Homework2_modernstat

Federico Veronesi

2023-10-05

```
library(ggplot2)

library(tidyverse)

library(mvtnorm)
```

Exercise 1

Preliminary commands

Function for gap-statistics

```
require(cluster)

## Caricamento del pacchetto richiesto: cluster

gapnc <- function(data,FUNcluster=kmeans,
                  K.max=10, B = 100, d.power = 2,
                  spaceH0 ="scaledPCA",
                  method ="globalSEmax", SE.factor = 2,...){
# As in original clusGap function the ... arguments are passed on
# to the clustering method FUNcluster (kmeans).
# Run clusGap
    gap1 <- clusGap(data,kmeans,K.max, B, d.power,spaceH0,...)
# Find optimal number of clusters; note that the method for
# finding the optimum and the SE.factor q need to be specified here.
    nc <- maxSE(gap1$Tab[,3],gap1$Tab[,4],method, SE.factor)
# Re-run kmeans with optimal nc.
    kmopt <- kmeans(data,nc,...)
    out <- list()
    out$gapout <- gap1
    out$nc <- nc
    out$kmopt <- kmopt
    out
}
```

Load data

```
oliveoil <- read.csv("C:/Users/Veronesi/Desktop/uniBo/Magistrale/Modern Statistics
and Big Data Analytics/Datasets/oliveoil.dat", sep = " ")
artdata2 <- read.csv("C:/Users/Veronesi/Desktop/uniBo/Magistrale/Modern Statistics
and Big Data Analytics/Datasets/artdata2.dat", sep = " ", header = F)
```

Point a: olive oil

```
set.seed(1234)
oliveoilclust <- scale(oliveoil[,3:10], T, T)
gap_output <- gapnc(oliveoilclust, K.max = 10, nstart=100, SE.factor = 2)

gap_output$nc
```

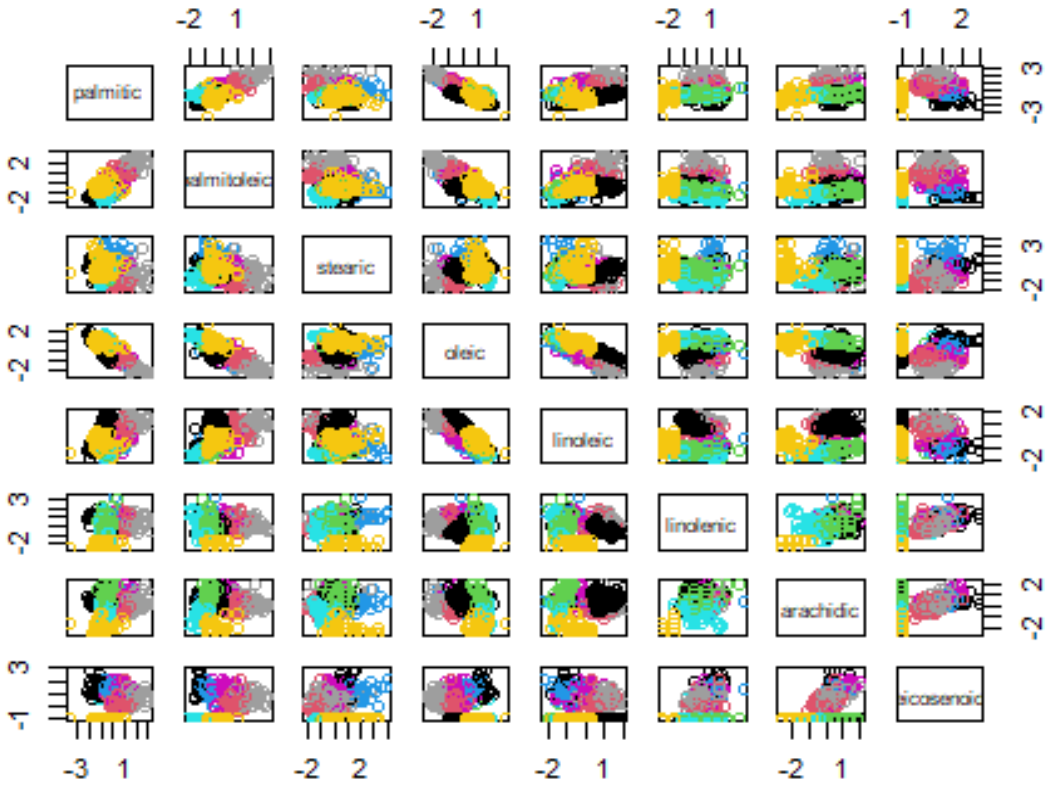
```
## [1] 9
```

```
km <- gap_output$kmopt
km
```

```
## K-means clustering with 9 clusters of sizes 98, 144, 35, 36, 58, 62, 60, 51, 28
##
## Cluster means:
##     palmitic palmitoleic     stearic      oleic   linoleic   linolenic
## 1 -0.7141153  -0.5590983 -0.07298179 -0.1077544  0.8896348 -0.36983477
## 2  0.7400028   0.9303122 -0.68558492 -0.8137655  0.7169267  0.17998032
## 3 -0.5241630  -0.7196986  0.39555425  0.9675748 -1.1515075  0.25537554
## 4  0.1219765  -0.4891523  2.07424366  0.1342351 -0.7970700  1.13098821
## 5 -0.8135311  -1.2544780 -0.85125141  1.5546846 -1.5497140 -0.05784551
## 6  0.5200600   0.1539787  0.18977650 -0.1857849 -0.3916507  0.76852182
## 7 -0.9732036  -0.4240659  0.72485134  0.9583751 -0.5073931 -2.07330862
## 8  1.7631814   1.7291064 -0.02995457 -1.6214070  1.0583322  0.28682382
## 9 -1.4004106  -1.2822079  0.46436840  1.2707426 -1.0996422  0.93393252
##      arachidic eicosenoic
## 1  0.684312137 -1.0184188
## 2 -0.003183092  0.4859325
## 3  0.903398650 -0.9897266
## 4  0.338518788  1.1023050
## 5 -0.700978421 -0.9895867
## 6  0.548315039  1.0611393
## 7 -2.190831665 -1.0105212
## 8  0.212726214  0.6691910
## 9  0.601865204  1.5320047
##
## Clustering vector:
##   [1] 9 9 9 9 9 9 9 9 9 9 5 9 9 9 9 6 9 6 9 9 9 9 9 9 9 6 6 6 6 6 6 4 6 6 6 6 4
##  [38] 6 4 6 6 4 4 4 4 6 6 6 4 4 4 4 4 4 4 6 4 6 3 6 6 6 6 6 6 6 6 6 6 6 4 4 4 4
##  [75] 6 4 4 4 4 6 6 2 2 2 2 2 2 8 8 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [112] 2 2 2 2 2 2 2 2 2 6 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 6 8 2 6 2 2 2 2 8 2
## [149] 2 2 2 2 2 8 8 8 8 8 8 8 2 2 8 2 2 2 8 8 2 2 2 2 8 2 6 8 8 2 2 2 2 2 2 2 2
## [186] 2 6 2 2 8 2 2 2 8 2 2 2 8 8 8 2 8 2 8 8 2 2 8 2 2 2 2 2 2 2 6 8 2 2 8 2 2
## [223] 2 2 2 2 6 2 2 6 2 2 6 2 2 2 6 2 2 2 2 8 2 2 2 8 2 2 8 2 2 2 2 2 8 6 4 6 8
## [260] 6 6 6 4 4 4 9 4 4 6 4 9 9 4 6 6 6 6 8 6 6 6 6 6 4 4 6 9 9 9 4 4 6 4 2 2 6
## [297] 2 2 8 8 8 2 2 2 2 8 8 2 8 2 8 8 8 8 8 2 8 8 8 8 8 2 1 1 1 1 1 1 1 1 1 1 1
## [334] 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 1 1 1 1 1 1
## [371] 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 1 1 1 1 1 1
## [408] 1 1 1 1 1 1 1 1 1 1 1 1 1 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [445] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 3 5 5 5 5 5 5 5 5 5 5 5 5 3 3 3 3 3 5 5 3 3
## [482] 3 3 3 3 3 3 3 3 3 3 3 7 3 7 3 3 3 3 3 3 7 7 7 5 3 7 7 7 3 3 5 5 3 3 3 3 3
## [519] 5 5 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
```

```
## [556] 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
##
## Within cluster sum of squares by cluster:
## [1] 110.66039 207.09211  69.24010  80.84766  87.13608 119.65169 160.47368
## [8] 135.37691  49.85361
##  (between_SS / total_SS =  77.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
plot(as.data.frame(oliveoilclust), col = km$cluster)
```



Point a: artificial dataset 2

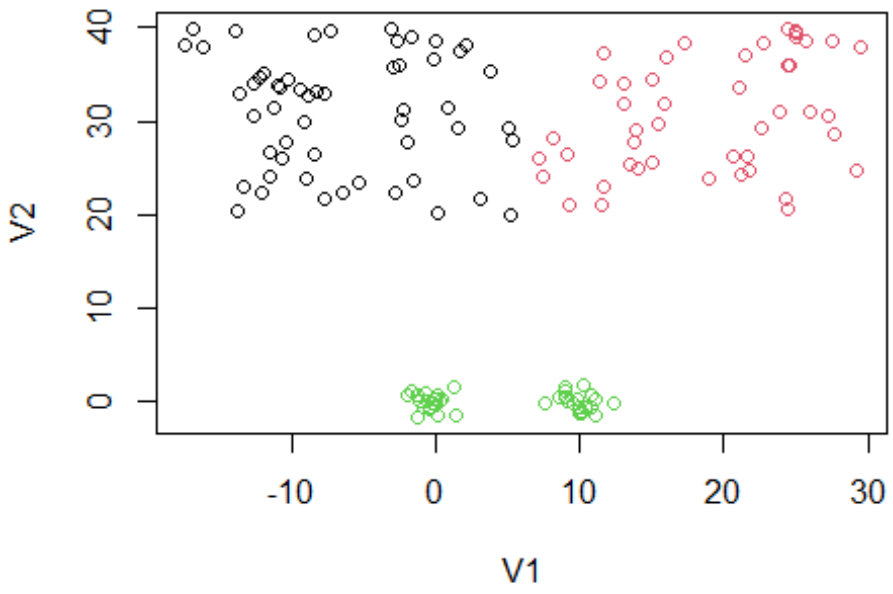
```
set.seed(1234)
gap_output <- gapnc(artdata2, K.max = 10, nstart=1, SE.factor = 2)
gap_output$nc
```

```
## [1] 3
```

```
km2 <- gap_output$kmopt
km2
```

```
## K-means clustering with 3 clusters of sizes 54, 46, 40
##
## Cluster means:
##          V1          V2
## 1 -6.397208 31.00742554
## 2 18.911236 30.57513973
## 3  4.825337 -0.01091477
##
## Clustering vector:
##   [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [38] 3 3 3 2 1 1 2 2 1 2 1 2 1 1 1 2 1 2 2 2 2 2 1 1 2 1 1 2 1 1 2 1 2 1 1 2
##  [75] 1 1 2 1 1 1 2 2 2 2 2 1 1 2 1 2 2 1 1 1 2 2 1 1 2 2 2 2 1 2 2 1 1 2 1 1 2
## [112] 1 1 2 2 1 1 2 1 1 1 1 2 1 1 1 2 1 2 1 2 2 1 1 2 1 1 1 2 1
##
## Within cluster sum of squares by cluster:
## [1] 4097.226 3557.411 1133.968
##  (between_SS / total_SS =  83.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
plot(artdata2, col = km2$cluster)
```



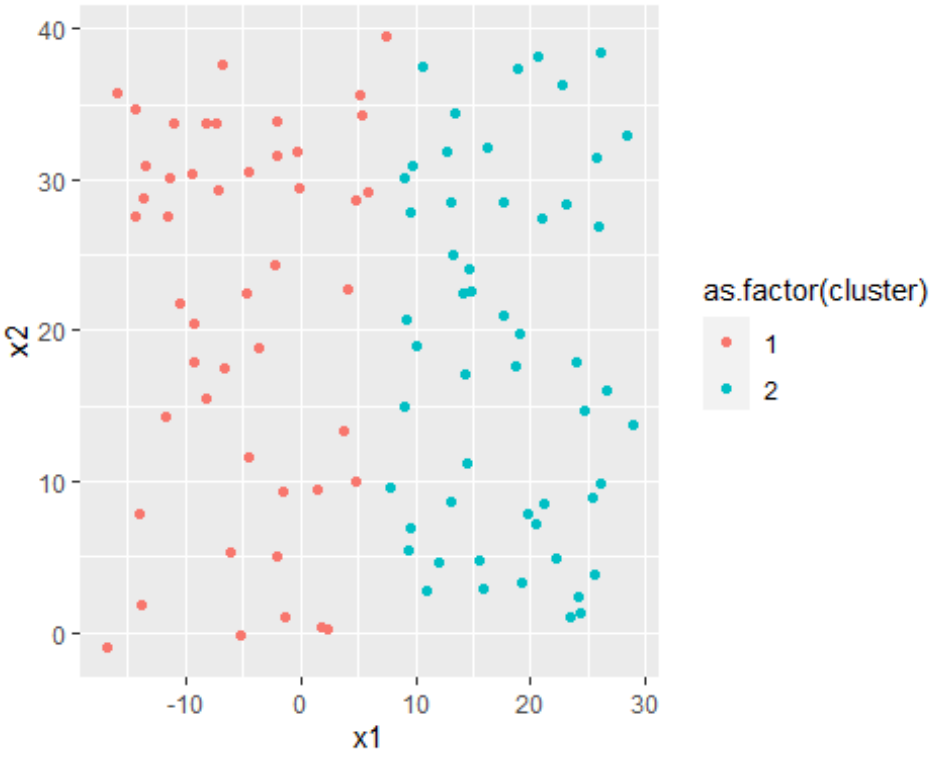
The function selects 3 as the best n° of clusters, treating the two groups below as parts of the same cluster and separating the data with higher values of V2.

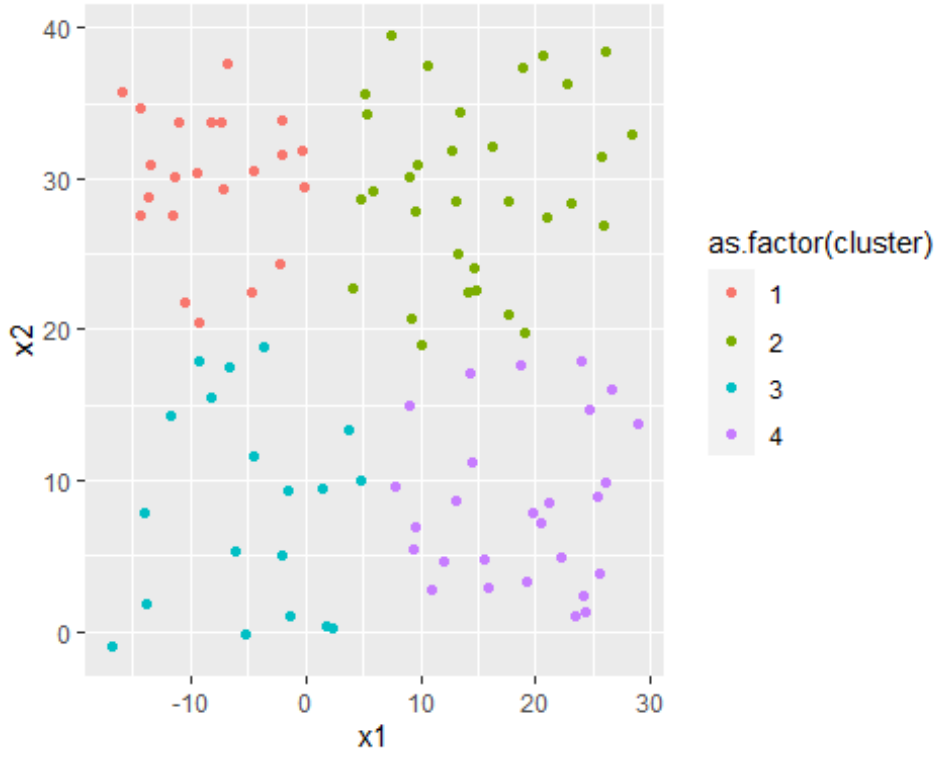
Point b

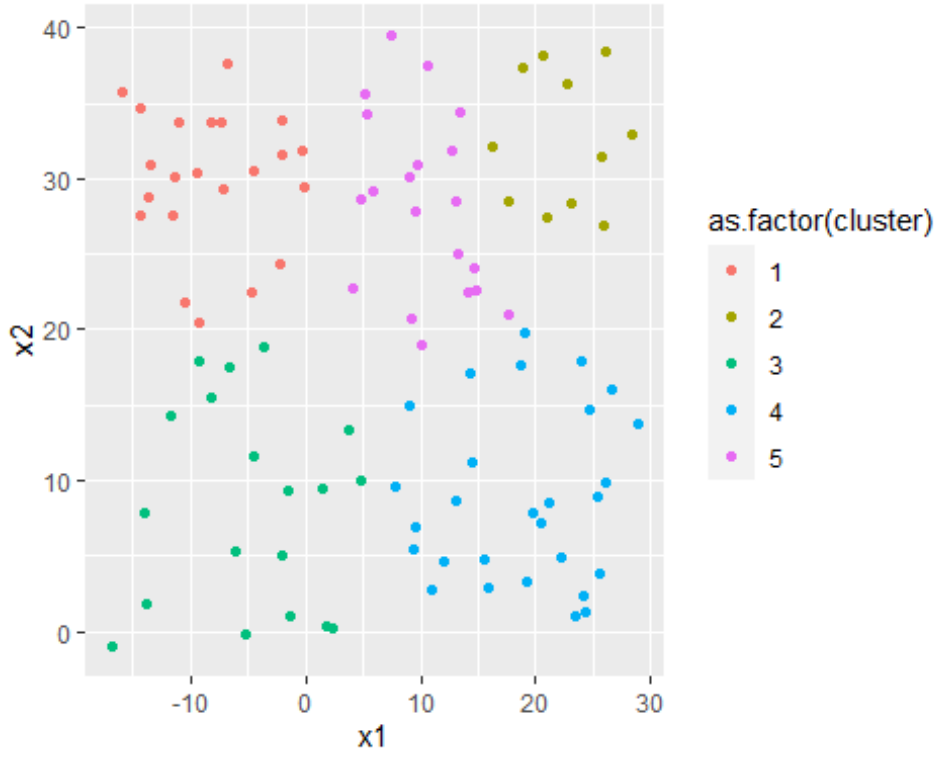
Generate the two dimensional dataset

```
set.seed(5555)
x1 <- runif(100, min(artdata2$V1), max(artdata2$V1))
x2 <- runif(100, min(artdata2$V2), max(artdata2$V2))
dataset_pointb <- cbind(x1, x2)
```

K-means up to K=10



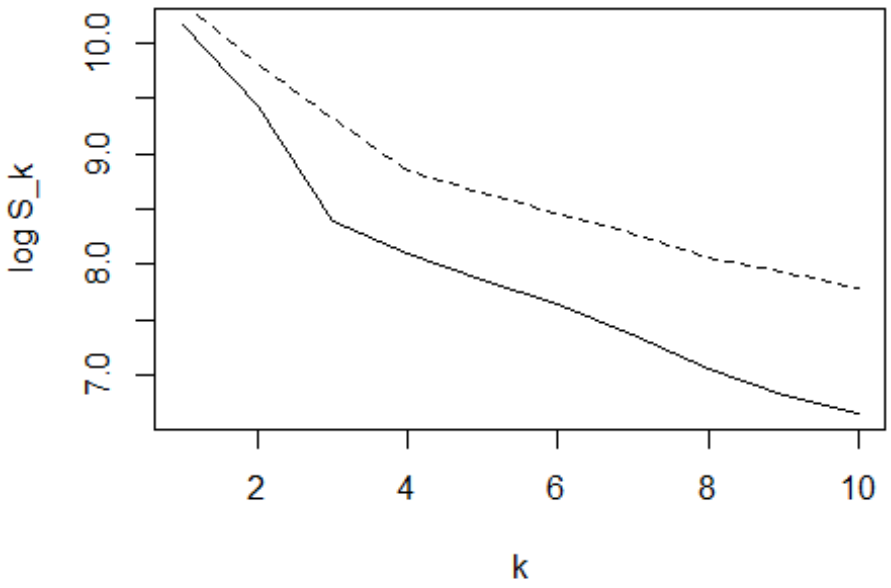




Plot the log(Sk)

```
cg1 <- clusGap(artdata2,kmeans,K.max=10,B=100,d.power=2,spaceH0="scaledPCA",nstart
=100)
Sk <- c()
```

```r
for (i in 1:10) {
  Sk[i] <- listkmeans[[i]]$tot.withinss
}

logsk <- log(Sk)

plot(1:10,cg1$Tab[,1],xlab="k",ylab="log S_k",type="l")
points(1:10,logsk,xlab="k",ylab="log S_k",type="l",lty=2)
legend(6,6.5,c("log S_k in artifcial_data 2","E(log S_k) unif.distributed dataset"
),lty=1:2)
```



Exercise 2

100 datasets (specifications of artificial dataset 1)

```r
library(sn)
```

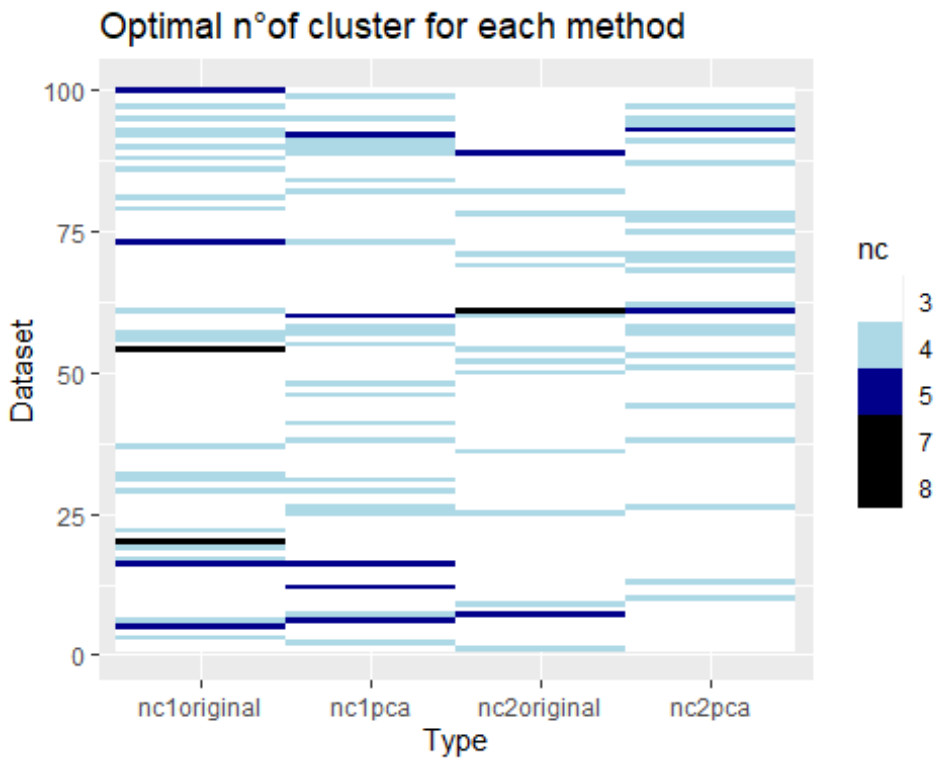
```
## Warning: il pacchetto 'sn' è stato creato con R versione 4.2.3
```

```
## Caricamento del pacchetto richiesto: stats4
```

```
##
## Caricamento pacchetto: 'sn'
```

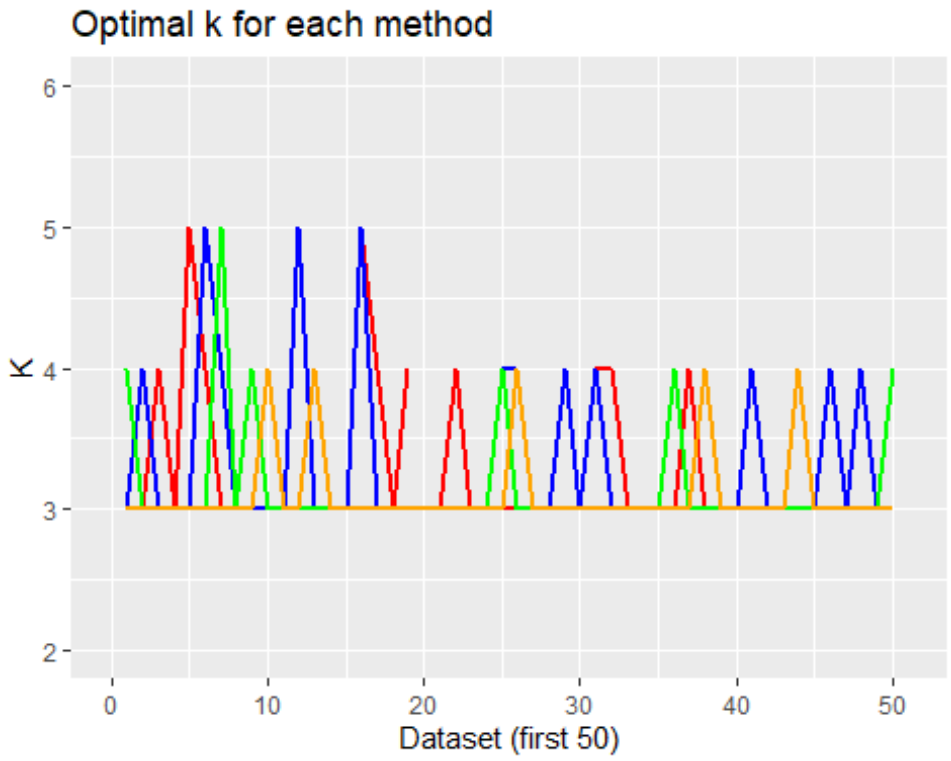
```
## Il seguente oggetto è mascherato da 'package:lubridate':
##
##     dst
```

```
## Il seguente oggetto è mascherato da 'package:stats':
##
##     sd

listdatasets <- list()
gap1original <- list()
gap2original <- list()
gap1pca <- list()
gap2pca <- list()
nc1original <- c()
nc2original <- c()
nc1pca <- c()
nc2pca <- c()

for (i in 1:100)
{
  v1 <- c(rnorm(50,0,1), rsn(70,5,1,8), rnorm(30,6,1))
  v2 <- c(rnorm(50,0,1), rsn(70,0,1,8), 8+rt(30,5))
  listdatasets[[i]] <- cbind(v1,v2)
  gap1original[[i]] <- gapnc(listdatasets[[i]], K.max = 10, nstart=1, SE.factor =
1, spaceH0 = "original")
  gap2original[[i]] <- gapnc(listdatasets[[i]], K.max = 10, nstart=1, SE.factor =
2, spaceH0 = "original")
  gap1pca[[i]] <- gapnc(listdatasets[[i]], K.max = 10, nstart=1, SE.factor = 1, sp
aceH0 = "scaledPCA")
  gap2pca[[i]] <- gapnc(listdatasets[[i]], K.max = 10, nstart=1, SE.factor = 2, sp
aceH0 = "scaledPCA")
  nc1original[i] <- gap1original[[i]]$nc
  nc2original[i] <- gap2original[[i]]$nc
  nc1pca[i] <- gap1pca[[i]]$nc
  nc2pca[i] <- gap2pca[[i]]$nc
}

ncluster <- cbind(nc1original, nc1pca, nc2original, nc2pca)
df <- c(1:100)
ncluster <- as.data.frame(ncluster)
ncluster$df <- df
ncluster <- pivot_longer(as.data.frame(ncluster), cols = c("nc1original", "nc1pca"
, "nc2original", "nc2pca"), names_to = "type", values_to = "nc") %>% mutate(nc = a
s.factor(nc))

values <- c(3,4,5,6,7)
custom_colors <- c("white", " light blue", "dark blue", "black", "black")

ggplot(ncluster, aes(x = type, y = df, fill = nc)) +
  geom_tile() +
  scale_fill_manual(values = custom_colors) +
  labs(title = "Optimal n°of cluster for each method", x = "Type", y = "Dataset")
```

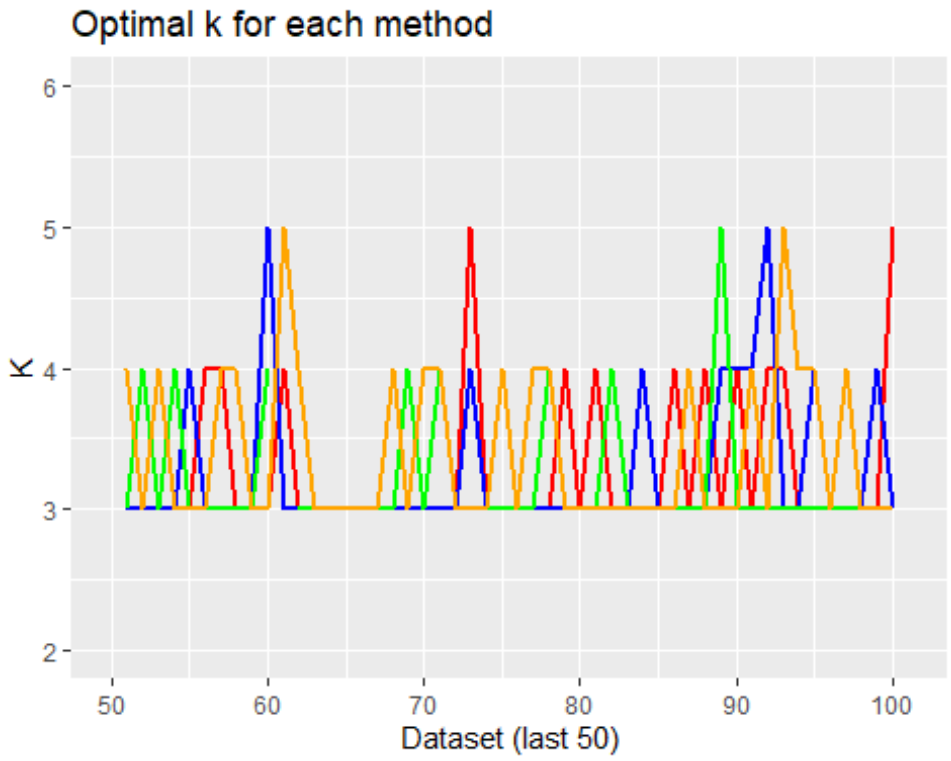



```
ncluster <- cbind(nc1original, nc1pca, nc2original, nc2pca)
df <- c(1:100)
ncluster <- as.data.frame(ncluster)
ncluster$df <- df

ggplot(ncluster[1:50,], aes(x = df)) +
  geom_line(aes(y = nc1original), color = "red", linetype = "solid", size = 1) +
  geom_line(aes(y = nc1pca), color = "blue", linetype = "solid", size = 1) +
  geom_line(aes(y = nc2original), color = "green", linetype = "solid", size = 1) +
  geom_line(aes(y = nc2pca), color = "orange", linetype = "solid", size = 1) +
  labs(title = "Optimal k for each method", x = "Dataset (first 50)", y = "K")+
  xlim(0,51)+
  ylim(2,6)
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



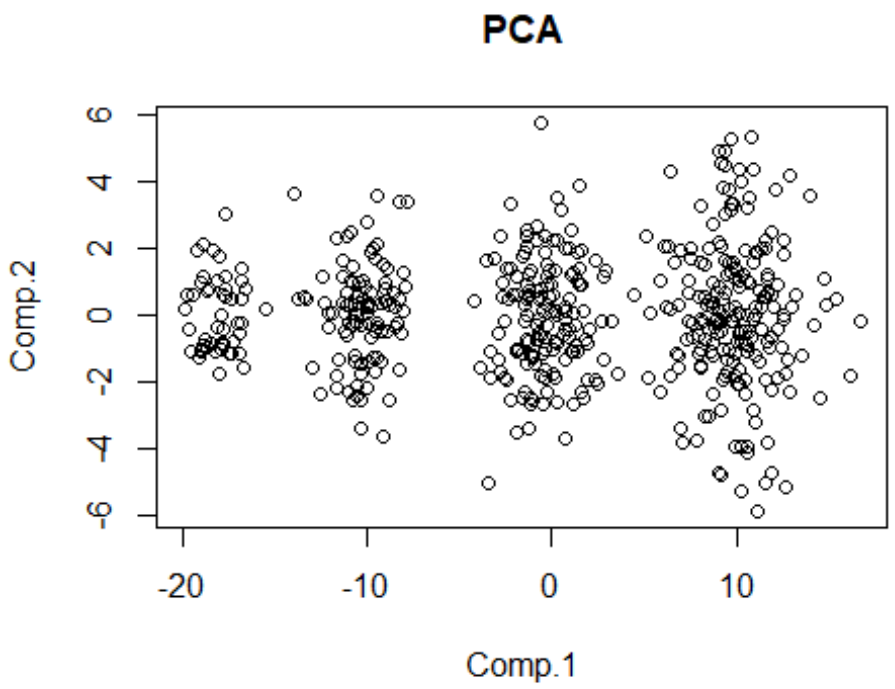

```
ggplot(ncluster[51:100,], aes(x = df)) +
  geom_line(aes(y = nc1original), color = "red", linetype = "solid", size = 1) +
  geom_line(aes(y = nc1pca), color = "blue", linetype = "solid", size = 1) +
  geom_line(aes(y = nc2original), color = "green", linetype = "solid", size = 1) +
  geom_line(aes(y = nc2pca), color = "orange", linetype = "solid", size = 1) +
  labs(title = "Optimal k for each method", x = "Dataset (last 50)", y = "K")+
  xlim(50,101)+
  ylim(2,6)
```



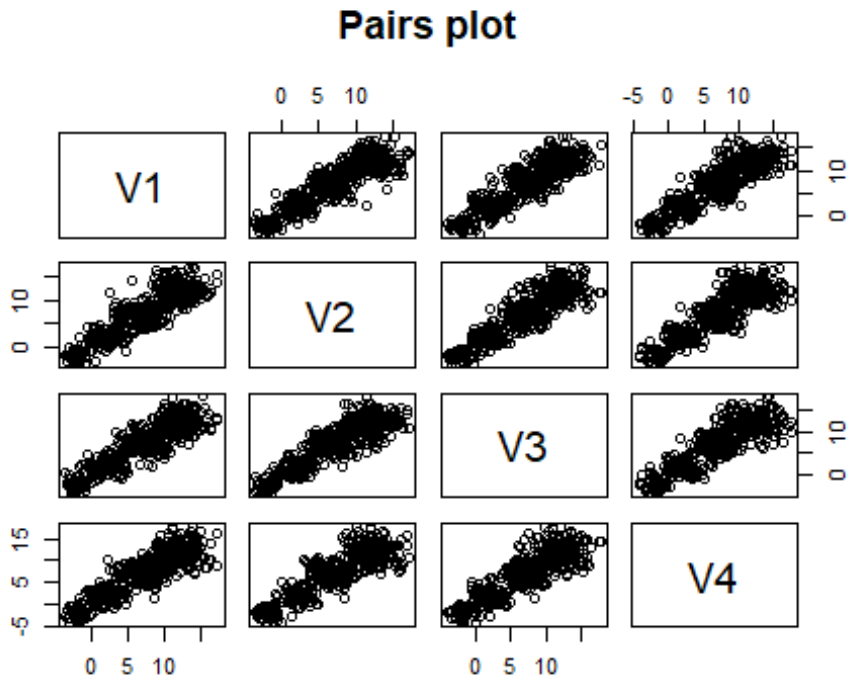
In this situation the of the ideal K for each of the 100 datasets varies from 3 to 5. We can see from the heatmap that the factor 2 for the SE is likely to produce a higher n°of clusters (4 or 5)

Exercise 2 part 2 (four-dimensional data)

```
set.seed (1000)
c1 <- rmvnorm (n=50, mean = rep(-2,4), sigma = diag(4))
c2 <- rmvnorm (n=100, mean = rep(2, 4), sigma = 2* diag(4))
c3 <- rmvnorm (n=150, mean = rep(7, 4), sigma = 3* diag(4))
c4 <- rmvnorm (n = 200, mean = rep (12,4), sigma = 4*diag(4))
data <- rbind(c1, c2, c3, c4)
data <- as.data.frame(data)
pcdata <- princomp(data) # PCA
plot(pcdata$scores, main = "PCA")
```

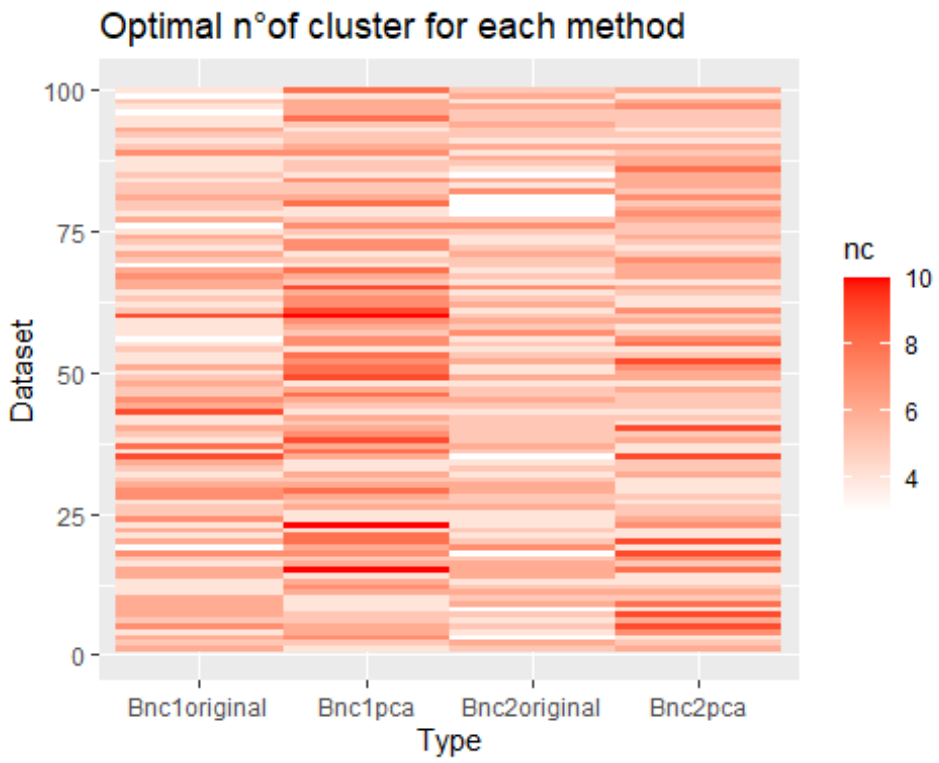



```
plot(data, main = "Pairs plot")
```

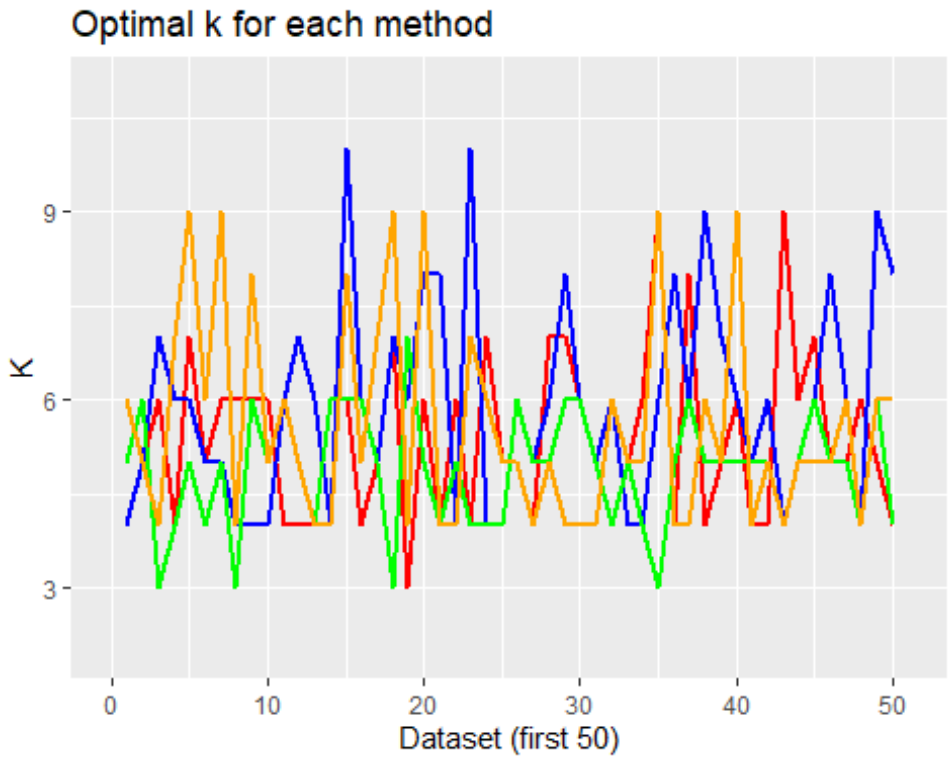



```
## Generate 100 datasets like the one above

lista <- list()
```

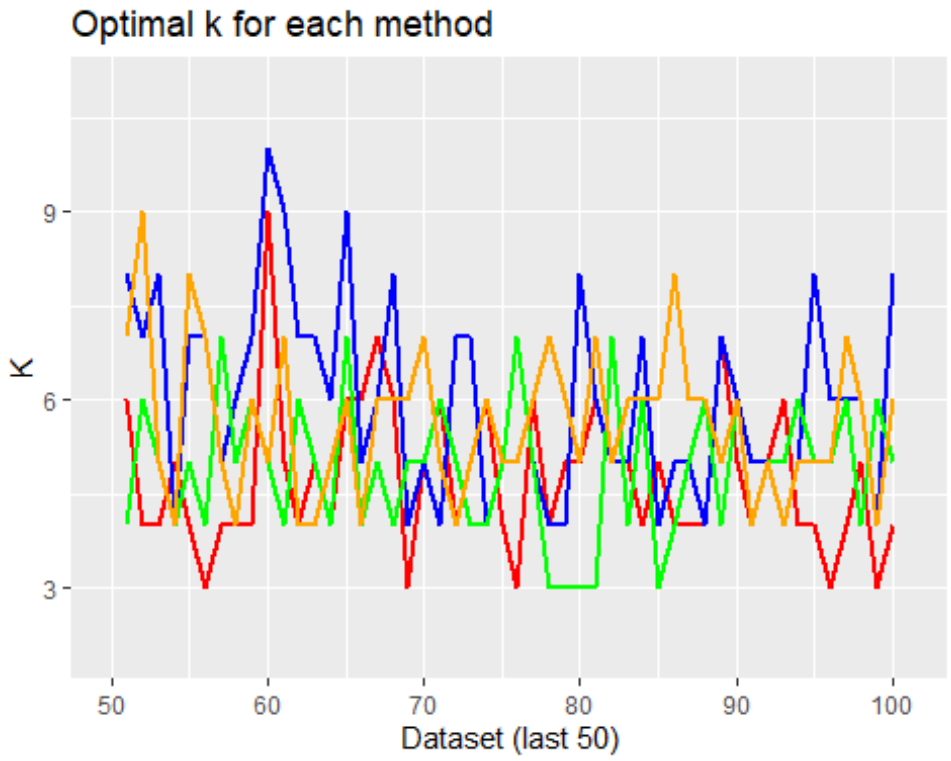
```r
for (i in 1:100){
c1 <- rmvnorm (n=50, mean = rep(-2,4), sigma = diag(4))
c2 <- rmvnorm (n=100, mean = rep(2, 4), sigma = 2* diag(4))
c3 <- rmvnorm (n=150, mean = rep(7, 4), sigma = 3* diag(4))
c4 <- rmvnorm (n = 200, mean = rep (12,4), sigma = 4*diag(4))
data <- as.data.frame(rbind(c1, c2, c3, c4))
lista[[i]] <- data
}

Bgap1original <- list()
Bgap2original <- list()
Bgap1pca <- list()
Bgap2pca <- list()
Bnc1original <- c()
Bnc2original <- c()
Bnc1pca <- c()
Bnc2pca <- c()

for (i in 1:100) {
  Bgap1original[[i]] <- gapnc(lista[[i]], K.max = 10, nstart=1, SE.factor = 1, spa
ceH0 = "original")
  Bgap2original[[i]] <- gapnc(lista[[i]], K.max = 10, nstart=1, SE.factor = 2, spa
ceH0 = "original")
  Bgap1pca[[i]] <- gapnc(lista[[i]], K.max = 10, nstart=1, SE.factor = 1, spaceH0
= "scaledPCA")
  Bgap2pca[[i]] <- gapnc(lista[[i]], K.max = 10, nstart=1, SE.factor = 2, spaceH0
= "scaledPCA")
  Bnc1original[i] <- Bgap1original[[i]]$nc
  Bnc2original[i] <- Bgap2original[[i]]$nc
  Bnc1pca[i] <- Bgap1pca[[i]]$nc
  Bnc2pca[i] <- Bgap2pca[[i]]$nc
}



nclusterB <- cbind(Bnc1original, Bnc1pca, Bnc2original, Bnc2pca)
df <- c(1:100)
nclusterB <- as.data.frame(nclusterB)
nclusterB$df <- df
nclusterB <- pivot_longer(as.data.frame(nclusterB), cols = c("Bnc1original", "Bnc1
pca", "Bnc2original", "Bnc2pca"), names_to = "type", values_to = "nc") %>% mutate(
nc = as.numeric(nc))



ggplot(nclusterB, aes(x = type, y = df, fill = nc)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "red") +
  labs(title = "Optimal n°of cluster for each method", x = "Type", y = "Dataset")
```




```
nclusterB <- cbind(Bnc1original, Bnc1pca, Bnc2original, Bnc2pca)
df <- c(1:100)
nclusterB <- as.data.frame(nclusterB)
nclusterB$df <- df

ggplot(nclusterB[1:50,], aes(x = df)) +
  geom_line(aes(y = Bnc1original), color = "red", linetype = "solid", size = 1) +
  geom_line(aes(y = Bnc1pca), color = "blue", linetype = "solid", size = 1) +
  geom_line(aes(y = Bnc2original), color = "green", linetype = "solid", size = 1)
+
  geom_line(aes(y = Bnc2pca), color = "orange", linetype = "solid", size = 1) +
  labs(title = "Optimal k for each method", x = "Dataset (first 50)", y = "K")+
  xlim(0,51)+
  ylim(2,11)
```




```
ggplot(nclusterB[51:100,], aes(x = df)) +
  geom_line(aes(y = Bnc1original), color = "red", linetype = "solid", size = 1) +
  geom_line(aes(y = Bnc1pca), color = "blue", linetype = "solid", size = 1) +
  geom_line(aes(y = Bnc2original), color = "green", linetype = "solid", size = 1)
+
  geom_line(aes(y = Bnc2pca), color = "orange", linetype = "solid", size = 1) +
  labs(title = "Optimal k for each method", x = "Dataset (last 50)", y = "K")+
  xlim(50,101)+
  ylim(2,11)
```



For these data the results for the four types of Gap functions are very different between each other. One possible explanation is that the separation between the four clusters is small, so the algorithm is likely to select an optimal K different from 4. In particular, all the four combinations of parameters tend to select a greater number of cluster (up to 10 in some situations). From the heatmap we can see that the use of spaceH0 = "scalesPCA" in the function returns, in general, an higher n°of clusters which, in this case, is not desirable since the "true" K is 4.

Exercise 3

$Gap(K*)$ and S_{K*} don't change their value for q=1 and q=2. Therefore, we can compare the second member of the disequality for the two different choices of q.

$[Gap(K*) - S_{K*}] - [Gap(K*) - 2S_{K*}] = Gap(K*) - S_{K*} - Gap(K*) + 2S_{K*} = S_{K*}$.
$S(K*) > 0$ by definition, so we can say that the right member of the disequality is greater when q=1. Now let's proceed in this way:
- First, compute $K_{0,1}$
- Then, in computing $K_{0,2}$ two situations are possible:

Situation 1): $\exists K < K_{0,1}$ such that $Gap(K*) - 2S_{K*} < Gap(K) \leq Gap(K*) - S_{K*}$. Then we will choose a $K_{0,2} < K_{0,1}$.

Situation 2): $\nexists K < K_{0,1}$ such that $Gap(K) > Gap(K*) - 2S_{K*}$. Then the first K that satisfies $Gap(K) > Gap(K*) - 2S_{K*}$ will be $K_{0,1}$, since $Gap(K_{0,1}) > Gap(K*) - S_{K*} > Gap(K*) - 2S_{K*}$. $K_{0,1}$ is the first K that satisfies the disequality for q=2, so in this case $K_{0,1} = K_{0,2}$.

Exericise 4

Description

A suitable dissimilarity measure could be related to the correlation of two units: in this manner, we will consider as similar units that show similiar behaviour (i.e. relatively small values or relatively big values for the same variables) without giving importance to the size of the values. In the litarature, a measure that satisfy these requirements is based on the Pearson coefficient. Formula:
$\delta_{ij} = (1 - \phi_{ij})/2$. In this way, units with high positive correlation will be similar while highly negative correlated units will have a dissimilarity near or equal to 1, which is the maximum value. Now, let's show that δ_{ij} fulfills the properties of non-negativity, identity and simmetry:

1) $\frac{1-\phi_{ij}}{2} \geq 0$ ->
   $1 - \phi_{ij} \geq 0$ ->
   $-\phi_{ij} \geq -1$ ->
   $\phi_{ij} \leq 1$.

The Pearson correlation coefficient varies from -1 to 1 by definition, so the non-negativity property is fulfilled for any couple of units x_i, x_j of any dataset.

2) Simmetry: $\phi_{ij} = \phi_{ij}$, that implies $\delta_{ij} = \delta_{ji}$

3) Identity: $\phi_{ii} = 1$, so we can conclude that $\delta_{ii} = 0$

Compute d*(x1, x2) and d*(x1, x3)

```
x1 <- c(1,4,5,4,2,1,1,4)
x2 <- c(2,3,2,2,3,3,3,3)
x3 <- c(7,11,11,12,9,8,8,12)

distcorr <- function(a, b){
d <- (1-cor(a,b))/2
return(d)
}

d12 <- distcorr(x1, x2)
d13 <- distcorr(x1, x3)

d12

## [1] 0.6447072

d13

## [1] 0.03577897

d12>d13

## [1] TRUE
```

The chosen measure satisfies our requirements for the given data.