LAB 10

TOPICS:

- Hierarchical clustering
- K-means clustering
- Exercises
- Multidimensional Scaling

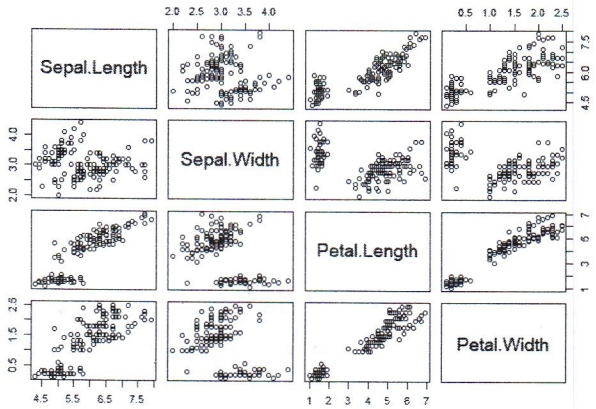
```
library(mvtnorm)
library(rgl)
library(car)
load("mcshapiro.test.RData")
```

1.

```
###------------------------------------------------------------------------
### Hierarchical Clustering
###------------------------------------------------------------------------
### Example 1: iris dataset
###------------------------------------------------------------------------

# Let's forget the labels (we perform a cluster analysis, not a discriminant one)

species.name <- iris[,5]
iris4        <- iris[,1:4]

x11()
pairs(iris4)
```

: we want to estimate the labels so we consider only the features of the dataset

← contains only the features



all the dots are black so we don't know the true labels. Are there clusters?
By the graphical exploration we can already see that there are 2 separated clusters

```
dev.off()
```

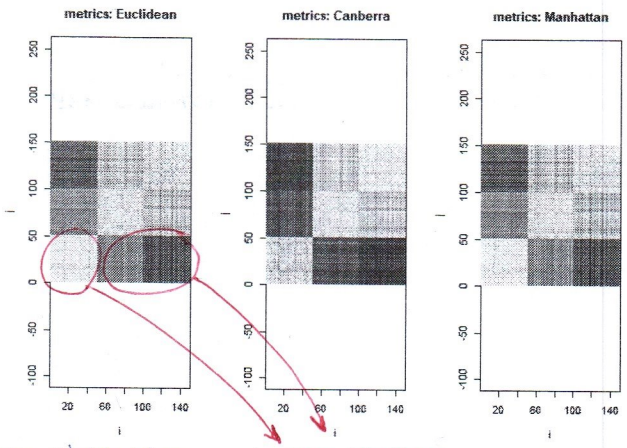
```
# compute the dissimilarity matrix of the data
# we choose the Euclidean metric (and then we look at other metrics)
help(dist)
```

```
## starting httpd help server ... done
```

```
iris.e <- dist(iris4, method='euclidean')

x11()
image(1:150,1:150,as.matrix(iris.e), main='metrics: Euclidean', asp=1, xlab='i', ylab='j')

# with other metrics:
iris.m <- dist(iris4, method='manhattan')
iris.c <- dist(iris4, method='canberra')

x11()
par(mfrow=c(1,3))
image(1:150,1:150,as.matrix(iris.e), main='metrics: Euclidean', asp=1, xlab='i', ylab='j' )
image(1:150,1:150,as.matrix(iris.c), main='metrics: Canberra', asp=1, xlab='i', ylab='j' )
image(1:150,1:150,as.matrix(iris.m), main='metrics: Manhattan', asp=1, xlab='i', ylab='j' )
```

: it takes the features (and the distance) as an input and returns the matrix of the distances (DISSIMILARITY MATRIX)

→ "image" works with matrices



light color = small values
dark color = larger values
(the diagonal ☑ have light colors because the distance of an element with itself is the lowest distance among the possible distances)

very similar among each others
(2 separate groups)
⇒ we expect 2 clusters
(Canberra and Manhattan confirm)

Note: the representation is different from the one we got used:

our: 1 … 150 (i) — 1 … 150

R: 150 … 1 (i) — 1 … 150

(the order of the rows is inverted)
so the two diagonals are:

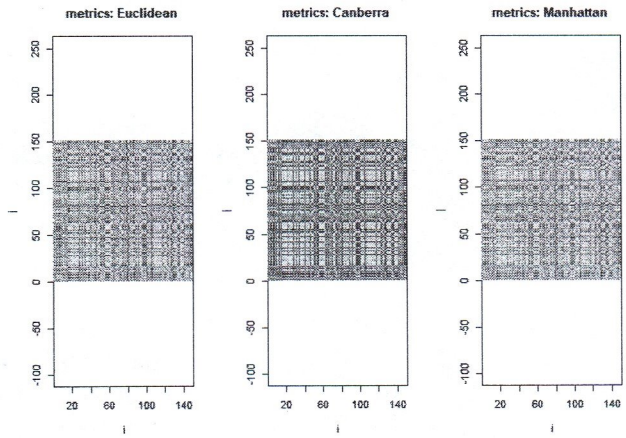
our: 

R: 

Note: we see something this clear because the dataset is ordered (the first observations are of the same species and so on), usually they're not so clear (see later)

```
# actually, the data are never ordered according to (unknown) labels
misc <- sample(150)
iris4 <- iris4[misc,]

iris.e <- dist(iris4, method='euclidean')
iris.m <- dist(iris4, method='manhattan')
iris.c <- dist(iris4, method='canberra')

x11()
par(mfrow=c(1,3))
image(1:150,1:150,as.matrix(iris.e), main='metrics: Euclidean', asp=1, xlab='i', ylab='j' )
image(1:150,1:150,as.matrix(iris.c), main='metrics: Canberra', asp=1, xlab='i', ylab='j' )
image(1:150,1:150,as.matrix(iris.m), main='metrics: Manhattan', asp=1, xlab='i', ylab='j' )
```

: we shuffle the data and see what happen



as expected, here is much harder to see which (and how many) are the clusters in the dataset

```
graphics.off()

# we now aim to perform hierarchical clustering of the dataset iris
# we limit to Euclidean distance

# Command hclust()
help(hclust)

iris.es <- hclust(iris.e, method='single')
iris.ea <- hclust(iris.e, method='average')
iris.ec <- hclust(iris.e, method='complete')

# if we want more detailed information on euclidean-complete
# clustering:
names(iris.ec)
```

we can choose the linkage function (notice that we've considering only Euclidean distance)

```
## [1] "merge"       "height"      "order"       "labels"      "method"
## [6] "call"        "dist.method"
```

```
iris.ec$merge # order of aggregation of statistical units / clusters
```

(description of the aggregation algorithm)

```
##       [,1] [,2]
##  [1,]  -38  -77
##  [2,] -106 -123
##  [3,]   -4  -54
##  [4,]  -92 -144
##  [5,]  -88 -102
##  [6,]   -2  -78
##  [7,]  -55  -75
##  [8,]  -65 -146
##  [9,]  -12  -42
## [10,]  -35 -149
## [11,]  -68  -85
## [12,]  -79 -138
## [13,]   -5   -8
## [14,]  -16  -33
## [15,]  -52 -101
## [16,]  -72 -110
## [17,]  -95 -143
## [18,] -118 -137
## [19,]  -63  -89
## [20,]  -14 -105
## [21,]  -70 -134
## [22,] -136    3
## ..
## [146,]  137  139
## [147,]  141  145
## [148,]  144  146
## [149,]  147  148
```

merge $\in \mathbb{R}^{(n-1)\times 2}$

row i describe the merging of clusters at step i of the clustering:

- if $j < 0 \Rightarrow$ the observation $-j$ was merged in a cluster
- if $j > 0 \Rightarrow$ the merge was with the cluster formed at the (earlier) stage j of the algorithm

at step **13** the units **5** and **8** have been aggregated

at step **146** have been aggregated the clusters produced at steps **137** and **139**

```
iris.ec$height # distance at which we have aggregations
```

```
##   [1] 0.0000000 0.1000000 0.1000000 0.1000000 0.1000000 0.1000000 0.1414214
##   [8] 0.1414214 0.1414214 0.1414214 0.1414214 0.1414214 0.1414214 0.1414214
##  [15] 0.1414214 0.1414214 0.1414214 0.1414214 0.1414214 0.1414214 0.1414214
##  [22] 0.1732051 0.1732051 0.1732051 0.1732051 0.1732051 0.1732051 0.2000000
##  [29] 0.2000000 0.2000000 0.2000000 0.2000000 0.2000000 0.2236068 0.2236068
##  [36] 0.2449490 0.2449490 0.2449490 0.2449490 0.2449490 0.2449490 0.2449490
##  [43] 0.2449490 0.2645751 0.2645751 0.2645751 0.2645751 0.2645751 0.2645751
##  [50] 0.2645751 0.2645751 0.2645751 0.2828427 0.2828427 0.3000000 0.3000000
##  [57] 0.3000000 0.3000000 0.3000000 0.3162278 0.3162278 0.3162278 0.3316625
##  [64] 0.3316625 0.3316625 0.3316625 0.3316625 0.3316625 0.3316625 0.3464102
##  [71] 0.3464102 0.3464102 0.3605551 0.3741657 0.3741657 0.3741657 0.3872983
##  [78] 0.3872983 0.3872983 0.4000000 0.4000000 0.4123106 0.4123106 0.4242641
##  [85] 0.4242641 0.4242641 0.4358899 0.4472136 0.4472136 0.4472136 0.4582576
##  [92] 0.4582576 0.4690416 0.4690416 0.5099020 0.5196152 0.5196152 0.5196152
##  [99] 0.5196152 0.5196152 0.5477226 0.5477226 0.5567764 0.5567764 0.5567764
## [106] 0.5916080 0.6000000 0.6082763 0.6082763 0.6164414 0.6164414 0.6480741
## [113] 0.6480741 0.6782330 0.6855655 0.7071068 0.7211103 0.7348469 0.7549834
## [120] 0.7549834 0.7615773 0.7874008 0.8062258 0.8062258 0.8062258 0.8246211 0.8306624
## [127] 0.9055385 0.9219544 1.0049876 1.0295630 1.0677078 1.0677078 1.1090537
## [134] 1.1747340 1.2124356 1.2247449 1.2727922 1.3892444 1.4071247 1.4491377
## [141] 1.4525839 1.4628739 1.6613248 1.7058722 2.2360680 2.4289916 3.2109189
## [148] 4.0249224 7.0851958
```

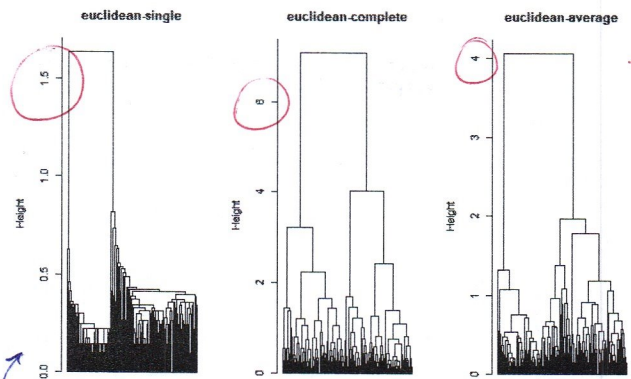
distance at which we have the aggregation ("Hight" at which we aggregate)

```
iris.ec$order  # ordering that allows to avoid intersections in the dendrogram
```

```
##   [1]  39  73  46 133 147  80  17 131  87 107  45 130  58 116 114  31 140  59
##  [19]  18 120 125   1  97  66 112  27  60  56  57  12  42  84  88 102 121  74
##  [37]  16  33 109  20 141  38  77  26  30  49  29  50  94  48  86  21  34  53
##  [55]  70 134 139 115 129  19 122   3  36  43   9  93  25  61  11  37  14 105
##  [73]  22 108  68  85 103  91 111  40 132  62 117  90  95 143   6  24  23  72
##  [91] 110 126  44 145 119  83 104 113  79 138  82  13  98  81  99 124  32  51
## [109]  69  96  65 146 100 142  76 150 128   2  78  67 127  10  92 144  64  63
## [127]  89  55  75 118 137 136   4  54  15 106 123 135  71  35 149 148  47  28
## [145]  41  52 101   7   5   8
```

order of the units that produces a readable dendrogram

```
# plot of the dendrograms
x11()
par(mfrow=c(1,3))
plot(iris.es, main='euclidean-single', hang=-0.1, xlab='', labels=F, cex=0.6, sub='')
plot(iris.ec, main='euclidean-complete', hang=-0.1, xlab='', labels=F, cex=0.6, sub='')
plot(iris.ea, main='euclidean-average', hang=-0.1, xlab='', labels=F, cex=0.6, sub='')
```



the values on the "Y" axis are different!

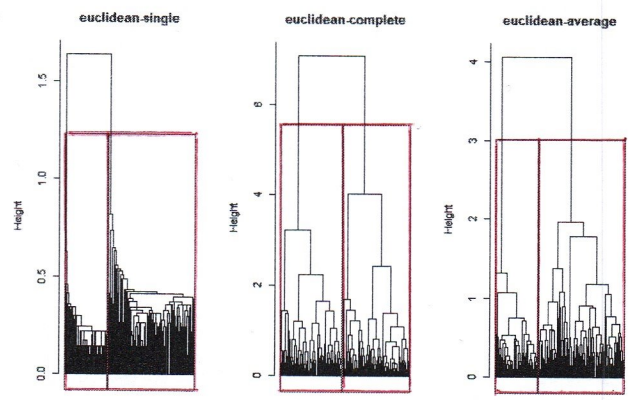
All the 3 dendrograms suggest a choice of 2 clusters

we can see the CHAINING effect (how the algorithm joined one statistical unit at the time concatenating single stat. units)

```
dev.off()
```

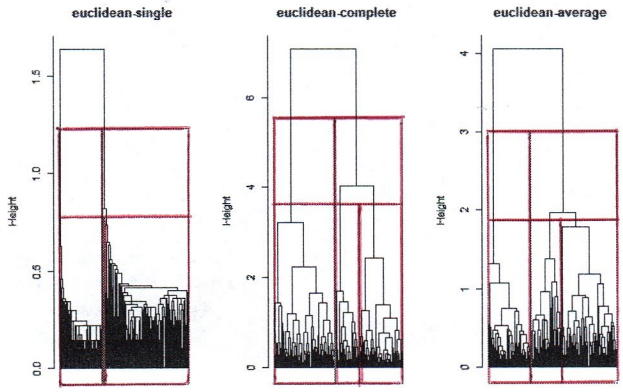
```
# plot dendrograms (2 clusters)
x11()
par(mfrow=c(1,3))
plot(iris.es, main='euclidean-single', hang=-0.1, xlab='', labels=F, cex=0.6, sub='')
rect.hclust(iris.es, k=2)
plot(iris.ec, main='euclidean-complete', hang=-0.1, xlab='', labels=F, cex=0.6, sub='')
rect.hclust(iris.ec, k=2)
plot(iris.ea, main='euclidean-average', hang=-0.1, xlab='', labels=F, cex=0.6, sub='')
rect.hclust(iris.ea, k=2)
```

: we want to CUT the dendrograms




```
dev.off()
```

```
# plot dendrograms (3 clusters)
x11()
par(mfrow=c(1,3))
plot(iris.es, main='euclidean-single', hang=-0.1, xlab='', labels=F, cex=0.6, sub='')
rect.hclust(iris.es, k=2)
rect.hclust(iris.es, k=3)
plot(iris.ec, main='euclidean-complete', hang=-0.1, xlab='', labels=F, cex=0.6, sub='')
rect.hclust(iris.ec, k=2)
rect.hclust(iris.ec, k=3)
plot(iris.ea, main='euclidean-average', hang=-0.1, xlab='', labels=F, cex=0.6, sub='')
rect.hclust(iris.ea, k=2)
rect.hclust(iris.ea, k=3)
```




```
dev.off()
```

```
# How to cut a dendrogram?
# We generate vectors of labels through the command cutree()
help(cutree)

# Fix k=2 clusters:
cluster.ec <- cutree(iris.ec, k=2) # euclidean-complete:
cluster.ec
```

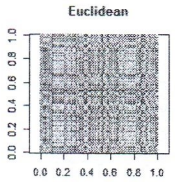
inputs: hclust object and number of clusters
output: labels

```
## 111  49  51   1   9 100  14  39  52  26  74 138  33  92  50 128 106 149  78 122
##   1   2   1   2   2   2   2   2   1   2   1   1   2   1   2   1   1   1   1   1
## 120  61  89  95  98 135 146  43 124 147 144  24 139  88  31  53  79 143 108  91
##   1   2   2   2   1   1   1   2   1   1   1   2   1   1   2   1   1   1   1   2
##   3 117  86  65 136 103   7  84 112 127  27   4  69  18  38 109 104 141 101 142
##   2   1   1   2   1   1   2   1   1   1   2   2   1   2   2   1   1   1   1   1
##  75  62   2  13  22 140  42  58  45  66  12  96 131  71   5  21 102  11  82 119
##   1   2   2   2   2   1   2   2   2   1   2   2   1   1   2   2   1   2   2   1
##  16  17  90 105  94 134 118 133  46  68  67  10  57  73  93  47 148  34  15  19
##   2   2   2   1   2   1   1   1   2   2   2   2   1   1   2   2   1   2   2   2
##  48 129 107  54  64   8 132  99 115  97  85 113  70 125  55 145  72  29  60 137
##   2   1   2   2   1   2   1   2   1   2   2   1   2   1   1   1   2   2   2   1
## 150  87  40  44 116  63  36  37  59 110 123  56 126  76  25  41  28  81  77 121
##   1   1   2   2   1   2   2   2   1   1   1   2   1   1   2   2   2   2   1   1
## 114   6  83  35  80  20 130  23  30  32
##   1   2   2   2   2   2   1   2   2   2
```

```
cluster.es <- cutree(iris.es, k=2) # euclidean-single
cluster.ea <- cutree(iris.ea, k=2) # euclidean-average

# Let's give a mark to the algorithms: did they aggregate coherently with
# the dissimilarity matrix or not?

# compute the cophenetic matrices
coph.es <- cophenetic(iris.es)
coph.ec <- cophenetic(iris.ec)
coph.ea <- cophenetic(iris.ea)

# compare with dissimilarity matrix (Euclidean distance)
x11()
layout(rbind(c(0,1,0),c(2,3,4)))
image(as.matrix(iris.e), main='Euclidean', asp=1 )
image(as.matrix(coph.es), main='Single', asp=1 )
image(as.matrix(coph.ec), main='Complete', asp=1 )
image(as.matrix(coph.ea), main='Average', asp=1 )
```

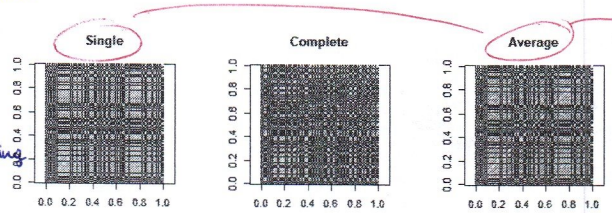
] let's see how good are the clusters

(graphically)



(cooler if we don't shuffle the data)

(*) this tells us that the complete linkage is using a cophenetic matrix which is less similar to the original distance matrix w.r.t. the single linkage and average linkage (suggesting us that single and average linkages are performing better w.r.t. complete linkage)



we can visually see more similar patterns between these two linkages while the complete linkage looks different. In order to have a quantification of this visual comparison we can compute the cophenetic correlation coefficients: (which are the correlations between the values of the original distances and the values of the cophenetic matrix)

(**) this evaluation has nothing to do with the number of clusters or the checking if there are clusters in the data

```
dev.off()
```

```
# compute cophenetic coefficients
es <- cor(iris.e, coph.es)
ec <- cor(iris.e, coph.ec)
ea <- cor(iris.e, coph.ea)

c("Eucl-Single"=es,"Eucl-Compl."=ec,"Eucl-Ave."=ea)
```

(to evaluate how good is the clustering)

```
## Eucl-Single Eucl-Compl.   Eucl-Ave.
##   0.8638787   0.7276283   0.8769561
```

all of them are quite high (⇒ the cophenetic matrix contains distances which are strongly correlated with the original distances) but Single and Average linkages have higher values. (*)

```
# interpret the clusters
table(label.true = species.name[misc], label.cluster = cluster.es)
```

: comparison with the true lables that we know (MISCLASSIFICATION TABLE)

(single)

```
##             label.cluster
## label.true   1  2
##   setosa     0 50
##   versicolor 50  0
##   virginica  50  0
```

→ perfect clustering

```
table(label.true = species.name[misc], label.cluster = cluster.ec)
```

(complete)

```
##             label.cluster
## label.true   1  2
##   setosa     0 50
##   versicolor 23 27
##   virginica  49  1
```

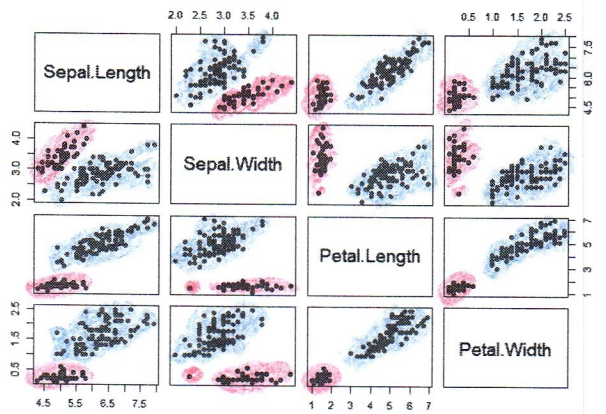
→ there are some problems

```
table(label.true = species.name[misc], label.cluster = cluster.ea)
```

(average)

```
##             label.cluster
## label.true   1  2
##   setosa     0 50
##   versicolor 50  0
##   virginica  50  0
```

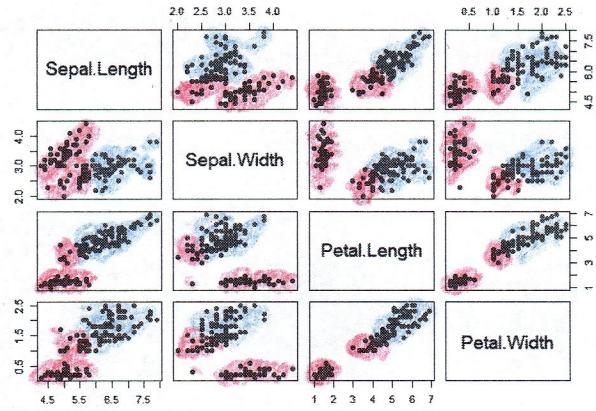
```
x11()
plot(iris4, col=ifelse(cluster.es==1,'red','blue'), pch=19)
```



single linkage

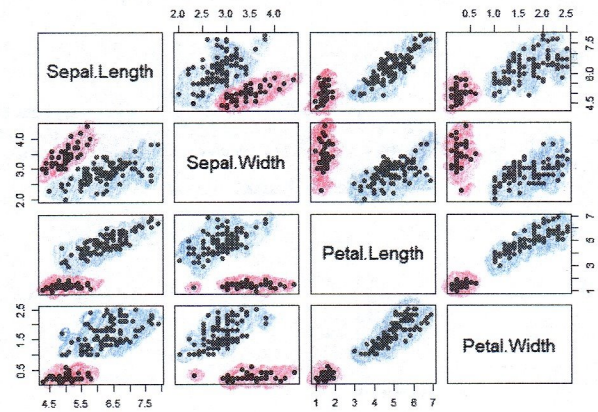
```
x11()
plot(iris4, col=ifelse(cluster.ec==1,'red','blue'), pch=19)
```

(**) through the hierarchicle algo. we produce another distance matrix (cophenetic matrix) and this is the distance matrix that the algorithm uses to produce the clustering. These coefficients (coph.'s) tells us how good the hierarchicle algorithm was in traslating the real dist. matrix into the cophenetic matrix. If the two are similar, the clustering will be coherent w.r.t. the original distances, if not, we have lost something and the final clustering will not represent at 100% the data



complete linkage

```
x11()
plot(iris4, col=ifelse(cluster.ea==1,'red','blue'), pch=19)
```



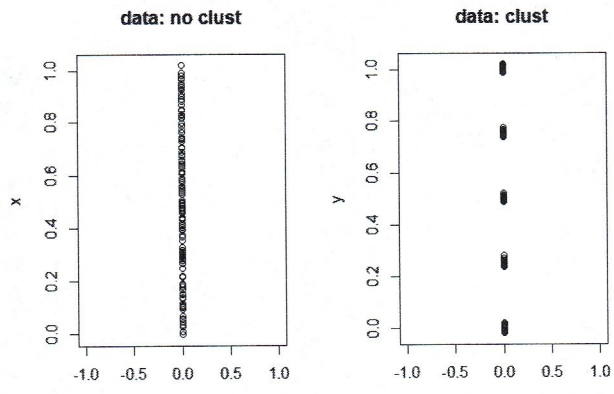
average linkage (identical to the single linkage)

```
graphics.off()

# Exercise: repeat the analysis with other metrics
```

2.

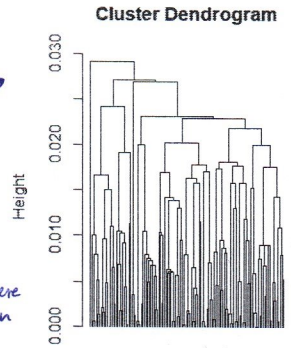
```
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### Example 2: simulated data
### ------------------------------------------------------------------------

### Univariate case
### ------------------------------------------------------------------------
set.seed(123)
# x : vector of NON clustered data
x <- 0:124/124 + rnorm(125, sd=0.01)
# y : vector of clustered data (5 clusters)
y <- c(rnorm(25, mean=0, sd=0.01), rnorm(25, mean=0.25, sd=0.01),
       rnorm(25, mean=0.5, sd=0.01), rnorm(25, mean=0.75, sd=0.01),
       rnorm(25, mean=1, sd=0.01))

x <- sample(x)
y <- sample(y)

x11()
par(mfrow=c(1,2))
plot(rep(0,125),x, main='data: no clust',xlab='')
plot(rep(0,125),y, main='data: clust',xlab='')
```

generation of data

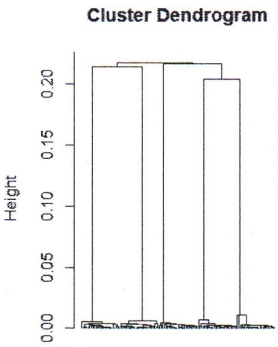


```
dx <- dist(x)
dy <- dist(y)
hcx<- hclust(dx, method='single')
hcy<- hclust(dy, method='single')

x11()
par(mfrow=c(1,2))
plot(hcx, labels=F, cex=0.5, hang=-0.1, xlab='', sub='x')
plot(hcy, labels=F, cex=0.5, hang=-0.1, xlab='', sub='y')
```

we don't specify the distance ⟹ Euclidean



no evidence of clusters within the data (there are very frequent splits, there are no wide ranges where the splits remain constant)



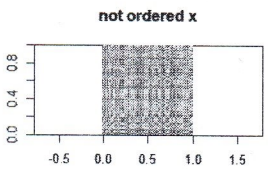
for a wide range of heights the dendrogram is clustered in 5 groups

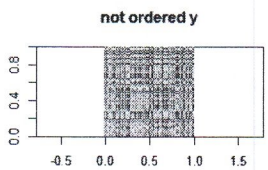
Remember: pay attention to the Y scale (it's different!)

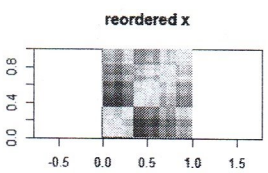
(not clustered) x

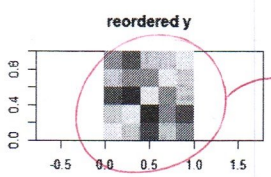
(clustered) y

```
x11()
par(mfrow=c(2,2))
image(as.matrix(dx), asp=1, main='not ordered x' )
image(as.matrix(dy), asp=1, main='not ordered y' )
image(as.matrix(dx)[hcx$order,hcx$order], asp=1, main='reordered x' )
image(as.matrix(dy)[hcy$order,hcy$order], asp=1, main='reordered y' )
```







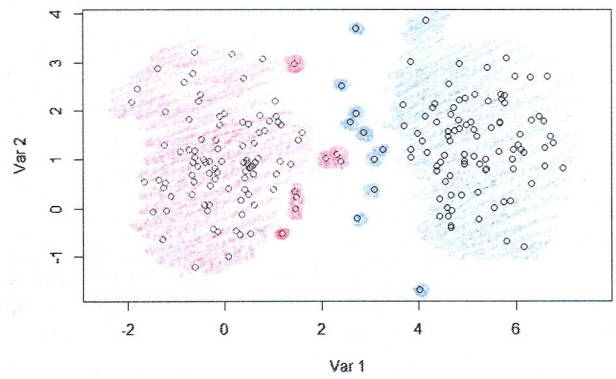


(order obtained by the clustering algo.)

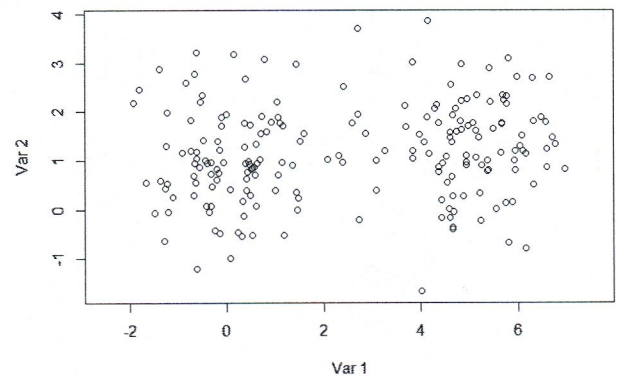
clear block structure (5 clusters)

```
graphics.off()


### Bivariate case (example of chaining effect)
###------------------------------------------------------------------
p <- 2
n <- 100

mu1 <- c(0,1)
mu2 <- c(5,1.2)
sig <- diag(rep(1,p))

set.seed(1)
X1 <- rmvnorm(n, mu1, sig)
X2 <- rmvnorm(n, mu2, sig)

X <- rbind(X1, X2)
# If we knew the Labels:
x11()
plot(X, xlab='Var 1', ylab='Var 2', col=rep(c('red','blue'),each=100), asp=1)
```

simulation / generation of data (bivariate example: we have 2 sets of points from bivariate gaussian distribution acting to different values of μ ($\mu_1 = [0,1]^T$, $\mu_2 = [5, 1.2]^T$) and same covariance structure)



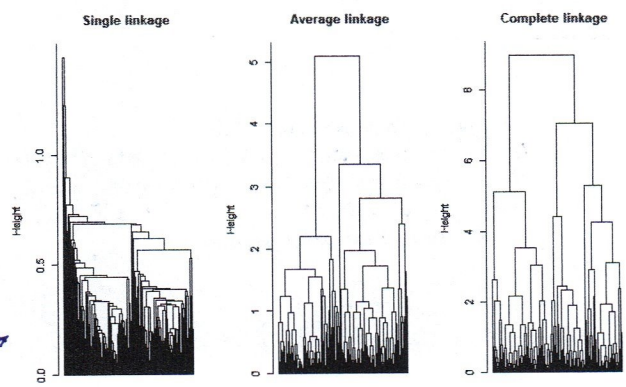

```
# How we actually see the data:
plot(X, xlab='Var 1', ylab='Var 2', asp=1)
```



we can already imagine the separation, the problem is that there is no "neat" separation between the two groups

```
dev.off()
```

```
# Let's compare the clustering results with Euclidean distance and three
# types of linkage
x.d <- dist(X, method = 'euclidean')

x.es <- hclust(x.d, method='single')
x.ea <- hclust(x.d, method='average')
x.ec <- hclust(x.d, method='complete')

x11()
par(mfrow=c(1,3))
plot(x.es, main = 'Single linkage'  , hang=-0.1, xlab='', labels=F, sub='')
plot(x.ea, main = 'Average linkage' , hang=-0.1, xlab='', labels=F, sub='')
plot(x.ec, main = 'Complete linkage', hang=-0.1, xlab='', labels=F, sub='')
```



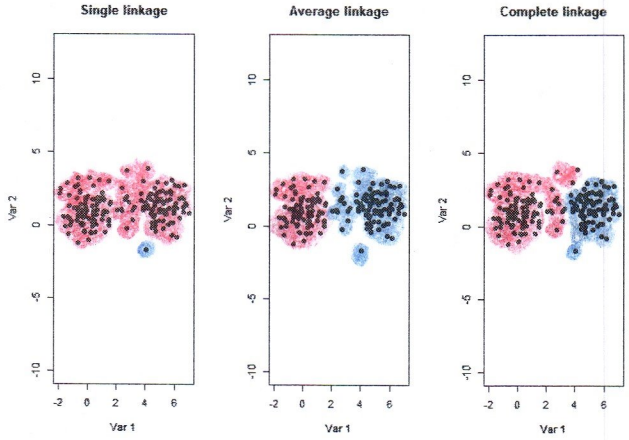
← not a clear structure

CHAINING EFFECT (many splits, merging almost 1 datum at step, heavy tail)

↑ indication: clusters = 2

```
# Let's cut the tree as to get 2 clusters
cluster.es <- cutree(x.es, k = 2)
cluster.ea <- cutree(x.ea, k = 2)
cluster.ec <- cutree(x.ec, k = 2)

x11()
par(mfrow=c(1,3))

plot(X, xlab='Var 1', ylab='Var 2', main = 'Single linkage', col=ifelse(cluster.es==1,'red','blue'), pch=16, asp=1)
plot(X, xlab='Var 1', ylab='Var 2', main = 'Average linkage', col=ifelse(cluster.ea==1,'red','blue'), pch=16, asp=1)
plot(X, xlab='Var 1', ylab='Var 2', main = 'Complete linkage', col=ifelse(cluster.ec==1,'red','blue'), pch=16, asp=1)
```

very similar to the true one (best one: in the dendrogram we clearly saw 2 clusters)

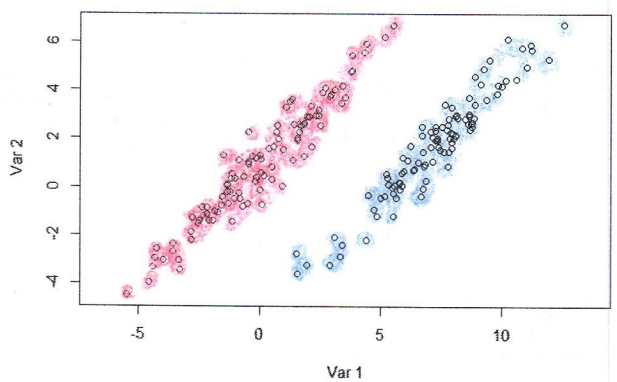


drastic result (due to the chain effect)

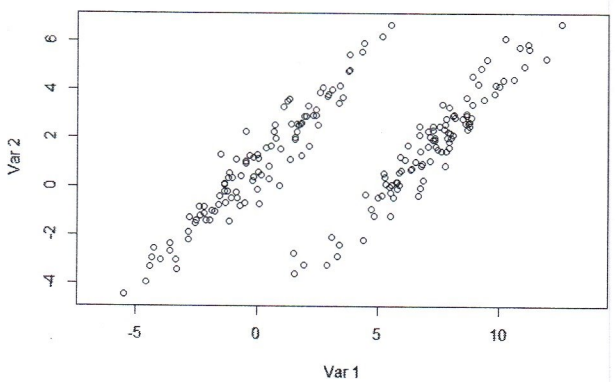
good, not optimal

```
graphics.off()

### Bivariate case (example of ellipsoidal clusters)
### ------------------------------------------------------------------------

p <- 2
n <- 100

mu1 <- c(0,1)
mu2 <- c(6.5,1)

e1 <- c(1,1)
e2 <- c(-1,1)
sig <- 5*cbind(e1)%*%rbind(e1)+.1*cbind(e2)%*%rbind(e2)

set.seed(2)
X1 <- rmvnorm(n, mu1, sig)
X2 <- rmvnorm(n, mu2, sig)

X <- rbind(X1, X2)

# If we knew the Labels:
x11()
plot(X, xlab='Var 1', ylab='Var 2', asp=1, col=rep(c('red','blue'),each=100))
```

simulation of data



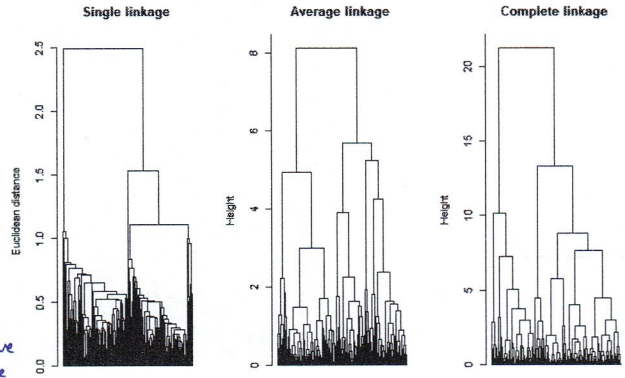

```
# How we actually see the data:
plot(X, xlab='Var 1', ylab='Var 2', asp=1)
```



again, from a graphical exploration we already see that there are 2 very well separated clusters

```
dev.off()
```

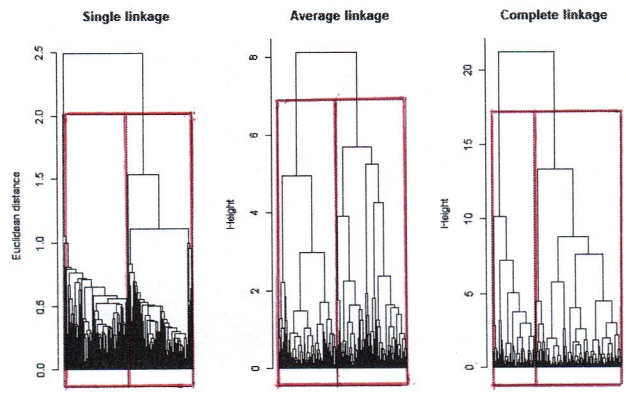
```r
## compare results with different linkage, when Euclidean distances is used

x.d <- dist(X, method = 'euclidean')

x.es <- hclust(x.d, method='single')
x.ea <- hclust(x.d, method='average')
x.ec <- hclust(x.d, method='complete')

x11()
par(mfrow=c(1,3))

plot(x.es, main = 'Single linkage', ylab='Euclidean distance', hang=-0.1, xlab='', labels=F, sub='')
plot(x.ea, main = 'Average linkage', hang=-0.1, xlab='', labels=F, sub='')
plot(x.ec, main = 'Complete linkage', hang=-0.1, xlab='', labels=F, sub='')
```



this time we don't have the heavy tail (no chain effect)

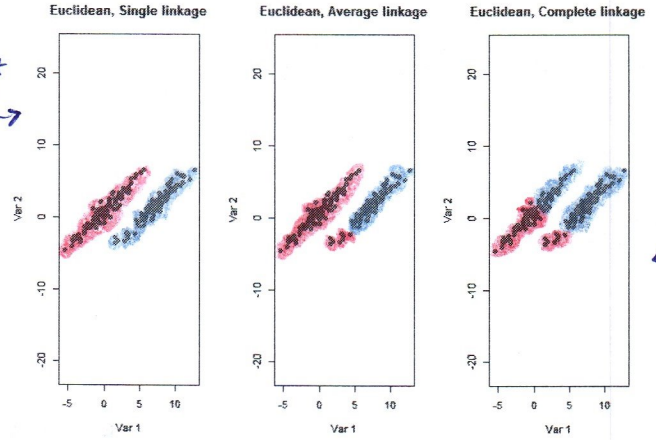
all the linkages give a suggestion for number of clusters = 2

```r
# Let's cut the tree to get 2 clusters
x11()
par(mfrow=c(1,3))

plot(x.es, main = 'Single linkage', ylab='Euclidean distance', hang=-0.1, xlab='', labels=F, sub='')
rect.hclust(x.es, k=2)
plot(x.ea, main = 'Average linkage', hang=-0.1, xlab='', labels=F, sub='')
rect.hclust(x.ea, k=2)
plot(x.ec, main = 'Complete linkage', hang=-0.1, xlab='', labels=F, sub='')
rect.hclust(x.ec, k=2)
```



we see that with complete linkage one group is smaller w.r.t. the other (in real application is not a problem, having in mind how we generated the data yes)

```r
cluster.es <- cutree(x.es, k = 2)
cluster.ea <- cutree(x.ea, k = 2)
cluster.ec <- cutree(x.ec, k = 2)

x11()
par(mfrow=c(1,3))

plot(X, xlab='Var 1', ylab='Var 2', main = 'Euclidean, Single linkage', col=cluster.es+1, pch=16, asp=1)
plot(X, xlab='Var 1', ylab='Var 2', main = 'Euclidean, Average linkage', col=cluster.ea+1, pch=16, asp=1)
plot(X, xlab='Var 1', ylab='Var 2', main = 'Euclidean, Complete linkage', col=cluster.ec+1, pch=16, asp=1)
```



perfect

← we have a separation which goes in the opposite direction to the one we have expected

```
graphics.off()

# Change the mean mu2
# Example:
# mu2 <- c(4.5,1.2)
```

Conclusions: there is no (general) correct choice of the linkage (look at all the examples)

3.

```
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### Example 3: earthquakes dataset
### ------------------------------------------------------------------------
help(quakes)
```

→ contains the locations of the earthquakes in Fiji

```
head(quakes)
```

```
##      lat   long depth mag stations
## 1 -20.42 181.62   562 4.8       41
## 2 -20.62 181.03   650 4.2       15
## 3 -26.00 184.10    42 5.4       43
## 4 -17.97 181.66   626 4.1       19
## 5 -20.42 181.96   649 4.0       11
## 6 -19.68 184.31   195 4.0       12
```

stations reporting the event

```
dim(quakes)
```

```
## [1] 1000    5
```

we want to work with the 3D coordinates of the earthquakes (latitude, longitude, depth) and try to find clusters among data (from previous knowledge we know that: # clusters = 2)

```
Q <- cbind(quakes[,1:2], depth = -quakes[,3]/100)
head(Q)
```

```
##      lat   long depth
## 1 -20.42 181.62 -5.62
## 2 -20.62 181.03 -6.50
## 3 -26.00 184.10 -0.42
## 4 -17.97 181.66 -6.26
## 5 -20.42 181.96 -6.49
## 6 -19.68 184.31 -1.95
```

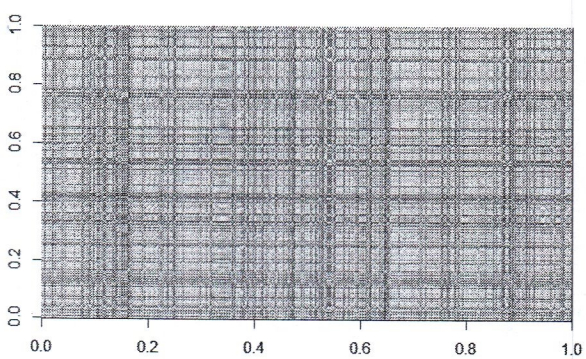
we transform this to obtain negative values (depth) not in km (just for physical meaning) (so the Euclidean distance makes sense)

Using the Euclidean dist with the untouched data would provide a clustering based (almost) only on depth since (because of the units of measure) the data seems to be more sparse in depth.

⇒ THINK about the meaning and UNITS OF MEASURE

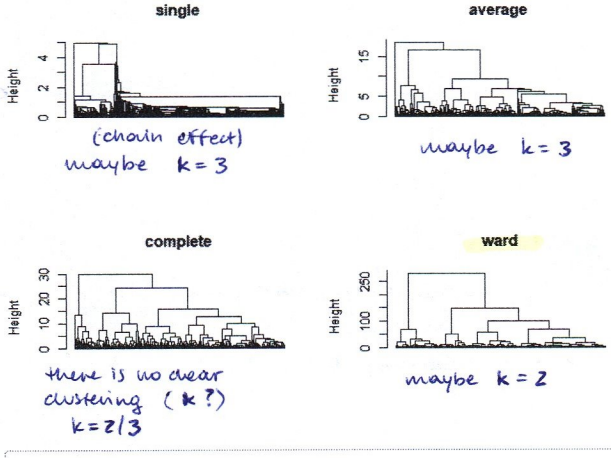
```
plot3d(Q, size=3, col='orange', aspect = F)

# dissimilarity matrix (Euclidean metric)
d <- dist(Q)

x11()
image(as.matrix(d))
```

[LOST IMAGE]
but it's the same as the next ones



from a dissimilarity matrix of real data is really hard to see some clustering structure

```
x11()
par(mfrow=c(2,2))

clusts <- hclust(d, method='single')
plot(clusts, hang=-0.1, labels=FALSE, main='single', xlab='', sub='')
# rect.hclust(clusts, k=2)
# rect.hclust(clusts, k=3)

clusta <- hclust(d, method='average')
plot(clusta, hang=-0.1, labels=FALSE, main='average', xlab='', sub='')
# rect.hclust(clusta, k=2)
# rect.hclust(clusta, k=3)

clustc <- hclust(d, method='complete')
plot(clustc, hang=-0.1, labels=FALSE, main='complete', xlab='', sub='')
# rect.hclust(clustc, k=2)
# rect.hclust(clustc, k=3)

clustw <- hclust(d, method='ward.D2')
plot(clustw, hang=-0.1, labels=FALSE, main='ward', xlab='', sub='')
```



(chain effect)
maybe k = 3

maybe k = 3

(k = # clusters)

there is no clear clustering (k ?)
k = 2/3

maybe k = 2

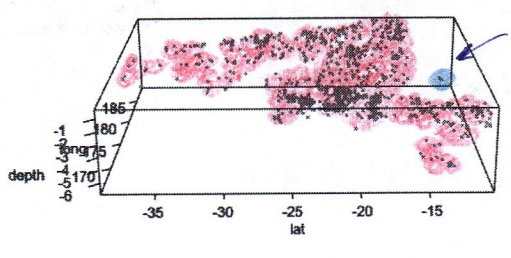
```
# rect.hclust(clustw, k=2)
# rect.hclust(clustw, k=3)

## Ward-Linkage (see J-W p. 692-693):
# Ward considered hierarchical clustering procedures based on minimizing the
# 'loss of information' from joining two groups. This method is usually implemented
# with loss of informations taken to be an increase in an error sum of squares
# criterion, ESS. First for a given cluster k, let ESS[k] be the sum of the
# squared deviations of every item in the cluster from the cluster mean (centroid).
# (ESS[k]=sum_{x.j in cluster k} t(x.j-x.mean[k])%*%(x.j-x.mean[k]), where
# x.mean[k] is the centroid of cluster k).
# If there are currently K clusterts, define ESS as the sum of the ESS[k]
# (ESS=ESS[1]+ESS[2]+...+ESS[K]). At each step in the analysis, the union of
# every possible pair of clusters is considered, and the two clusters whose
# combination results in the smallest increase in ESS (minimum loss of information)
# are joined.
# Initially, each cluster consists of a single item and, if there are N items,
# ESS[k]=0, k=1,2,...,N, so ESS=0. At the other extreme, when all the clusters are
# combined in a single group of N items, the value of ESS is given by
# ESS=sum_j(t(x.j-x.mean)%*%(x.j-x.mean)), where x.j is the multivariate measurement
# associated with the jth item and x.mean is the mean of all items.

open3d()
```

```
# single linkage
clusters <- cutree(clusts, 2)
plot3d(Q, size=3, col=clusters+1, aspect = F)
rglwidget()
```

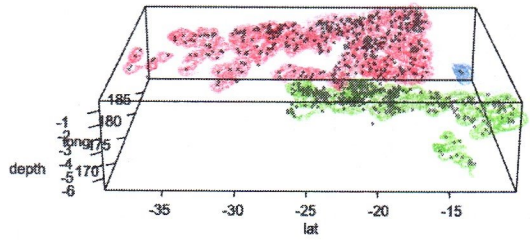
result of the chaining effect

only 2 point

but the indication from dendr. was k = 3

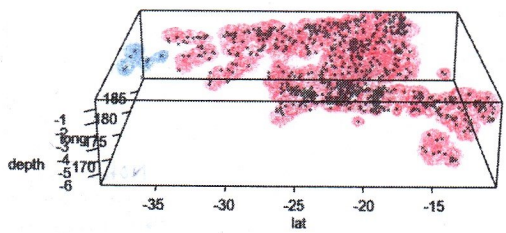



```
clusters <- cutree(clusts, 3)
plot3d(Q, size=3, col=clusters+1, aspect = F)
rglwidget()
```

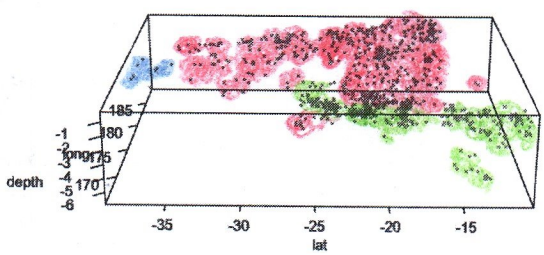


(looking at the 3D structure we see that it's a good separation)

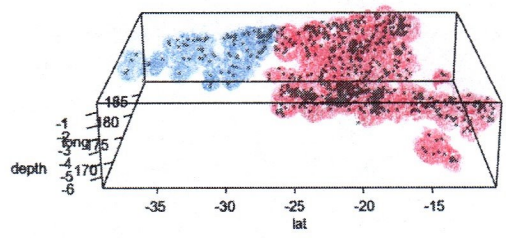
```
# average linkage
clustera <- cutree(clusta, 2)
plot3d(Q, size=3, col=clustera+1, aspect = F)
rglwidget()
```



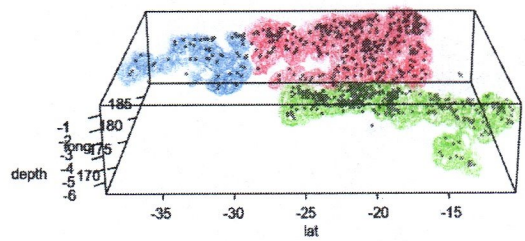

```
clustera <- cutree(clusta, 3)
plot3d(Q, size=3, col=clustera+1, aspect = F)
rglwidget()
```

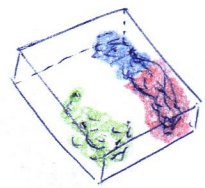



```
# complete linkage
clusterc <- cutree(clustc, 2)
plot3d(Q, size=3, col=clusterc+1, aspect = F)
rglwidget()
```



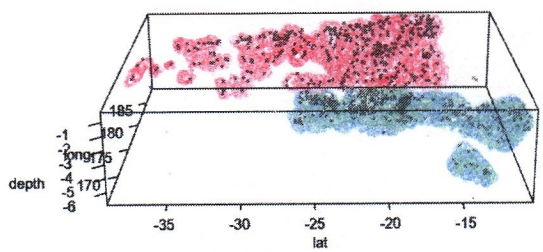

```
clusterc <- cutree(clustc, 3)
plot3d(Q, size=3, col=clusterc+1, aspect = F)
rglwidget()
```

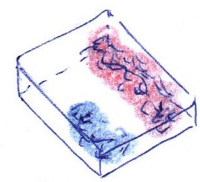




Note: the complete linkage tends to separate the elongated groups

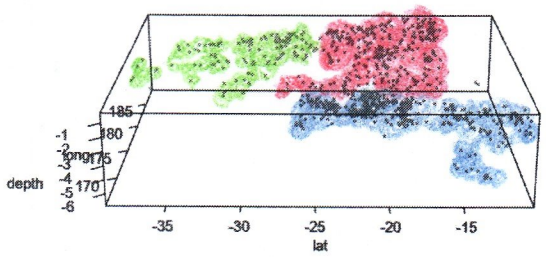
```
# ward linkage
clusterw <- cutree(clustw, 2)
plot3d(Q, size=3, col=clusterw+1, aspect = F)
rglwidget()
```



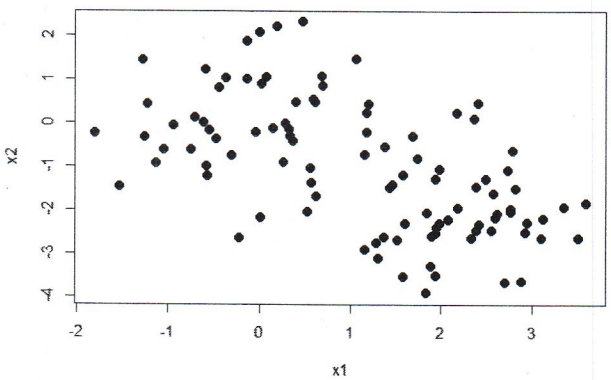


we have the clustering we expect (this is the only linkage that identify with k=2 the clusters we're looking for)

```
clusterw <- cutree(clustw, 3)
plot3d(Q, size=3, col=clusterw+1, aspect = F)
rglwidget()
```




```
### ---------------------------------------------------------------------------
### K-means method

################################################################################
# Recall K-means algorithm:

# simulated data
n <- 100
set.seed(1)
x <- matrix(rnorm(n*2), ncol=2)
x[1:(n/2),1] <- x[1:(n/2),1]+2
x[1:(n/2),2] <- x[1:(n/2),2]-2

x11()
plot(x,pch=20,cex=2,xlab='x1',ylab='x2')
```

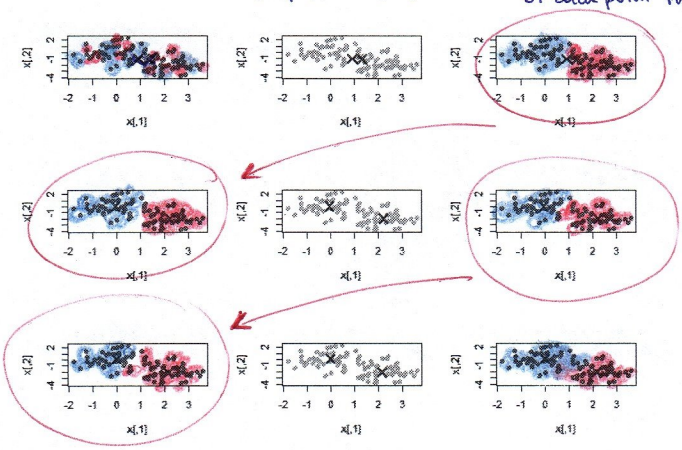



```
# k-means algorithm

k <- 2
cluster <- sample(1:2, n, replace=TRUE)
iter.max <- 3

colplot <- c('royalblue','red')
colpoints <- c('blue4','red4')

x11()
par(mfrow = c(iter.max,3))

for(i in 1:iter.max)
{
  C <- NULL
  for(l in 1:k)
    C <- rbind(C, colMeans(x[cluster == l,]))

  plot(x, col = colplot[cluster],pch=19)
  line <- readline()

  points(C, col = colpoints, pch = 4, cex = 2, lwd = 2)
  line <- readline()

  plot(x, col = 'grey',pch=19)
  points(C, col = colpoints, pch = 4, cex = 2, lwd = 2)
  line <- readline()

  QC <- rbind(C, x)
  Dist <- as.matrix(dist(QC, method = 'euclidean'))[(k+1):(k+n),1:k]
  for(j in 1:n)
    cluster[j] <- which.min(Dist[j,])

  plot(x, col = colplot[cluster],pch=19)
  points(C, col = colpoints, pch = 4, cex = 2, lwd = 2)
  line <- readline()

}
```

- random lable
- centroids

- forget about the previous lables but keep the centroids

- assign new labdes looking at the distance of each point from the centroid



then, we iterate (at each iteration the starting point is the last clustering from the previous iteration)

we go on with iterations until we don't see any changes anymore between the starting and ending clusters

```
graphics.off()

#############################################################################

# back to the earthquakes dataset

### in automatic, command kmeans()
help(kmeans)

result.k <- kmeans(Q, centers=2) # Centers: fixed number of clusters

names(result.k)
```

- **inputs**: dataset (Q) and centers (which is the number of clusters (we have to say it in advance)

```
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
result.k$cluster      # labels of clusters
```

```
##    [1] 1 1 1 1 1 1 2 1 1 1 1 2 1 1 2 1 2 1 2 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 2
##   [38] 1 1 2 1 1 1 1 2 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 2 1
##   [75] 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 2 2 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1
##  [112] 1 1 1 1 1 2 2 2 2 2 1 1 1 1 2 1 1 1 1 1 1 2 1 1 2 1 1 1 1 2 1 2 1 2 1 1 2
##  [149] 1 1 1 2 1 2 2 1 2 1 2 2 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [186] 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
##  [223] 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 2 2 2 1 2 1 1 1 2 1
##  [260] 1 1 1 2 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1
##  [297] 1 1 1 2 2 1 1 1 2 1 1 1 1 1 2 2 1 1 1 1 1 2 1 2 2 1 1 1 2 1 1 1 2 1 2 1 1
##  [334] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 2 1 1 2 1 1 1 1 1 1 1
##  [371] 1 1 1 1 1 1 1 1 1 1 2 2 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1
##  [408] 2 1 1 1 1 2 1 1 2 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 2 1 2 1
##  [445] 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1
##  [482] 1 1 1 1 1 1 1 1 2 1 2 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 2 2 2 1 1 1 1 1 1 1 2 1
##  [519] 1 1 1 1 1 2 1 1 2 2 1 1 2 2 1 2 1 2 1 2 2 1 2 2 2 2 2 2 2 1 1 1 1 2 2 1 1
##  [556] 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 2 1 1 1 1 1
##  [593] 2 2 1 2 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 2 1 1 1 1 2 1 2 1 1 1 1 1 2 1 1 1 2
##  [630] 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 2 2 1 1 1 1 1 1 1 1 1 1
##  [667] 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1
##  [704] 1 1 1 1 1 1 1 1 1 2 1 2 1 2 1 1 1 1 1 2 1 1 1 1 2 2 1 1 1 1 1 1 1 1 2 1 1 1 1
##  [741] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 2 1 1 1 1 1 1 2 2 1 1 2 1 1 1 1 1 1 1
##  [778] 1 2 1 2 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1
##  [815] 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1
##  [852] 2 2 1 1 1 2 1 1 2 1 1 1 1 2 2 1 1 2 2 1 2 2 1 1 1 1 1 1 1 1 1 2 2 1 1 2 2
##  [889] 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 2 1 1 2 2 1 1 1 1 1 2 1 1 1 1
##  [926] 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
##  [963] 2 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 2 1 2 1 1 1 1 2 2 2 1 1 1 1 1 1 1 1 2 1 1
## [1000] 2
```

```
result.k$centers      # centers of the clusters
```

```
##         lat     long     depth
## 1 -21.87143 182.3506 -3.479182
## 2 -15.87785 168.2598 -1.696390
```

```
result.k$totss        # tot. sum of squares
```

```
## [1] 66706.31
```

```
result.k$withinss     # sum of squares within clusters
```

```
## [1] 24075.266  3899.542
```

```
result.k$tot.withinss # sum(sum of squares nei cluster)
```

```
## [1] 27974.81
```

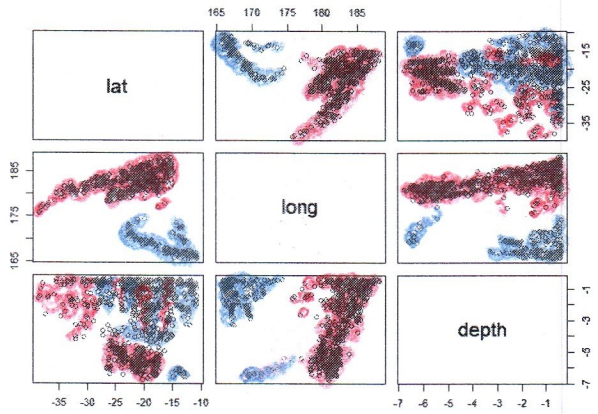
```
result.k$betweenss    # sum of squares between clusters
```

```
## [1] 38731.5
```

```
result.k$size         # dimention of the clusters
```

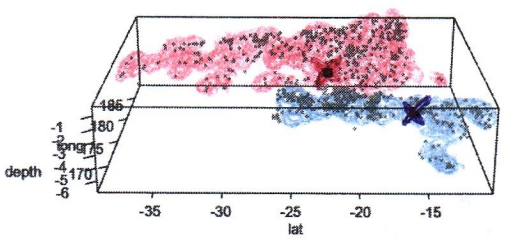
```
## [1] 795 205
```

```
x11()
plot(Q, col = result.k$cluster+1)
```

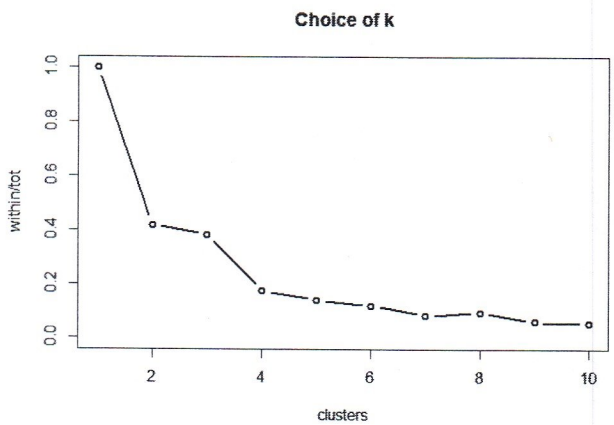



```
open3d()
```

```
plot3d(Q, size=3, col=result.k$cluster+1, aspect = F)
points3d(result.k$centers,size=10)
rglwidget()
```

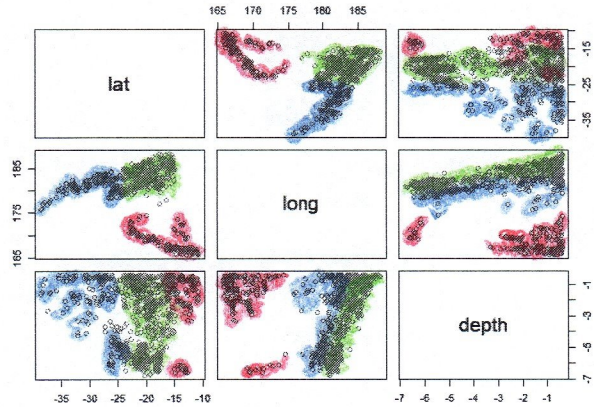



```
### How to choose k:
### 1) evaluate the variability between the groups with respect to
###    the variability within  the groups
### 2) evaluate the result of hierarchical clustering (not recommended,
###     quite computationally expensive)

# (we've just seen method 2) and suggested k = 2)

# method 1)
b <- NULL
w <- NULL
for(k in 1:10){

  result.k <- kmeans(Q, k)
  w <- c(w, sum(result.k$wit))
  b <- c(b, result.k$bet)

}

x11()
matplot(1:10, w/(w+b), pch='', xlab='clusters', ylab='within/tot', main='Choice of k', ylim=c(0,1))
lines(1:10, w/(w+b), type='b', lwd=2)
```



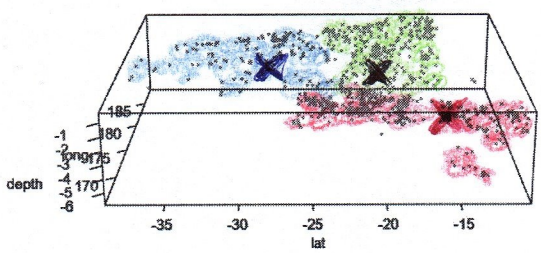
← in this case we should pick k = 2 (/ k = 3) since moving to k = 4 would provide much more complex representation (k=2 already has a significant decrease of the within variability)

```
# this method seems to suggest k = 2 or 3
# let's try also k=3:
result.k <- kmeans(Q, 3)

x11()
plot(Q, col = result.k$cluster+1)
```




```
open3d()
```

```
plot3d(Q, size=3, col=result.k$cluster+1, aspect = F)
points3d(result.k$centers, size = 10)
rglwidget()
```



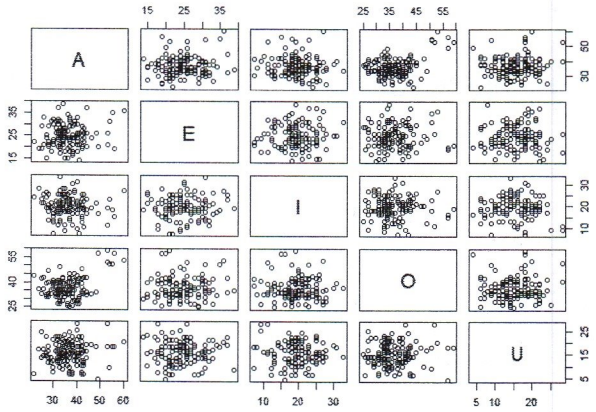

```
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### Exercises
### ------------------------------------------------------------------------
### Problem 3 of July 1, 2009
### ------------------------------------------------------------------------
# The Veritatis Diagram is a mysterious work attributed to Galileo. Some
# semiologists believe that some pages of the book hide a coded message;
# they also believe that these pages are characterized by an abnormal
# numerosity of some letters of the alphabet. The veritatis.txt file
# lists, for 132 pages of the book, the absolute frequencies of the five
# vowels of the Latin alphabet.
# a) By using an agglomerative clustering algorithm (Manhattan distance,
#    average linkage), identify two clusters and report suspicious pages;
# b) assuming that, for each of the two clusters, the five absolute
#    frequencies are (approximately) normally distributed with the same
#    covariance matrix perform a test to prove the existence of a difference
#    in the mean value of the two distributions;
# c) using five Bonferroni intervals of global confidence 90%, comment the
#    results of the test at point (b).

vowels <- read.table('veritatis.txt', header=T)
head(vowels)
```

```
##    A  E  I  O  U
## 1 41 26 20 31 15
## 2 37 29 19 43  6
## 3 32 32 11 34 16
## 4 38 16 23 42 16
## 5 33 28 27 27 14
## 6 38 24 23 38 18
```

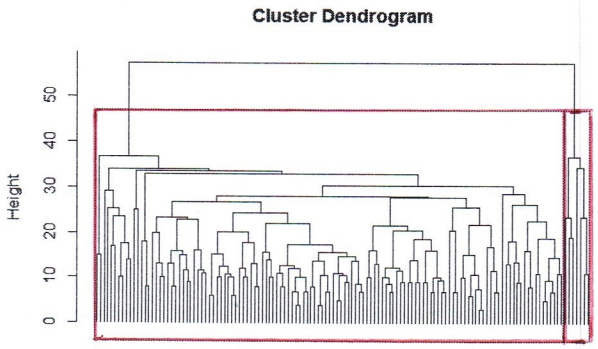
```
dim(vowels)
```

```
## [1] 132   5
```

```
### question a)
x11()
plot(vowels)
```




```
HC <- hclust(dist(vowels, method='manhattan'), method = 'average')

x11()
plot(HC, hang=-0.1, sub='', labels=F, xlab='')

# we cut the dendrogram at k=2 clusters
rect.hclust(HC, k=2)
```



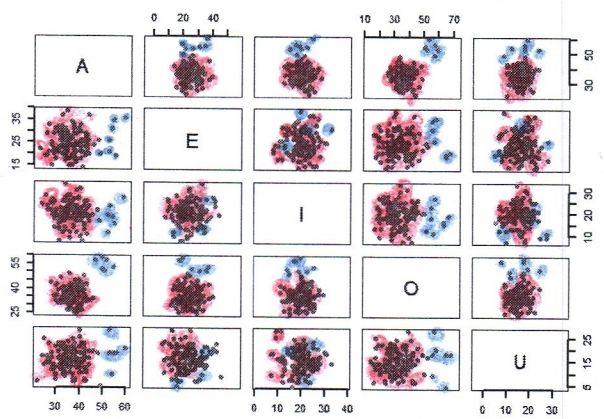

```
pag <- cutree(HC, k=2)
table(pag)
```

```
## pag
##   1   2
## 125   7
```

```
which(pag==2)
```

```
##  33  54  79  93 102 105 117
##  33  54  79  93 102 105 117
```

```
x11()
plot(vowels , col=pag+1, asp=1, pch=16, lwd=2)
```



```
### question b)
p  <- 5
n1 <- table(pag)[1]
n2 <- table(pag)[2]

# Verify gaussianity
load("./mcshapiro.test.RData")

mcshapiro.test(vowels[pag=='1',])
```

```
## $Wmin
## [1] 0.9792442
##
## $pvalue
## [1] 0.6064
##
## $devst
## [1] 0.009770958
##
## $sim
## [1] 2500
```

```
mcshapiro.test(vowels[pag=='2',])
```

```
## $Wmin
## [1] 0.5959427
##
## $pvalue
## [1] 0.9064
##
## $devst
## [1] 0.005825428
##
## $sim
## [1] 2500
```

```
# Test for independent Gaussian populations
t1.mean <- sapply(vowels[pag=='1',],mean)
t2.mean <- sapply(vowels[pag=='2',],mean)
t1.cov  <-  cov(vowels[pag=='1',])
t2.cov  <-  cov(vowels[pag=='2',])
Sp      <- ((n1-1)*t1.cov + (n2-1)*t2.cov)/(n1+n2-2)

# Test: H0: mu.1-mu.2==0 vs H1: mu.1-mu.2!=0
delta.0 <- c(0,0,0,0,0)
Spinv   <- solve(Sp)
T2 <- n1*n2/(n1+n2) * (t1.mean-t2.mean-delta.0) %*% Spinv %*% (t1.mean-t2.mean-delta.0)
P <- 1 - pf(T2/(p*(n1+n2-2)/(n1+n2-1-p)), p, n1+n2-1-p)
P
```

```
##      [,1]
## [1,]    0
```

← the p-value is even smaller than the 0 of R → there is a difference in the means of the 2 populations

```
### question c)
alpha <- 0.1
IC <- cbind(t2.mean-t1.mean - sqrt(diag(Sp)*(1/n1+1/n2)) * qt(1 - alpha/(p*2), n1+n2-2),
            t2.mean-t1.mean,
            t2.mean-t1.mean + sqrt(diag(Sp)*(1/n1+1/n2)) * qt(1 - alpha/(p*2), n1+n2-2))
IC
```

since we want 90%

: correction

```
##        [,1]      [,2]      [,3]
## A 14.267514 18.949714 23.631915
## E -2.418594  2.201143  6.820880
## I -5.791398 -1.195429  3.400541
## O 13.936536 18.565714 23.194893
## U -3.277573  1.244571  5.766716
```

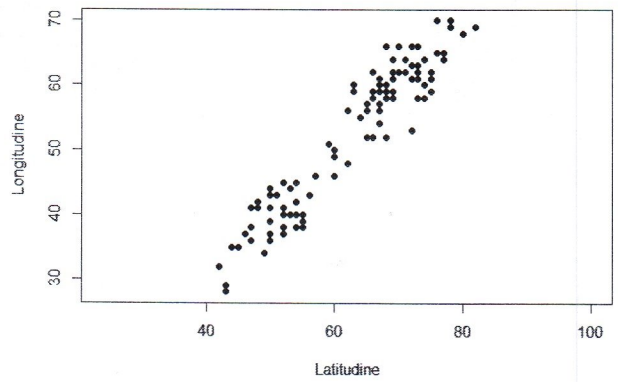
only 2 of them do not contain the origin ("A", "O"). This tells us that the previous results (conclusions of point b) are justified by the values of the first feature and the fourth feature (the absolute frequency of letter A and O are significantly different in the 2 groups)

```
graphics.off()
```

```
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### Problem 2 of February 18, 2009
### ------------------------------------------------------------------------
# Friday, October 17, 2008, in Black Fortune skies a crash occurred between
# two artificial satellites. About a hundred debris were found on the
# ground (satellite.txt file). However, due to friction with the atmosphere it
# was not possible to define the origin of any debris. To clarify the
# accident, the US Air Force requests to estimate the relative position of
# the two points of impact on the ground.
# Assuming that the debris coming from the same satellite are scattered on
# the ground according to a normal law of unknown mean and covariance matrix:
# a) attribute the debris to the two satellites via a hierarchical clustering
#    algorithm that uses the Euclidean distance and the average linkage;
# b) report the numerosity of the two clusters and the value of the cophenetic
#    coefficient;
# c) introducing the appropriate hypotheses, identifying the points of impact
#    on the ground with the mean of the two normal distributions and assuming
#    correct the allocation of the debris to the two satellites, provide an elliptical
#    confidence region (global level 99%) for the relative position of the two
#    points of impact on the ground.

satellite <- read.table('satellite.txt', header=T)
head(satellite)
```

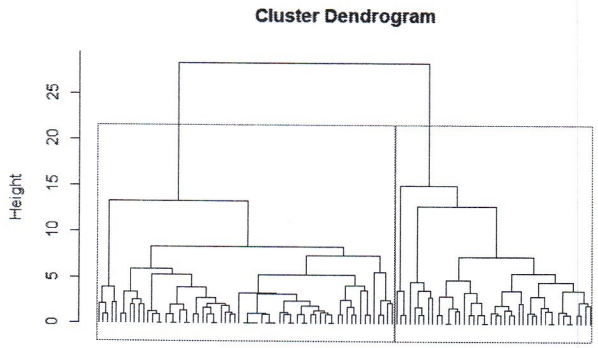
```
##   Latitudine Longitudine
## 1         55          38
## 2         43          29
## 3         56          43
## 4         72          63
## 5         50          44
## 6         75          62
```

```
x11()
plot(satellite, asp = 1, pch=16)
```

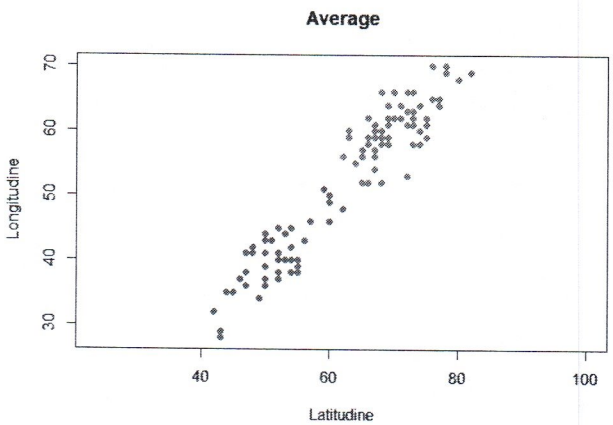



```
dev.off()
```

```
### question a)

D.s <- dist(satellite)

HCa <- hclust(D.s, method = 'average')

x11()
plot(HCa, hang=-0.1, sub='', xlab='', labels=F)

# we know that there are 2 clusters, so we cut the dendrograms
# accordingly:
rect.hclust(HCa, k=2)
```




```
sata <- cutree(HCa, k=2)

x11()
plot(satellite , col=sata+1, asp=1, pch=16, main='Average')
```




```
dev.off()
```

```
### question b)

table(sata)
```

```
## sata
##  1  2
## 44 66
```

```r
coph.a <- cophenetic(HCa)

coph.sat <- cor(D.s, coph.a)
coph.sat
```

```
## [1] 0.8608342
```

```r
### question c)

p  <- 2
n1 <- table(sata)[1]
n2 <- table(sata)[2]

# Assumptions:
# - Normality
# - Independent populations
# - Homogeneity of covariance structures

# Verificy Gaussian assumption
mcshapiro.test(satellite[sata=='1',])$pvalue
```

```
## [1] 0.8472
```

```r
mcshapiro.test(satellite[sata=='2',])$pvalue
```

```
## [1] 0.4696
```

```r
t1.mean <- sapply(satellite[sata=='1',],mean)
t2.mean <- sapply(satellite[sata=='2',],mean)
t1.cov  <-  cov(satellite[sata=='1',])
t2.cov  <-  cov(satellite[sata=='2',])

# Homogeneity of covariances
t1.cov
```

```
##             Latitudine Longitudine
## Latitudine    23.33827    19.50740
## Longitudine   19.50740    25.80761
```

```r
t2.cov
```

```
##             Latitudine Longitudine
## Latitudine    19.29627    12.00909
## Longitudine   12.00909    17.40909
```

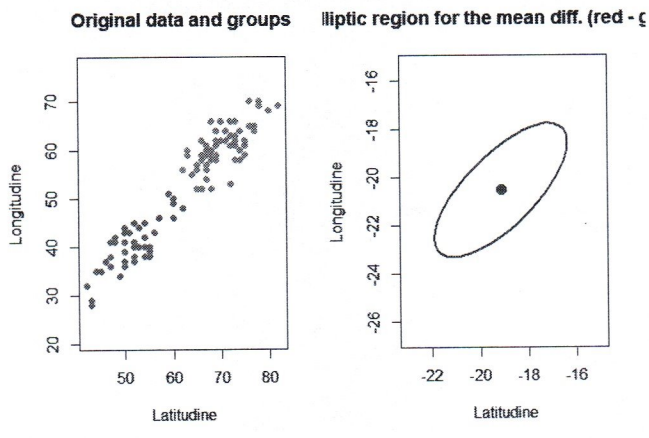
```r
Sp      <- ((n1-1)*t1.cov + (n2-1)*t2.cov)/(n1+n2-2)

# Elliptic confidence region at 99%
alpha <- 0.01
cfr.fisher <- (p*(n1+n2-2)/(n1+n2-1-p))*qf(1-alpha, p, n1+n2-1-p)

# Characterize the ellipse:
# Directions of the axes
eigen(Sp)$vector
```

```
##           [,1]      [,2]
## [1,] -0.708904  0.705305
## [2,] -0.705305 -0.708904
```

```r
# Radius
r <- sqrt(cfr.fisher)

# Length of the semi-axes
r*sqrt(eigen(Sp)$values*(1/n1+1/n2))
```

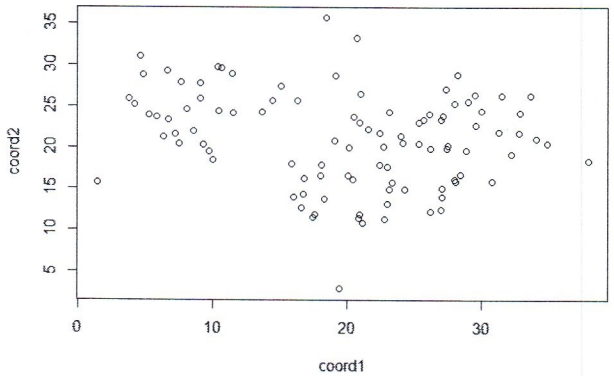
```
## [1] 3.629574 1.464779
```

```r
x11(width=14, height = 7)
par(mfrow=c(1,2))
plot(satellite , col=sata+1, asp=1, pch=16, main='Original data and groups')
plot(satellite, xlim=c(-23,-15), ylim=c(-25,-17), pch='', asp=1,
     main='Elliptic region for the mean diff. (red - green)')
# confidence region and sample mean in blue
ellipse(center=t1.mean-t2.mean, shape=Sp*(1/n1+1/n2), radius=sqrt(cfr.fisher),
        lwd=2, col='blue')
```

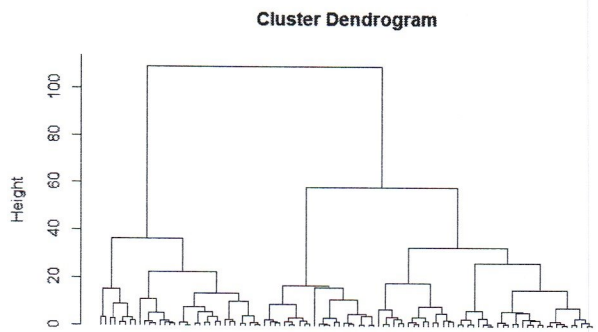


```
graphics.off()
```

```
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### Problem 2 of February 28, 2013
### ------------------------------------------------------------------------
# In view of the week of haute couture in Paris, the fashion house La Boutin
# has decided to create a unique-edition pair of shoes decorated with
# gemstones instead of crystals. During the processing of the pair of shoes,
# a bumbling craftsman caused the fall of the gemstones and of a
# box of crystals. The file preziosi.txt collects data related to the Cartesian
# coordinates of the found stones.
# a) Through a hierarchical clustering algorithm (Euclidean distance and
#    Ward linkage), identify the two clusters of stones.
# b) reporting the numerosity of the two groups identified at point a) and
#    the value of the cophenetic coefficient.
# c) Identify the gemstones with the smaller group. Having introduced
#    (and verified) the appropriate assumptions, estimate an elliptical region
#    containing 99% of gemstones (report the center, the length and the direction
#    of the principal axes of the ellipse).

stones <- read.table('preziosi.txt',header=TRUE)
plot(stones)
```




```
# question a)
p.e <- dist(stones, method='euclidean')
p.ew <- hclust(p.e, method='ward.D2')

plot(p.ew, hang=-0.1, sub='', xlab='', labels=F)
```



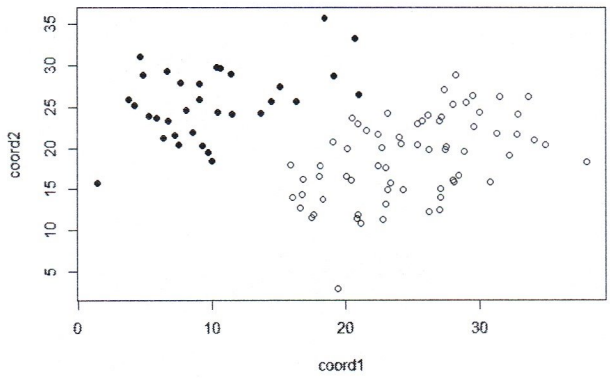

```
cl.ew <- cutree(p.ew, k=2) # euclidea-ward

# question b)
table(cl.ew)
```

```
## cl.ew
##  1  2
## 67 33
```

```
coph.ew <- cophenetic(p.ew)
ew <- cor(p.e, coph.ew)
ew
```

```
## [1] 0.7424942
```

```
# question c)
p.pr<-stones[which(cl.ew==2),]
n <- dim(p.pr)[1]
p <-dim(p.pr)[2]

plot(stones)
points(p.pr, pch=19)
```




```
# Verify gaussian assumptions
mcshapiro.test(p.pr)
```

```
## $Wmin
## [1] 0.9762131
##
## $pvalue
## [1] 0.77
##
## $devst
## [1] 0.00841665
##
## $sim
## [1] 2500
```

```
M <- sapply(p.pr, mean)
S <- cov(p.pr)
alpha <- 0.01
cfr.chisq <- qchisq(1-alpha,p)

# Characterize the ellipse:
# Axes directions:
eigen(S)$vectors
```

```
##            [,1]       [,2]
## [1,] -0.8108739  0.5852209
## [2,] -0.5852209 -0.8108739
```

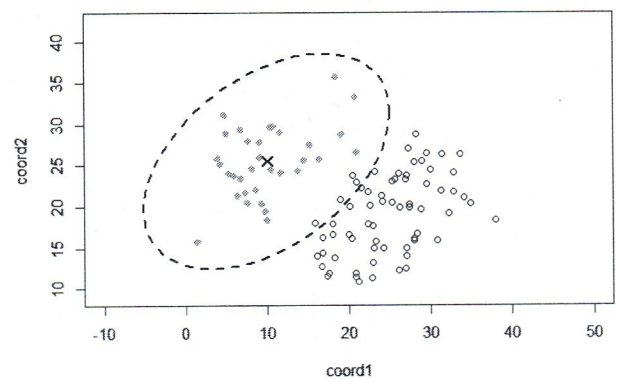
```
# Center:
M
```

```
##    coord1    coord2
##  9.974545 25.487576
```

```
# Radius of the ellipse:
r <- sqrt(cfr.chisq)
# Length of the semi-axes:
r*sqrt(eigen(S)$values)
```

```
## [1] 17.13797 10.25600
```

```
x11()
plot(p.pr, asp = 1, col='gold', pch=19, xlim=c(-10,50))
points(M[1], M[2], pch = 4, cex = 1.5, lwd = 2)

ellipse(center=M, shape=S, radius=sqrt(cfr.chisq), col = 'black', lty = 2, center.pch = 4)
points(stones[which(cl.ew==1),])
```




```
graphics.off()
```

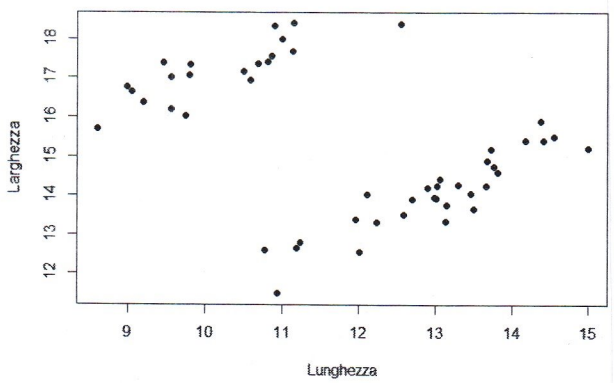
```
### ---------------------------------------------------------------------------
### ---------------------------------------------------------------------------
### Problem 3 of February 28, 2007
### ---------------------------------------------------------------------------
# The dataset Pb3.txt reports Length (cm) Width (cm) of the chest of
# 50 sparrows. The biologist who has collected the measurements aims to
# demonstrate that the sparrows can be divided into two distinct groups
# in terms of length and width of the chest. Help him to prove his theory
# by implementing and commenting on the following analyses:
# (a) By means of an agglomerative hierarchical clustering algorithm that
#     uses the Manhattan distance and the Single linkage state if it is
#     reasonable to cluster the data into two groups.
# (b) Implement a test to prove the difference of the means of the two groups.
# (c) Identify and comment the four Bonferroni intervals with global confidence
#     90% (lower bound, central value, upper bound) for:
#     - The difference of the mean of the variable length.
#     - The difference of the mean of the variable width.
#     - The difference of the mean of the sum of the variables length and width.
#     - The difference of the mean of the difference of variable length and
#       width.
# (d) verify the assumptions necessary to the implementation of the test.

sparrows <- read.table('Pb3.txt')
head(sparrows)
```

```
##   Lunghezza Larghezza
## 1     9.801    17.070
## 2     8.591    15.711
## 3    14.179    15.420
## 4    13.506    13.656
## 5    13.809    14.595
## 6    12.238    13.312
```

```
dim(sparrows)
```

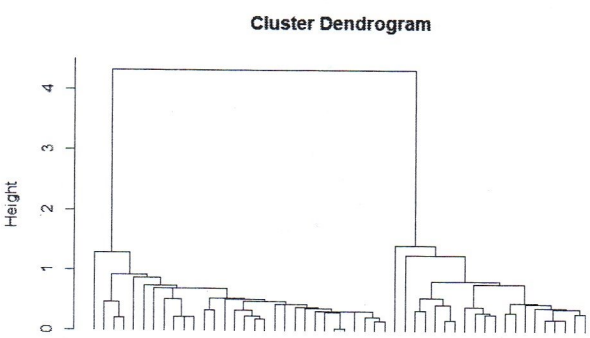
```
## [1] 50  2
```

```
### question (a)

x11()
plot(sparrows, pch=16)
```

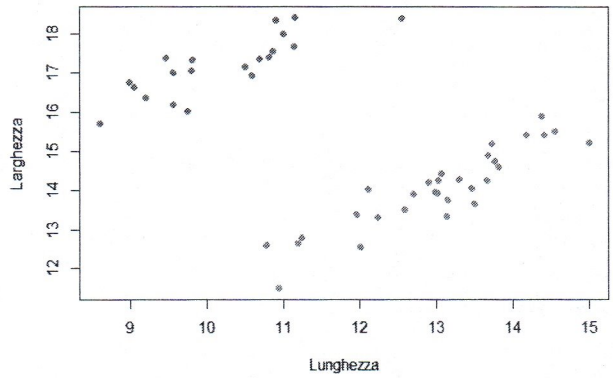



```
dev.off()
```

```
gruppi <- hclust(dist(sparrows, method='manhattan'), method='single')
plot(gruppi, hang=-0.1, sub='', xlab='', labels=F)
```




```
# cut in two groups
cluster <- cutree(gruppi, k=2)

plot(sparrows, pch=16, col = as.vector(cluster)+1)
```




```
### question (b)
g1 <- sparrows[cluster==1,]
g2 <- sparrows[cluster==2,]
g1
```

```
##    Lunghezza Larghezza
## 1      9.801    17.070
## 2      8.591    15.711
## 7     10.995    17.987
## 12    11.146    18.404
## 15    10.679    17.343
## 16    10.810    17.414
## 19    10.896    18.352
## 25    11.132    17.674
## 26     9.742    16.026
## 28     9.815    17.329
## 30     9.460    17.374
## 31    12.545    18.387
## 35     8.979    16.761
## 37    10.501    17.150
## 41     9.195    16.366
## 43     9.554    17.004
## 44    10.584    16.933
## 47     9.043    16.649
## 48    10.866    17.556
## 50     9.562    16.187
```

```
g2
```

```
##    Lunghezza Larghezza
## 3     14.179    15.420
## 4     13.506    13.656
## 5     13.809    14.595
## 6     12.238    13.312
## 8     13.722    15.185
## 9     12.109    14.023
## 10    13.138    13.335
## 11    13.304    14.280
## 13    12.898    14.194
## 14    13.153    13.762
## 17    10.771    12.602
## 18    12.583    13.501
## 20    13.023    14.242
## 21    14.374    15.904
## 22    12.014    12.559
## 23    10.938    11.480
## 24    14.553    15.505
## 27    13.676    14.897
## 29    13.460    14.053
## 32    13.662    14.259
## 33    15.000    15.216
## 34    14.411    15.417
## 36    13.762    14.740
## 38    11.186    12.655
## 39    11.960    13.378
## 40    13.059    14.434
## 42    11.241    12.807
## 45    13.014    13.932
## 46    12.987    13.942
## 49    12.695    13.908
```

```
# Test: H0: mu.1-mu.2==0 vs H1: mu.1-mu.2!=0
p  <- 2
n1 <- dim(g1)[1]
n2 <- dim(g2)[1]
alpha <- 0.10

mean1 <- sapply(g1,mean)
mean2 <- sapply(g2,mean)
cov1  <-  cov(g1)
cov2  <-  cov(g2)
Sp      <- ((n1-1)*cov1 + (n2-1)*cov2)/(n1+n2-2)

delta.0 <- c(0,0)
Spinv   <- solve(Sp)

T2 <- n1*n2/(n1+n2) * (mean1-mean2-delta.0) %*% Spinv %*% (mean1-mean2-delta.0)

cfr.fisher <- (p*(n1+n2-2)/(n1+n2-1-p))*qf(1-alpha,p,n1+n2-1-p)

pvalue <- 1 - pf(T2/(p*(n1+n2-2)/(n1+n2-1-p)), p, n1+n2-1-p)
pvalue
```

```
##      [,1]
## [1,]    0
```

```
### question (c)

dm <- (mean1-mean2)
A  <- rbind(c(1,0), c(0,1), c(1,1), c(1,-1))
k  <- dim(A)[1]

A.s2 <- diag(A%*%Sp%*%t(A))
A.dm <- A%*%(mean1-mean2)

Bonf <- cbind(inf=A.dm - qt(1-(alpha/(2*k)), n1+n2-2) * sqrt( A.s2*(1/n1+1/n2) ),
              center=A.dm,
              sup=A.dm + qt(1-(alpha/(2*k)), n1+n2-2) * sqrt( A.s2*(1/n1+1/n2) ))
Bonf
```

```
##            [,1]       [,2]      [,3]
## [1,] -3.5160953 -2.8193667 -2.122638
## [2,]  2.5233414  3.1440833  3.764825
## [3,] -0.9524155  0.3247167  1.601849
## [4,] -6.2957694 -5.9634500 -5.631131
```

```
### question (d)
mcshapiro.test(g1)$pvalue
```

```
## [1] 0.2692
```

```
mcshapiro.test(g2)$pvalue
```

```
## [1] 0.9508
```

```
cov1
```

```
##           Lunghezza Larghezza
## Lunghezza 0.9462719 0.6157970
## Larghezza 0.6157970 0.5898402
```

```
cov2
```

```
##           Lunghezza Larghezza
## Lunghezza  1.180818  1.006807
## Larghezza  1.006807  1.042966
```

```
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### Multidimensional Scaling (principal coordinate analysis)
### ------------------------------------------------------------------------
### Given the distances (dissimilarities) among n statistical units, look for
### the k-dimensional representation (k small) of the n statistical units
### such that the distances (dissimilarities) among the representations
### of the n units are as close as possible to the original distances
### (dissimilarities) among the n units.

### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### Example 1: European cities
### ------------------------------------------------------------------------

help(eurodist)
eurodist
```

: it contains the distances between European cities (measured by road-distances (km))

```
##                Athens Barcelona Brussels Calais Cherbourg Cologne Copenhagen
## Barcelona        3313
## Brussels         2963      1318
## Calais           3175      1326      204
## Cherbourg        3339      1294      583    460
## Cologne          2762      1498      206    409       785
## Copenhagen       3276      2218      966   1136      1545     760
## Geneva           2610       803      677    747       853    1662      1418
## Gibraltar        4485      1172     2256   2224      2047    2436      3196
## Hamburg          2977      2018      597    714      1115     460       460
## Hook of Holland  3030      1490      172    330       731     269       269
## Lisbon           4532      1305     2084   2052      1827    2290      2971
## Lyons            2753       645      690    739       789     714      1458
## Madrid           3949       636     1558   1550      1347    1764      2498
## Marseilles       2865       521     1011   1059      1101    1035      1778
## Milan            2282      1014      925   1077      1209     911      1537
## Munich           2179      1365      747    977      1160     583      1104
## Paris            3000      1033      285    280       340     465      1176
## Rome              817      1460     1511   1662      1794    1497      2050
## Stockholm        3927      2868     1616   1786      2196    1403       650
## Vienna           1991      1802     1175   1381      1588     937      1455
##                Geneva Gibraltar Hamburg Hook of Holland Lisbon Lyons Madrid
## Barcelona
## Brussels
## Calais
## Cherbourg
## Cologne
## Copenhagen
## Geneva
## Gibraltar         1975
## Hamburg           1118      2897
## Hook of Holland    895      2428      550
## Lisbon            1936       676     2671            2280
## Lyons              158      1817     1159             863   1178
## Madrid            1439       698     2198            1730    668  1281
## Marseilles         425      1693     1479            1183   1762   320  1157
## Milan              328      2185     1238            1098   2250   328  1724
## Munich             591      2565      805             851   2507   724  2010
## Paris              513      1971      877             457   1799   471  1273
## Rome               995      2631     1751            1683   2700  1048  2097
## Stockholm         2068      3886      949            1500   3231  2108  3188
## Vienna            1019      2974     1155            1205   2937  1157  2409
##                Marseilles Milan Munich Paris Rome Stockholm
## Barcelona
## Brussels
## Calais
## Cherbourg
## Cologne
## Copenhagen
## Geneva
## Gibraltar
## Hamburg
## Hook of Holland
## Lisbon
## Lyons
## Madrid
## Marseilles
## Milan                618
## Munich              1109   331
## Paris                792   856    821
## Rome                1011   586    946  1476
## Stockholm           2428  2187   1754  1827 2707
## Vienna              1363   898    428  1249 1209      2105
```

```
# road distances among European cities

# R function for multidimensional scaling: cmdscale
help(cmdscale)

location <- cmdscale(eurodist, k=2)
location
```

inputs: distances matrix (type "dist") and dimension of the space on which we want to project the data

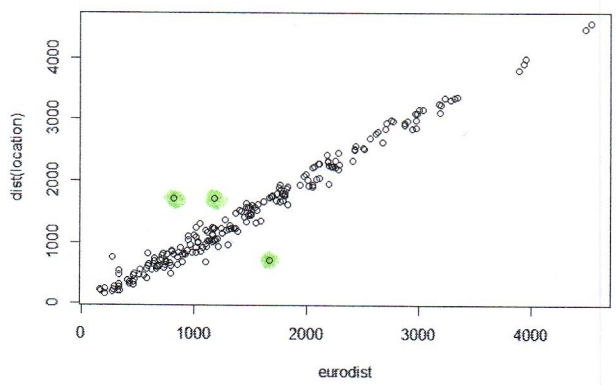
```
##                      [,1]       [,2]
## Athens         2290.274680  1798.80293
## Barcelona      -825.382790   546.81148
## Brussels         59.183341  -367.08135
## Calais          -82.845973  -429.91466
## Cherbourg      -352.499435  -290.90843
## Cologne         293.689633  -405.31194
## Copenhagen      681.931545 -1108.64478
## Geneva           -9.423364   240.40600
## Gibraltar    -2048.449113   642.45854
## Hamburg        561.108970  -773.36929
## Hook of Holland  164.921799  -549.36704
## Lisbon       -1935.040811    49.12514
## Lyons          -226.423236   187.08779
## Madrid       -1423.353697   305.87513
## Marseilles     -299.498710   388.80726
## Milan           260.878046   416.67381
## Munich          587.675679    81.18224
## Paris          -156.836257  -211.13911
## Rome            709.413282  1109.36665
## Stockholm       839.445911 -1836.79055
## Vienna          911.230500   205.93020
```

← we obtain the coordinates of the 21 cities on the new plane that we had obtained with multidimensional scaling (with k=2)

```
# I have to set asp=1 (equal scales on the two axes)
# to correctly represent Euclidean distances
x11()
plot(location[,1], location[,2], type='n', asp=1, axes=FALSE, main="MDS of European cities",xlab='',ylab='')
text(location[,1], location[,2], labels=colnames(as.matrix(eurodist)), cex = 0.75, pos = 3)
```


```
# change the sign to get the North in the upper part of the plot
x11()
plot(location[,1], -location[,2], type='n', asp=1, axes=FALSE, main="MDS of European cities",xlab='',ylab='')
text(location[,1], -location[,2], labels=colnames(as.matrix(eurodist)), cex = 0.75, pos = 3)
```

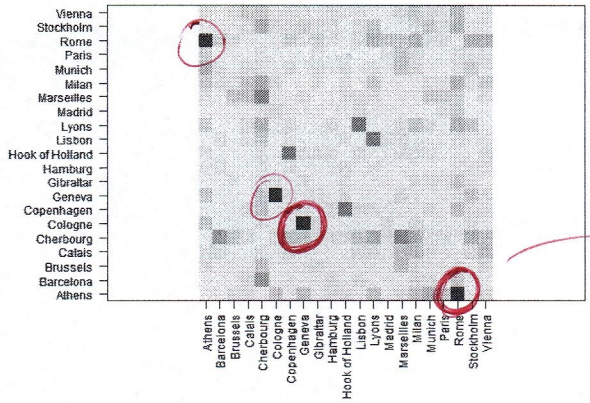

```
# compare the original matrix d_ij = d(x_i,x_j) and delta_ij = d(y_i,y_j)
x11()
plot(eurodist, dist(location))
```



every point is a couple of cities, the x coordinate is the original distance (road distance), on the y axe we have the distance in the new 2-dimensional space (the Euclidean distance between the two points)

we see a couple of outliers (●)

```
# visualize the most different distances
x11()
par(cex = 0.75, mar = c(10,10,2,2))
image(1:21, 1:21, asp=1, abs(as.matrix(dist(location)) - as.matrix(eurodist)), axes = F, xlab = '', ylab ='')
axis(1, at = 1:21, labels = colnames(as.matrix(eurodist)), las = 2, cex = 0.75)
axis(2, at = 1:21, labels = colnames(as.matrix(eurodist)), las = 1, cex = 0.75)
box()
```



outliers
(Attention: only 2 outliers, the matrix is symmetric!)

```
# Rome-Athens
as.matrix(eurodist)[19,1]
```

```
## [1] 817
```

```
as.matrix(dist(location))[19,1]
```

```
## [1] 1724.658
```

```
# Cologne-Geneve
as.matrix(eurodist)[6,8]
```

```
## [1] 1662
```

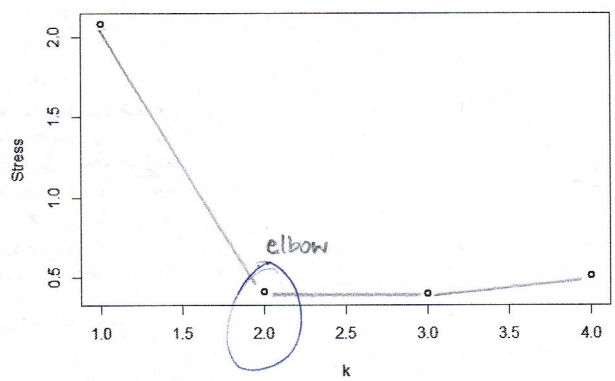
```
as.matrix(dist(location))[6,8]
```

```
## [1] 713.3226
```

(to check how good is the matching between old and new matrix of distance)

(we can use this to choose the dimension of the new representation of the data: we vary k and for each k we compute the stress

```
# Compute the "stress": the higher it is, the worse
# the matching between original distances and their
# geometrical representation through MDS
Stressk <- NULL
for(k in 1:4)
{
  location.k <- cmdscale(eurodist, k)
  Stress <- (sum( (as.vector(eurodist) - as.vector(dist(location.k)))^2)  /
               sum( as.vector(location.k)^2))^(1/2)
  Stressk <- c(Stressk, Stress)
}

x11()
plot(1:4,Stressk,xlab='k',ylab='Stress',lwd=2)
```



we have an indication for: k = 2

```
# the stress increases for k>2 because of numerical problems
# (the representation of minimal stress is not always found)

graphics.off()
```