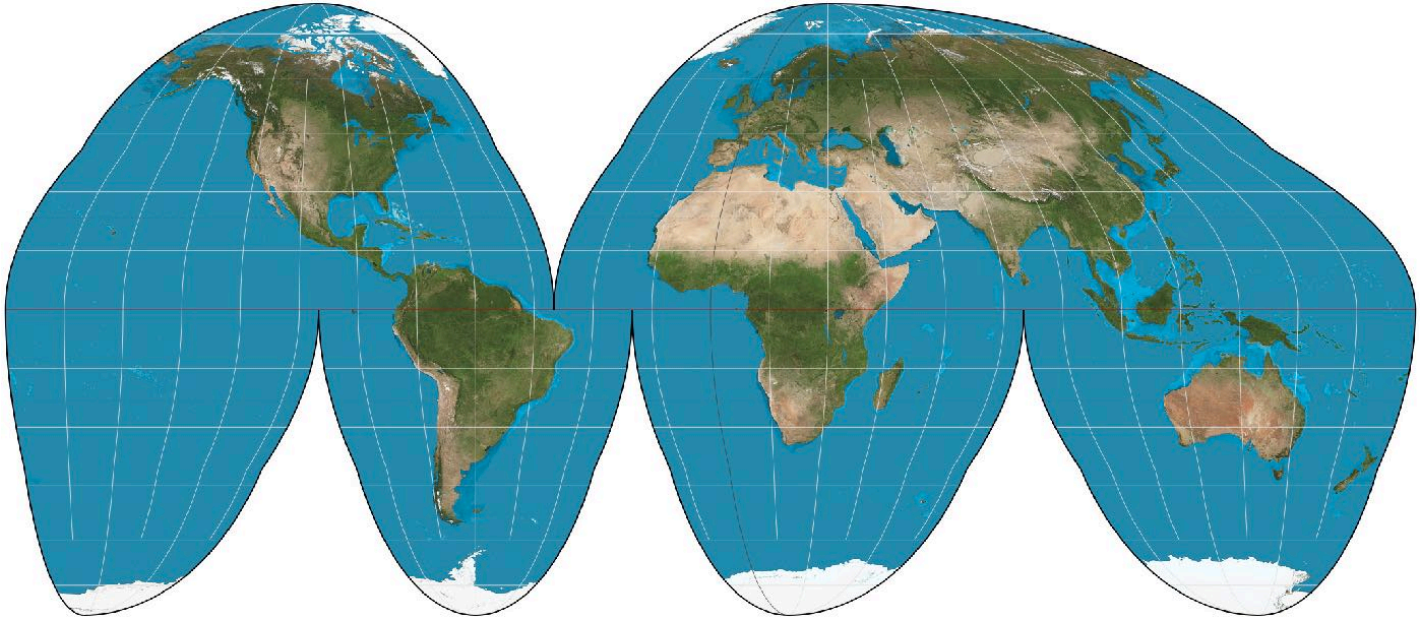
Venna et al. 2006, https://doi.org/10.1016/j.neunet.2006.05.014

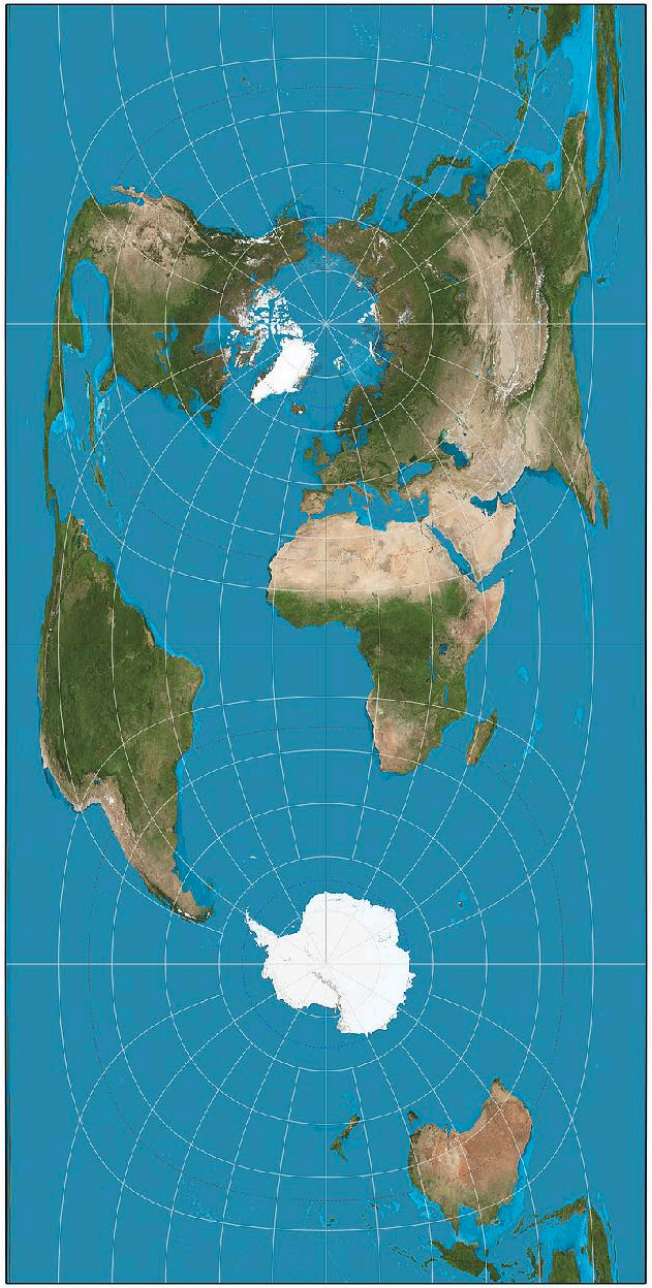
Which is the right world map?

https://en.wikipedia.org/wiki/List_of_map_projections



Which properties are important to retain in projection?





Recap

- PCA and MDS variants will struggle with non-linear manifolds
- PCA/Torgerson scaling is a linear projection
- large distances dominate the cost function in MDS methods

- techniques specifically designed to flatten manifolds
  - ISOMAP
  - LLE
  - Laplacian eigenmap
  - local multidimensional scaling
  - many more exist...
- either redefine the distance or look only at the vicinity of individual points
- practical issues: distortions, may be computationally expensive

Problem with lack of guidance

- The previous methods have one major problem: they produce an embedding given some (technical criteria). The result may or may not be what user wants.

- One way to tune the embedding is to add **guidance**: find embedding such that it maximises dependency with respect to some other variable
- Assume that in the original (high-dimensional) data consists of pairs of variables (x,y), where x is data variable and y is response variable (e.g., class).

- **Problem:** Find embedding X such that y depends mainly on X.

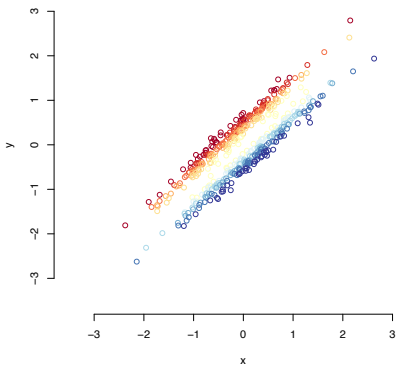
Supervised PCA

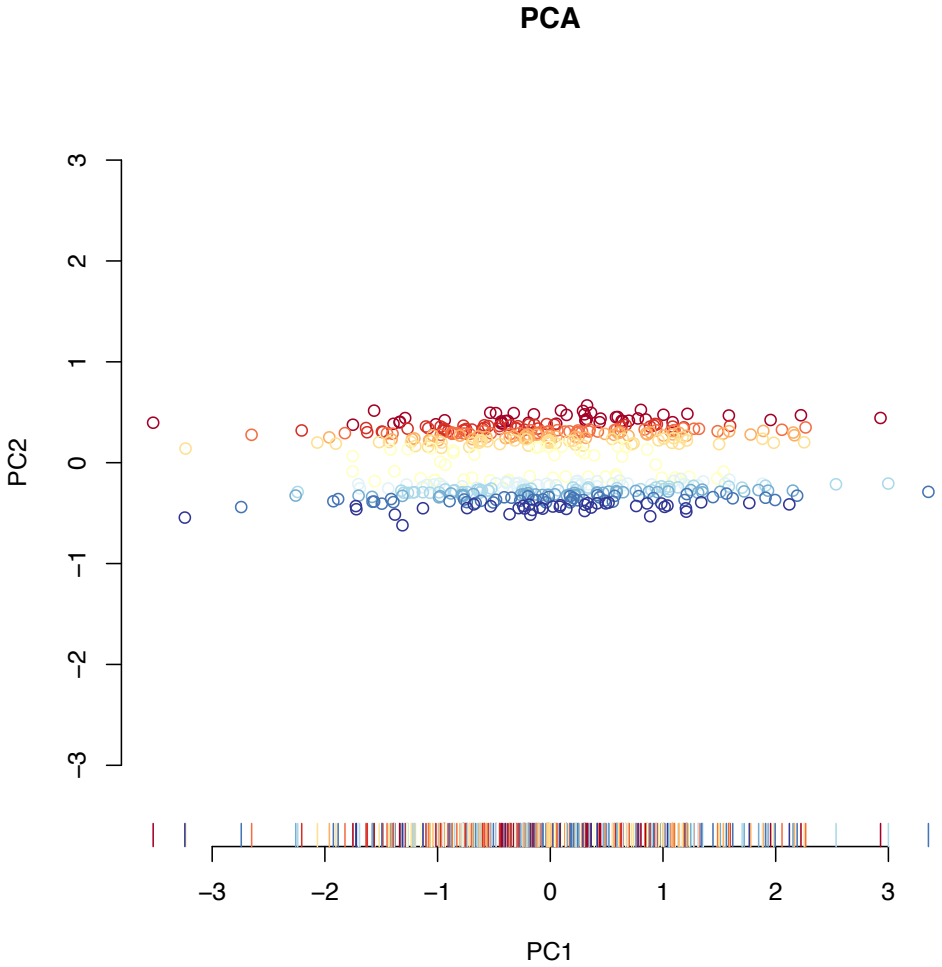
- At simplest,
  let X be $n \times m$ data matrix (with zero mean columns) and Y be $n \times m'$ matrix of response variables.
  - Use largest eigenvectors of $X^T Y Y^T X$ to project into lower dimensions
  - If $YY^T = \mathbf{1}$ this reduces to PCA

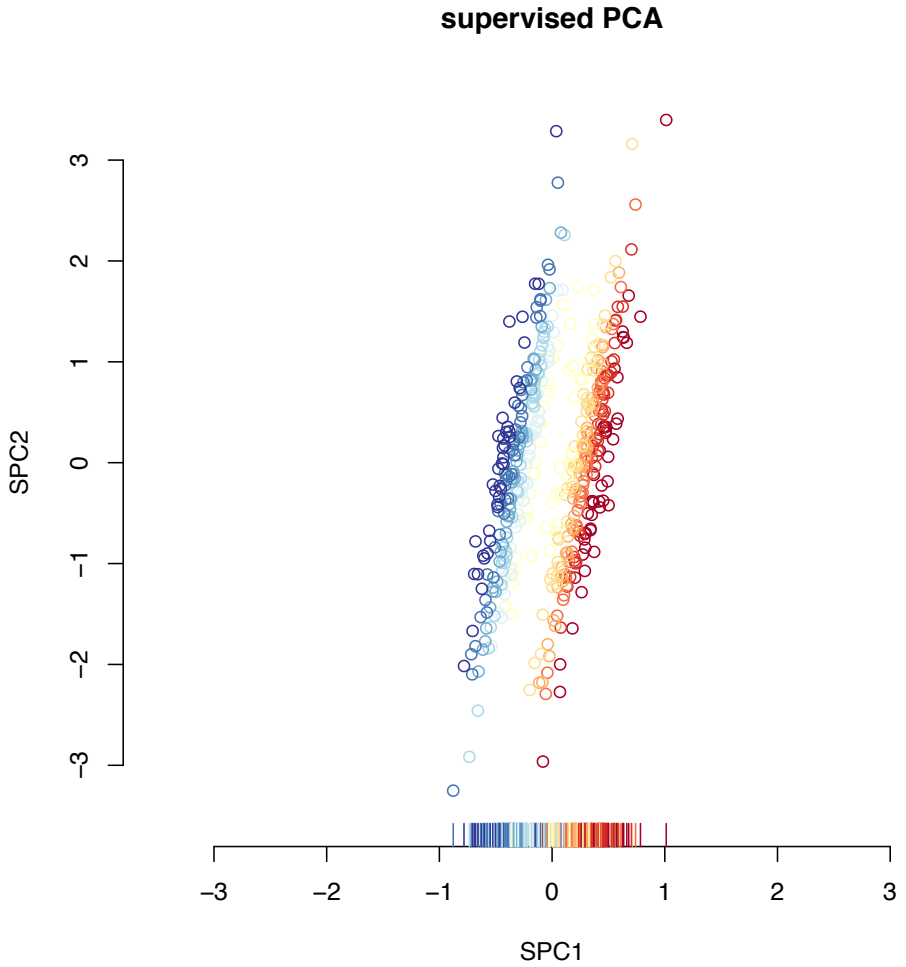
- For details and fancier variants see Barshan et al. 2011, https://doi.org/10.1016/j.patcog.2010.12.015

Supervised PCA

- Same 2-cluster data as before
- Y is $n \times 1$ matrix and $Y_{i1}=-1$ or 1 if i is in red or blue cluster, respectively
  (i.e. Y gives a classification of the data)

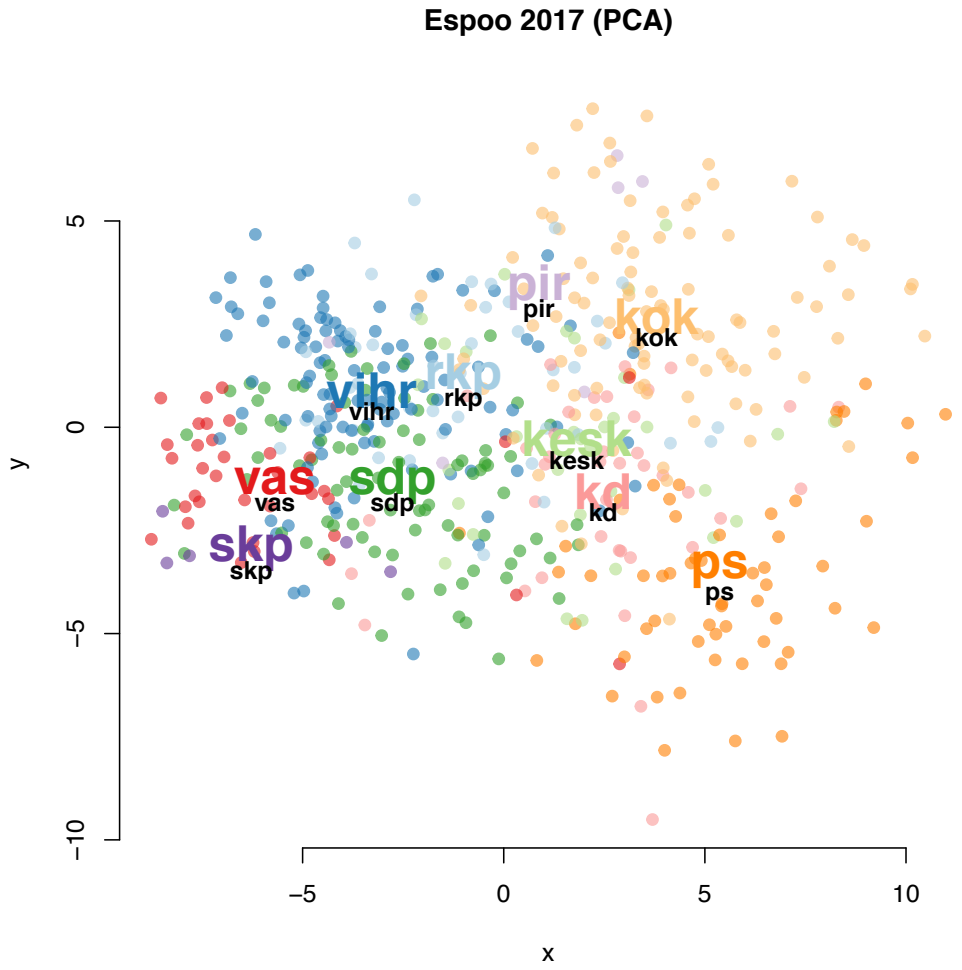


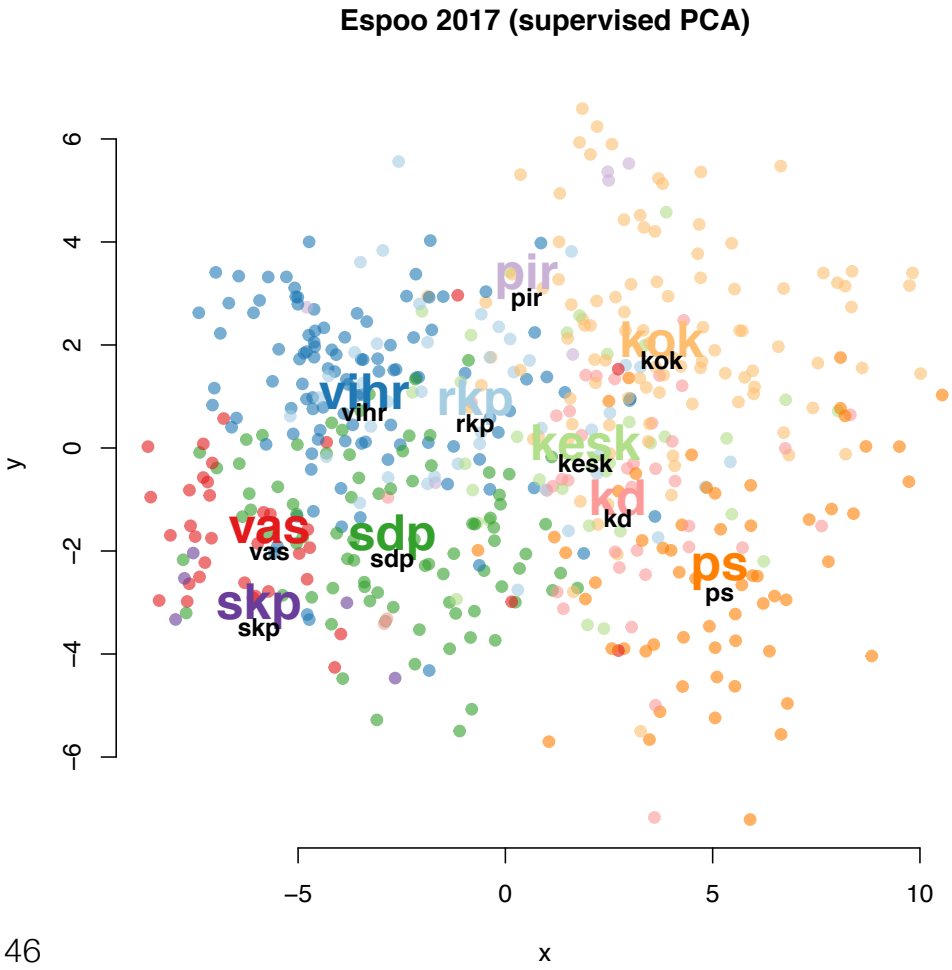




Supervised PCA

- Supervise PCA to **separate the following parties**: *vihr, rkp, sdp, vas*
- Y is 515 x 4 matrix where
  $Y_{i1} = 1$ if candidate i is in *virh*, $Y_{i2} = 1$ if candidate i is in *rkp*,
  $Y_{i3} = 1$ if candidate i is in *sdp*, $Y_{i4} = 1$ if candidate i is in *vas*,
  otherwise $Y_{ij} = 0$.





Guided Locally Linear Embedding

- It is possible to guide also other methods
- The principles in **guided** LLE (GLLE) are similar as for supervised PCA
- For details see Alipanahi et al. 2011, https://doi.org/10.1016/j.patrec.2011.02.002

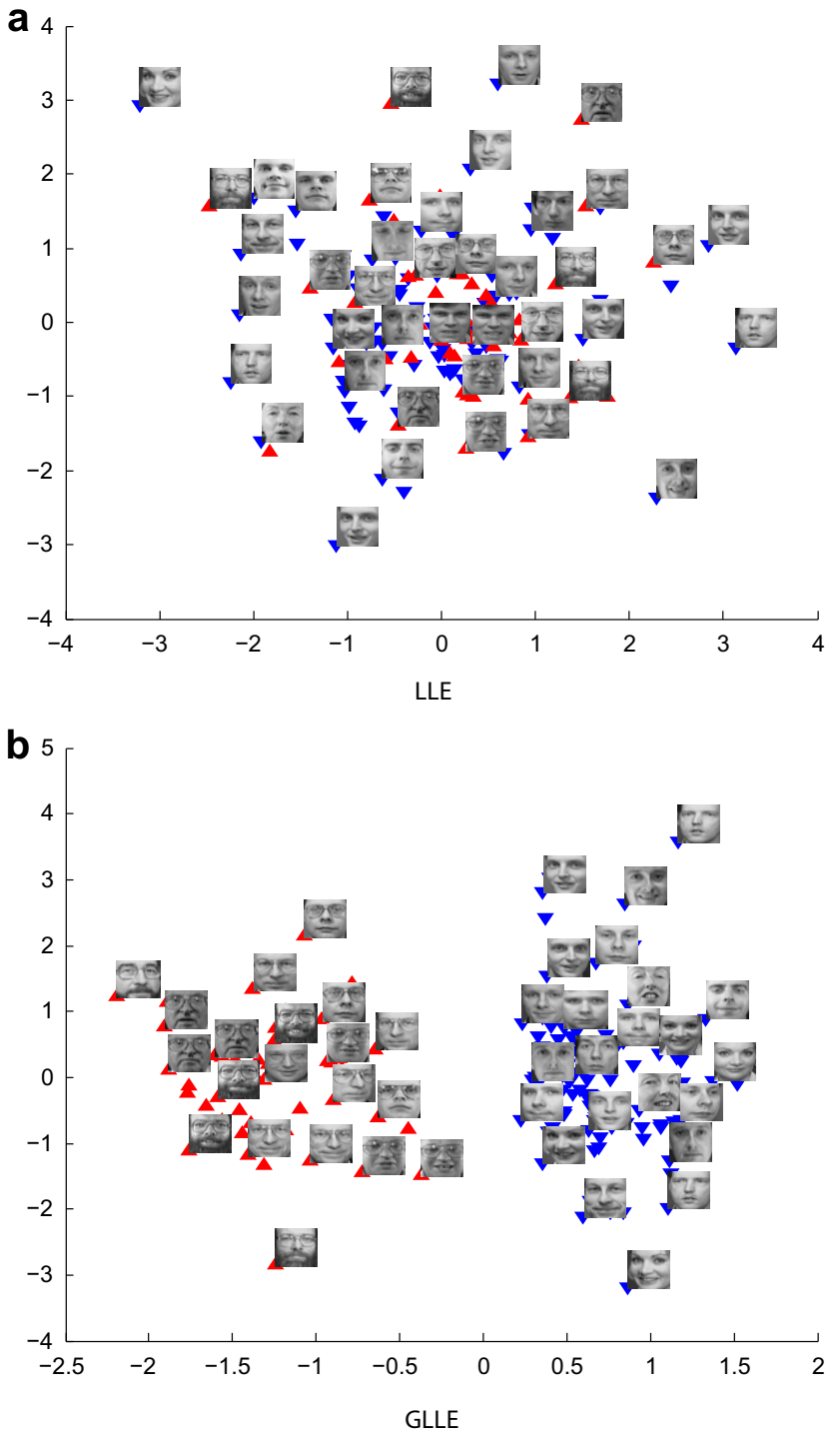


Fig. 2. Comparison of visualizations acquired by LLE and 0.5-GLLE ($k = 50$). There are two groups: persons with and without glasses.

Problem with lack of interaction

- "Controllability and interaction are two concepts that are mostly absent from dimensionality reduction." (Verleysen et al. 2013)

- The previous methods have one major problem: they produce an embedding given some (technical) criteria. The result may or may not be what user wants.
- ***New problem:*** How to create efficient interactions such that the user can in an understandable way modify the embedding? (e.g., by noticing cluster structures or outliers and asking to show something different, by must-link or cannot-link constraints etc.)
- **Visually controllable data mining.** Extension of Furnas' effective view navigation to the context of having automated analysis. Puolamäki et al. 2010, https://doi.org/10.1109/ICDMW.2010.141

Interactive knowledge-based kernel PCA

Paurat et al. 2013

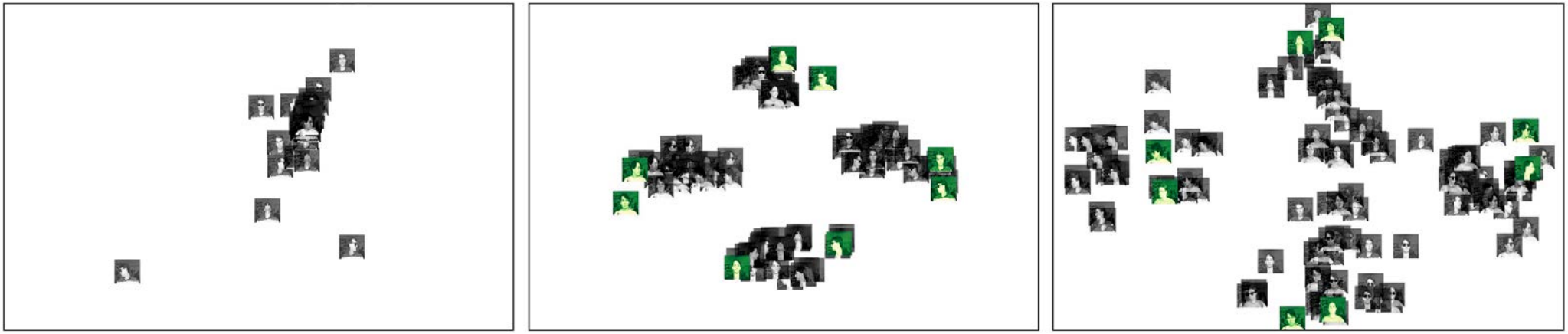
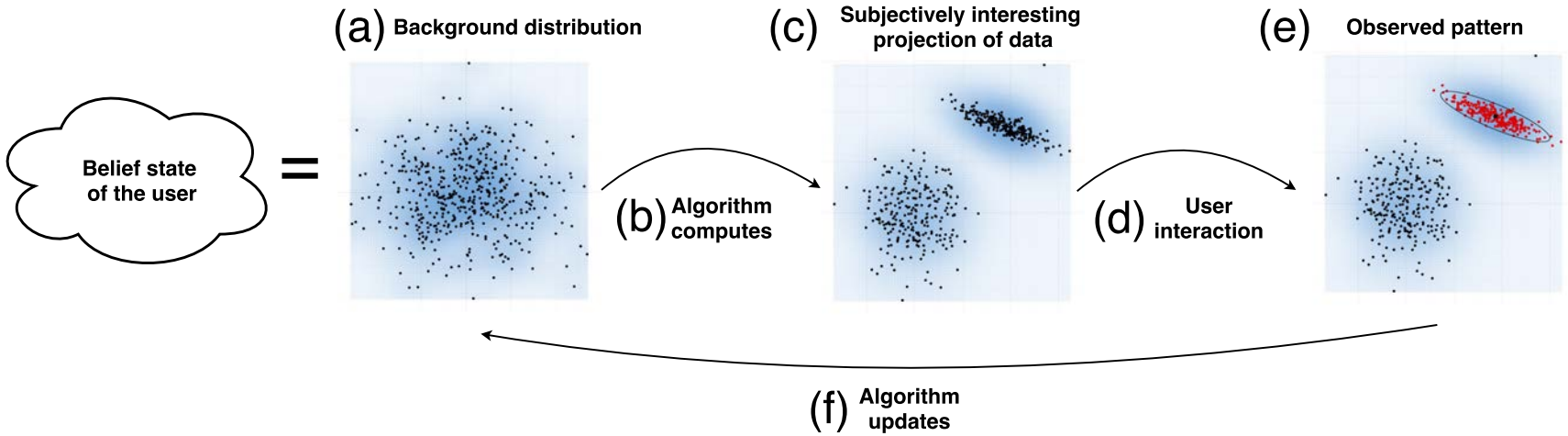


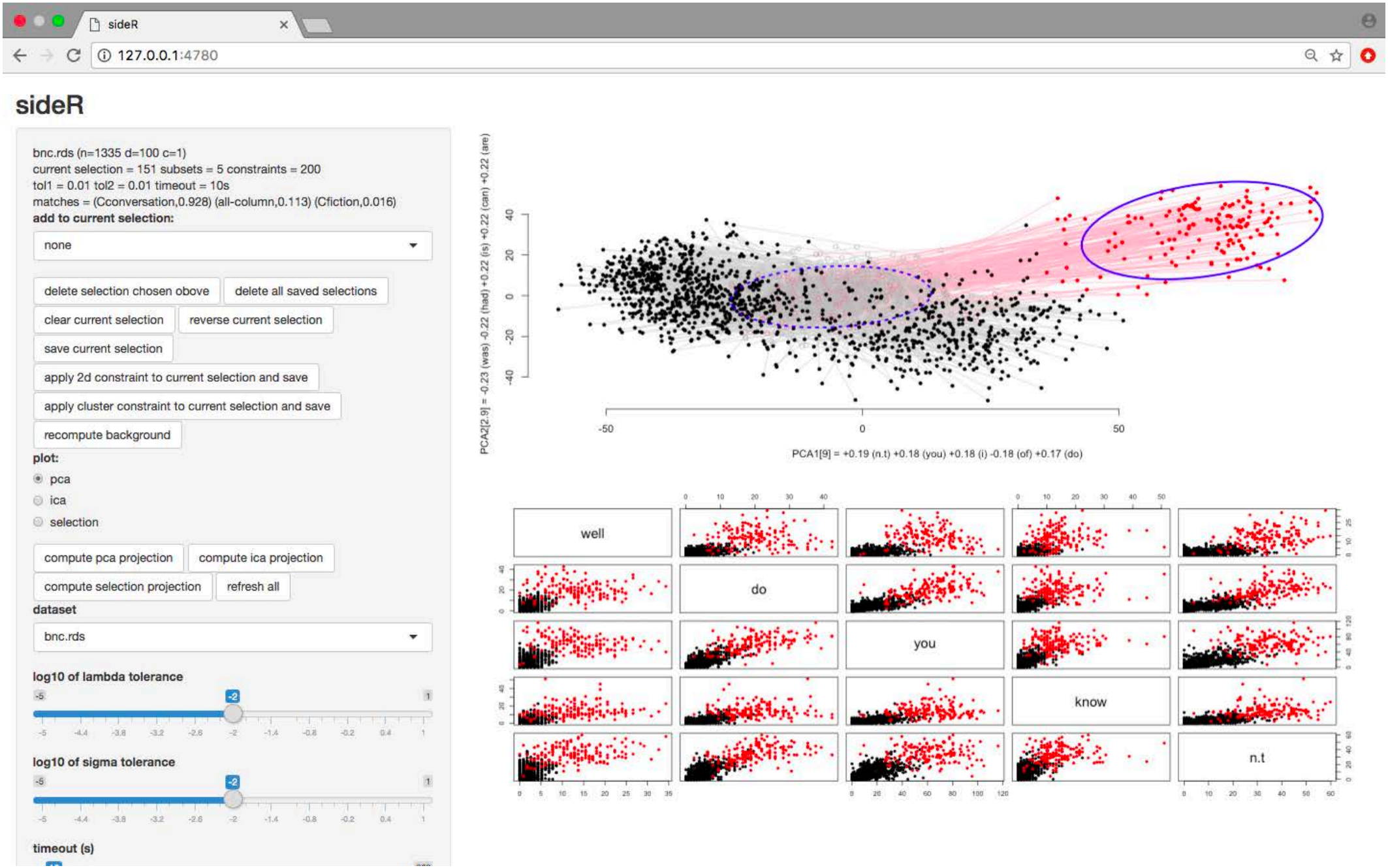
Fig. 3. A dataset of facial images embedded in different ways. The left figure shows a plain PCA embedding, while the other two figures use LSP to group the control points by person and by pose (*looking-straight*, *-up*, *-left* and *-right*), respectively.

- Variant of kernel PCA where user can add, e.g. must-link constraints to modify the embedding in a computationally efficient way (so that it is usable in interactive systems!)
- Paurat et al. 2013, https://doi.org/10.1007/978-3-642-40994-3_52
  Oglic et al. 2014, https://doi.org/10.1007/978-3-662-44851-9_32

Tell the me something I don't know



- We model user's knowledge of the data (*background model*)
- We show the user the view in which the data and the background model differs most
- Each time the user observes something marks it (e.g., cluster, outlier) the background model is updated accordingly
- Uses dimensionality reduction to produce views (tuned to show maximal difference between data distributions)
- Demo (implemented by R Shiny) http://www.iki.fi/kaip/sider.html
- Puolamäki et al. 2017, https://arxiv.org/abs/1710.08167



http://www.iki.fi/kaip/sider.html

Give feedback!

This and following lectures

- Today: dimensionality reduction
- Thu 28 March: guest lecture on "Visualisation of Networked Information" by Tomi Kauppinen
- Mon 29 March: presentation of selected student assignments, more on dimensionality reduction(?)