Multidimensional Scaling

STAT 140/240 Final Project

Adam Birnbaum
Kent Jitpatima
Ruiqi Li
Rebekah Rodeback
Peng Xu
Xuyang Zhao

December 18th, 2020

General Outline

- Motivation
- Concepts
- Theory

- Packages
- Code
- Applications

Motivation and Overview

Quick Review: Principal Component Analysis

- Principal Component Analysis (PCA) is used in exploratory data analysis and for making predictive models.
- Its main goal is to reduce the dimensionality of a data set, making it not only interpretable, but also maintaining as much ‘correlation’ of the original data set.
- The “Principal Components” are linear functions of the variables in the original dataset. Thus they are eigenvectors of the data’s covariance matrix.

Quick Review: Principal Component Analysis

- Once we find the first and second principal components (linear functions of the two highest ‘rank/importance’) , we can use them as axes and and observe the correlation between them on a 2-D plot.
- Now data that had originally *N* items, can be visualized on a simpler 2-dimension plot.
- Note: Principal Components are ranked in order of importance, thus the distance between points on the ‘PC1’ is more important than the ‘PC2’

Multidimensional Scaling: Motivation

- What if we are also interested in other ways our observations are related?
- Is correlation the only way to measure how *‘similar’* or *‘dissimilar’* our data is?
- Of course not! Multidimensional scaling is a method of converting “distances” among samples into a 2-D graph.
- Multidimensional scaling is very similar to PCA, except now we can measure ‘distance/similarity’ of the data in another metric besides correlation.

Multidimensional Scaling: Motivating Example

- Suppose ten subjects rate the similarities of six automobiles. That is, each subject rates the similarity of each of the fifteen possible pairs. The ratings are on a scale from 1 to 10, with “1” meaning that the cars are identical in every way and “10” meaning that the cars are as different as possible. The ratings are averaged across subjects, forming a *similarity matrix*. MDS provides the marketing researcher with a map (scatter plot) of the six cars that summarizes the results visually.
- https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Multidimensional_Scaling.pdf

Multidimensional Scaling : Concept

- For *N* items, there are *M= N(N-1)/*2 distances/dissimilarities between the pairs of items.
- Note: If similarities cannot be quantified easily, we can use non-metric multidimensional scaling. We will cover that later.
- These similarities can be arranged in a strictly ascending order
  - $s_{i_1 k_1} < s_{i_2 k_2} < \dots < s_{i_M k_M}$
  - $s_{i_1 k_1}$ is the smallest value of our M similarities while $i_1 k_1$ represents the pair of items that are least similar (rank 1 in this similarity ordering)

Multidimensional Scaling : Concept

- There are two main types of MDS:
  - Metric
  - Non-Metric

- In metric MDS, we want to find a *q*-dimensional configuration of the N items such that the distances in *q*-dimensions, denoted ${d_{ik}}^{(q)}$, match the original observed distances as closely as possible

- In non-metric MDS, we want to find a *q*-dimensional configuration of the N items so that the ordering of the ${d_{ik}}^{(q)}$ correspond to the ordering of the observed dissimilarities as closely as possible
  - We are concerned more with preserving the ordering than the actual quantitative dissimilarities

Metric Multidimensional Scaling : Concept

- While PCA uses correlation among variables to measure similarity, metric multidimensional scaling lets us use different distances such as
  - Euclidean Distance, but this is the same as PCA since minimizing linear Euclidean distance is the same thing as maximizing the linear correlations.
  - Manhattan Distance: measure of the sum of the distances along each dimension.
  - Canberra Distance : absolute difference divided by the sum of the values.
  - Minkowski Distance: the pth root of the sum of the pth powers of the differences of the components.
    - Note: if p =1 this is Manhattan, if p = 2 this is Euclidean.

Classical MDS

Motivation

- Classical MDS is a classical approach on dimensionality reduction, that is used to represent the pairwise dissimilarity between different objects from mapping the original position of data in multidimensional space to a reduced number of dimensions.

Distance Matrix

A= (x_{ij}) with $1 \leq i, j \leq n$ is a distance matrix for a metric distance, then

i. the entries on the main diagonal are all zero,i.e. $x_{ii} = 0$ for all $1 \leq i \leq N$
ii. all the off-diagonal entries are positive ($x_{ij} > 0$ if $i \neq j$)
iii. the matrix is a symmetric matrix ($x_{ij} = x_{ji}$)
iv. for any i and j, $x_{ij} \leq x_{ik} + x_{kj}$ for all k (the triangle inequality).

Given a set of dissimilarities, one can ask whether these values are distances and, moreover, whether can be interpreted as Euclidean distances

Euclidean Distance Matrix

- Euclidean distance matrix is an n × n matrix representing the spacing of a set of n points in Euclidean space. For points $x_1, x_2, \ldots, x_n$ in k-dimensional space $\mathbb{R}^k$, the elements of their Euclidean distance matrix A are given by squares of distances between them. That is

  $A=(a_{ij})$

  $a_{ij}=d^2_{ij}=\|x_i-x_j\|^2$

$$A = \begin{bmatrix} 0 & d_{12}^2 & d_{13}^2 & \dots & d_{1n}^2 \\ d_{21}^2 & 0 & d_{23}^2 & \dots & d_{2n}^2 \\ d_{31}^2 & d_{32}^2 & 0 & \dots & d_{3n}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{n1}^2 & d_{n2}^2 & d_{n3}^2 & \dots & 0 \end{bmatrix}$$

Multidimensional Scaling (MDS)

- The basic idea of MDS is to transform the original distance matrix into a cross-product matrix and then perform the eigen-decomposition that would gives a principal component analysis(PCA).
- Given a Distance matrix A, MDS tries to map x1,x2,…,xn $\in \mathbb{R}^k$ in a reduced dimension P (ideally p=2)so that gives d_{ij}=llx_i-$x_j$ll as close as possible.

Classical MDS

- Classical MDS is also known as Principal Coordinates Analysis (PCoA), Torgerson Scaling or Torgerson–Gower scaling.
- In classical MDS, it find a configuration in a reduced dimension P such that the distance between the points in the configuration, d_{ij}, are close to in value to the observed distances δ_{ij}. In other words, it takes a distance matrix A giving dissimilarities between pairs of points and outputs a coordinate matrix whose configuration minimizes a loss called strain.

Centered configuration and Strain

- Classical scaling is based on inner products which, unlike distances, depend on the origin. W.l.o.g. one can center configurations at the origin, $\Sigma_i xi = 0$, so their inner product matrices have zero marginal means, $\Sigma_k <x_k, x_j>= \Sigma_k <x_i , x_k> = 0$.
- The solution of Classical MDS is not unique, but the assumption of centered configuration, i.e. $\Sigma_i x_{ik}=0$ serves well for dimension reduction. And applying the assumption, it is possible for us to get an unique solution.
- General function of loss function called Stress in distance MDS and strain in classical MDS. We call "Strain" any loss function that measures the lack of fit between inner products $<x_i , x_j>$ and inner-product data B_{ij} .
- $\text{Strain}_D(x_1,x_2,..,X_n)=(\frac{\Sigma_{i,j}(A_{ij}-<x_i,x_j>)^2}{\Sigma_{i,j}A^2_{ij}})^{1/2}$ where A_{ij} are the terms of the matrix A,
- which is equivalent to performing eigenvalue decomposition on a Gram matrix constructed from the dissimilarity measure. When the dissimilarity measure is Euclidean distance, classical MDS is the same as Principal Components Analysis.

Steps of the classical MDS

Theclassical MDS finds the centered configuration $x_1, . . . , x_n \in \mathbb{R}^k$ for some $q \geq n - 1$ so that their pairwise distances are the same as those corresponding distances in D.

- Given a similarity matrix A, the relationship between a similarity and distance is defined as $d_{ij}=(A_{ij}-2A_{ij}+A_{jj})^{1/2}$ from $\|x_i-x_j\|^2=x_i^Tx_i+x_j^Tx_j-2x_i^Tx_j$
- Consider the matrix B_{ij}, with entries $B_{ij}=-\frac{1}{2}d^2{}_{ij}$
- By assumption of centered configuration,

$$\sum_{i=1}^{n} bij = \sum_{i=1}^{n}\sum_{k=1}^{q} x_{ik}x_{jk} = \sum_{k=1}^{q} x_{jk}\sum_{i=1}^{n} x_{ik} = 0,\ for\ j=1,\ldots,n$$

Steps of the classical MDS

- $With\ trace(B) = \sum_{i=1}^{n} b_{ii}$

  $$\sum_{i=1}^{n} d_{ij}^{2} = tr(B) + nb_{jj}, \sum_{j=1}^{n} d_{ij}^{2} = tr(B) + nb_{ii}, \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^{2} = 2ntr(B)\ .$$

- *Combining the two equalities, it gives an unique solution:*

  $$b_{ij} = -\frac{1}{2}\left(d_{ij}^{2} - d_{.i}^{2} - d_{.j}^{2} + d_{..}^{2}\right)\ or\ B = -\frac{1}{2}CD_{2}C.$$

- *Then perf orm the eigendecomposition on* B, *that is* $B = Q \wedge Q^{T}$

  it gives a solution $X = \wedge^{\frac{1}{2}} Q^{T}$

Steps of the classical MDS

- The space which X lies is the eigenspace where the first coordinate contains the largest variation and is identified with $\mathbb{R}^k$.
- If we wish to reduce the dimension to $p \leq k$, then the first p rows of $X_{(K)}$ best preserves the distances d_{ij} among all other linear dimension reduction of X (to p). Then $X_{(p)} = \Lambda_p^{1/2} Q_p^T$
- The dimension reduction from X to $X_{(k)}$ to $X_{(p)}$ is the same as PCA

Steps of a Classical MDS algorithm

In summary, Classical MDS uses the fact that the coordinate matrix X can be derived by eigenvalue decomposition from $B=XX^T$. And the matrix B can be computed from proximity matrix D by using double center.

The steps to solve metric MDS follow as:

1. Set up the squared proximity matrix $D_{(2)}=[d^2_{ij}]$
2. Apply double centering: $B=-\frac{1}{2}CD_{(2)}C$ using assumption of centered configuration or centering matrix $J=I-\frac{1}{n}11^T$, where n is the number of objects.
3. Find the p largest eigenvalues λ_1 , λ_2 ,…, λ_p of B and corresponding eigenvectors L = (L(1), L(2),…,L(p)) of B (where p is the number of dimensions desired for the output, ideally p=2)
4. Compute the coordinates of n points in the Euclidean space given by $X=\Lambda_j^{1/2}Q^T$ where $Q^T$is the matrix of m eigenvector and Λ_j is the diagonal matrix of m eigenvalue of B

Classical MDS assumes Euclidean distances, so this is not applicable for direct dissimilarity ratings.

Non-Metric MDS

NMDS: Motivation

While classical MDS is appropriate if the data consist of true distances (or true distances are easily obtained from the data), the procedure may be inappropriate for other types of data. Consider a psychological experiment in which letters are flashed on a screen very briefly and subjects are then asked to state which letter was shown; imagine the following data were obtained:

- the letters 'Q' and 'O' were mistaken for one another 20 times
- the letters 'E' and 'F' were mistaken for one another 10 times

We might feel it's valid to say that 'Q' and 'O' are more similar than 'E' and 'F' but don't necessarily believe that the former are twice as similar as the latter.

- In this case, we can turn to non-metric MDS to visualize the relationships between the letters

NMDS: Concept

Non-Metric MDS:

- given n objects, let δ_{ij} denote some measure of "dissimilarity" between objects i and j
    - one possible definition for the example on the previous slide could be:

      $$\delta_{ij} = K - C_{ij}$$

      where C_{ij} is the number of times objects i and j were mistaken for one another and $K = \max\limits_{i,j:i \neq j} C_{ij}$
- we want to find $X_1, X_2, \ldots, X_n \in \mathbb{R}^q$ such that the *ranking* of the distances between the Xs matches the ranking of the dissimilarities between the n objects as closely as possible
    - the actual distances in $\mathbb{R}^q$ are irrelevant, we only seek to match the rank order
    - as with metric MDS, it is often useful to choose q to be 2 or 3 so that scatterplot visualizations are possible

NMDS: Concept

- Note that rank order of the distances is translation- and scale-invariant
  - i.e. we can shift all points by a constant or scale all distances by a constant and maintain the same rank order of the distances
- therefore, there is no unique solution to the NMDS objective
- it is common to constrain the placement of the X_i such that the mean location is the origin and the mean Euclidean distance from the origin is 1

NMDS: Concept

For a candidate placement of $X_1, X_2, \ldots, X_n$ in $\mathbb{R}^q$ let d_{ij} denote the Euclidean distance between X_i and X_j. As a simple example, consider a case where $n = 3$ and:

$$\delta_{12} < \delta_{13} < \delta_{23}$$

Then we want to place X_1, X_2, and X_3 in $\mathbb{R}^q$ such that:

$$d_{12} < d_{13} < d_{23}$$

Note that for large n and small q, it may not be possible to perfectly match the rank order

NMDS: Concept

Shephard and Kruskal noted that if there is a monotonic relationship between the δ_{ij} and the d_{ij} then the candidate placement perfectly preserves the rank order of the observed dissimilarities.

- A candidate placement of the X_i can be visually assessed by plotting the δ_{ij} against the d_{ij}
- An ideal placement is visualized below:

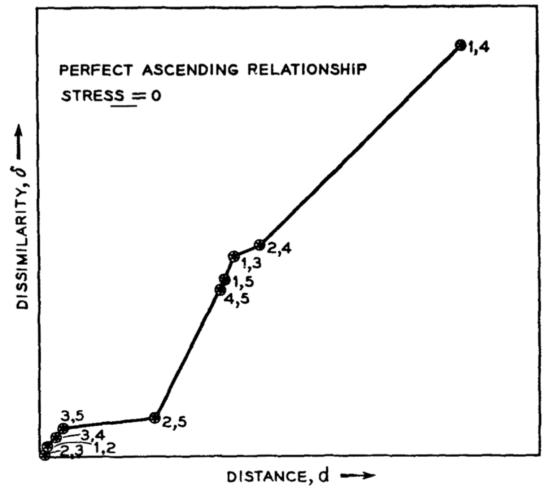


Figure: Possible ideal configuration for $n = 5$

NMDS: Methods

Let $\hat{d}_{ij}$, $i = 1, 2, \ldots, n-1$, $j = 1, 2, \ldots, n$ be a set of numbers whose rank order perfectly matches the rank order of the δ_{ij}. Kruskal defined the stress of a configuration $X_1, X_2, \ldots, X_n$ as:

$$S(X_1, ..., X_n) = \min_{\hat{d}_{ij}\ \forall i,j} \sqrt{\frac{\sum_{i<j}(d_{ij} - \hat{d}_{ij})^2}{\sum_{i<j} d_{ij}^2}}$$

to calculate the stress we need to find the set of numbers $\hat{d}_{ij}$ that minimize the sum of squares subject to the constraint that they have a monotonic relationship with the δ_{ij}

- we can obtain the $\hat{d}_{ij}$ by performing isotonic regression of the d_{ij} on the δ_{ij} and setting $\hat{d}_{ij}$ equal to the fitted values from this regression

NMDS: Methods

- a stress of zero means that the candidate placement perfectly preserves the rank order of the observed dissimilarities
- as stress increases, the candidate placement deviates more from the rank order of the observed dissimilarities

Therefore, given q, we want to find the placement of the X_i that minimizes the stress; we can formally define the target placement of the X_i in q-dimensions as:

$$X_1, \ldots, X_n = \underset{\tilde{X}_1, \ldots, \tilde{X}_n}{\arg\min}\ S(\tilde{X}_1, \ldots, \tilde{X}_n)$$

Given a set of observed dissimilarities and a desired number of dimensions q, how do we find this placement?

NMDS: Methods

Since each $\tilde{X}_i$ has q entries, we can think of $S(\tilde{X}_1, \dots, \tilde{X}_n)$ as a function of nq variables. To that end define:

$$Y = (\tilde{x}_{11}, \dots, \tilde{x}_{1q}, \tilde{x}_{21}, \dots, \tilde{x}_{2q}, \dots, \tilde{x}_{nq})^T \in \mathbb{R}^{nq}$$

And consider $S(\cdot)$ to be a function of Y. Kruskal suggested obtaining the optimal placement $X_1, \dots, X_n$ by picking some initial value of Y and then performing gradient descent to find the minimum of S.

- note that there is no guarantee that $S(\cdot)$ is a strictly convex function; we may have multiple local minima in addition to the target global minimum
- in practice, one might start from multiple initial values of Y and choose the one that yields the smallest value of $S(\cdot)$

NMDS: Methods

While gradient descent was the initial method proposed by Kruskal, de Leeuw later proposed a majorization-minimization algorithm:

- the algorithm specifies a majorizing function $g(Y; Y^{(t)})$ which satisfies:

$$g(Y; Y^{(t)}) \geq S(Y) \ \forall Y$$
$$g(Y^{(t)}; Y^{(t)}) = S(Y^{(t)})$$

- once the majorizing function is specified, we then iteratively minimize $g(Y; Y^{(t)})$ and update the candidate placement by:

$$Y^{(t+1)} = \arg\min_{Y} \, g(Y; Y^{(t)})$$

- this procedure will also optimize $S(Y)$ since:

$$S(Y^{(t)}) = g(Y^{(t)}; Y^{(t)}) \geq g(Y^{(t+1)}; Y^{(t)}) \geq S(Y^{(t+1)})$$

NMDS: Methods

- Ideally, we want to select a $g(\cdot; Y^{(t)})$ that is easier to optimize than the objective function $S(\cdot)$
- See de Leeuw and Mair (2009) for full specification of the majorizing function for minimizing stress in MDS
    - it turns out the stress can be majorized at each iteration by a simple quadratic function
- de Leeuw refers to this algorithm as SMACOF (**S**caling by **MA**jorizing a **CO**mplicated **F**unction)

NMDS: Methods

A high-level overview of the algorithm for NMDS is given below:

1. Choose the desired number of dimensions, q
2. Initialize: $Y^{(0)} = [X_1^{(0)\,T}, X_2^{(0)\,T}, \ldots, X_n^{(0)\,T}]^T$
3. For: $t = 0, 1, \ldots$
    1. $d_{ij}^{(t)} \leftarrow \sqrt{\sum_{k=1}^{q}(x_{ik}^{(t)} - x_{jk}^{(t)})^2}$ for all i, j $i \neq j$
    2. Obtain $\hat{d}_{ij}^{(t)}$ by performing isotonic regression of the $d_{ij}^{(t)}$ on the δ_{ij}
    3. Obtain $Y^{(t+1)}$ via gradient descent or majorization-minimization

The looping step (3) is repeated until convergence

Packages & Code & Applications

MDS methodology

- Calculate a (dis)similarity matrix among pairs of objects (observations, individuals, samples, etc.)
- Apply MDS method/model to obtain low-dimensional representation
- MDS methods/models
    - Metric MDS (Principal Coordinate Analysis or Classical Scaling): any approach where analyzed matrix is based on metric distance; suitable for quantitative data
    - Non-metric MDS (ordinal MDS): suitable for qualitative data; NOT metric distance value, but its value in relation to distances between other pairs of objects

Metric (Classical) MDS

Step1 Calculate Distance matrix from original dataset

Step2 Reduce dimensions of data

Step3 Plot clusters of low-dimensional data from Step2

Datasets:
swiss (data on 47 French-speaking provinces in Switzerland in 1888)

wine (data on 3 types of wine grown in the same region in Italy from 3 different cultivars)

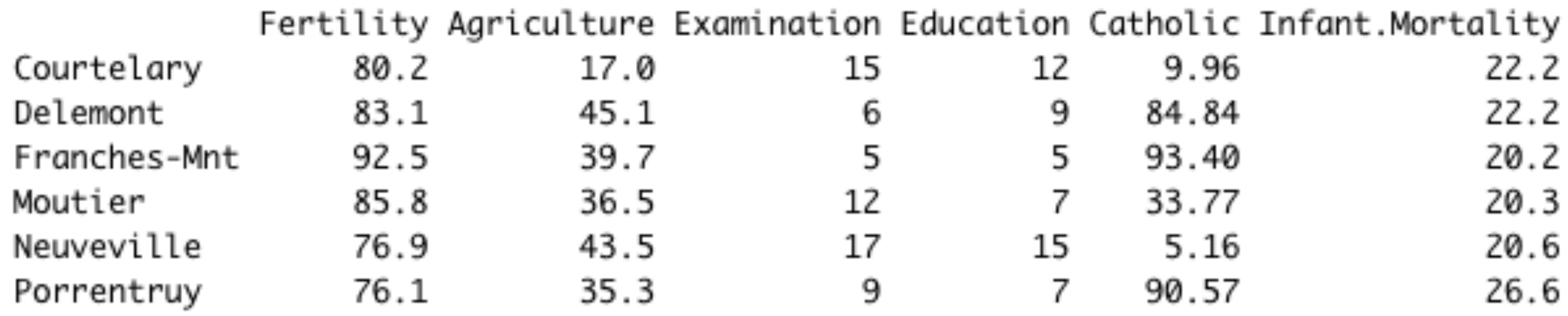
Metric MDS *or Principal Coordinate Analysis* or *Classical Scaling*

possible functions & packages:

cmdscale()	**from stats** by R Development Core Team
wcmdscale()	**from vegan** by Jari Oksanen *et al*
smacofSym()	**from smacof** by Jan de Leeuw and Patrick Mair
pco()	from ecodist by Sarah Goslee and Dean Urban
pco()	from labdsv by David W. Roberts
pcoa()	from ape by Emmanuel Paradis *et al*
dudi.pco()	from ade4 by Daniel Chessel *et al*

swiss dataset

data on 47 French-speaking provinces in Switzerland in 1888



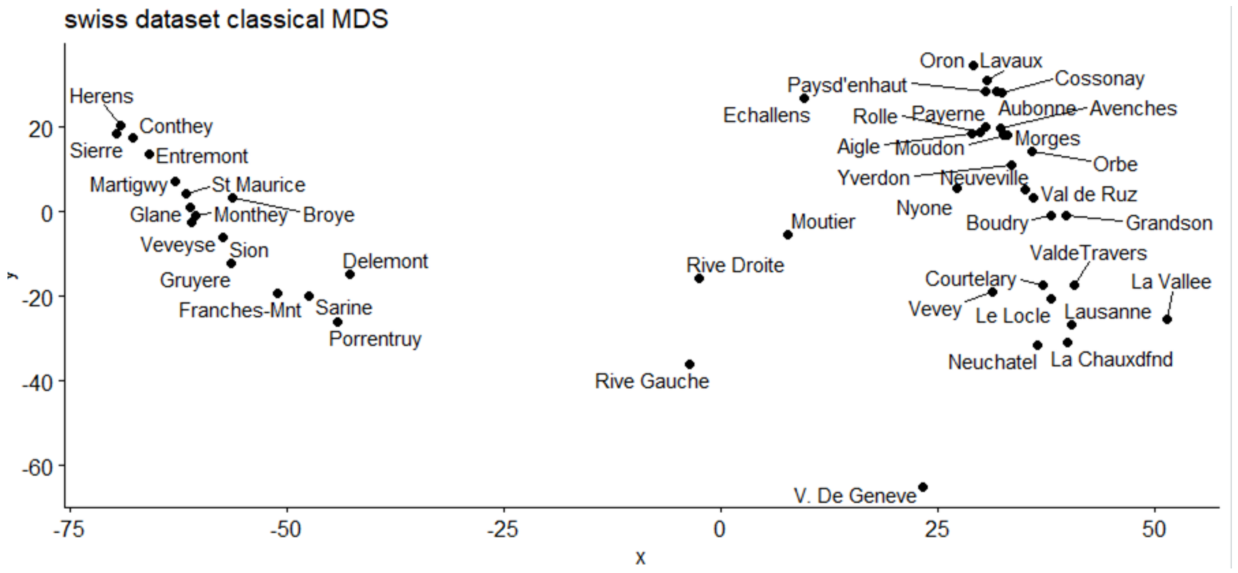
```
             Fertility Agriculture Examination Education Catholic Infant.Mortality
Courtelary        80.2        17.0          15        12     9.96             22.2
Delemont          83.1        45.1           6         9    84.84             22.2
Franches-Mnt      92.5        39.7           5         5    93.40             20.2
Moutier           85.8        36.5          12         7    33.77             20.3
Neuveville        76.9        43.5          17        15     5.16             20.6
Porrentruy        76.1        35.3           9         7    90.57             26.6
```

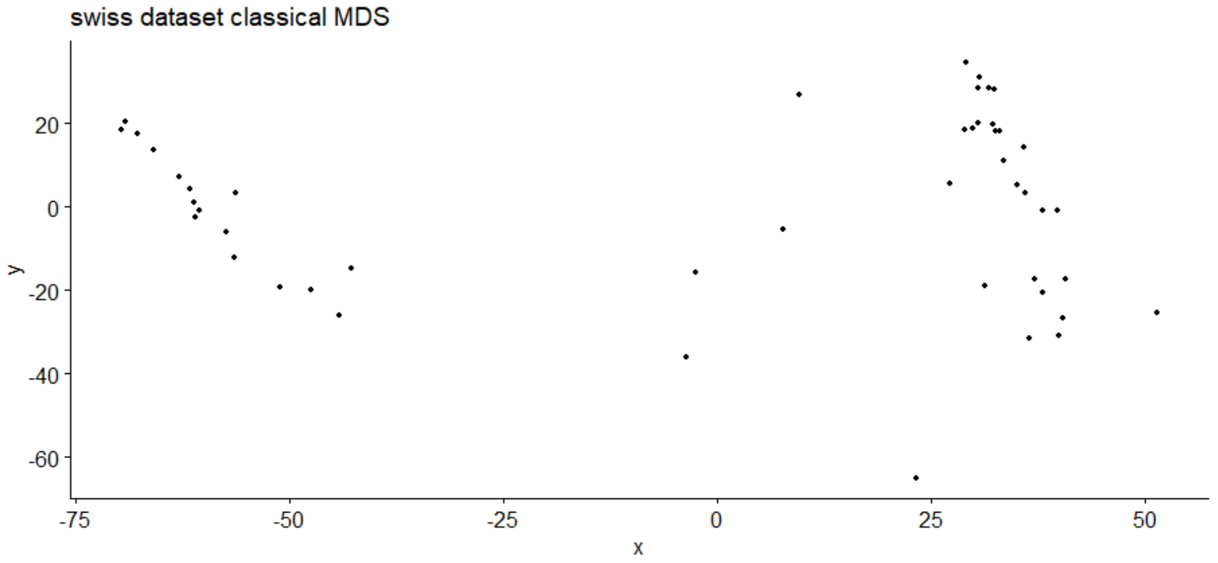
fertility:	*Ig* (common standardized fertility measure)
agriculture:	% males involved in agriculture as occupation
examination:	% draftees receiving highest mark on army exam
education:	% education beyond primary school for draftees
Catholic:	% Catholic (opposed to Protestant)
infant.mortality:	live births living less than one year

Mosteller, F. & Tukey, J.W. (1977). *Data Analysis and Regression: A Second Course in Statistics,* Project “16P5”, pages 549-51. Addison-Wesley, Reading Mass.

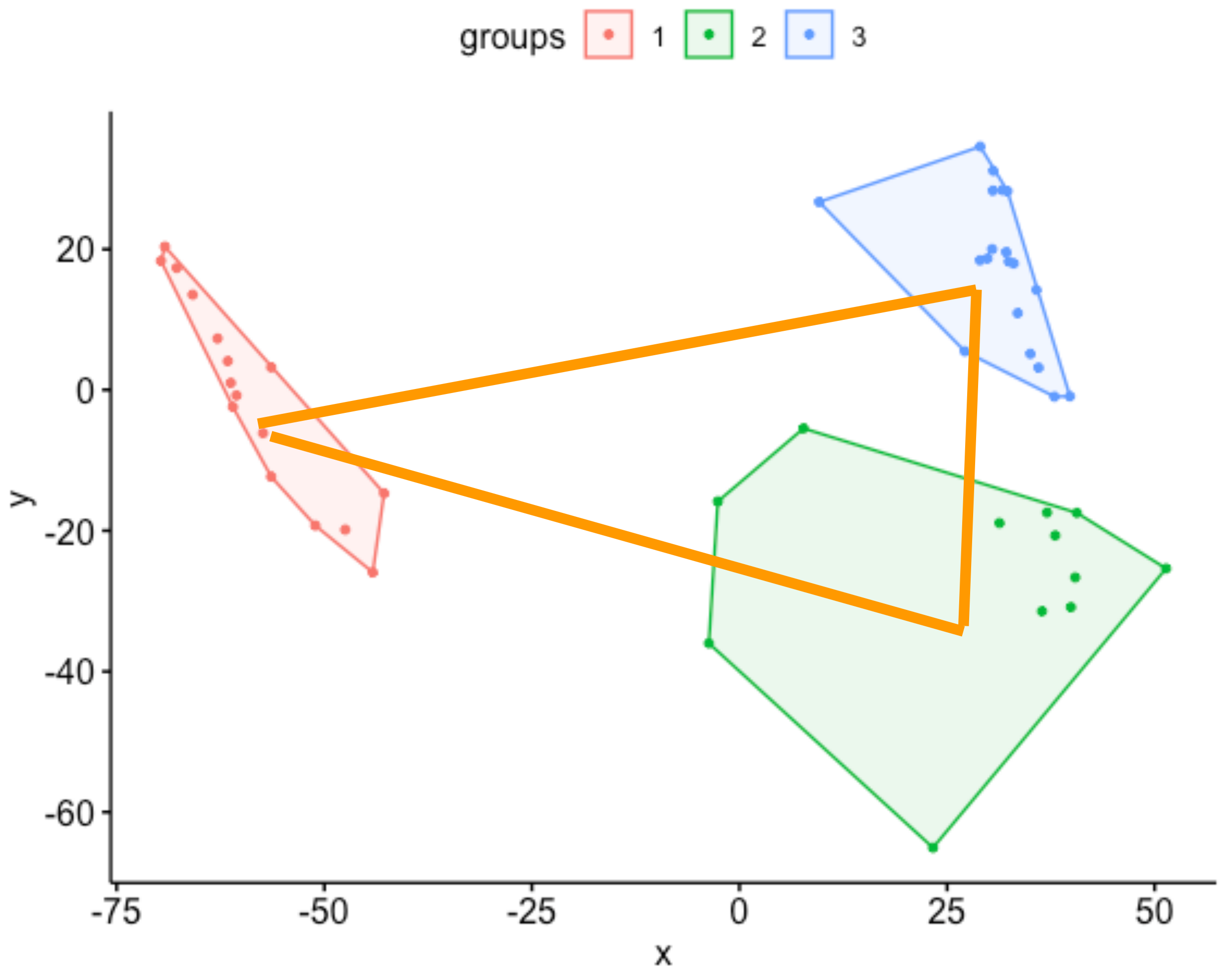
Example with cmdscale() -- swiss

```r
d <- dist(swiss) #Euclidean
mds <- cmdscale(d) # default: k = 2
mds_data <- as_tibble(mds)
colnames(mds_data) <- c("Dim.1","Dim.2")

# Plot MDS
# Plot1 with label & Plot2 without label
ggpubr::ggscatter(mds_data, x = "Dim.1", y = "Dim.2",
          label = rownames(swiss),
          xlab = "x",
          ylab = "y",
          title = "swiss dataset classical MDS",
          size = 2,
          repel = TRUE)

ggpubr::ggscatter(mds_data, x = "Dim.1", y = "Dim.2",
                xlab = "x",
                ylab = "y",
                title = "swiss dataset classical MDS",
                size = 1,
                repel = TRUE)
```





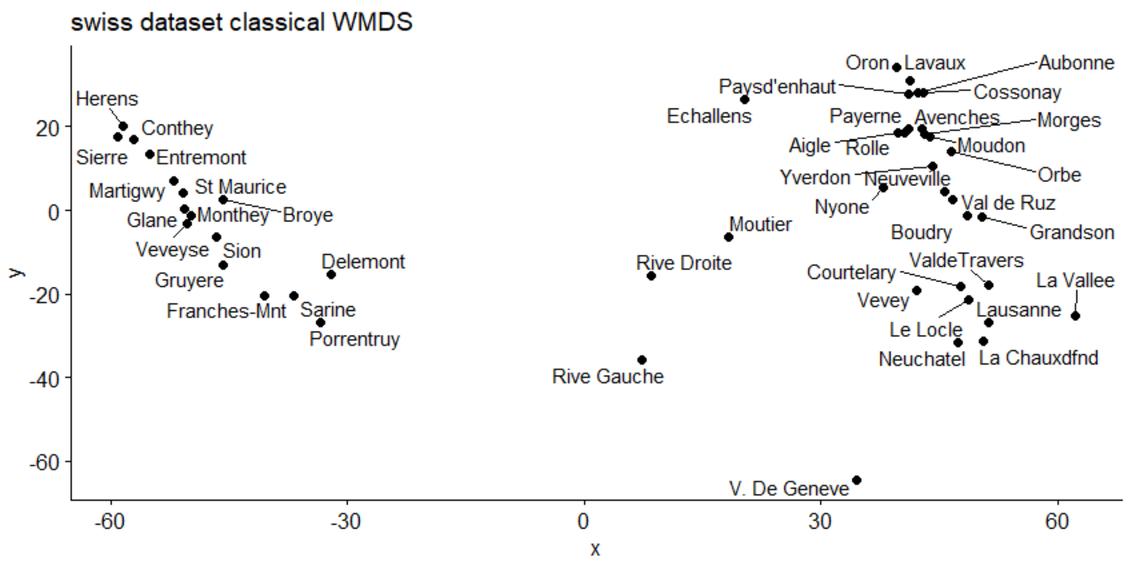
Example with cmdscale() -- swiss

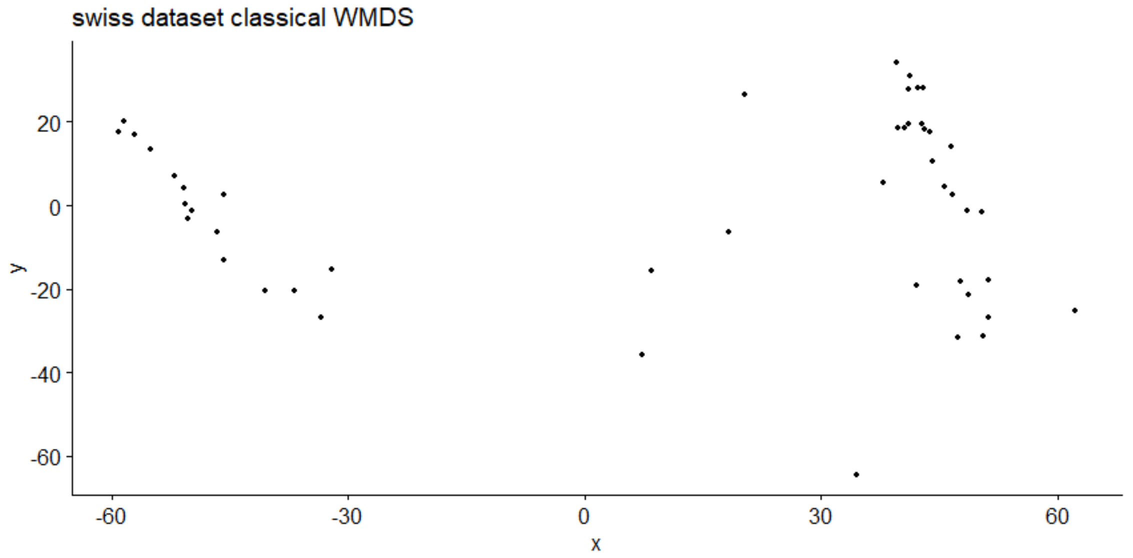


```
# Plot MDS
# Plot clusters
clust <- kmeans(mds_data, 3)$cluster %>%
  as.factor()
mds <- mds_data %>%
  mutate(groups = clust)
# Plot and color
ggpubr::ggscatter(mds, x = "Dim.1", y = "Dim.2",
          color = "groups",
          xlab = "x",
          ylab = "y",
          size = 1,
          ellipse = TRUE,
          ellipse.type = "convex",
          repel = TRUE)
```

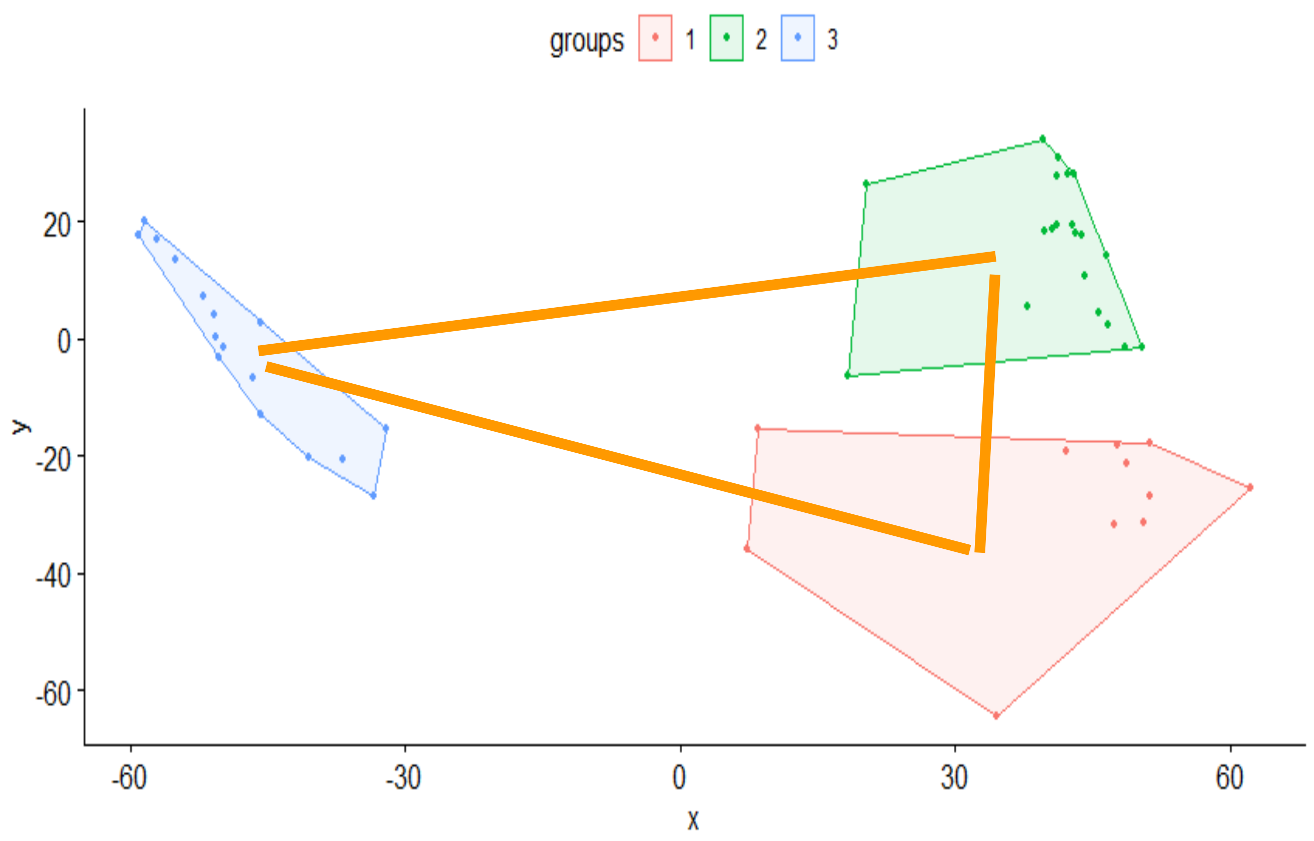
Example with wcmdscale() -- swiss

```r
weight <- rowSums(swiss)/sum(swiss) #weight of each row(ob) in swiss

d <- dist(swiss) #Euclidean
wmds <- wcmdscale(d, w = weight) #default: k = 2
wmds_data <- as_tibble(wmds)
colnames(wmds_data) <- c("Dim.1", "Dim.2")


# Plot WMDS
# Plot1 with label and Plot2 without label
ggpubr::ggscatter(wmds_data, x = "Dim.1", y = "Dim.2",
                  label = rownames(swiss),
                  xlab = "x",
                  ylab = "y",
                  title = "swiss dataset classical WMDS",
                  size = 2,
                  repel = TRUE)

ggpubr::ggscatter(wmds_data, x = "Dim.1", y = "Dim.2",
                  xlab = "x",
                  ylab = "y",
                  title = "swiss dataset classical WMDS",
                  size = 1,
                  repel = TRUE)
```





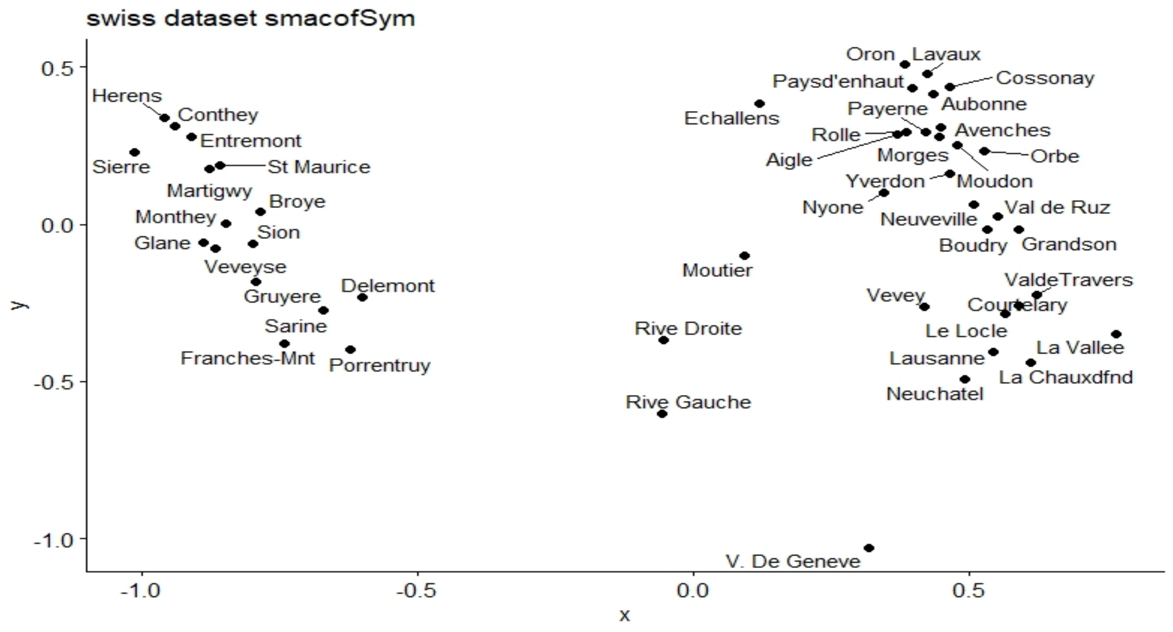
Example with wcmdscale() -- swiss

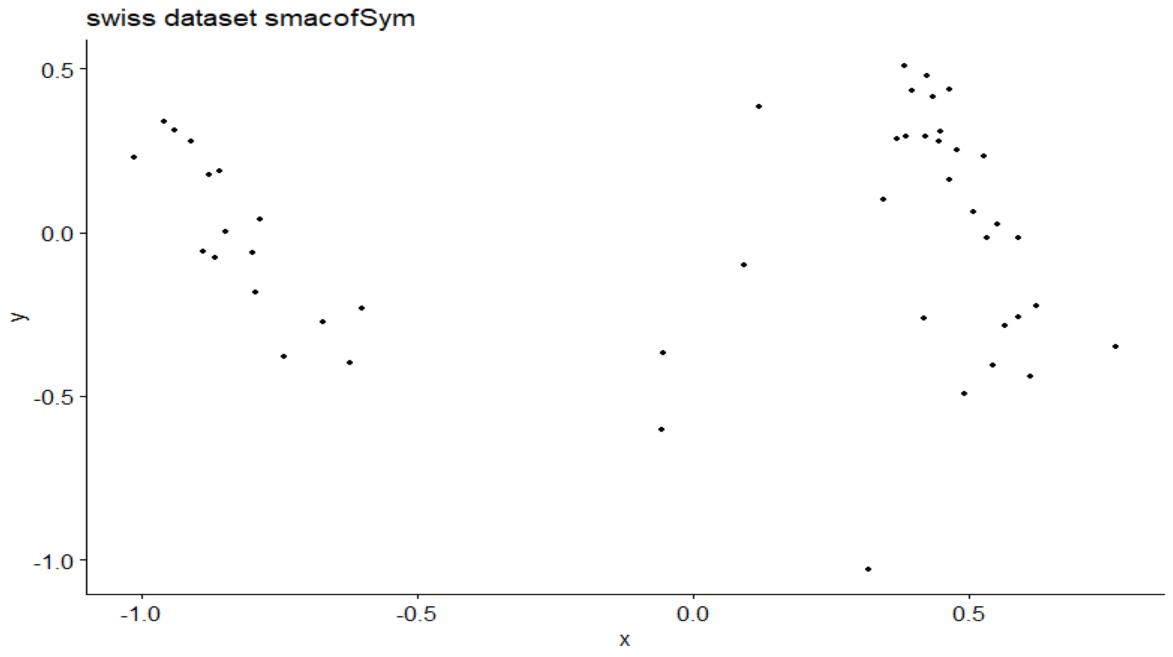


```
# Plot WMDS
# Plot Clusters
wclust <- kmeans(wmds_data, 3)$cluster %>%
  as.factor()
wmds <- wmds_data[,1:2] %>%
  mutate(groups = wclust)
# Plot and color
ggpubr::ggscatter(wmds, x = "Dim.1", y = "Dim.2",
                  color = "groups",
                  xlab = "x",
                  ylab = "y",
                  size = 1,
                  ellipse = TRUE,
                  ellipse.type = "convex",
                  repel = TRUE)
```

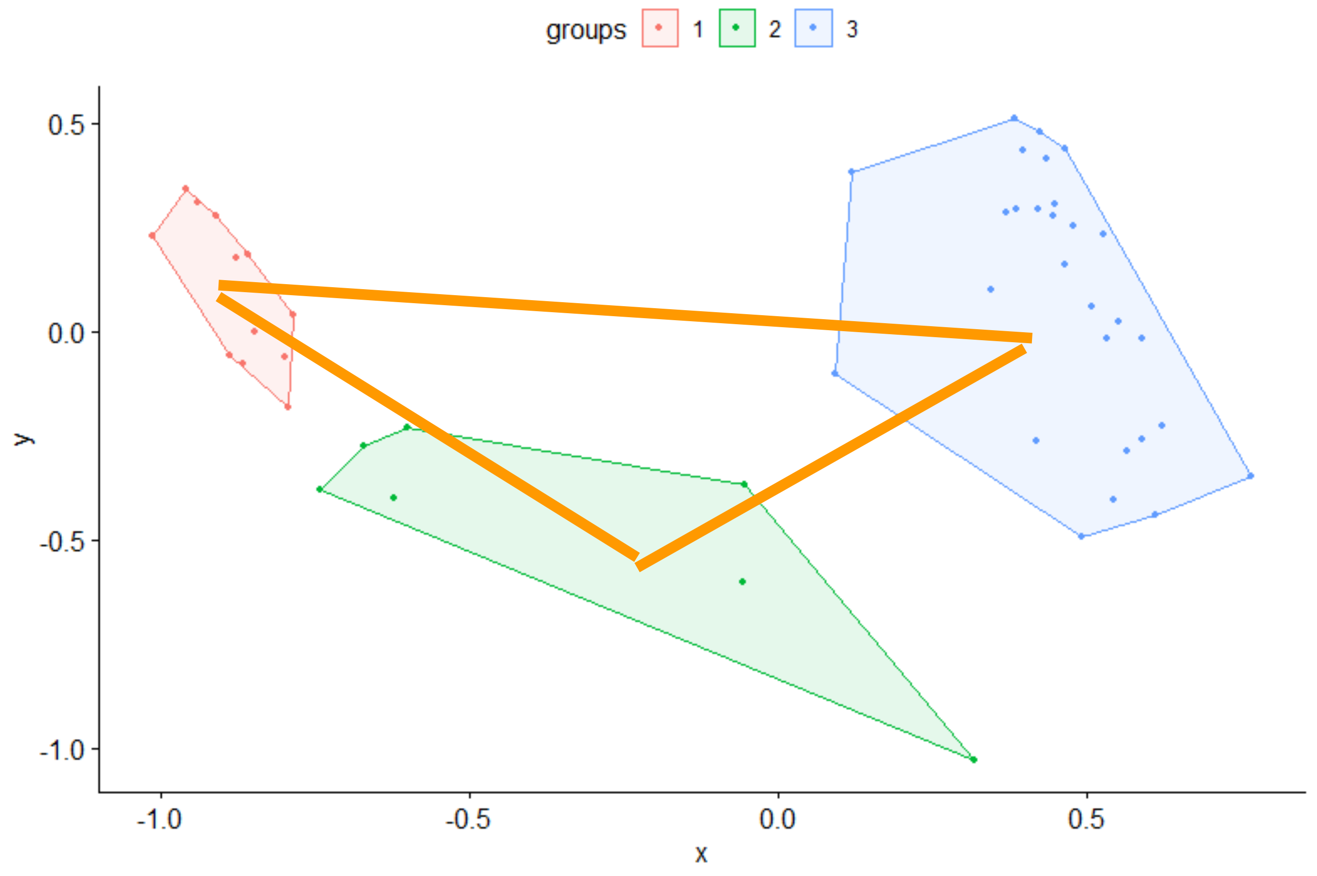
Example with smacofSym() -- swiss

```
d <- dist(swiss)
smds <- smacofSym(d, ndim=2)
#summary(mds)
smds_data <- as_tibble(smds$conf)
colnames(smds_data) <- c("Dim.1","Dim.2")

# Plot MDS
# Plot1 with label & Plot2 without label
ggpubr::ggscatter(smds_data, x = "Dim.1", y = "Dim.2",
                  label = rownames(swiss),
                  xlab = "x",
                  ylab = "y",
                  title = "swiss dataset smacofSym",
                  size = 2,
                  repel = TRUE)

ggpubr::ggscatter(smds_data, x = "Dim.1", y = "Dim.2",
                  xlab = "x",
                  ylab = "y",
                  title = "swiss dataset smacofSym",
                  size = 1,
                  repel = TRUE)
```





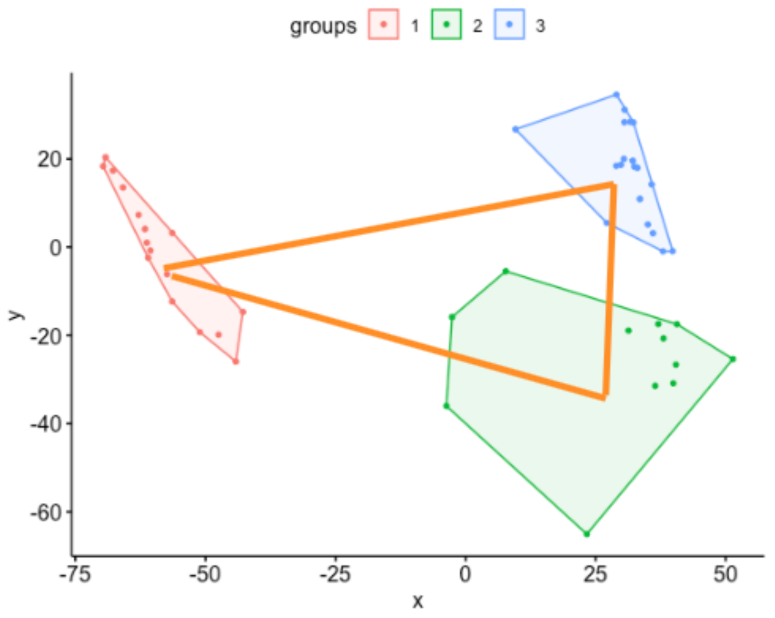
Example with smacofSym() -- swiss



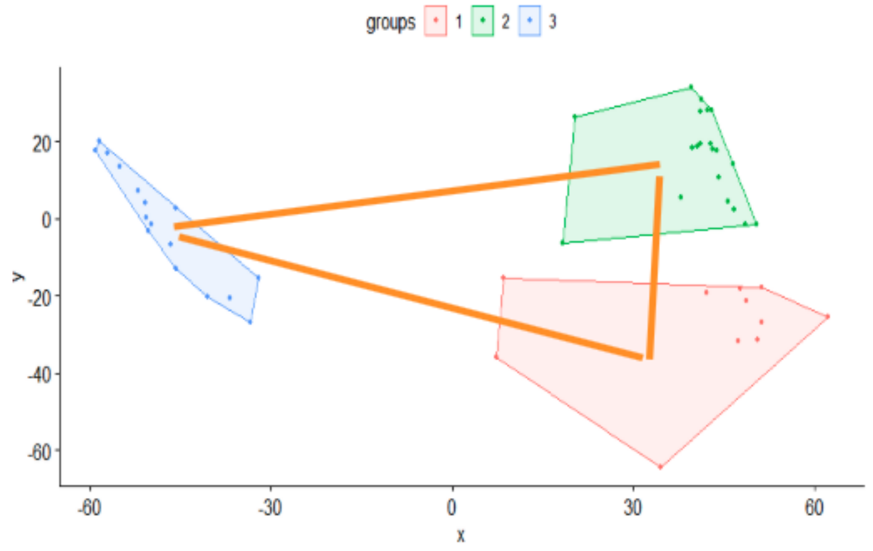
```
# Plot SMDS
# Plot Clusters
sclust <- kmeans(smds_data, 3)$cluster %>%
  as.factor()

smds <- smds_data %>%
  mutate(groups = sclust)

# Plot and color
ggpubr::ggscatter(smds, x = "Dim.1", y = "Dim.2",
                  color = "groups",
                  xlab = "x",
                  ylab = "y",
                  size = 1,
                  ellipse = TRUE,
                  ellipse.type = "convex",
                  repel = TRUE)
```

Comparison with cmdscale() wcmdscale() smacofSym()

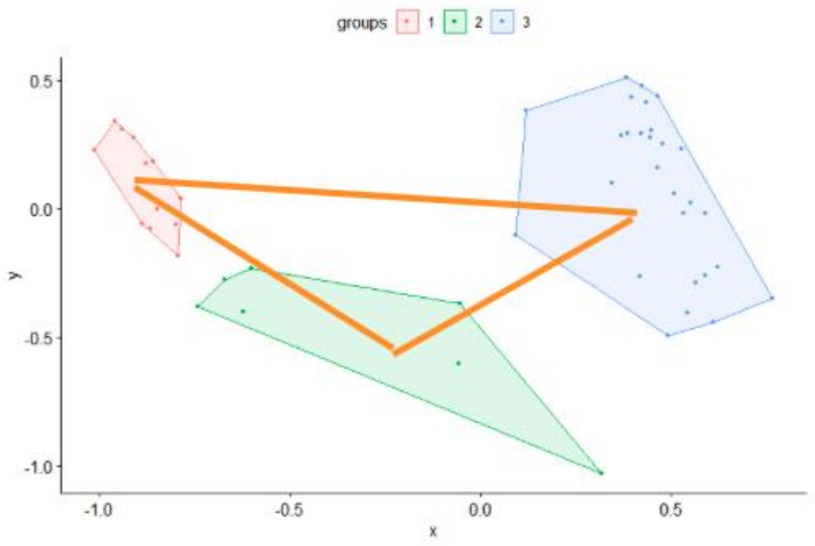
Example with cmdscale() -- swiss



Example with wcmdscale() -- swiss

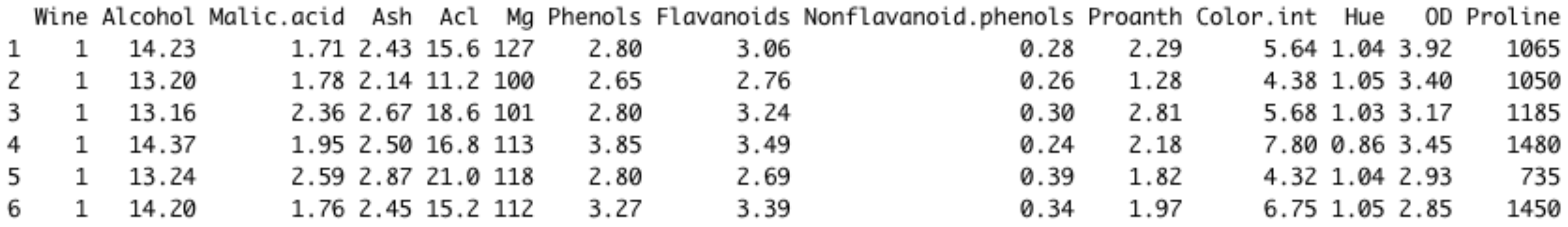


Example with smacofSym() -- swiss



wine dataset

data on 3 types (classes) of wine grown in same region in Italy from 3 different cultivars



```
  Wine Alcohol Malic.acid  Ash  Acl  Mg Phenols Flavanoids Nonflavanoid.phenols Proanth Color.int  Hue   OD Proline
1    1   14.23       1.71 2.43 15.6 127    2.80       3.06                 0.28    2.29      5.64 1.04 3.92    1065
2    1   13.20       1.78 2.14 11.2 100    2.65       2.76                 0.26    1.28      4.38 1.05 3.40    1050
3    1   13.16       2.36 2.67 18.6 101    2.80       3.24                 0.30    2.81      5.68 1.03 3.17    1185
4    1   14.37       1.95 2.50 16.8 113    3.85       3.49                 0.24    2.18      7.80 0.86 3.45    1480
5    1   13.24       2.59 2.87 21.0 118    2.80       2.69                 0.39    1.82      4.32 1.04 2.93     735
6    1   14.20       1.76 2.45 15.2 112    3.27       3.39                 0.34    1.97      6.75 1.05 2.85    1450
```

variables: 13 chemical constituents found in wine

1) Alcohol
2) Malic acid
3) Ash
4) Alcalinity of ash
5) Magnesium
6) Total phenols
7) Flavanoids
8) Nonflavanoid phenols
9) Proanthocyanins
10)Color intensity
11)Hue
12)OD280/OD315 of diluted wines
13)Proline

Forina, M. et al., *PARVUS - An Extendible Package for Data Exploration, Classification and Correlation.* Institute of Pharmaceutical and Food Analysis and Technologies, Genoa, Italy.

Example with cmdscale() -- wine

```
mds <- wine[,2:14] %>%
  dist() %>%
  cmdscale() %>%
  as_tibble()
colnames(mds) <- c("Dim.1", "Dim.2")
```

(a) plot with labels:

```
# Plot MDS
# Plot with label
ggpubr::ggscatter(mds, x = "Dim.1", y = "Dim.2",
                  label = wine[,1],
                  xlab = "x",
                  ylab = "y",
                  shape = 2,
                  title = "Wine Dataset Classical MDS",
                  size = 1,
                  repel = TRUE)
```

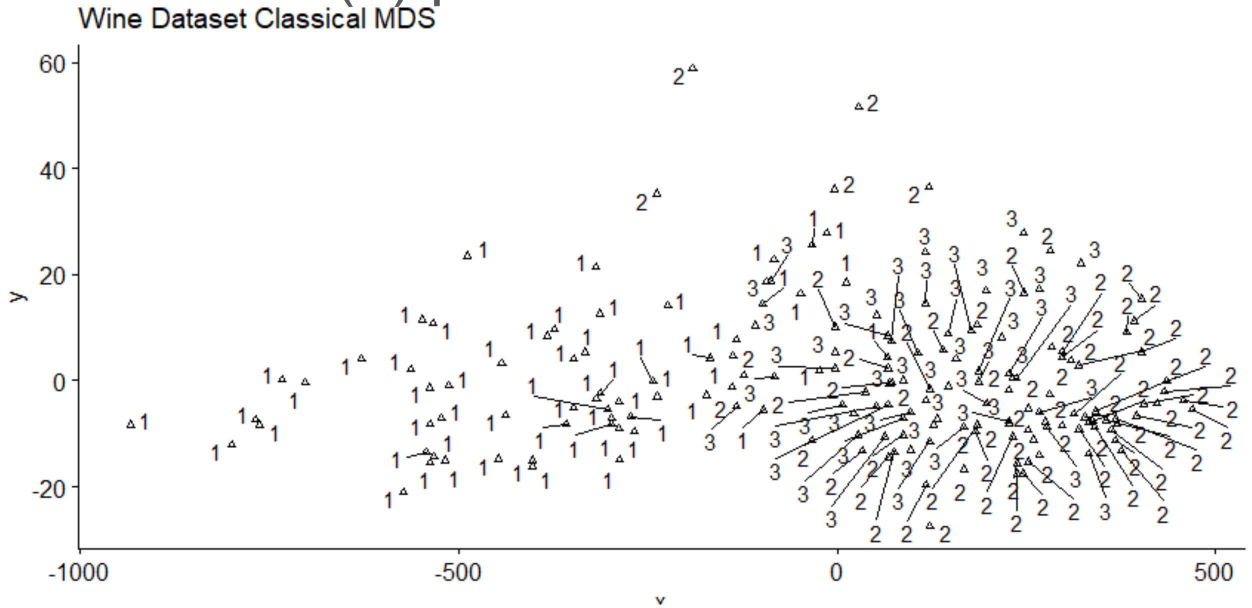
(b) plot clusters:

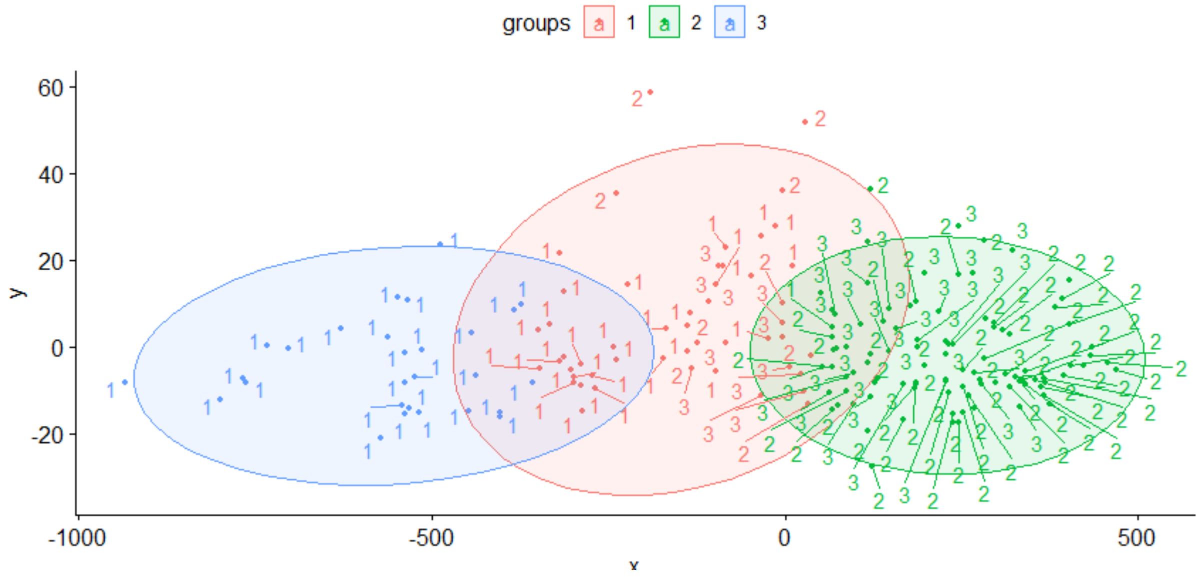
```
# Plot MDS
# Plot with clusters
clust <- kmeans(mds, 3)$cluster %>%
  as.factor()
mds <- mds %>%
  mutate(groups = clust)
class <- wine[,1]
mds$wine <- class
ggpubr::ggscatter(mds, x = "Dim.1", y = "Dim.2",
                  label = "wine",
                  color = "groups",
                  #palette = c("#00AFBB", "#E7B800", "#FC4E07"),
                  xlab = "x",
                  ylab = "y",
                  size = 1,
                  ellipse = TRUE,
                  #ellipse.type = "convex",
                  repel = TRUE)
```

Example with cmdscale() -- wine

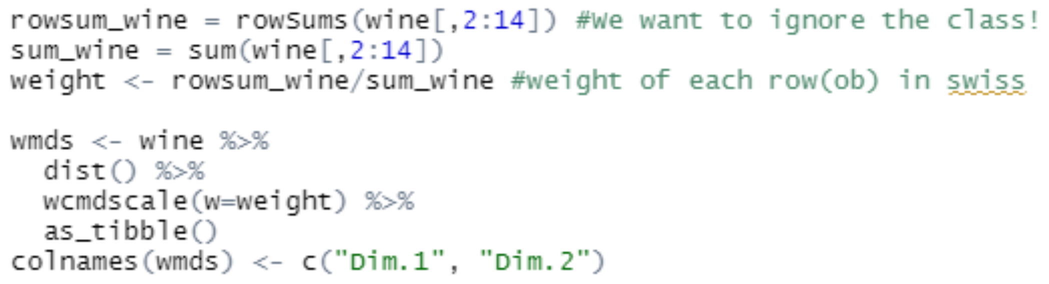
(a) plot with labels:

(b) plot clusters:



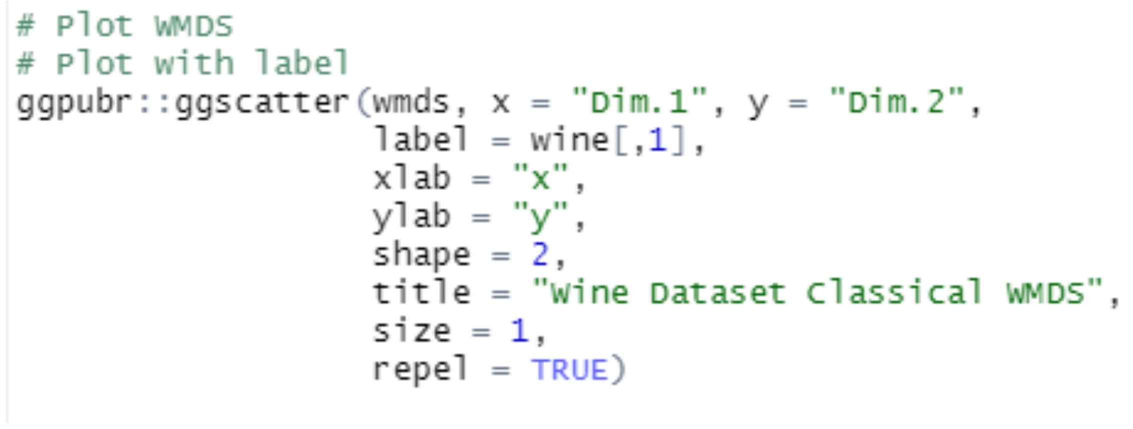


Example with wcmdscale() -- wine



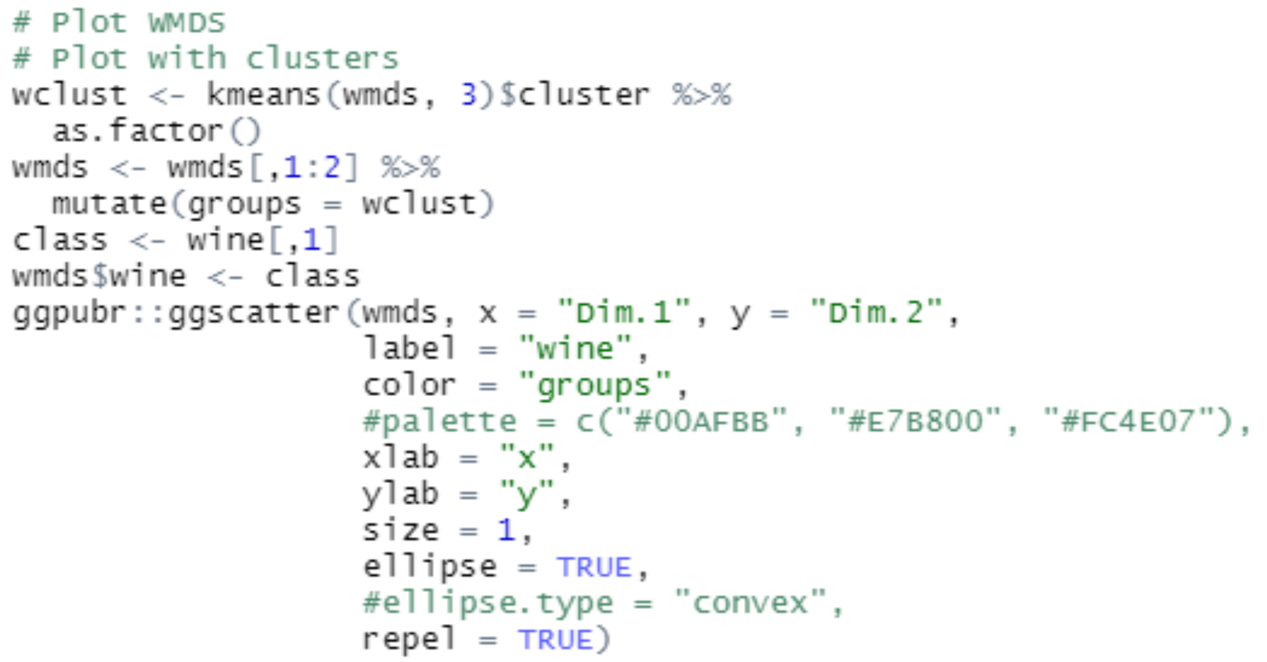
```
rowsum_wine = rowSums(wine[,2:14]) #We want to ignore the class!
sum_wine = sum(wine[,2:14])
weight <- rowsum_wine/sum_wine #weight of each row(ob) in swiss

wmds <- wine %>%
  dist() %>%
  wcmdscale(w=weight) %>%
  as_tibble()
colnames(wmds) <- c("Dim.1", "Dim.2")
```

(a) plot with labels:



```
# Plot WMDS
# Plot with label
ggpubr::ggscatter(wmds, x = "Dim.1", y = "Dim.2",
                  label = wine[,1],
                  xlab = "x",
                  ylab = "y",
                  shape = 2,
                  title = "Wine Dataset Classical WMDS",
                  size = 1,
                  repel = TRUE)
```

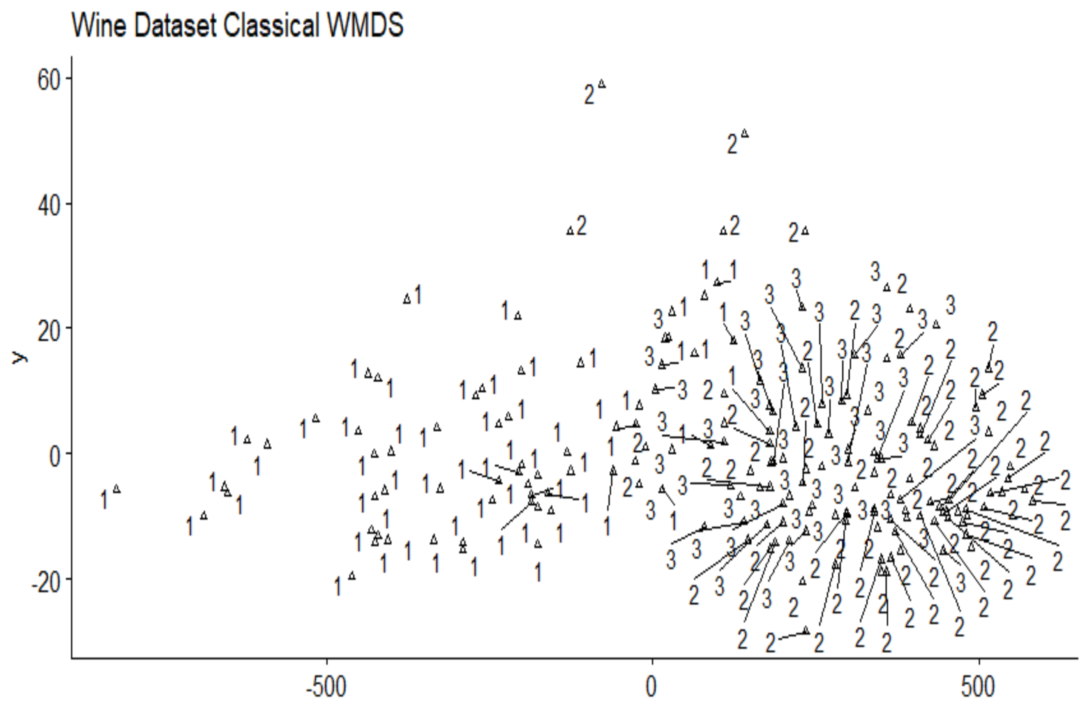
(b) plot clusters:

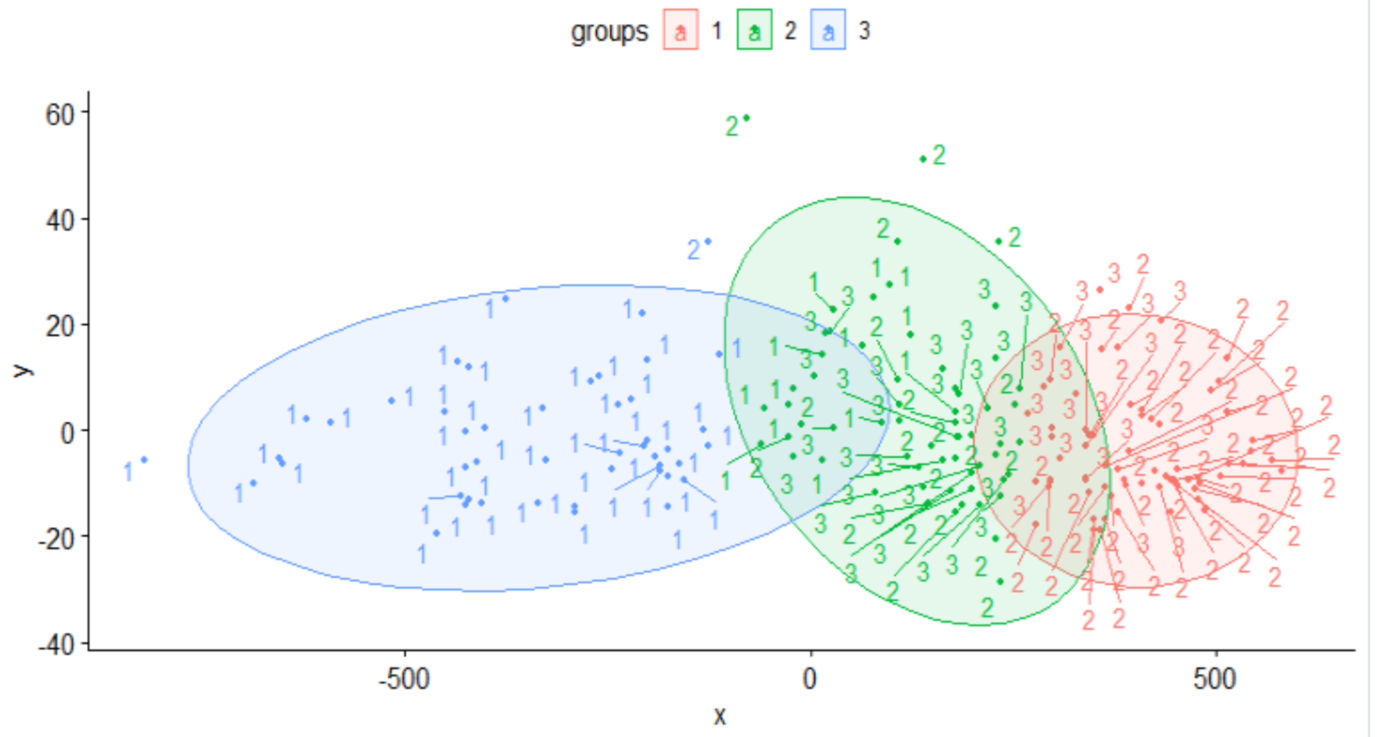


```
# Plot WMDS
# Plot with clusters
wclust <- kmeans(wmds, 3)$cluster %>%
  as.factor()
wmds <- wmds[,1:2] %>%
  mutate(groups = wclust)
class <- wine[,1]
wmds$wine <- class
ggpubr::ggscatter(wmds, x = "Dim.1", y = "Dim.2",
                  label = "wine",
                  color = "groups",
                  #palette = c("#00AFBB", "#E7B800", "#FC4E07"),
                  xlab = "x",
                  ylab = "y",
                  size = 1,
                  ellipse = TRUE,
                  #ellipse.type = "convex",
                  repel = TRUE)
```

Example with wcmdscale() -- wine

(a) plot with labels:





Example with smacofSym() -- wine

```r
d <- dist(wine[,2:14])
smds <- smacofSym(d, ndim=2)
smds_data <- as_tibble(smds$conf)
colnames(smds_data) <- c("Dim.1","Dim.2")
```

(a) plot with labels:

```r
# Plot WMDS
# Plot with label
ggpubr::ggscatter(smds_data, x = "Dim.1", y = "Dim.2",
                  label = wine[,1],
                  xlab = "x",
                  ylab = "y",
                  shape = 2,
                  title = "Wine Dataset Classical WMDS",
                  size = 1,
                  repel = TRUE)
```

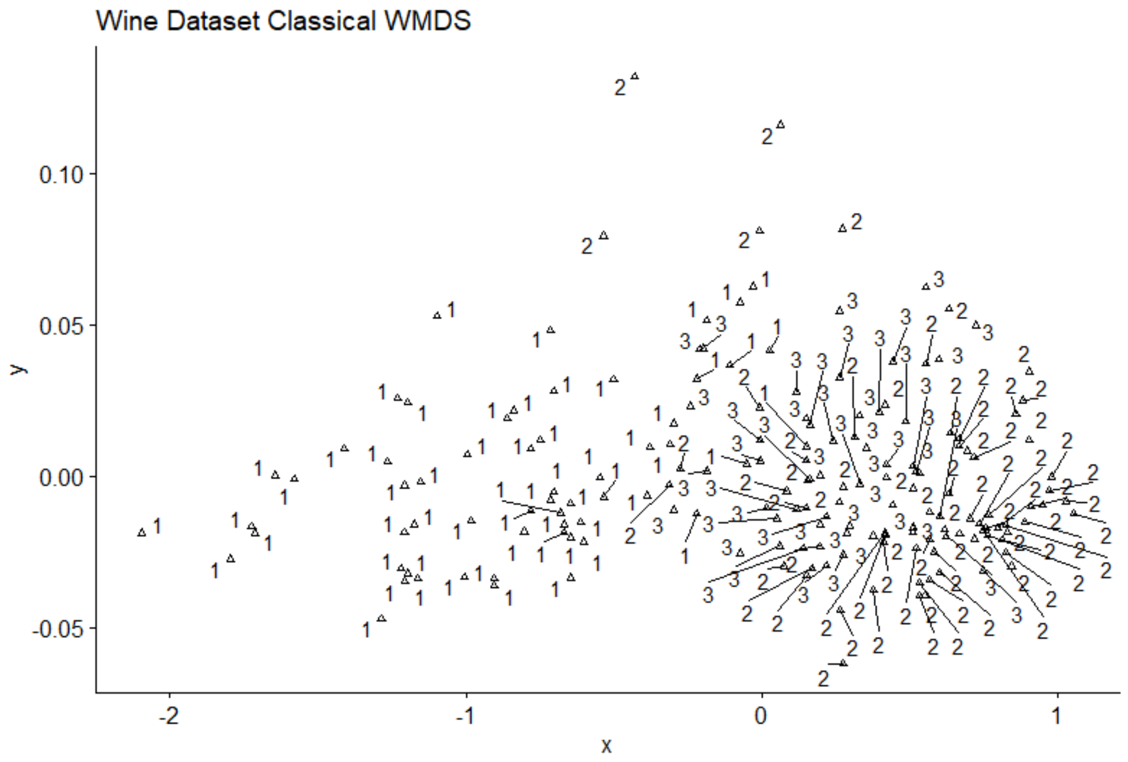
(b) plot clusters:

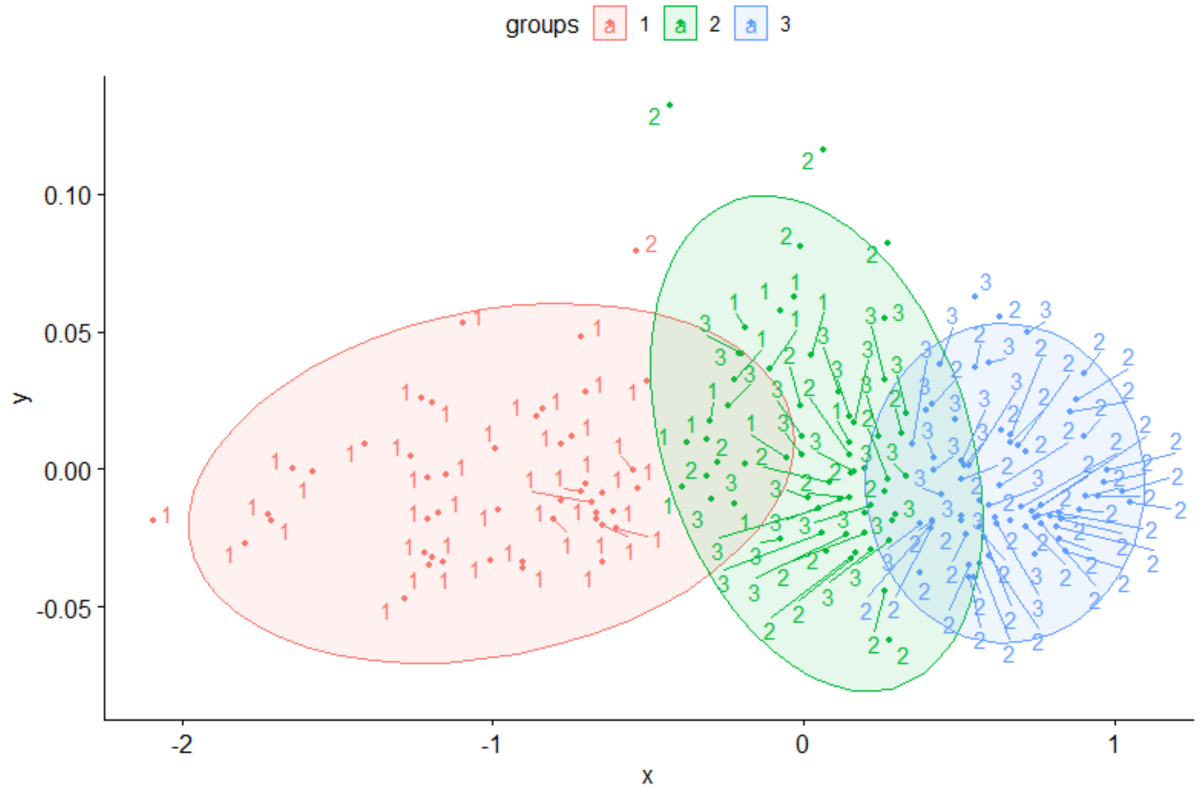
```r
# Plot SMDS
# Plot with clusters
sclust <- kmeans(smds_data, 3)$cluster %>%
  as.factor()
smds <- smds_data %>%
  mutate(groups = sclust)
class <- wine[,1]
smds$wine <- class
ggpubr::ggscatter(smds, x = "Dim.1", y = "Dim.2",
                  label = "wine",
                  color = "groups",
                  #palette = c("#00AFBB", "#E7B800", "#FC4E07"),
                  xlab = "x",
                  ylab = "y",
                  size = 1,
                  ellipse = TRUE,
                  #ellipse.type = "convex",
                  repel = TRUE)
```

Example with smacofSym() -- wine

(a) plot with labels:

(b) plot clusters:

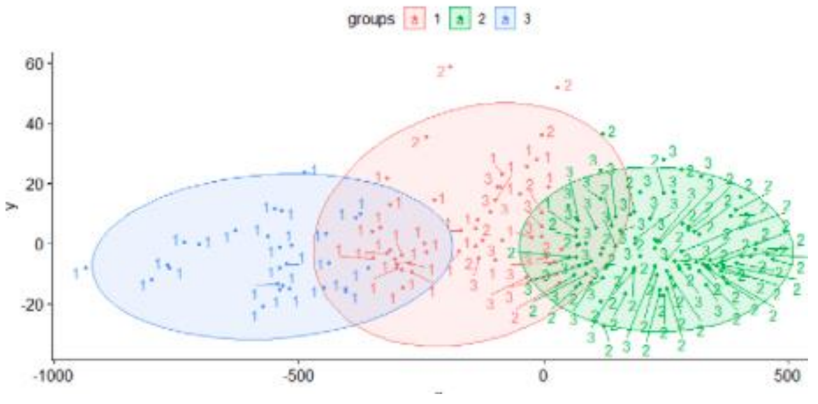




Comparison with cmdscale() wcmdscale() smacofSym()

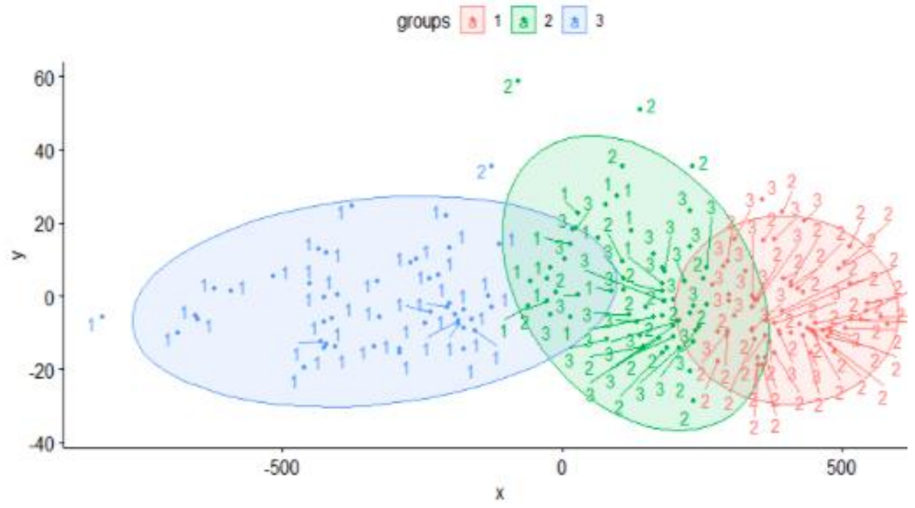
Example with cmdscale() -- wine

(b) plot clusters:



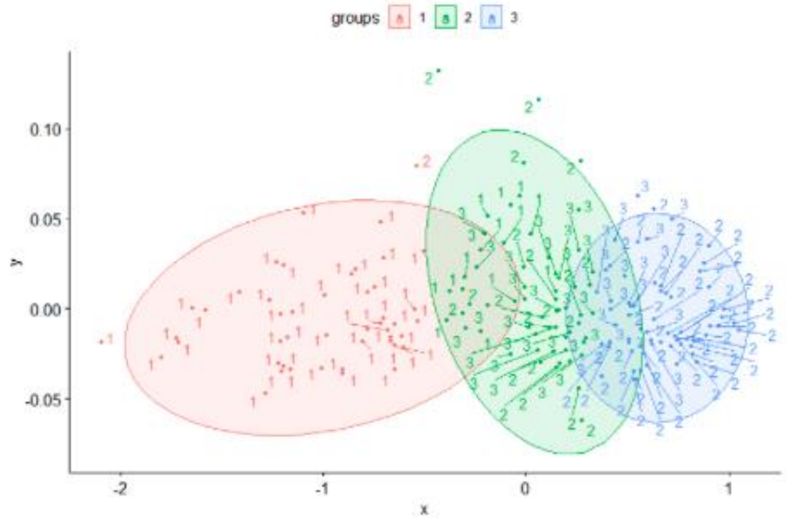
Example with wcmdscale() -- wine

(b) plot clusters:



Example with smacofSym() -- wine

(b) plot clusters:



Non-metric MDS

Step1 Calculate Distance matrix or Dissimilarity matrix from original dataset
Packages: **vegan** - **vegdist()** (for Dissimilarity matrix)
vegan - **rankindex()** (find the best pattern of dissimilarity matrix)
Step2 Reduce dimensions of data
Packages: **MASS** - **isoMDS()** ; **vegan** - **monoMDS()** and **metaMDS()**
Step3 Cluster low-dimensional data from Step2
Packages: **factoextra** - **fviz_cluster()** ; **k-means()** (for cluster)

Dataset:
swiss (fertility and socio-economic data on 47 French-speaking provinces in Switzerland)
wine (quantities of 13 constituents found in each of the 3 types of wines grown in the Same region but derived from 3 different cultivars.

Example with isoMDS() -- swiss

(a) **Euclidean** distance matrix:

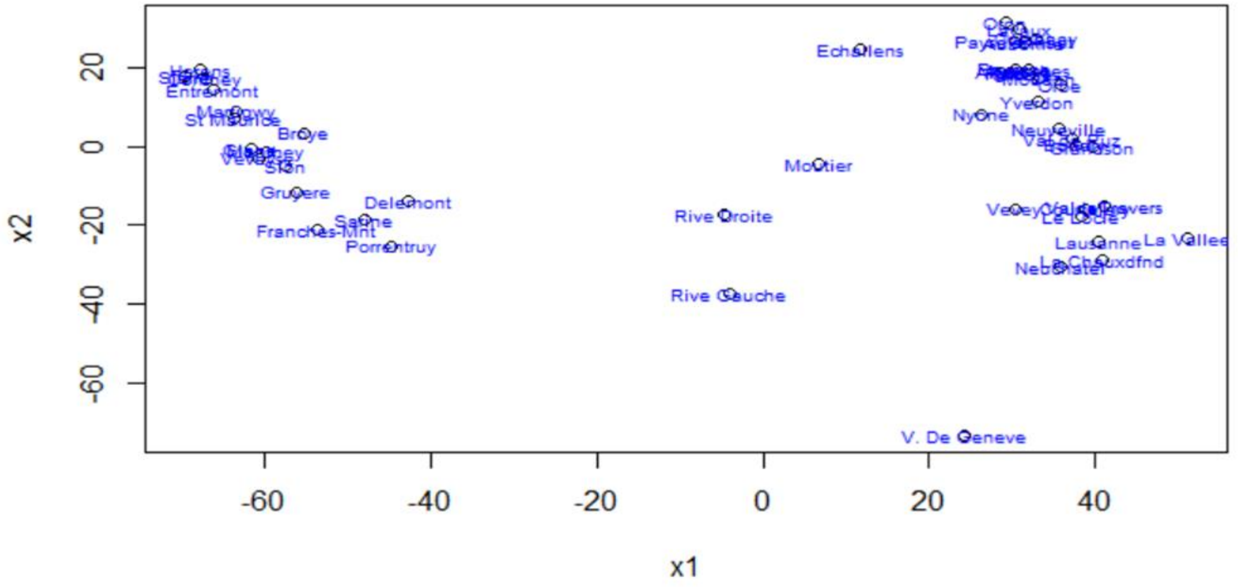
```
d<-dist(data1) #Euclidean
MDS1<-isoMDS(d,k=2)
x1<-MDS1$points[,1]
x2<-MDS1$points[,2]
plot(x1,x2)
x00<-cbind(x1,x2)
text(x1,x2,labels=swiss$Location,col="blue",cex = 0.6, sub="swiss-isoMDS-Euclidean")
km.res = kmeans(x00, 3)
library(facteoextra)## Visualize clusters using factoextra
fviz_cluster(km.res, x00)
```

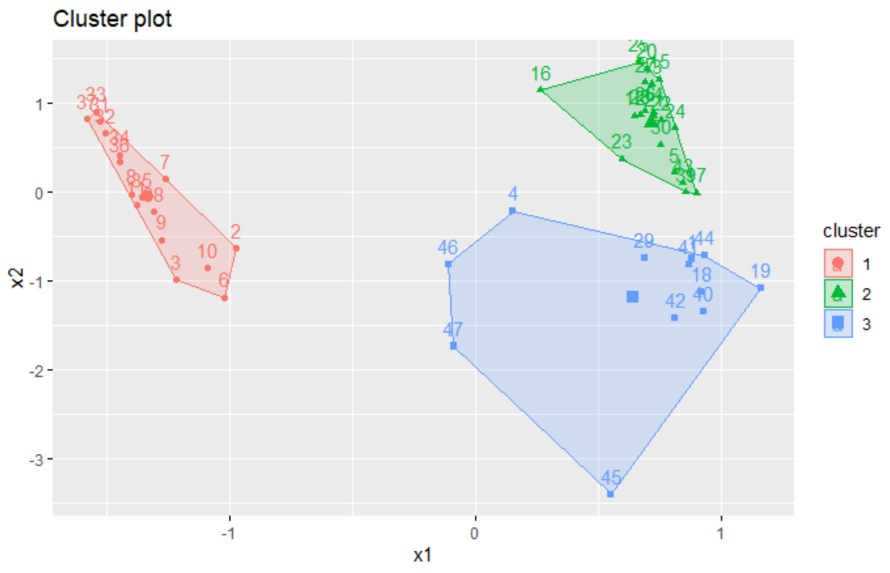
(b)**Bray-Curtis** is a commonly used index when calculate dissimilarity matrix:

```
data1.dis<- vegdist(data1) #Bray-Curtis
MDS1<-isoMDS(data1.dis,k=2)
x1<-MDS1$points[,1]
x2<-MDS1$points[,2]
plot(x1,x2)
x00<-cbind(x1,x2)
text(x1,x2,labels=swiss$Location,col="blue",cex = 0.6, sub="swiss-isoMDS- Bray-Curtis")
km.res = kmeans(x00, 3)
library(facteoextra)## Visualize clusters using factoextra
fviz_cluster(km.res, x00)
```

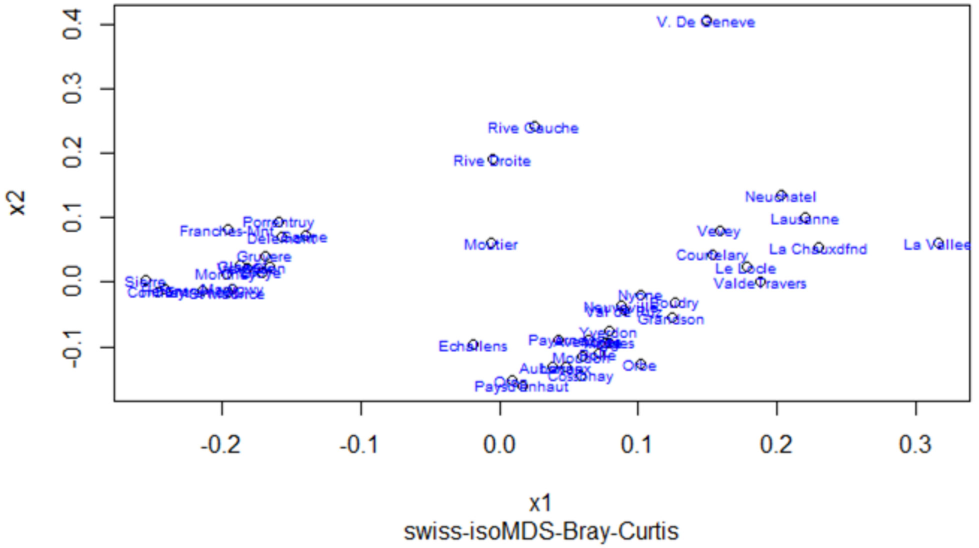
Example with isoMDS() -- swiss

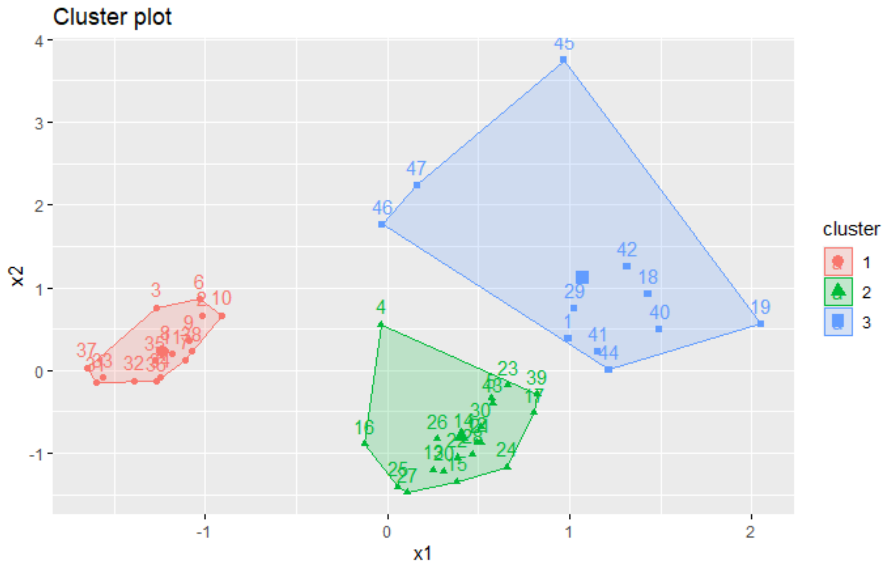
(a) **Euclidean** distance matrix





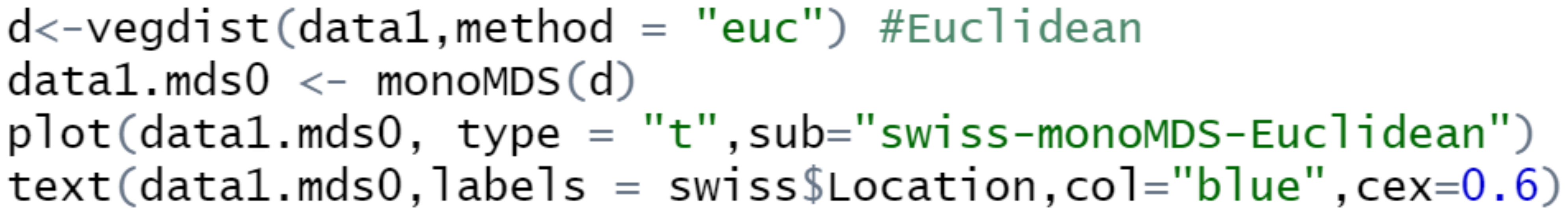
(b) **Bray-Curtis**





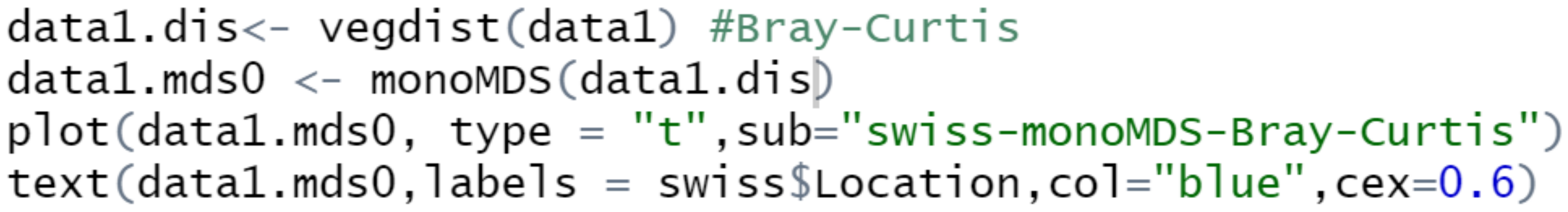
Example with monoMDS() -- swiss

(a) **Euclidean** distance matrix



```
d<-vegdist(data1,method = "euc") #Euclidean
data1.mds0 <- monoMDS(d)
plot(data1.mds0, type = "t",sub="swiss-monoMDS-Euclidean")
text(data1.mds0,labels = swiss$Location,col="blue",cex=0.6)
```

(b) **Bray-Curtis**

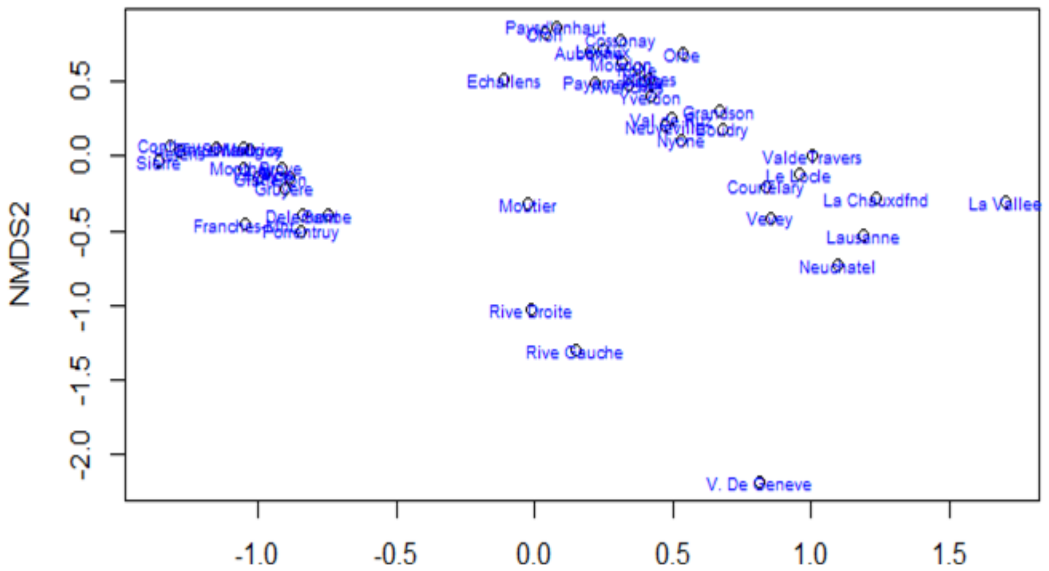


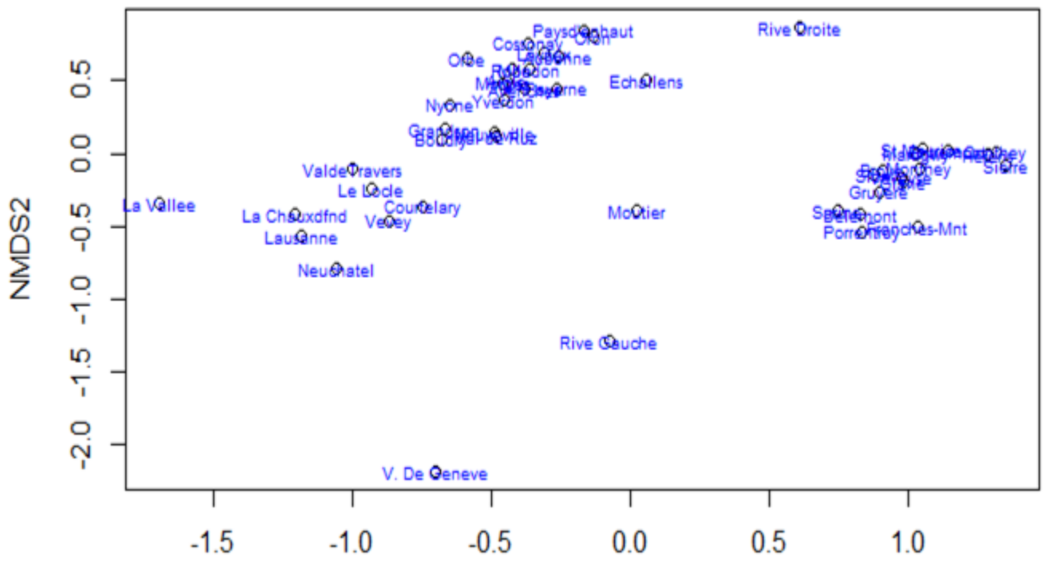
```
data1.dis<- vegdist(data1) #Bray-Curtis
data1.mds0 <- monoMDS(data1.dis)
plot(data1.mds0, type = "t",sub="swiss-monoMDS-Bray-Curtis")
text(data1.mds0,labels = swiss$Location,col="blue",cex=0.6)
```

Example with monoMDS() -- swiss

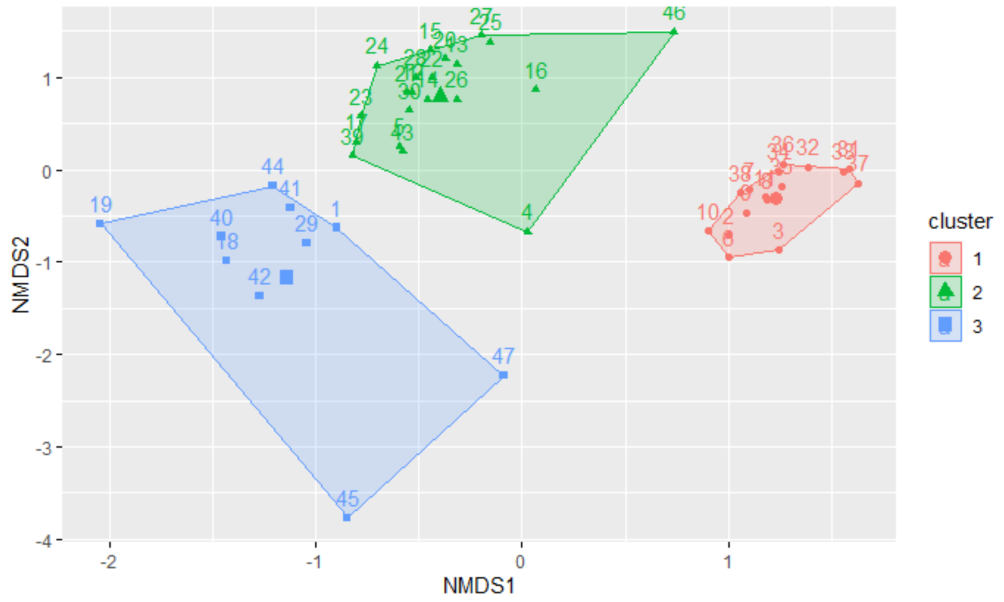
(a) **Euclidean** distance matrix

(b) **Bray-Curtis**

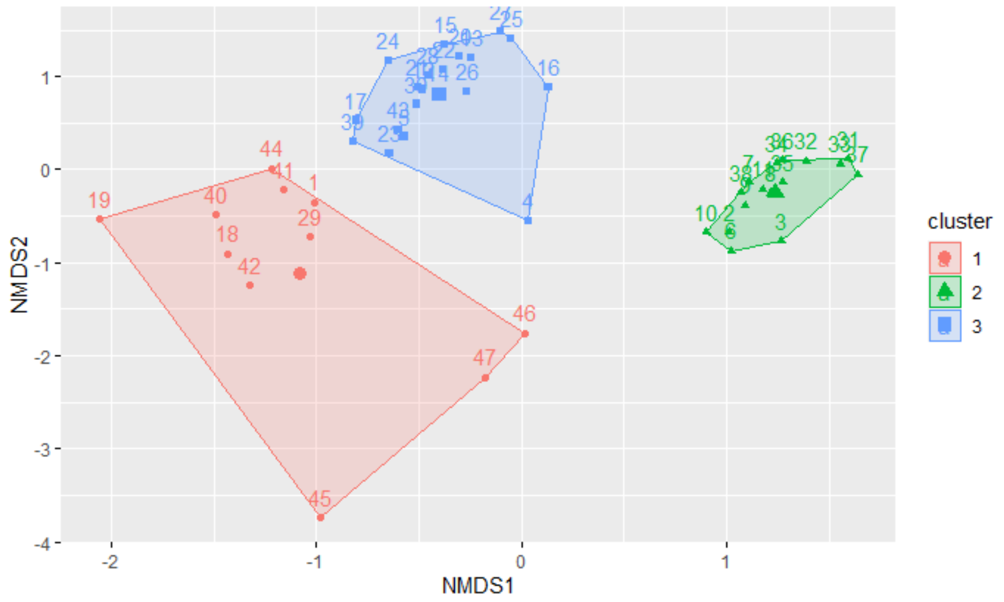




Cluster plot

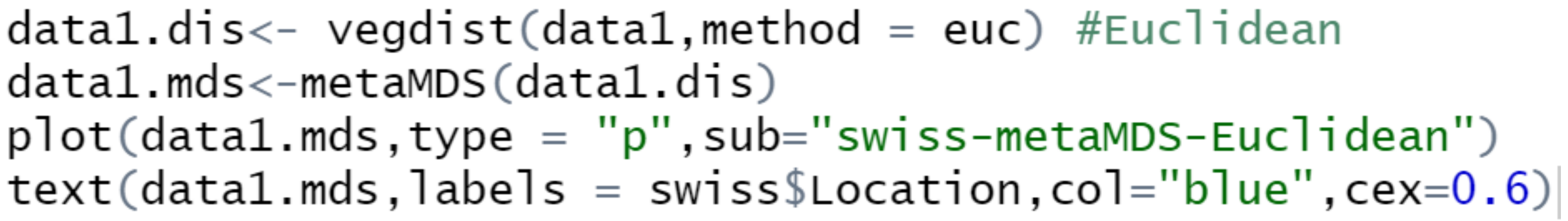


Cluster plot



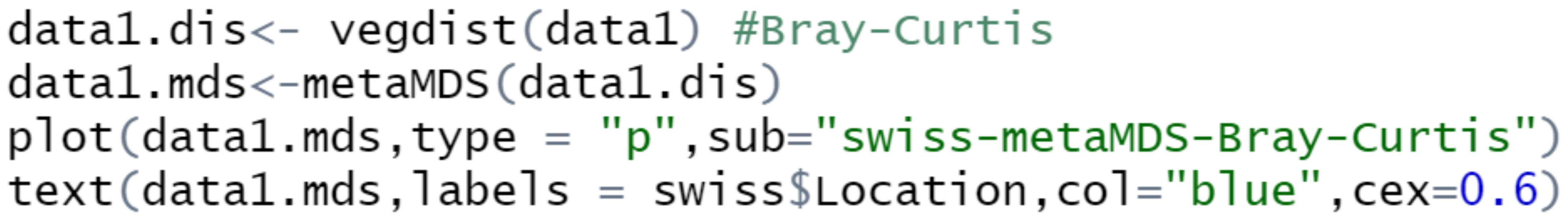
Example with metaMDS() -- swiss

(a) **Euclidean** distance matrix



```r
data1.dis<- vegdist(data1,method = euc) #Euclidean
data1.mds<-metaMDS(data1.dis)
plot(data1.mds,type = "p",sub="swiss-metaMDS-Euclidean")
text(data1.mds,labels = swiss$Location,col="blue",cex=0.6)
```

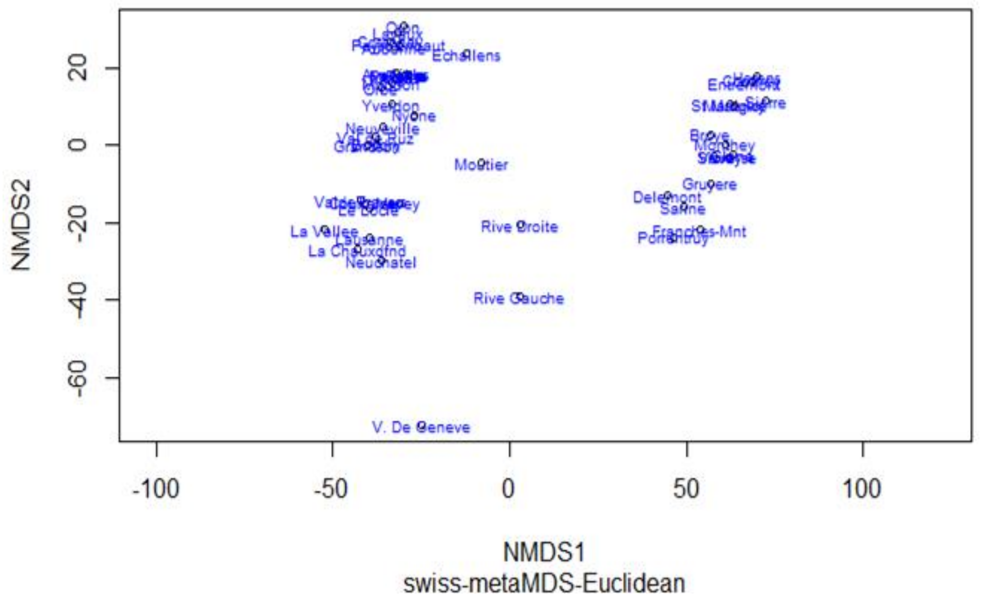
(b) **Bray-Curtis**

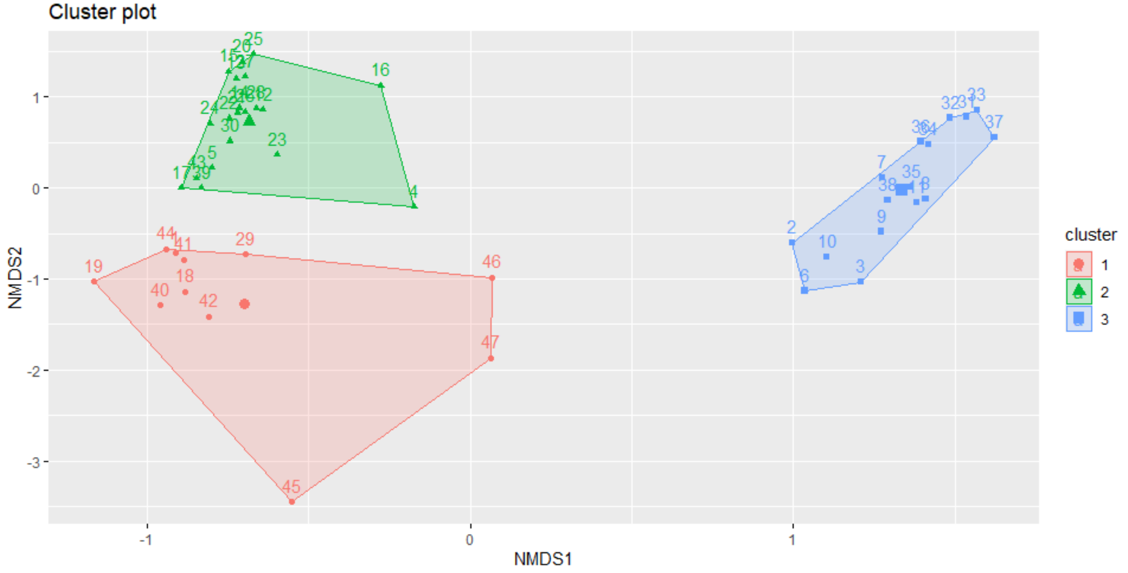


```r
data1.dis<- vegdist(data1) #Bray-Curtis
data1.mds<-metaMDS(data1.dis)
plot(data1.mds,type = "p",sub="swiss-metaMDS-Bray-Curtis")
text(data1.mds,labels = swiss$Location,col="blue",cex=0.6)
```

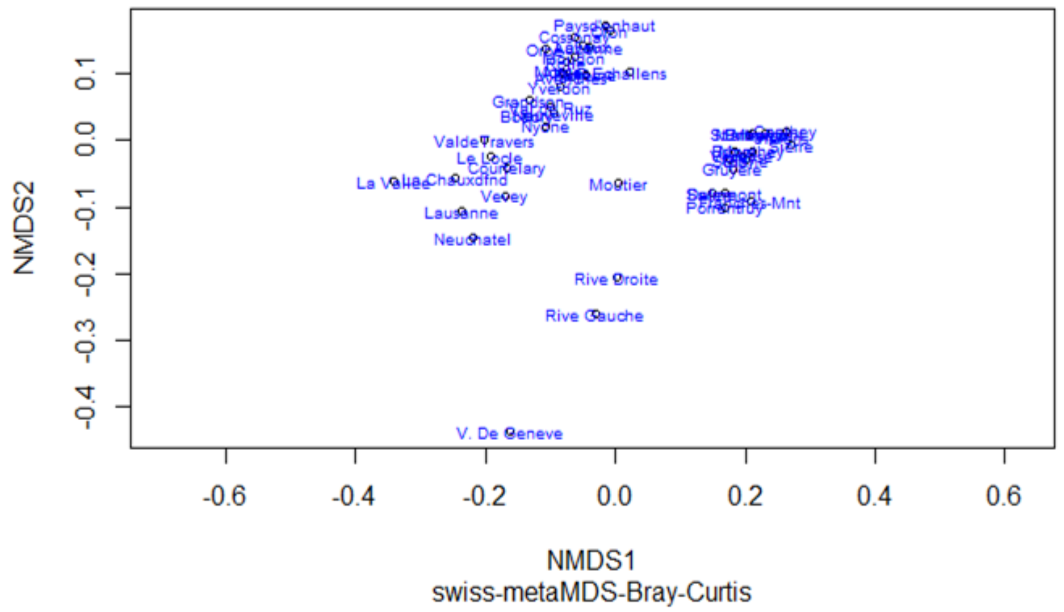
Example with metaMDS() -- swiss

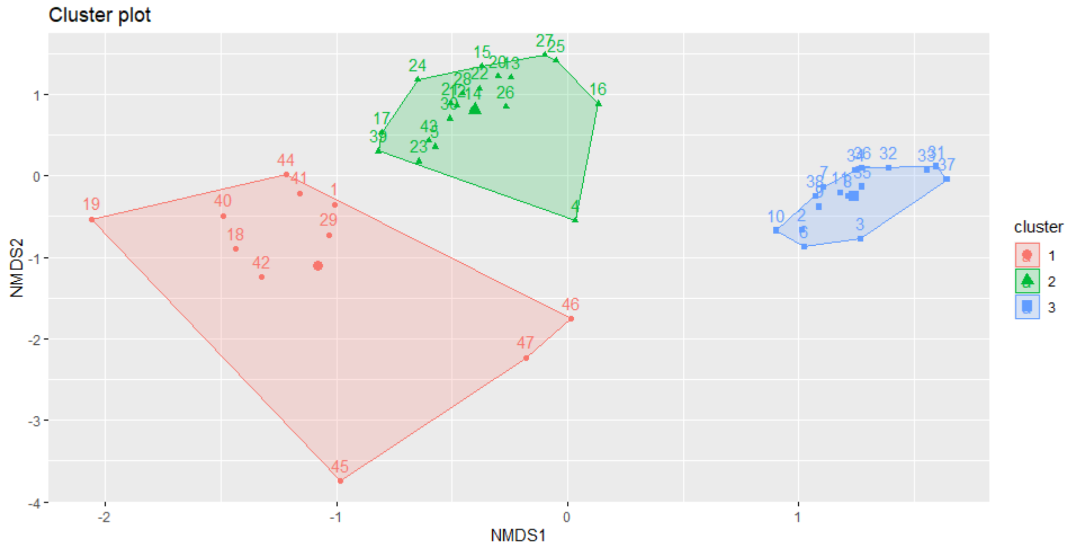
(a) **Euclidean** distance matrix





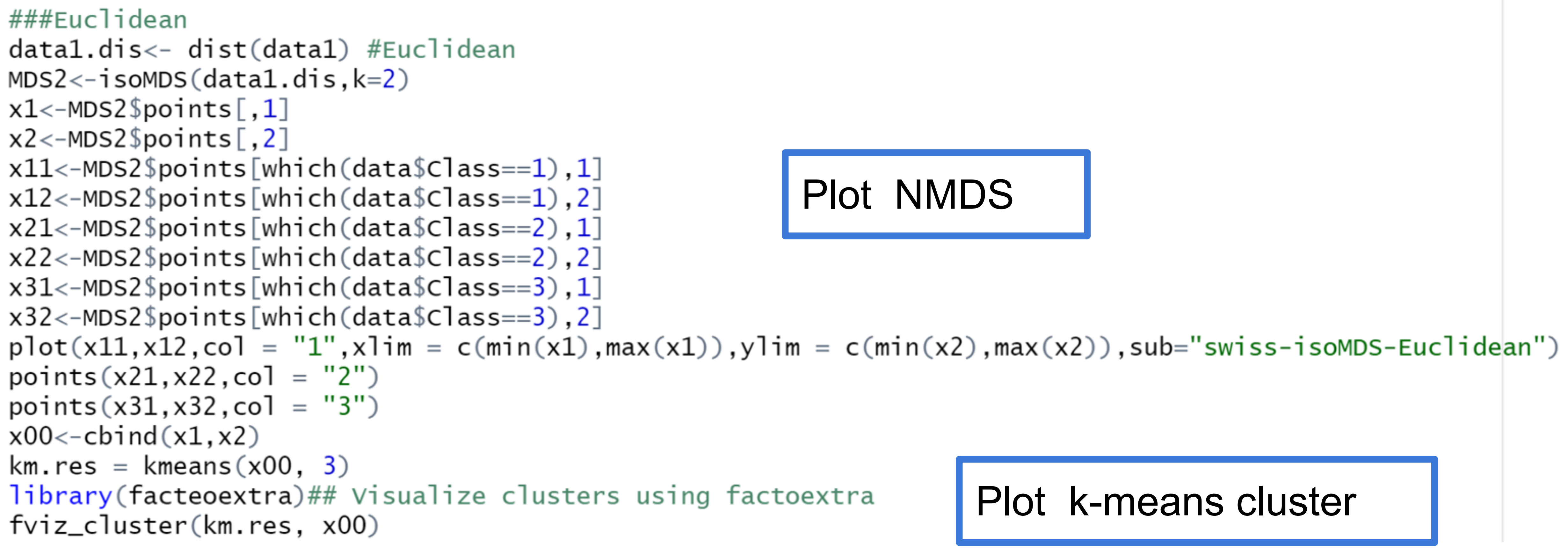
(b) **Bray-Curtis**





Example with isoMDS() -- wine

(a) **Euclidean** distance matrix



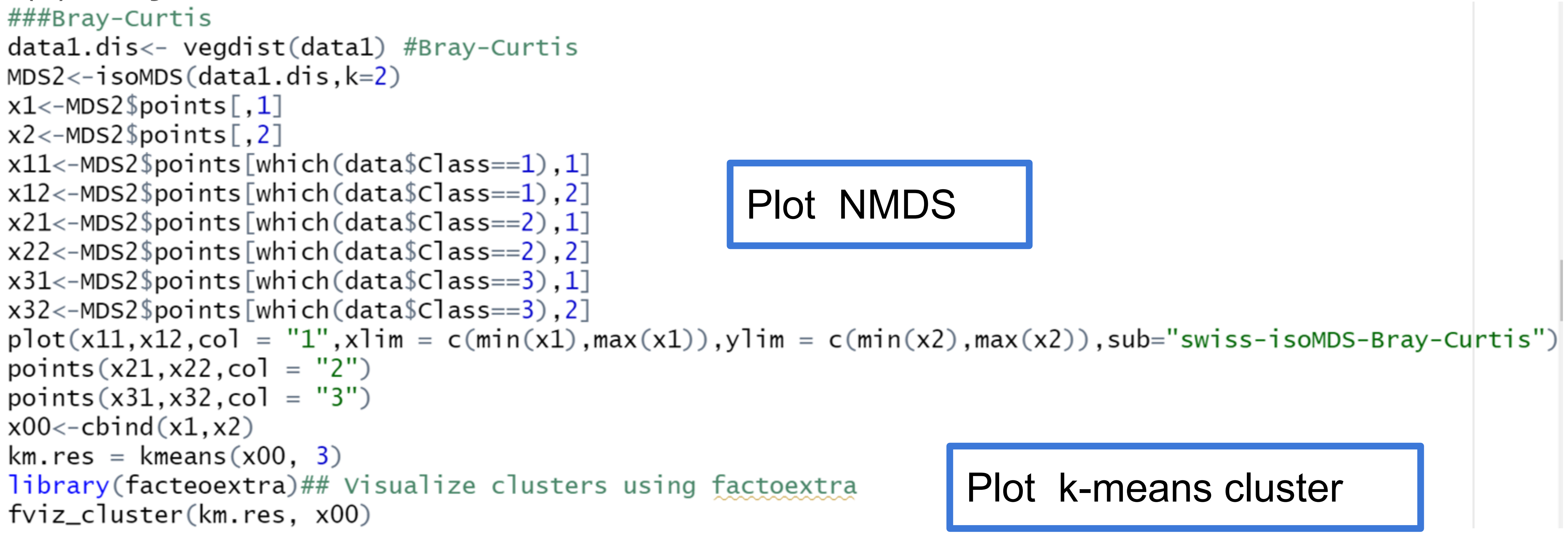
```
###Euclidean
data1.dis<- dist(data1) #Euclidean
MDS2<-isoMDS(data1.dis,k=2)
x1<-MDS2$points[,1]
x2<-MDS2$points[,2]
x11<-MDS2$points[which(data$Class==1),1]
x12<-MDS2$points[which(data$Class==1),2]
x21<-MDS2$points[which(data$Class==2),1]
x22<-MDS2$points[which(data$Class==2),2]
x31<-MDS2$points[which(data$Class==3),1]
x32<-MDS2$points[which(data$Class==3),2]
plot(x11,x12,col = "1",xlim = c(min(x1),max(x1)),ylim = c(min(x2),max(x2)),sub="swiss-isoMDS-Euclidean")
points(x21,x22,col = "2")
points(x31,x32,col = "3")
x00<-cbind(x1,x2)
km.res = kmeans(x00, 3)
library(facteoextra)## Visualize clusters using factoextra
fviz_cluster(km.res, x00)
```

Plot NMDS

Plot k-means cluster

Example with isoMDS() -- wine

(b) **Bray-Curtis**



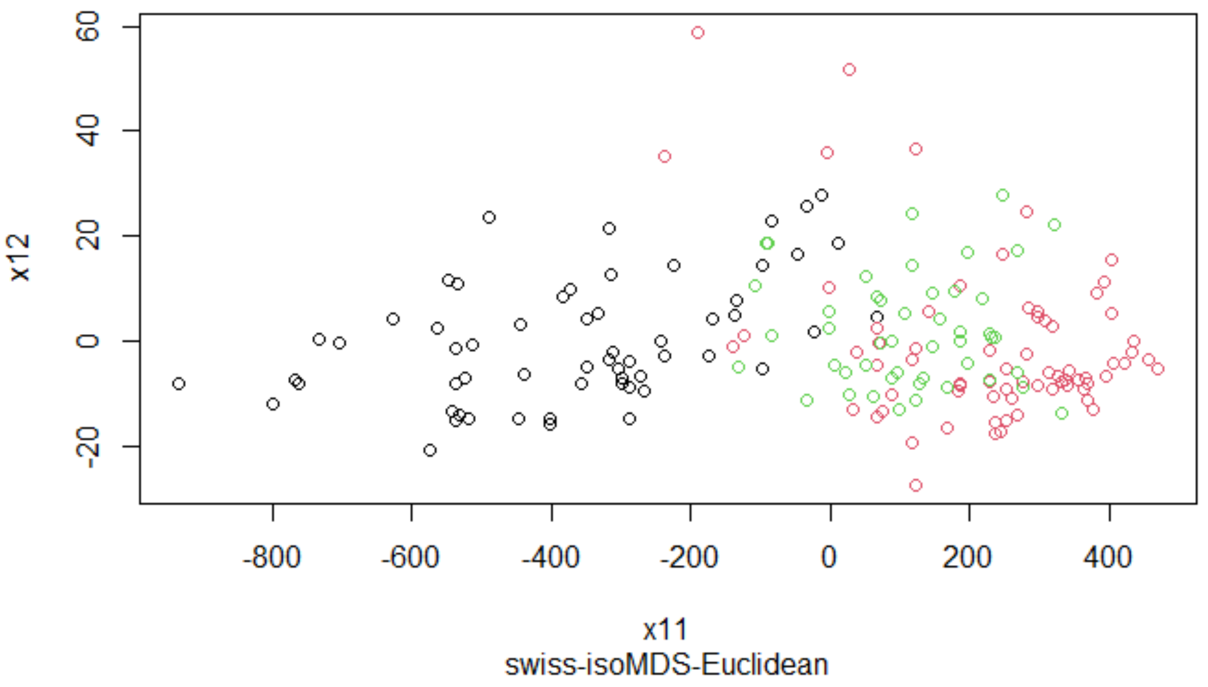
```
###Bray-Curtis
data1.dis<- vegdist(data1) #Bray-Curtis
MDS2<-isoMDS(data1.dis,k=2)
x1<-MDS2$points[,1]
x2<-MDS2$points[,2]
x11<-MDS2$points[which(data$Class==1),1]
x12<-MDS2$points[which(data$Class==1),2]
x21<-MDS2$points[which(data$Class==2),1]
x22<-MDS2$points[which(data$Class==2),2]
x31<-MDS2$points[which(data$Class==3),1]
x32<-MDS2$points[which(data$Class==3),2]
plot(x11,x12,col = "1",xlim = c(min(x1),max(x1)),ylim = c(min(x2),max(x2)),sub="swiss-isoMDS-Bray-Curtis")
points(x21,x22,col = "2")
points(x31,x32,col = "3")
x00<-cbind(x1,x2)
km.res = kmeans(x00, 3)
library(facteoextra)## Visualize clusters using factoextra
fviz_cluster(km.res, x00)
```

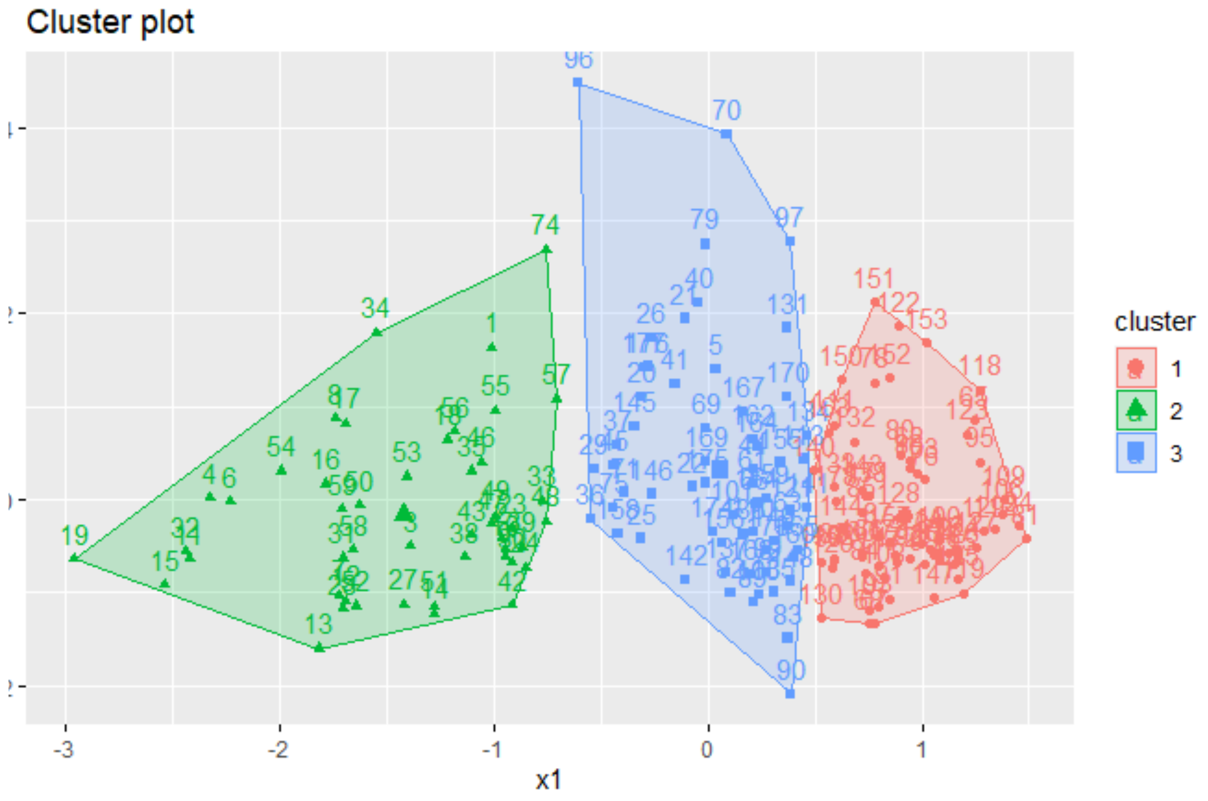
Plot NMDS

Plot k-means cluster

Example with isoMDS() -- wine

(a) **Euclidean** distance matrix

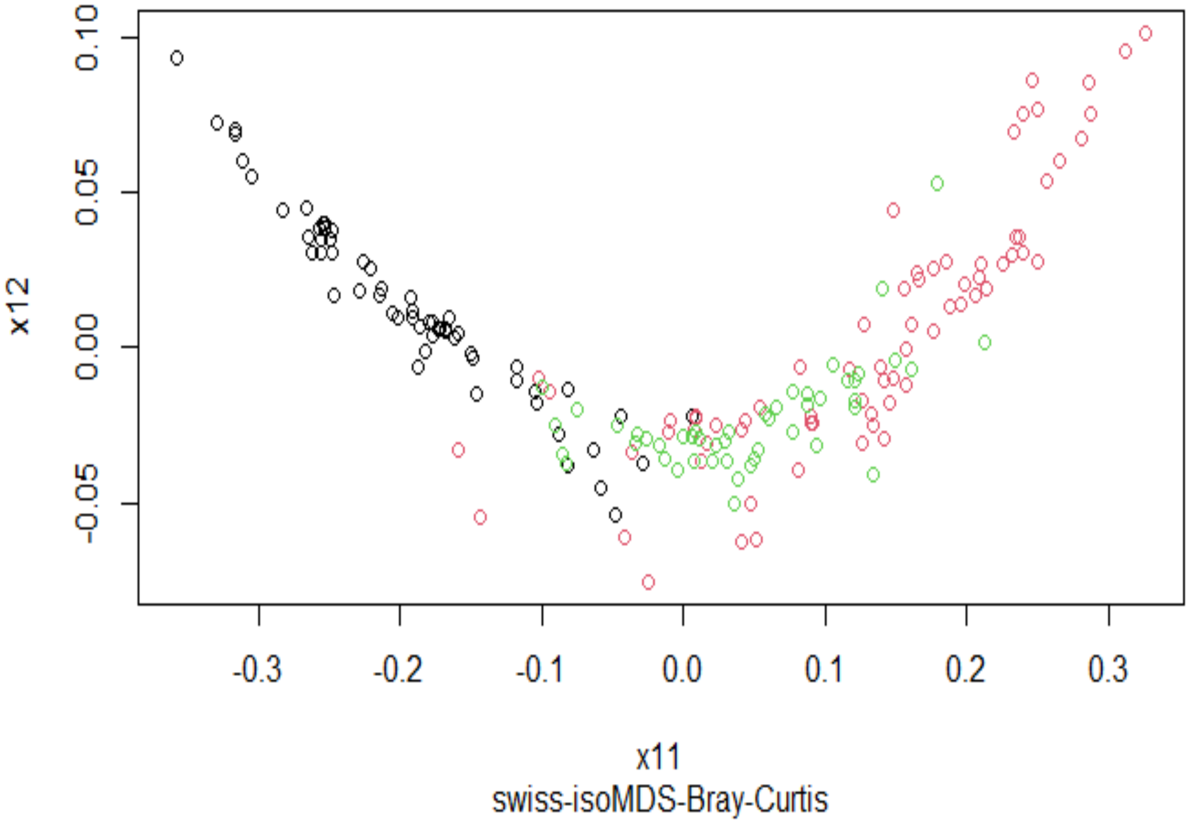


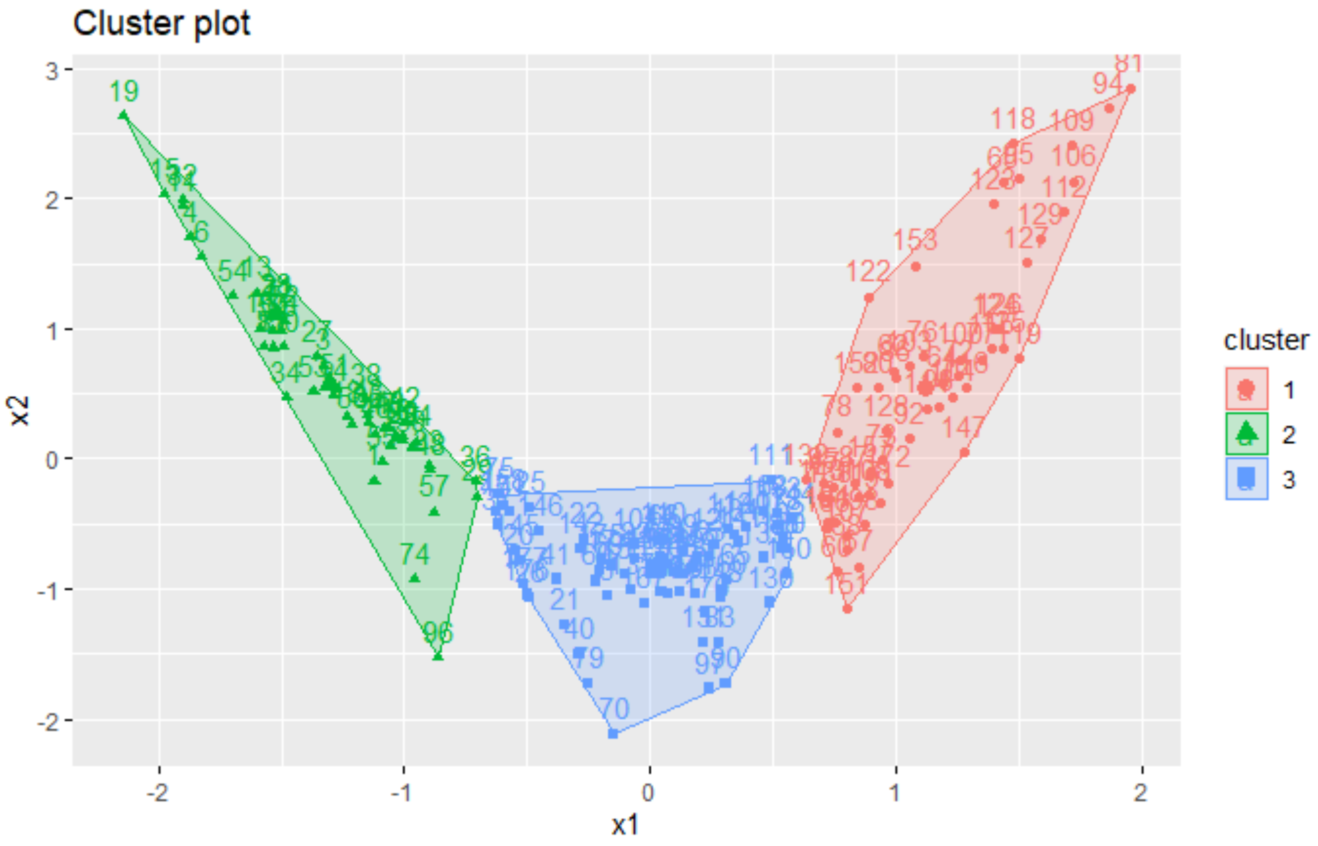


Example with isoMDS() -- wine

(b) **Bray-Curtis**

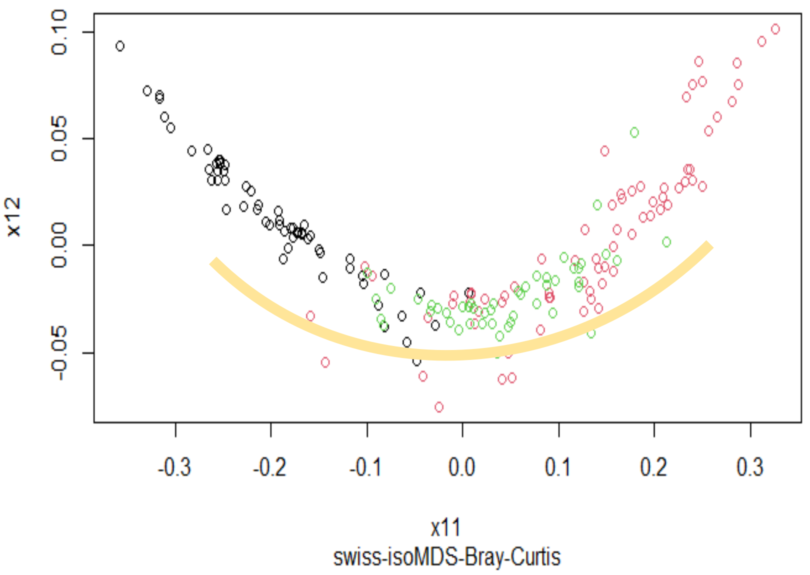
The outputs in wine dataset change a lot when using different methods.
[Why?]



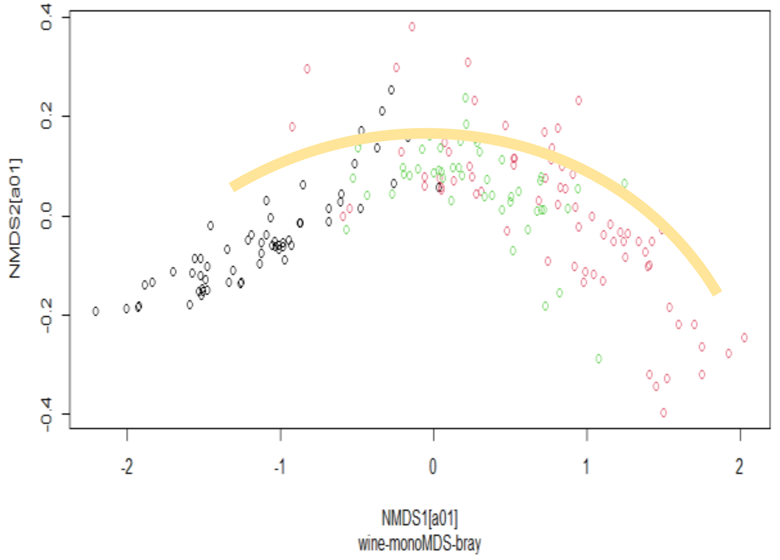


Compare isoMDS() nomoMDS() and metaMDS() -- wine--Bray-Curtis

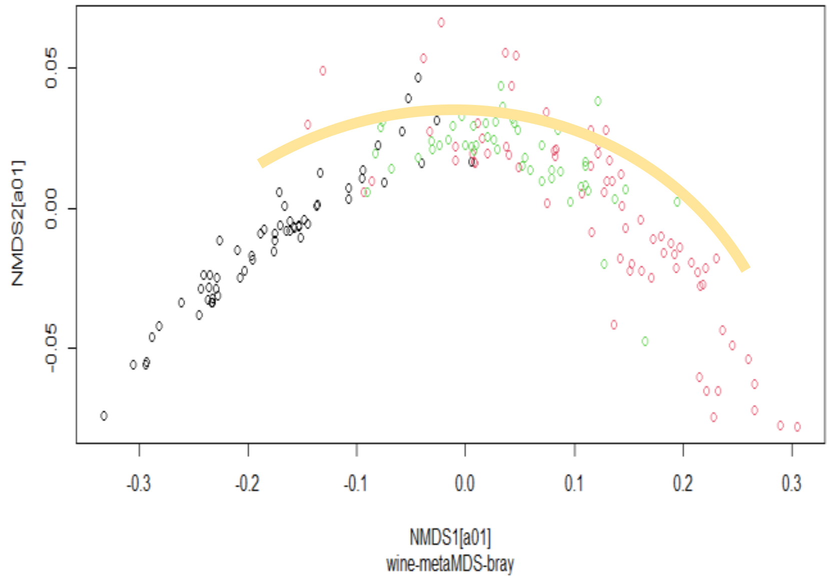
(A) isoMDS()



(B) nomoMDS()



(C) metaMDS



Compare metric & non-metric methods

swiss dataset

Similar metric and non-metric MDS outputs

No large differences in relative location or size of clusters

Outputs are still similar when using Bray-Curtis dissimilarity matrix for the Euclidean one in NMDS

wine dataset

Both use Euclidean Distance as input

Similar three clusters represented and mainly vary across x-dimension

Metric MDS outputs have more similar cluster sizes than non-metric MDS

Bray-Curtis non-metric MDS varied much more on y-dimension than metric MDS outputs

The pattern of scatter plot and cluster plot of different methods are different, but the output of cluster is still robust

Summary

Metric and non-metric methods output similar results

Metric method approach can be used directly as it outputs “true distance” data

Metric MDS can be same as PCA if using Euclidean Distances as inputs

Non-metric method approach is more appealing as it utilizes dissimilarities

Bray-Curtis dissimilarity matrix used in non-metric MDS can output similar results as using Euclidean distance matrix does, and it is applicable in more generalized situation.

References

- Principal Component Analysis Step-by-step, https://www.youtube.com/watch?v=FgakZw6K1QQ

- MDS and PCoA, https://www.youtube.com/watch?v=GEn-_dAyYME

Reference

- Cox, T. F. and Cox, M. A. A. (2001). *Multidimensional Scaling (2n ed.)* Chapter 3. London: Chapman and Hall/CRC. https://www.bristol.ac.uk/media-library/sites/cmm/migrated/documents/chapter3.pdf
- Cox, R. F., and Cox, M. A. A. (1994), Multidimensional Scaling, London: Chapman & Hall.
- Buja, A., Swayne, D., Littman, M., Dean, N.,Hofmann, H., and Chen, L.(2007). *Data Visualizationwith Multidimensional Scaling. http://www.stat.yale.edu/~lc436/papers/JCGS-mds.pdf*
- Jung, S.,(2013). University of Pittsburg https://www.stat.pitt.edu/sungkyu/course/2221Fall13/lec8_mds_combined.pdfgh
- Taylor, J., (2012). Standford University http://statweb.stanford.edu/~jtaylo//courses/stats202/restricted/notes/mds.pdf

NMDS: References

1. Kruskal, J.B. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika 29, 1–27 (1964). https://doi.org/10.1007/BF02289565
2. Kruskal, J.B. Nonmetric multidimensional scaling: A numerical method. Psychometrika 29, 115–129 (1964). https://doi.org/10.1007/BF02289694
3. de Leeuw, J., & Mair, P. (2009). Multidimensional Scaling Using Majorization: SMACOF in R. Journal of Statistical Software, 31(3), 1 - 30. doi:http://dx.doi.org/10.18637/jss.v031.i03

Packages+Code Reference

http://www.gastonsanchez.com/visually-enforced/how-to/2013/01/23/MDS-in-R/

http://www.rdatamining.com/examples/multidimensional-scaling

http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/122-multidimensional-scaling-essentials-algorithms-and-r-code/

https://rpubs.com/flitaiff/MDS

http://r-statistics.co/Multi-Dimensional-Scaling-With-R.html

https://www.rdocumentation.org/packages/smacof/versions/1.2-3/topics/smacofSym

https://www.rdocumentation.org/packages/vegan/versions/2.4-2/topics/wcmdscale

http://finzi.psych.upenn.edu/R/library/vegan/html/wcmdscale.html

https://www.sciencedirect.com/science/article/pii/S0898122113000953

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

https://www.tandfonline.com/doi/abs/10.1198/106186008X318440

http://www.pinlue.com/article/2018/11/2405/557612805257.html

https://site.douban.com/182577/widget/notes/11806604/note/261772270/