t-SNE and MDS

Julien JACQUES

Université Lyon 2

Introduction

Multi-Dimensional Scaling (MDS)

t-Distributed Stochastic Neighbor Embedding (t-SNE)

Uniform Manifold Approximation and Projection (UMAP)

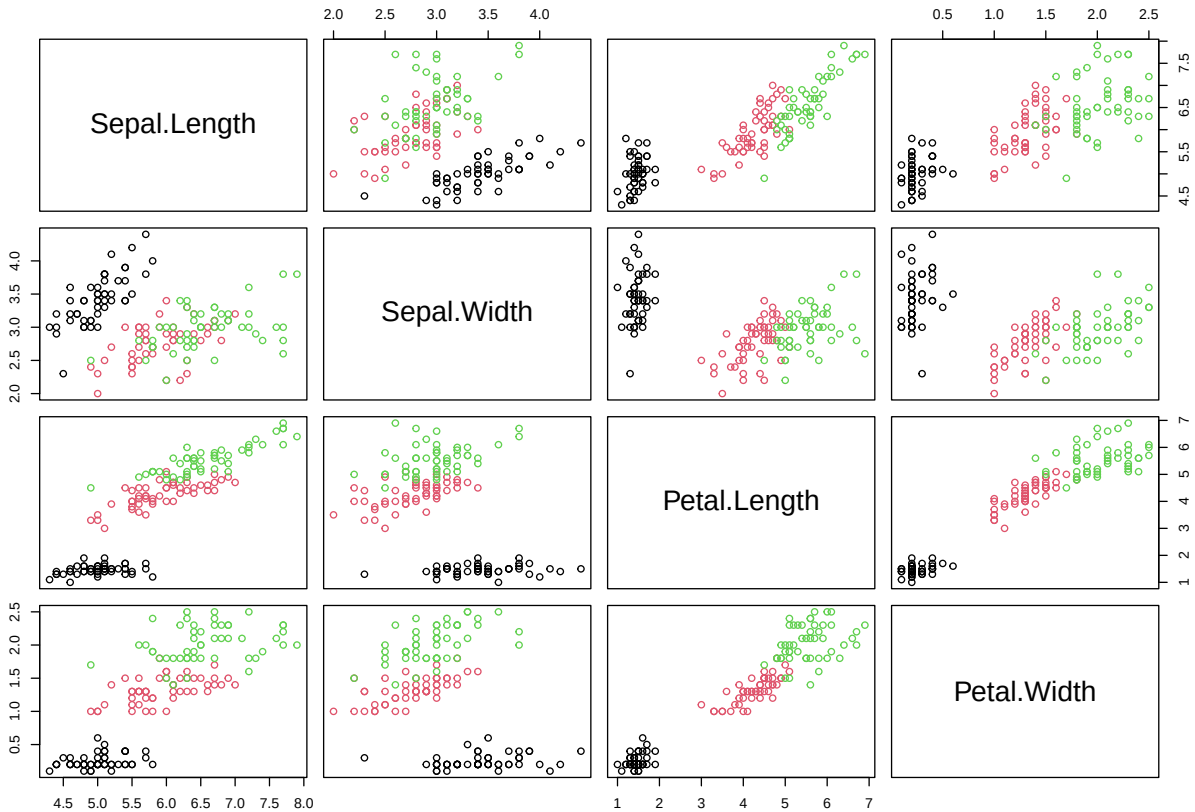
Introduction

Low dimensional data representation

- Visualizing data is crucial for many machine learning application.
- Representing a data set $X \in \mathbf{R}^{n \times p}$ is difficult since $p \geq 3$.
- Several projection methods have been introduced to project the data points from $\mathbf{R}^p$ in a space a smaller dimension (typically $\mathbf{R}^2$):
  - Principal Component Analysis (PCA)
  - Multi Dimensional Scaling (MDS)
  - t-Distributed Stochastic Neighbor Embedding (t-SNE)
  - Uniform Manifold Approximation and Projection (UMAP)

Iris data set

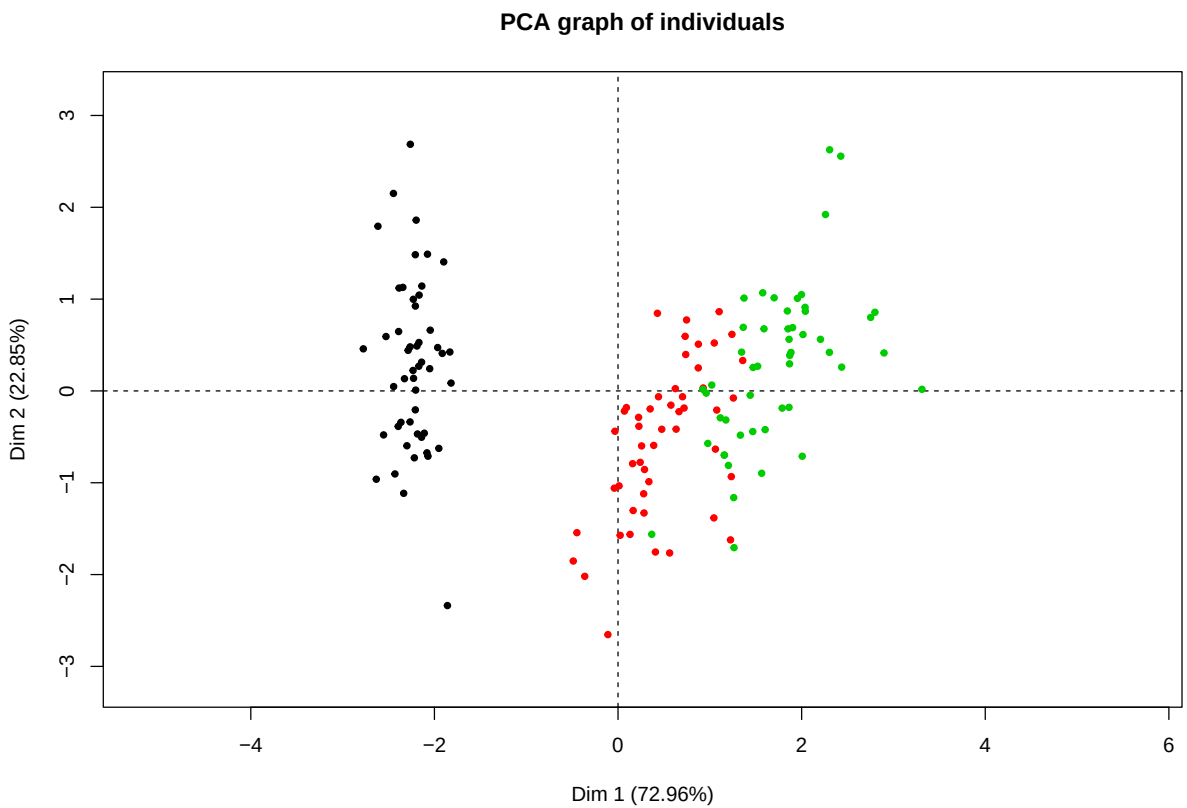
```
plot(iris[,-5],col=iris$Species)
```



PCA

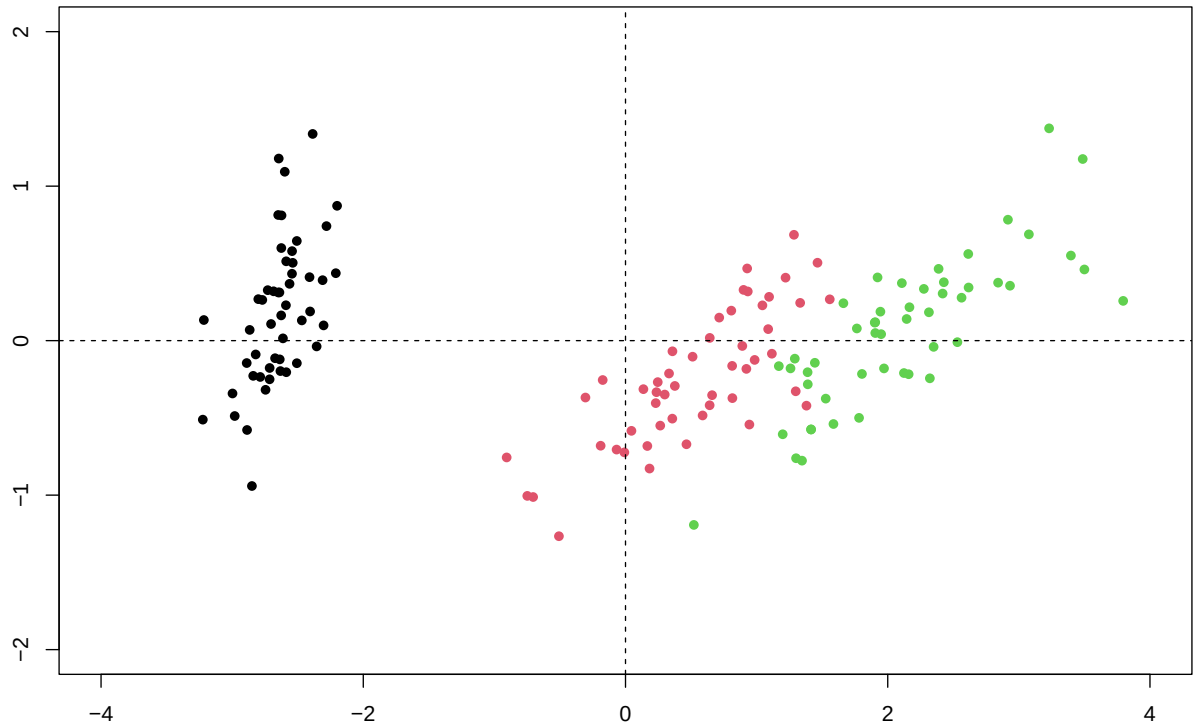
PCA projects the data points s.t. the variance after projection be maximum $\Leftrightarrow$ the Euclidean distance between points are maximally preserved

```
library("FactoMineR")
res.pca <- PCA(iris[,-5],graph = F)
plot(res.pca,choix="ind",col.ind=iris$Species,
     graph.type = "classic",label='none')
```



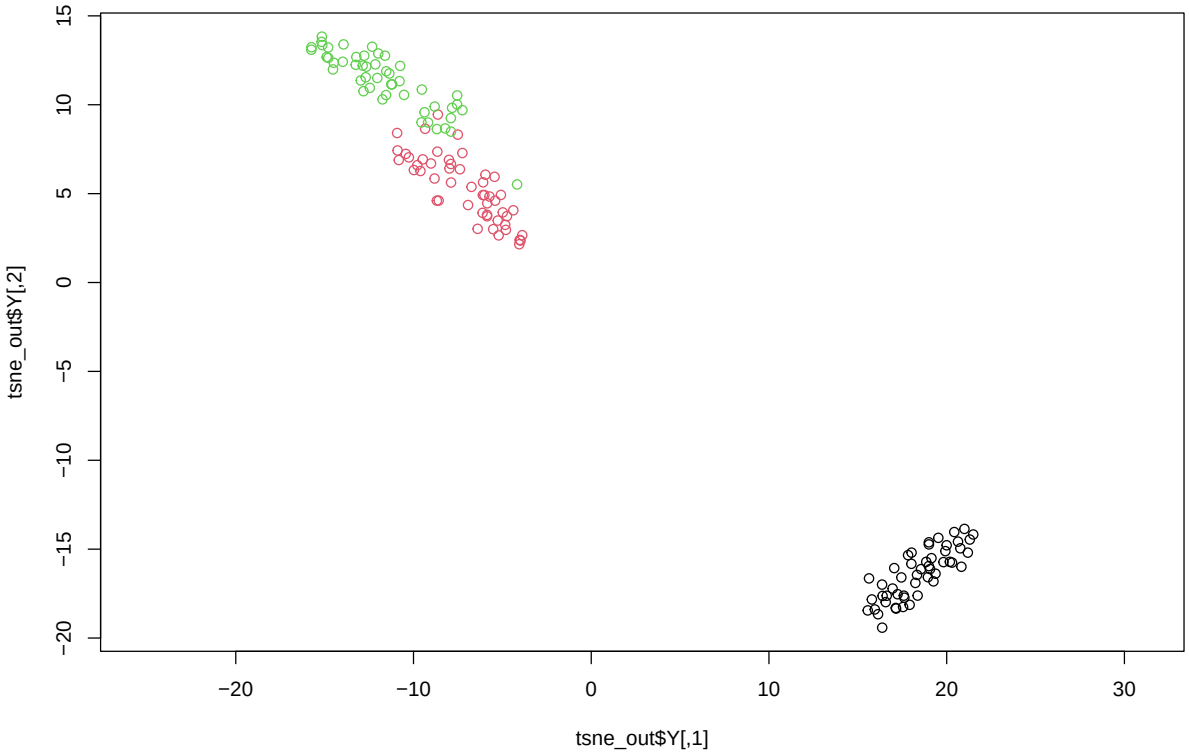
MDS

```
loc <- cmdscale(dist(iris[,-5]))
plot(loc,xlab="",ylab="",col=iris$Species,
     ylim=c(-2,2),xlim=c(-4,4),pch=16)
abline(v=0,lty=2);abline(h=0,lty=2)
```



t-SNE

```
library("Rtsne")
iris_unique <- unique(iris) # Remove duplicates
iris_matrix <- as.matrix(iris_unique[,1:4])
tsne_out <- Rtsne(iris_matrix,pca=FALSE,perplexity=30,theta
plot(tsne_out$Y,col=iris_unique$Species, asp=1)
```



Multi-Dimensional Scaling (MDS)

MDS

We do not use (know ?) the data matrix

$$X = (x_{ij})_{1 \leq i \leq n, 1 \leq j \leq p}$$

but only the matrix of distances (or dissimilarities) between individuals

$$D = (d_{ij})_{1 \leq i \leq n, 1 \leq j \leq n}$$

Interest:

- to be able to graphically observe the same data set through different "optics" and even to compare the representations. Each optic is defined by the way we measure distances or dissimilarities between objects
- to visualize link between variables (from matrix of correlations)

Distance

$D = (d_{ij})_{1 \leq i \leq n, 1 \leq j \leq n}$ is a matrix of distance if:

- $d_{ii} = 0$
- $d_{ij} = d_{ji} \geq 0$ for all $i \neq j$
- $d_{ij} \leq d_{ik} + d_{kj}$

Examples:

- Euclidean distance:

$$d_{ij} = (\sum_{\ell=1}^{p} (x_{i\ell} - x_{j\ell})^2)^{1/2}$$

- Manhattan distance :

$$d_{ij} = \sum_{\ell=1}^{p} |x_{i\ell} - x_{j\ell}|$$

- Mahalanobis distance (when variables are of different scales):

$$d_{ij} = (\sum_{\ell=1}^{p} \frac{1}{\sigma_\ell^2} (x_{i\ell} - x_{j\ell})^2)^{1/2}$$

About the choice among distances

To go further, have a look to the following paper which explain, before introducing new distances, that the Manhattan distance is preferable to the Euclidean distance in some high dimension machine learning task (as clustering):

Aggarwal, C.C., Hinneburg, A., Keim, D.A. (2001). On the Surprising Behavior of Distance Metrics in High Dimensional Space. In: Van den Bussche, J., Vianu, V. (eds) Database Theory — ICDT 2001. ICDT 2001.

Dissimilarity

$D = (d_{ij})_{1 \leq i \leq n, 1 \leq j \leq n}$ is a matrix of dissimilarity if:

- $d_{ij} = d_{ji} \geq d_{ii}$

Dissimilarity are especially useful for binary variables:

- Jaccard dissimilarity:

$$1 - \frac{a_{ij}}{p - d_{ij}}$$

  where:
  - $0 \leq a_{ij} \leq p$ is the number variables equal to 1 for individuals i and j
  - $0 \leq d_{ij} \leq p$ is the number variables equal to 0 for individuals i and j
- Concordance dissimilarity: $1 - \frac{a_{ij}+d_{ij}}{p}$
- Dice dissimilatiry: $1 - \frac{2a_{ij}}{a_{ij}+p-d_{ij}}$

Exercice 1

Compute the different dissimilarity indices for individuals $(1, 1, 0, 0, 0)$ and $(1, 0, 1, 0, 0)$.

Goal of Classical MDS

Given a dissimilarity (distance) matrix D, **Classical MDS** seeks to find $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n \in \mathbf{R}^m$ (*principal coordonates*) such that

$$d_{ij} \simeq ||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j||_2$$

in the sense that we try to minimize

$$\sum_{i,j} (d_{ij} - ||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j||_2)^2$$

Rk: the representation $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n$ is not unique, since adding any constant c does not change the distances.

Classical MDS algorithm

Let assume the we have a matrix D of Euclidean distances. The principal coordinates of MDS can be obtained by:

1. Set up the squared proximity matrix: $D^{(2)} = (-\frac{1}{2} d_{ij}^2)_{i,j}$
2. Apply double centering: $B = CD^{(2)}C$ with $C = I_n - \frac{1}{n}\mathbf{1}_n$ (I_n is the $n \times n$ identity matrix and $\mathbf{1}_n$ is the $n \times n$ matrix of 1.)
3. Determine the m largest eigenvalues $\lambda_1, \ldots, \lambda_m$ and eigenvectors $e_1, \ldots, e_m$ of B
4. Compute the projection (principal coordinates) $\hat{X} = E_m \Lambda_m^{1/2}$ where E_m is the matrix of eigenvectors and Λ_m the diagonal matrix of eignevalues

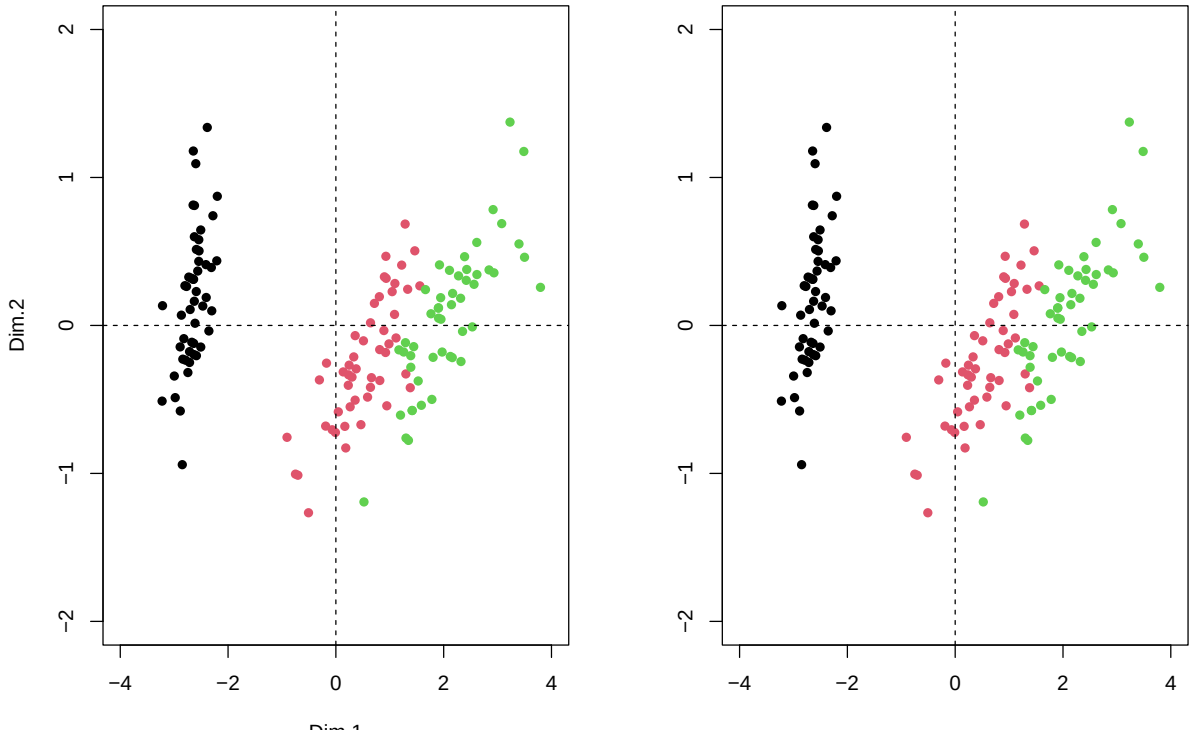
Classical MDS and PCA

When D is the Euclidean distance, PCA (not normalized) and Classical MDS are equivalent. The principal coordinates $\hat{X}^{(m)}$ are equal to $\sqrt{n}$ times the principal components of the PCA.

The interest of MDS is to use other distance or dissimilarities.

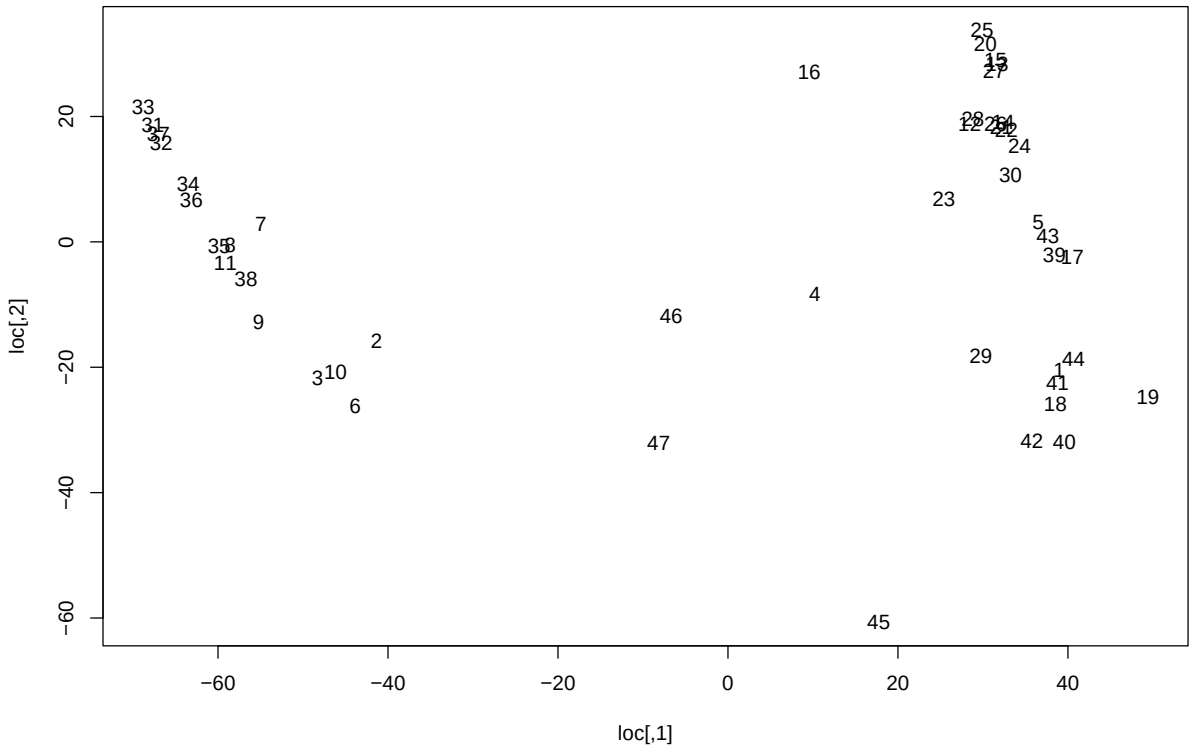
Classical MDS and PCA

```
par(mfrow=c(1,2))
res.pca <- PCA(iris[,-5],graph = F,scale.unit =F)
plot(res.pca$ind$coord[,1:2],col=iris$Species,
     ylim=c(-2,2),xlim=c(-4,4),pch=16)
loc <- cmdscale(dist(iris[,-5]))
abline(v=0,lty=2);abline(h=0,lty=2)
plot(loc,xlab="",ylab="",col=iris$Species,
     ylim=c(-2,2),xlim=c(-4,4),pch=16)
abline(v=0,lty=2);abline(h=0,lty=2)
```



Classical MDS in R

```
swiss.x <- as.matrix(swiss[, -1])
loc <- cmdscale(dist(swiss.x))
plot(loc,type="n")
text(loc,labels=as.character(1:nrow(swiss.x)))
```



Metric MDS

Metric MDS seeks to find $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n \in \mathbf{R}^m$ such that

$$d_{ij} \simeq \hat{d}_{ij} = d(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j)$$

where d is any distance in the projected space.

The proximity between d_{ij} and $\hat{d}_{ij}$ is measured by a stresse function, which usually can be the squared loss stress function:

$$\mathcal{L}(\hat{d}_{ij}) = \left(\frac{\sum_{i<j} (d_{ij} - \hat{d}_{ij})^2}{\sum d_{ij}^2} \right)^{1/2}$$

Metric MDS: Sammon mapping

Other stress function can be considered.

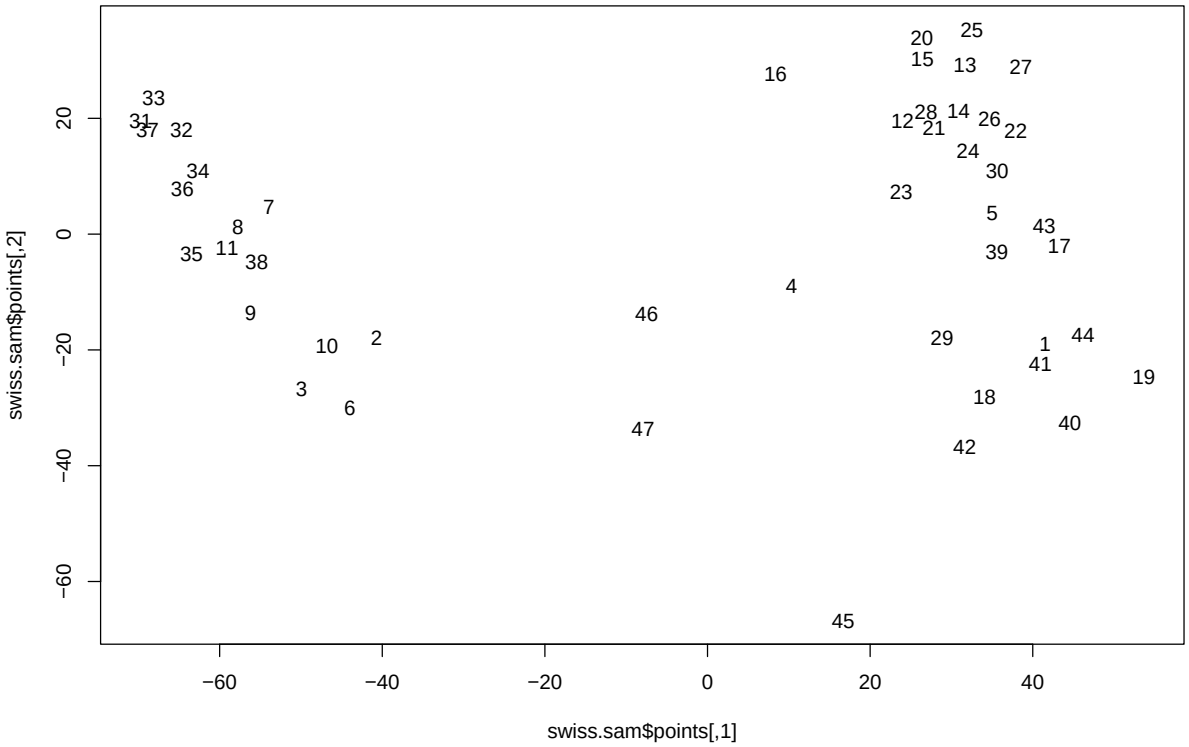
Sammon mapping consider an alternative stress:

$$\mathcal{L}_S(\hat{d}_{ij}) = \frac{1}{\sum_{i<j} d_{ij}} \sum_{i<j} \frac{(d_{ij} - \hat{d}_{ij})^2}{d_{ij}}$$

By dividing each $(d_{ij} - \hat{d}_{ij})^2$ by d_{ij}, Sammon mapping preserves the small d_{ij}, giving them a greater degree of importance in the fitting procedure than for larger values of d_{ij}

Sammon mapping in R

```
library(MASS)
swiss.x=as.matrix(swiss[, -1])
swiss.sam=sammon(dist(swiss.x),trace=F)
plot(swiss.sam$points,type="n")
text(swiss.sam$points,labels=as.character(1:nrow(swiss.x)))
```



Non-Metric MDS

Non-Metric MDS seeks to find $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n \in \mathbf{R}^m$ such that, for some monotonic function f, $f(\hat{d}_{ij})$ be as close as possible from d_{ij}:

$$d_{ij} \simeq f(\hat{d}_{ij})$$

Consequently, only the order of dissimilarities is important rather than the amount of dissimilarities. Indeed:

$$d_{ij} < d_{i\ell} \Leftrightarrow f(d_{ij}) < f(d_{i\ell})$$

The $f(d_{ij})$ are called *disparities*.

Non-metric MDS should then be used when the distance or dissimilarities are only known from an **ordinal** point of view: the values d_{ij} is arbitrary, just the fact that d_{ij} is lower or greater than d_{ik} is important.

Kruskal's Non-metric MDS

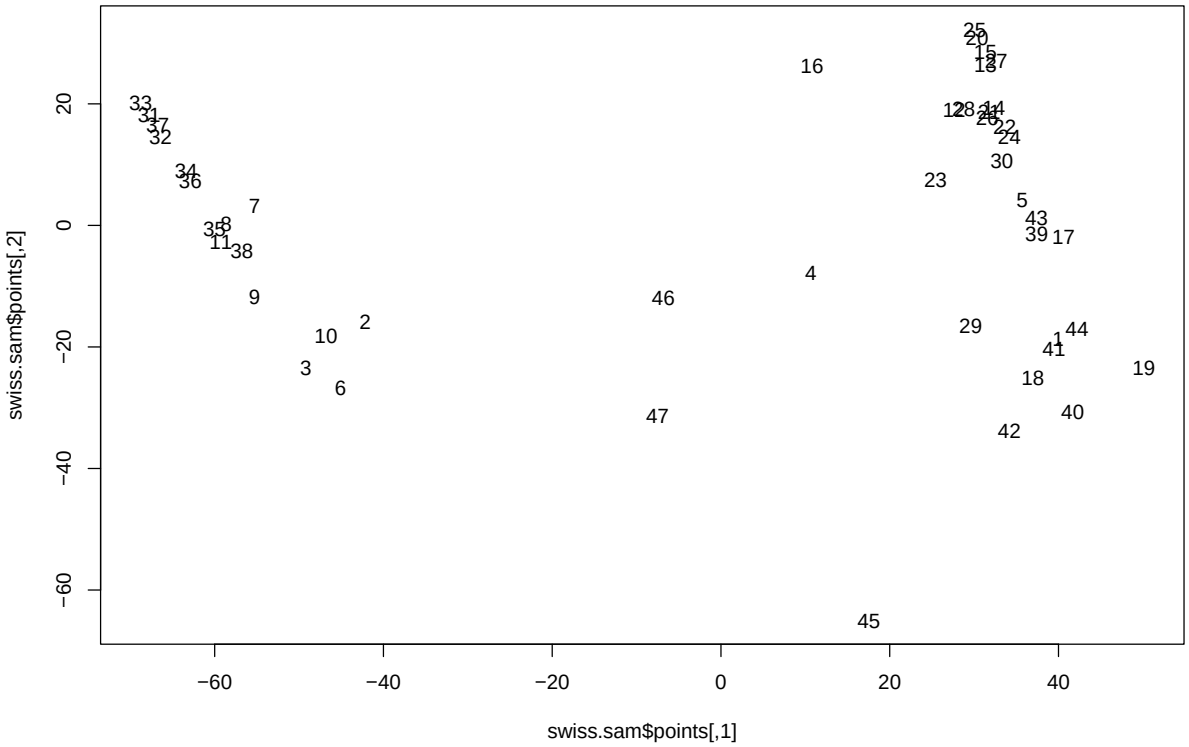
Using the usual squared loss *stress*, Kruskal's Non-metric MDS seek to minimize:

$$\mathcal{L}(\hat{d}_{ij}) = \left(\frac{\sum_{i<j}(d_{ij} - f(\hat{d}_{ij}))^2}{\sum d_{ij}^2} \right)^{1/2}$$

according to both $\hat{d}_{ij}$ (i.e. $\hat{\mathbf{x}}_i$'s) and f.

Kruskal's Non-metric MDS in R

```
library(MASS)
swiss.x=as.matrix(swiss[, -1])
swiss.sam=isoMDS(dist(swiss.x),trace=F)
plot(swiss.sam$points,type="n")
text(swiss.sam$points,labels=as.character(1:nrow(swiss.x)))
```



MDS

Conclusion:

- when working with Euclidean distance and when only D is available (and not X), **classical MDS** can be used in place of PCA
- when working with another dissimilarity (or non Euclidean distance), **Metric MDS** should be used.
- when in D only the order is significant and not the value of the distances, **Non-metric MDS** should be used.

To go further:

B. Ghojogh et al, Multidimensional Scaling, Sammon Mapping, and Isomap: Tutorial and Survey, arXiv, 2020.

Application 1: Airlines distances

```
library(cluster.datasets)
data(airline.distances.1966)
print(airline.distances.1966[1:6,1:6])
```

```
##   code AZ BD BN BY BS
## 1   AZ  0 39 22 59 54
## 2   BD 39  0 20 20 81
## 3   BN 22 20  0 39 74
## 4   BY 59 20 39  0 93
## 5   BS 54 81 74 93  0
## 6   CO 33  8 18 27 73
```

Represent into a 2-dimensional space the principal cities of the world from their airline distances.

Application 2: Letter recognition

Wolford and Hollingsworth (1974) were interested in the confusions made when a person attempts to identify letters of the alphabet viewed for some milliseconds only. A confusion matrix was constructed that shows the frequency with which each stimulus letter was mistakenly called something else. A section of this matrix is shown in the table below.

Letter	C	D	G	H	M	N	Q	W
C	–							
D	5	–						
G	12	2	–					
H	2	4	3	–				
M	2	3	2	19	–			
N	2	4	1	18	16	–		
Q	9	20	9	1	2	8	–	
W	1	5	2	5	18	13	4	–

- is-it a dissimilarity matrix?
- which MDS method is appropriated for representing these data?

t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE

- We seek for a matrix $\hat{X} \in \mathbb{R}^q$ which correctly represent $X \in \mathbb{R}^p$ ($q < p$).
- Neighbors $\hat{\mathbf{x}}_j$ of $\hat{\mathbf{x}}_i$ in the reduced (projection) space should be the same than neighbors $\mathbf{x}_j$ of $\mathbf{x}_i$ in the initial space.
- Neighborhood of $\mathbf{x}_i$ is represented by the conditional probability $p_{ij} = p(\mathbf{x}_j|\mathbf{x}_i)$ than $\mathbf{x}_j$ be a neighbor of $\mathbf{x}_i$
- So we look for projection $\hat{\mathbf{x}}_i$ s.t. $\hat{p}_{ij} \simeq p_{ij}$

L.v.d. Maaten and G. Hinton. Visualizing Data using t-SNE. Journal of Machine Learning Research, 9(86) :2579–2605, 2008.

t-SNE

- The conditional probability than $\mathbf{x}_j$ is a neighbor of $\mathbf{x}_i$ is evaluated thanks to the value of the Gaussian density, centered in $\mathbf{x}_i$, evaluated in $\mathbf{x}_j$:

$$p_{ij} = \frac{\exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2)}{\sum_{k\neq i}\exp(-||\mathbf{x}_i - \mathbf{x}_k||^2/2\sigma^2)}$$

  with the convention $p_{ii} = 0$ s.t. $\sum_j p_{ij} = 1$. The variance σ^2 is an hyper-parameter of the method, called *perplexity*

- In the reduced space, the $\hat{p}_{ij}$ are computed using a Student distribution with 1 degree of freedom:

$$\hat{p}_{ij} = \frac{(1 + ||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j||^2)^{-1}}{\sum_{k\neq i}(1 + ||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_k||^2)^{-1}}$$

  Using this heavy-tailed distribution allows a better distingstion of farest neighbors

t-SNE optimization

The proximity between distribution $\hat{p}_{ij}$ and p_{ij} is measured with the Kullback-Leibler divergence:

$$KL(p|\hat{p}) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{\hat{p}_{ij}}$$

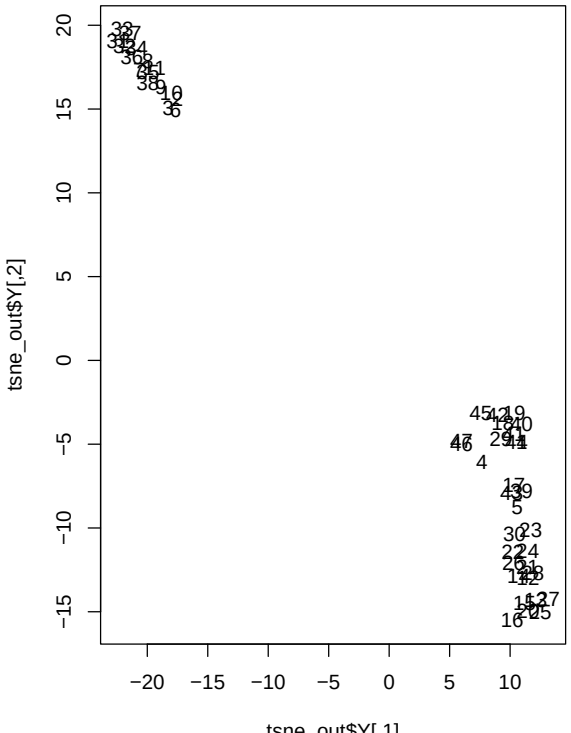
Gradient descent optimization is considered for minimizing $KL(p|\hat{p})$ in function of $\hat{p}$ (i.e. $\hat{\mathbf{x}}_i$'s)

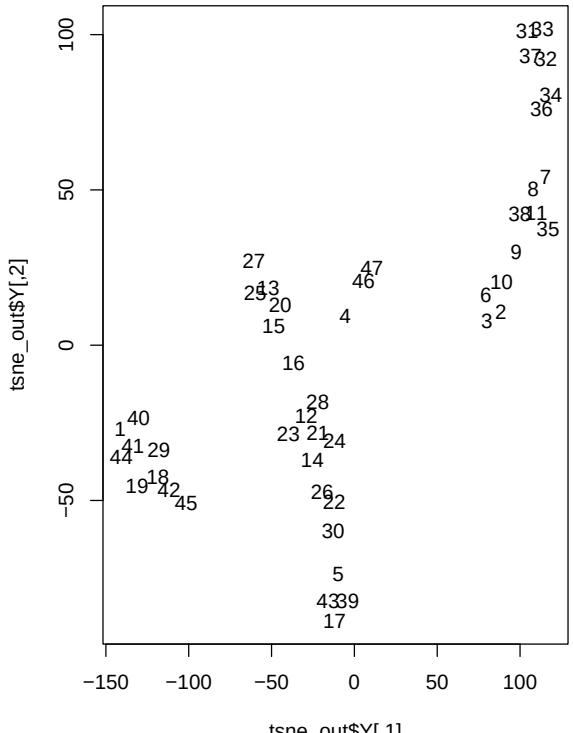
Some limits of t-SNE

- Optimization is stochastic (depend on the initialization)
- Representation depends on some optimization hyper-parameter and on the perplexity
- It is not possible to project a new points: the otpimization should be reload
- It is not possible to come back to the initial point from the reduced space.

t-SNE in R

```
library("Rtsne")
swiss.x=as.matrix(swiss[, -1])
par(mfrow=c(1,2))
tsne_out <- Rtsne(swiss.x,pca=FALSE,perplexity=10,theta=0.0)
plot(tsne_out$Y,,type="n")
text(tsne_out$Y,labels=as.character(1:nrow(swiss.x)))
tsne_out <- Rtsne(swiss.x,pca=FALSE,perplexity=5,theta=0.0)
plot(tsne_out$Y,,type="n")
text(tsne_out$Y,labels=as.character(1:nrow(swiss.x)))
```





Application 3: MNIST

Use different representation method (PCA, MDS, t-SNE) for representing (a subset of) the MNIST data set:
http://yann.lecun.com/exdb/mnist/

Uniform Manifold Approximation and Projection (UMAP)

UMAP

UMAP is a recent competitors of t-SNE which should:

- be faster
- best preserve the global structure of data

McInnes et al., (2020). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, arXiv 1802.03426.

What does UMAP

Similarly to t-SNE, UMAP:

- build a high-dimensional neighborhood graph (*fuzzy simplicial complex*), which is a weighted graph with weight depending of the probability that 2 points are connected
  - 2 points are connected if the balls of radius r centred in these points overlaps
  - the choice of r is locally adapted depending of the distance to the k nearest neighboors
- optimize a low-dimensional graph as closest as possible to the high-dimensional one (similar to t-SNE but with some tricks to speed up the optimisation)

UMAP hyper-parameters

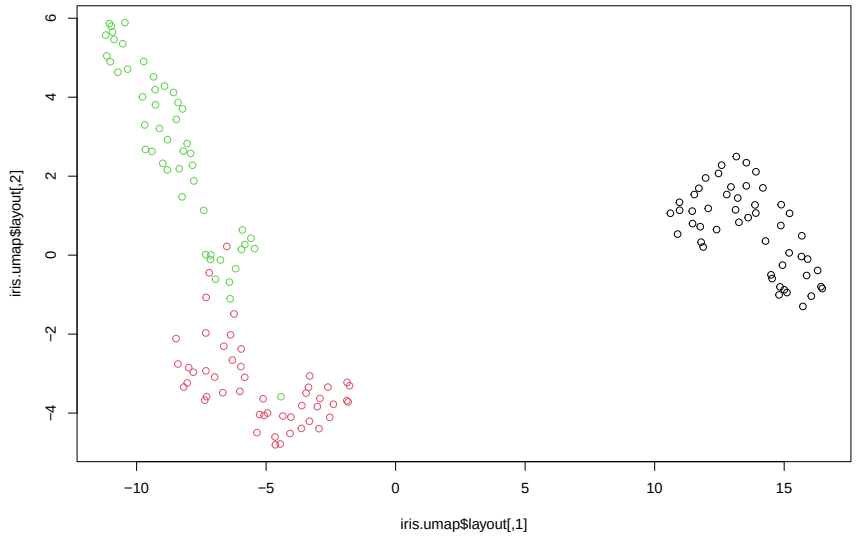
- nearest neighboors:
  - low values lead to focus on local structure
  - high values lead to focus on global structure
- min-dist: the minimum distance between points in low-dimensional space

Let's have a look to the following animations for an idea of these hyper-parameter :
https://pair-code.github.io/understanding-umap/

UMAP in R

Example of use with the iris data set

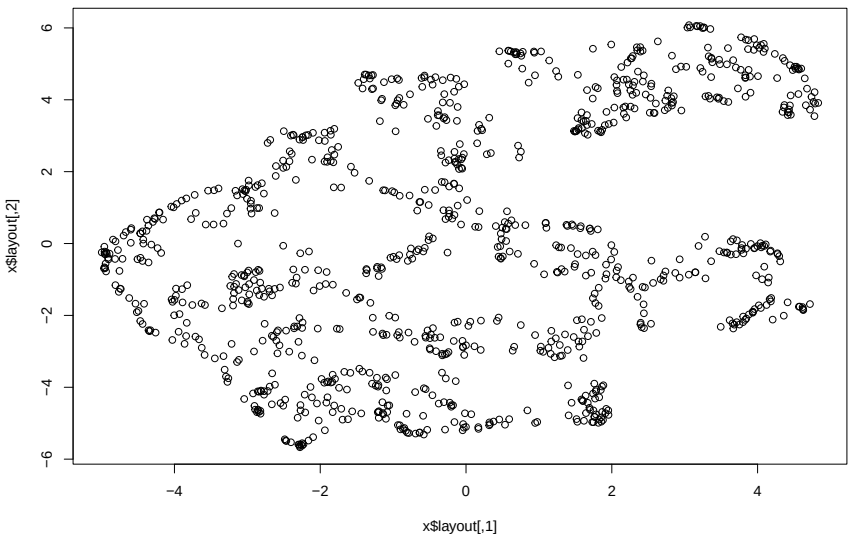
```
library("umap")
custom.config <- umap.defaults
custom.config$n_neighbors=10
custom.config$min_dist=.5

iris.umap = umap(iris[,1:4], config=custom.config)
plot(iris.umap$layout,col=iris$Species)
```



UMAP in R

But be careful with such representations, which can sometimes exhibit some structures where none exists . . .

```
data=matrix(runif(3000),1000,3)
x = umap(data)
plot(x$layout)
```



Application 4: MNIST

Compare UMAP to the other representation method for a MNIST sample.

Play with hyper-parameters.