EXERCISE 2

Riccardo Petrella

2023-10-07

1.a Use the gap statistic method to estimate the number of clusters for the olive oil data, and for Artificial Data Set 2 (you can use Kmax = 10 but you can go higher if you want). Comment on the solutions.

In order to run K-means properly, it is better to keep only the continuos variables in the dataset. Moreover, since the axes have very different ranges, it is useful to standardize the variables as well.

```r
setwd("C:/Users/ricky/Desktop/secondo_anno/modern stat")
library(cluster)
set.seed(123456)

oliveoil <- read.table("oliveoil.dat",header=TRUE, stringsAsFactors = TRUE)

str(oliveoil)
```

```
## 'data.frame':    572 obs. of  10 variables:
##  $ macro.area : Factor w/ 3 levels "Centre.North",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ region     : Factor w/ 9 levels "Apulia.north",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ palmitic   : int  1075 1088 911 966 1051 911 922 1100 1082 1037 ...
##  $ palmitoleic: int  75 73 54 57 67 49 66 61 60 55 ...
##  $ stearic    : int  226 224 246 240 259 268 264 235 239 213 ...
##  $ oleic      : int  7823 7709 8113 7952 7771 7924 7990 7728 7745 7944 ...
##  $ linoleic   : int  672 781 549 619 672 678 618 734 709 633 ...
##  $ linolenic  : int  36 31 31 50 50 51 49 39 46 26 ...
##  $ arachidic  : int  60 61 63 78 80 70 56 64 83 52 ...
##  $ eicosenoic : int  29 29 29 35 46 44 29 35 33 30 ...
```

```r
# The chemical variables:
olive <- oliveoil[,3:10]
solive <- scale(olive)
solive <- as.data.frame(solive)
```

We now use "clusGap" to compute the gap statistics with the following parameters:

- K.max indicates the maximum number of clusters;
- B=100 indicates 100 simulations from uniform distribution; d.power=2 indicates squared Euclidean distances (as in the original paper and in class),
- spaceH0="scaledPCA" specifies the way the uniform distribution is simulated.

Then we print the output of "cg1" which yields:

- the optimal number of clusters
- $log(s_k)$, $E[log(s_k)]$, gap and the simulated Standard Error

```r
cg1 <- clusGap(solive,kmeans,K.max=10,B=100,d.power=2,spaceH0="scaledPCA",
               nstart=100, iter.max = 100)
print(cg1,method="globalSEmax",SE.factor=2)
```

```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = solive, FUNcluster = kmeans, K.max = 10, B = 100, d.power = 2, spaceH0 = "scaledPCA", ns
## B=100 simulated reference sets, k = 1..10; spaceH0="scaledPCA"
##  --> Number of clusters (method 'globalSEmax', SE.factor=2): 9
##           logW   E.logW       gap     SE.sim
##  [1,] 7.733684 8.693623 0.9599395 0.01843618
##  [2,] 7.308810 8.329372 1.0205617 0.01562368
##  [3,] 7.056186 8.178244 1.1220579 0.01413717
##  [4,] 6.804664 8.052789 1.2481254 0.01356304
##  [5,] 6.585133 7.979268 1.3941343 0.01221699
##  [6,] 6.482607 7.913866 1.4312590 0.01236158
##  [7,] 6.394681 7.857392 1.4627111 0.01252318
##  [8,] 6.303561 7.806990 1.5034283 0.01306546
##  [9,] 6.234736 7.762993 1.5282564 0.01344674
## [10,] 6.178046 7.723784 1.5457380 0.01369074
```

```r
nc <- maxSE(cg1$Tab[,3],cg1$Tab[,4], method = "globalSEmax", SE.factor=2)
print(paste("number of optimal clusters", nc))
```

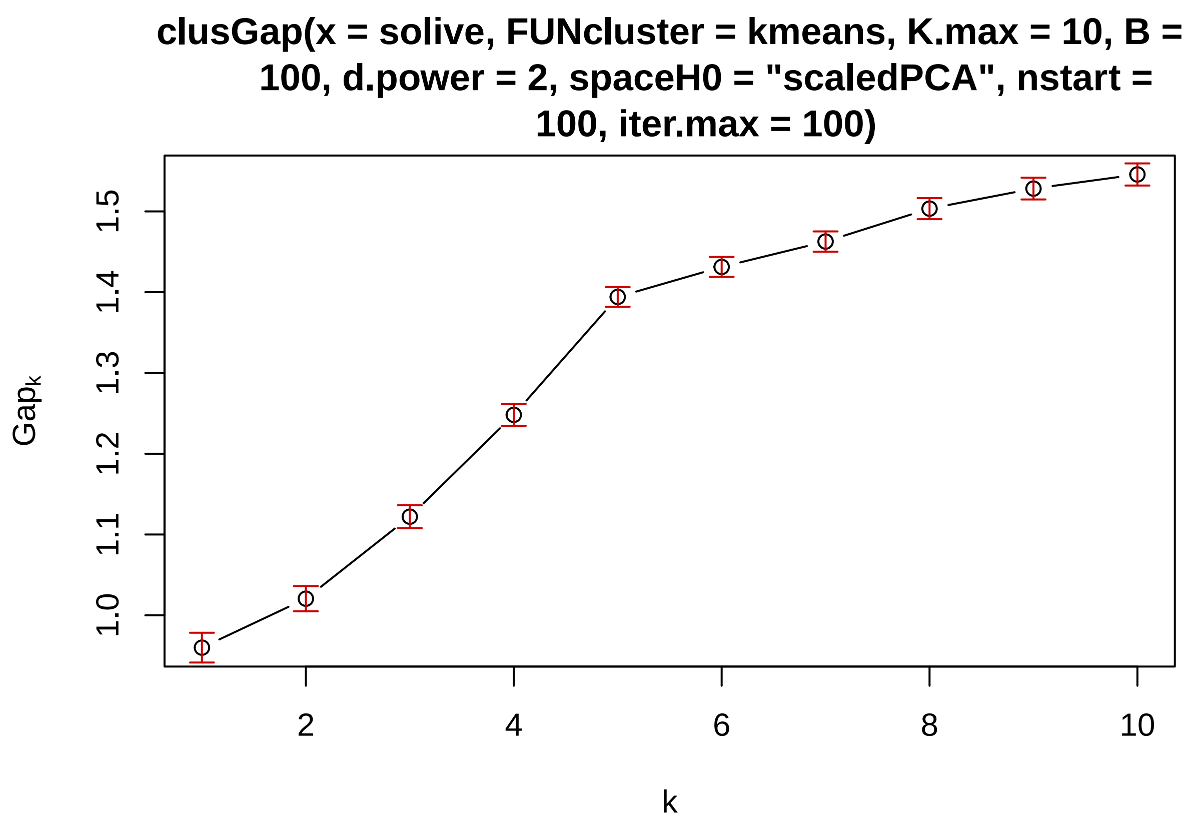
```
## [1] "number of optimal clusters 9"
```

We want to visualize the output of "cg1":

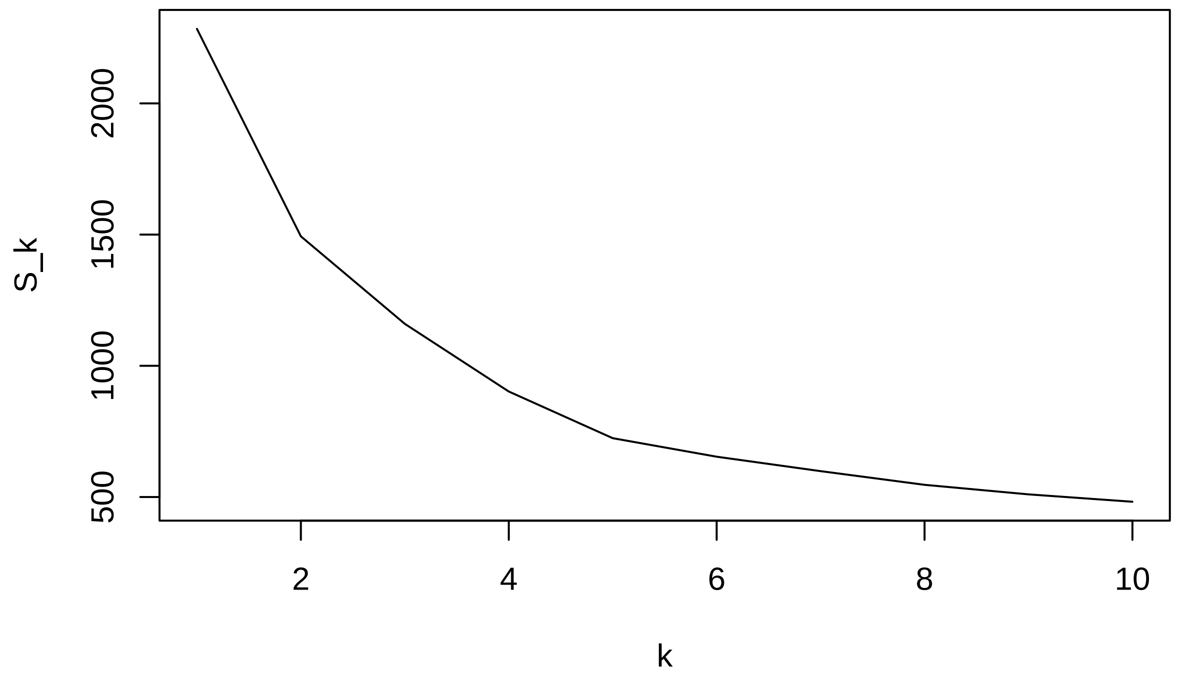
- The first plot shows the gap statistics over K. As we can see, it keeps growing even at $K = 10$.
- The second plot shows S_k over K. The elbow is not visible.
- The third plot represents the difference between $log(s_k)$ and $E[log(s_k)]$. As we can see, with our dataset, we are doing much better than with the uniform distribution.

Finally we apply K-Means with the optimal $K_0 = 9$ to the dataset and we plot the scatterplots with the new clusters. A brief visual comparision with the true clusters (i.e. the regions of oliveoil) reveals that the clusters are disposed in a different way in every scatterplot, and this phenomenon is particularly noticeable in the elliptic clusters.

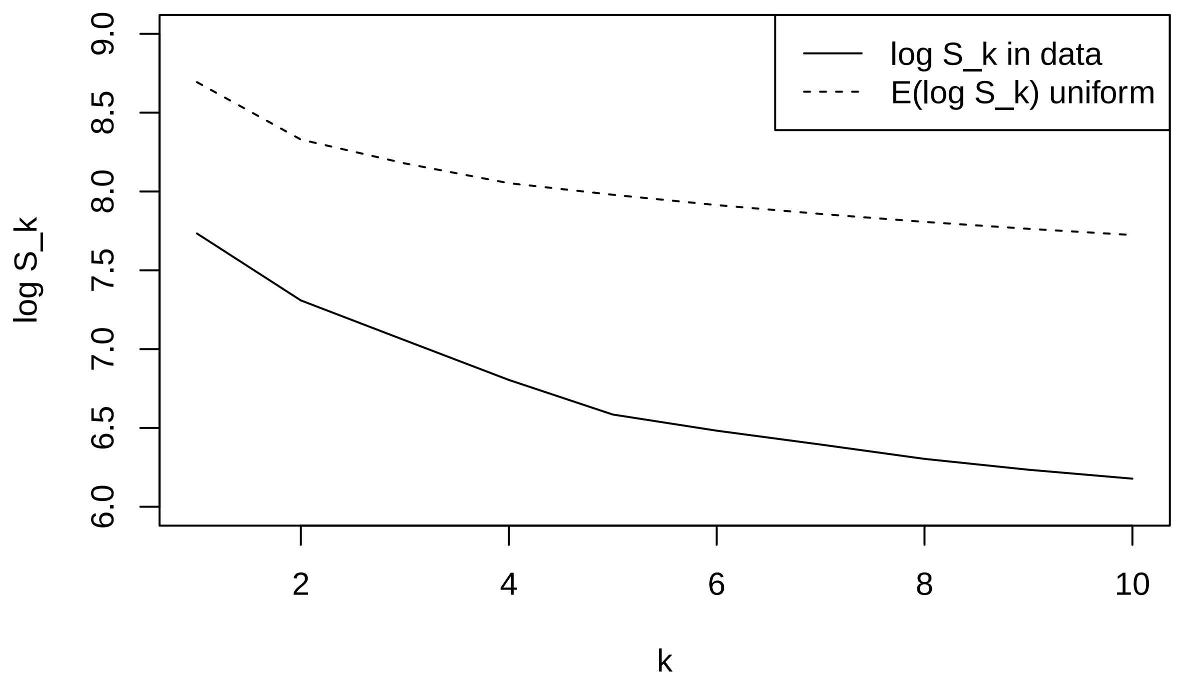
```r
# Values of gap
plot(cg1)
```



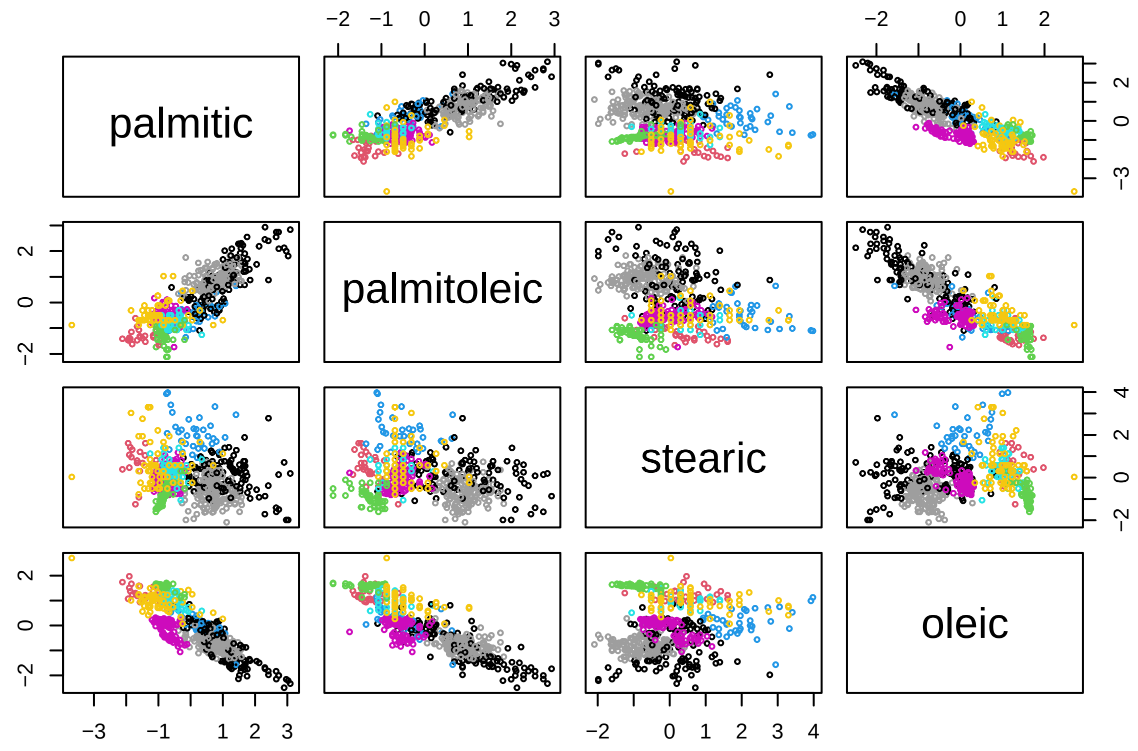
```
# Values of S_k; cg1$Tab[,1] has values of log(S_k), therefore the exp.
plot(1:10,exp(cg1$Tab[,1]),xlab="k",ylab="S_k",type="l")
```



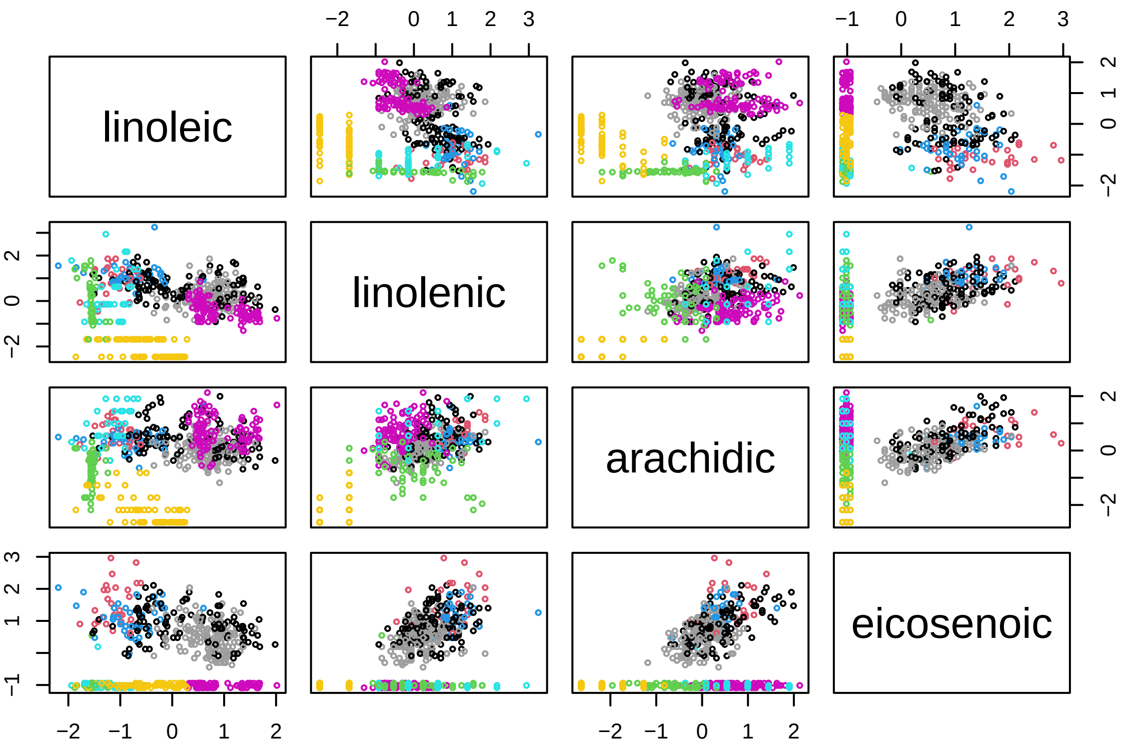
```
# Values of log(S_k) and its expectation under uniform distribution
# The latter are in cg1$Tab[,2].
plot(1:10,cg1$Tab[,1],xlab="k",ylab="log S_k",type="l", ylim = c(6, 9))

points(1:10,cg1$Tab[,2],xlab="k",ylab="log S_k",type="l",lty=2)

legend("topright",c("log S_k in data","E(log S_k) uniform"),lty=1:2)
```



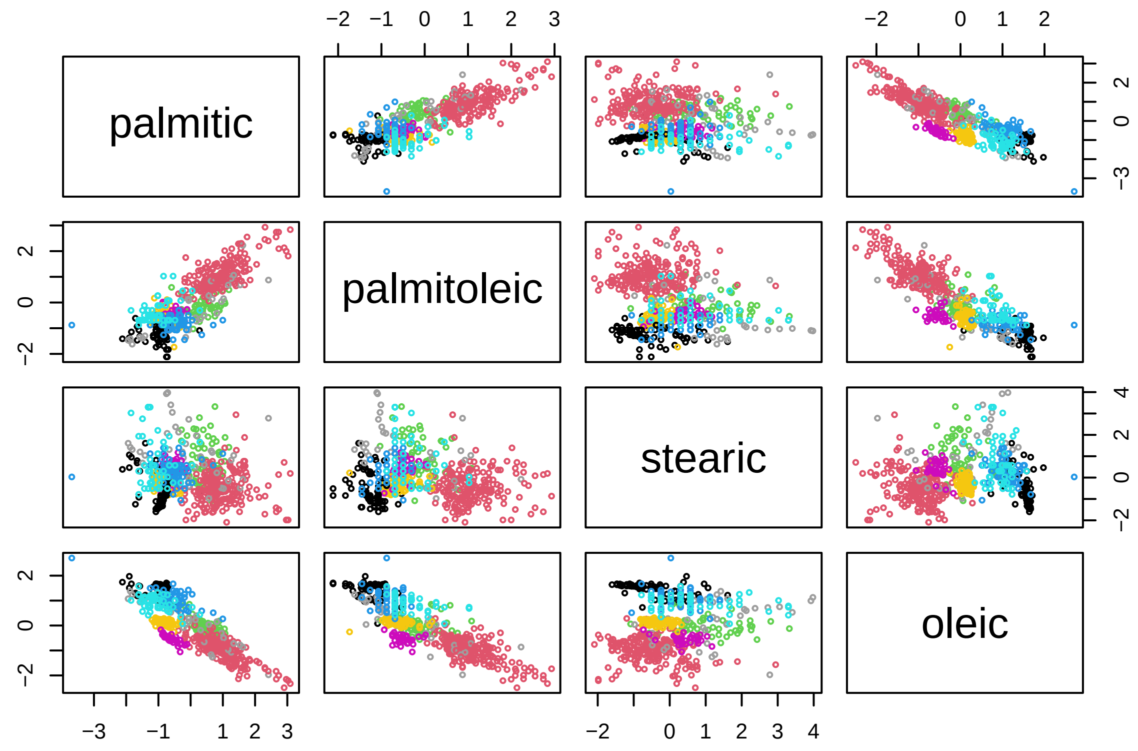
```
# new clusters with k-means
km_solive <- kmeans(solive, 9, nstart=100)
pairs(solive[,1:4], col=km_solive$cluster, cex=0.5)
```



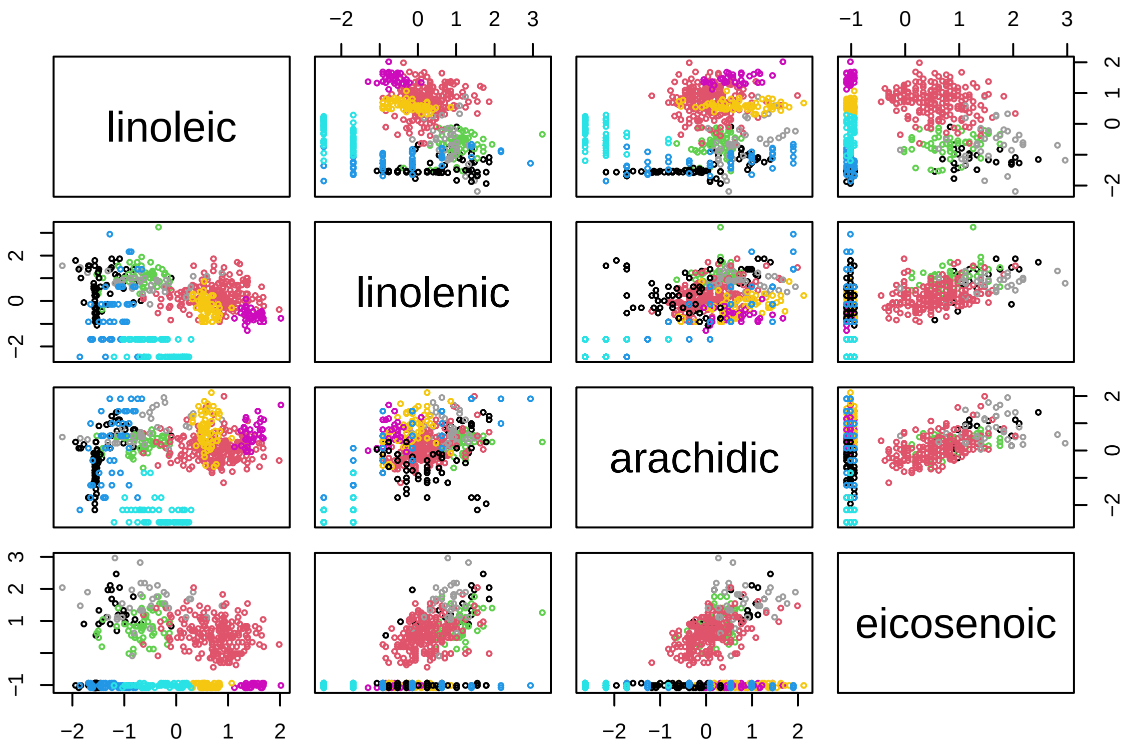
```r
pairs(solive[,5:8], col=km_solive$cluster, cex=0.5)
```



```
# original true clusters
pairs(solive[,1:4], col=oliveoil[,2], cex = 0.5)
```



```r
pairs(solive[,5:8], col=oliveoil[,2], cex = 0.5)
```



We repeat the same process for "Artificial Data set 2". We notice that with the gap statistics the optimal number of clusters is still 9.

```r
art <- read.table("clusterdata2.dat",header=TRUE,stringsAsFactors = TRUE)
str(art)
```

```
## 'data.frame':    139 obs. of  2 variables:
##  $ X.1.69447949793728: num  0.494 -0.753 -0.16 -0.375 1.276 ...
##  $ X1.15561482046003 : num  0.398 0.86 0.213 -0.853 1.493 ...
```

```r
cg2 <- clusGap(art,kmeans,K.max=10,B=100,d.power=2,spaceH0="scaledPCA",
         nstart=100, iter.max = 100)
print(cg2,method="globalSEmax",SE.factor=2)
```

```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = art, FUNcluster = kmeans, K.max = 10, B = 100, d.power = 2, spaceH0 = "scaledPCA", nstar
## B=100 simulated reference sets, k = 1..10; spaceH0="scaledPCA"
##  --> Number of clusters (method 'globalSEmax', SE.factor=2): 9
##            logW    E.logW         gap     SE.sim
##  [1,] 10.153516 10.045110 -0.10840581 0.05867565
##  [2,]  9.419738  9.495736  0.07599844 0.06301865
##  [3,]  8.382931  9.027491  0.64456004 0.05969282
##  [4,]  8.085169  8.619151  0.53398255 0.05261272
##  [5,]  7.853534  8.388555  0.53502038 0.05138633
##  [6,]  7.631342  8.184578  0.55323547 0.04899025
##  [7,]  7.369366  8.002002  0.63263660 0.05280375
##  [8,]  7.055152  7.836583  0.78143035 0.04872541
##  [9,]  6.817640  7.690823  0.87318306 0.04864986
## [10,]  6.651424  7.558758  0.90733457 0.05074262
```

```
nc2 <- maxSE(cg2$Tab[,3],cg2$Tab[,4], method = "globalSEmax", SE.factor=2)
print(paste("number of optimal clusters", nc2))
```

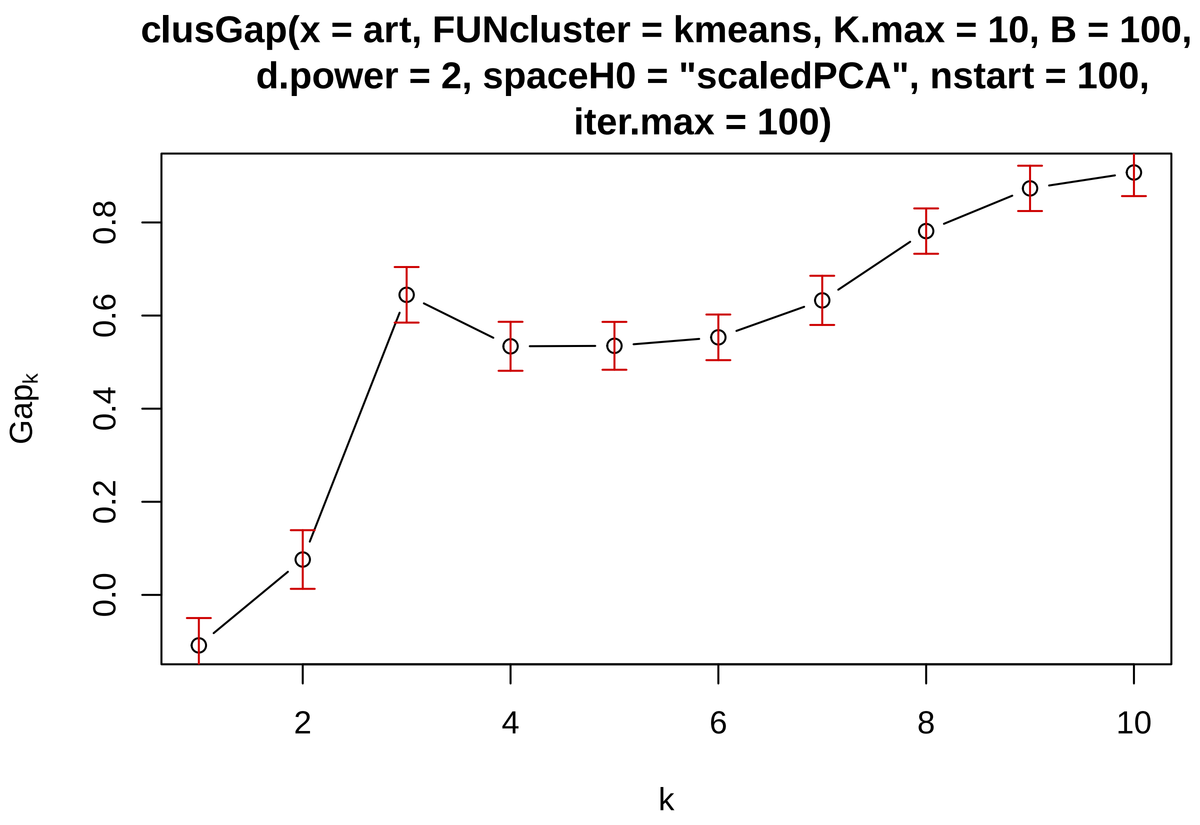
```
## [1] "number of optimal clusters 9"
```

We plot the output of "cg2" as well. We can say that $K_0 = 9$ because $\text{Gap}(K = 9) > \text{Gap}(K = 10) - 2s_K$ and $\text{Gap}(K = 8) < \text{Gap}(K = 9) - 2s_K$.

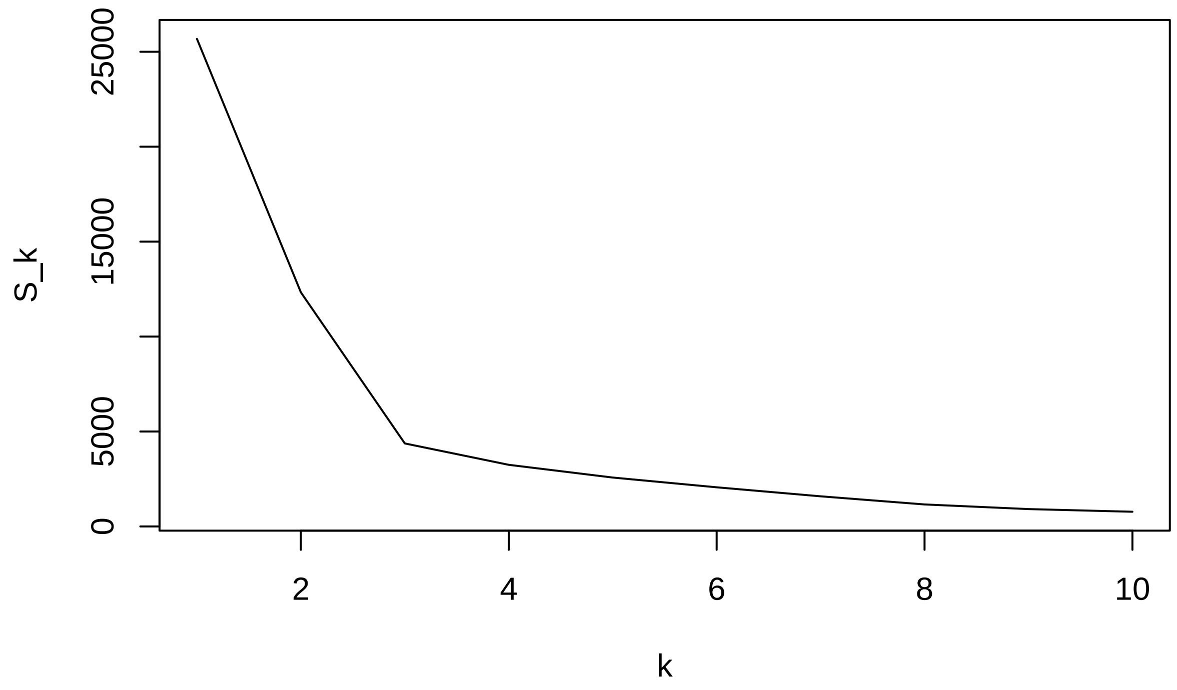
Remark: if we had used the elbow plot as a method to select K_0, we would have chosen $K_0 = 3$.

When we recompute the K-Means with the optimal K, we notice that the new clusters do not match with the underlying distributions which have been used to generate the Artificial Data Set 2. That is because the underlying distributions were 5, while $K_0 = 9$.

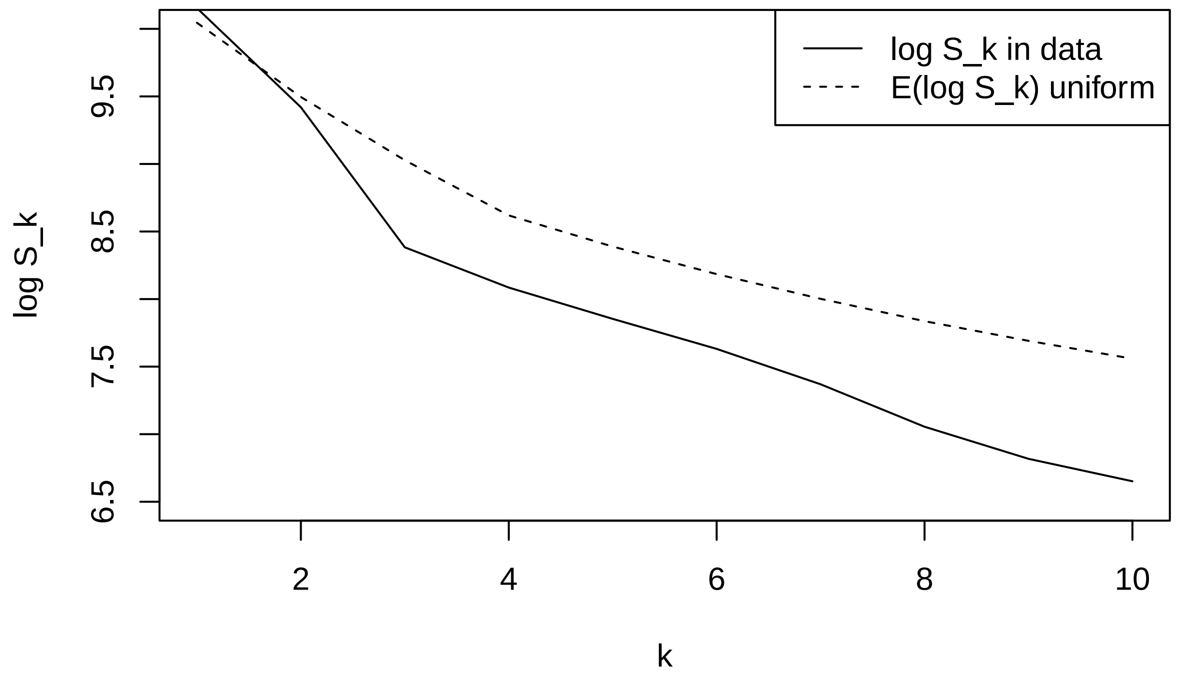
```
# Values of gap
plot(cg2)
```



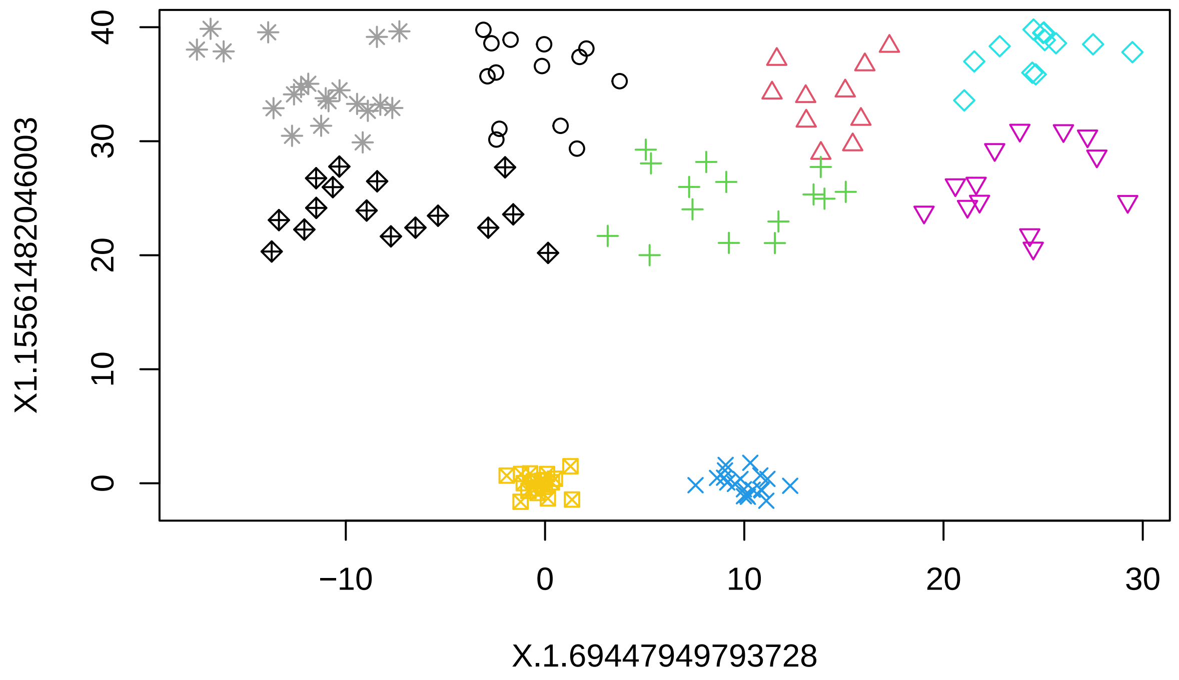
```
# Values of S_k; cg2$Tab[,1] has values of log(S_k), therefore the exp.
plot(1:10,exp(cg2$Tab[,1]),xlab="k",ylab="S_k",type="l")
```



```
# Values of log(S_k) and its expectation under uniform distribution
# The latter are in cg2$Tab[,2].
plot(1:10,cg2$Tab[,1],xlab="k",ylab="log S_k",type="l", ylim = c(6.5, 10))

points(1:10,cg2$Tab[,2],xlab="k",ylab="log S_k",type="l",lty=2)

legend("topright",c("log S_k in data","E(log S_k) uniform"),lty=1:2)
```



```r
km_art <- kmeans(art, 9, nstart=100)
plot(art, col = km_art$cluster, pch =  km_art$cluster)
```



1.b Generate a dataset from a two-dimensional uniform distribution on the rectangle [min x1, max x1] × [min x2, max x2], where x1, x2 are the values of the first and second variable of Artificial Data Set 2. Compute K-means clusterings for K from 1 to 10 (obviously for K = 1 it's just all points in the same cluster) and show at least three scatterplots of the dataset with clusterings with different K. Compare the values of log Sk from clustering Artificial Data Set 2 in (a) with those from clustering the uniformly distributed dataset in a plot like the ones on p.94 and 100, right side, on the course slides (you don't need to plot E(log Sk) from the B uniform datasets generated by the clusGap function in part (a)).

We generate the new dataset as requested, starting from Artificial Data Set 2. Remark: 139 is the number of observations in the original dataset.

```r
head(art)
```

```
##   X.1.69447949793728 X1.15561482046003
## 1          0.4944459         0.3977652
## 2         -0.7534610         0.8599461
## 3         -0.1599999         0.2133719
## 4         -0.3752007        -0.8525524
## 5          1.2756643         1.4925222
## 6          0.1000833         0.8119199
```

```r
y1 <- runif(139,min(art[,1]),max(art[,1]))
y2 <- runif(139,min(art[,2]),max(art[,2]))
data <- cbind(y1,y2)
```

We now compute the K-Means for $K = 1, 2, ..., 10$. Then we create some of the scatterplots as example.

```r
sk <- numeric(0)
kclusterings <- list()

for (k in 1:10){
  kclusterings[[k]] <- kmeans(data, k, nstart=100)
}
kclusterings[2:10]
```

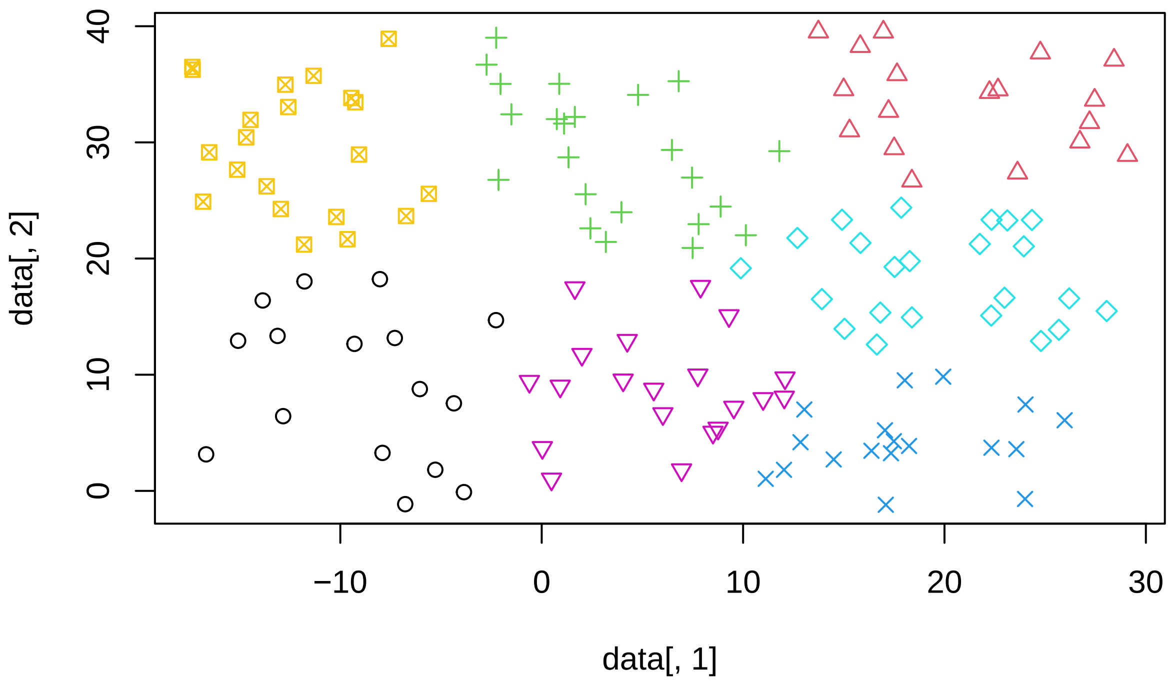
```
## [[1]]
## K-means clustering with 2 clusters of sizes 60, 79
##
## Cluster means:
##          y1       y2
## 1 -6.168215 22.41836
## 2 16.774324 17.40816
##
## Clustering vector:
##   [1] 2 2 1 1 2 1 1 1 1 2 2 2 1 1 2 2 2 2 2 1 2 1 1 1 2 2 1 1 2 2 1 2 2 2 2 1 2
##  [38] 2 1 2 1 1 2 1 2 2 2 2 1 2 2 2 2 2 2 1 1 2 2 2 2 1 1 1 1 2 2 2 2 2 1 1 2 1
##  [75] 2 2 2 2 1 1 1 1 2 2 2 1 1 1 2 2 2 1 2 2 2 2 2 2 1 2 2 1 1 2 2 1 2 1 1 1 2
## [112] 2 1 2 1 1 1 2 1 2 2 1 1 2 1 1 1 2 2 2 2 1 1 1 2 2 1 1 2
##
## Within cluster sum of squares by cluster:
## [1] 10763.94 13567.88
##  (between_SS / total_SS =  43.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
##
## [[2]]
## K-means clustering with 3 clusters of sizes 47, 45, 47
##
## Cluster means:
##          y1        y2
## 1 18.470614 25.799252
## 2  9.753303  6.274011
## 3 -7.488058 26.073428
##
## Clustering vector:
##   [1] 1 1 3 3 1 3 3 3 3 2 1 1 3 3 2 1 1 2 1 3 2 2 3 3 2 1 3 3 2 1 3 1 1 2 2 2 1
##  [38] 1 3 2 3 3 1 1 2 2 1 2 2 2 1 1 1 2 1 3 3 1 1 2 2 2 3 3 3 2 2 1 1 2 2 3 1 3
##  [75] 1 1 2 2 2 3 3 1 2 1 1 2 3 3 1 2 1 3 1 1 1 1 1 2 2 3 1 2 2 3 2 2 3 2 3 3 3 1
## [112] 1 3 1 3 3 3 1 2 1 2 3 3 2 2 3 3 1 1 1 2 3 3 3 1 2 2 3 2
##
## Within cluster sum of squares by cluster:
## [1] 4721.349 4522.764 5737.781
##  (between_SS / total_SS =  65.3 %)
##
## Available components:
##
```

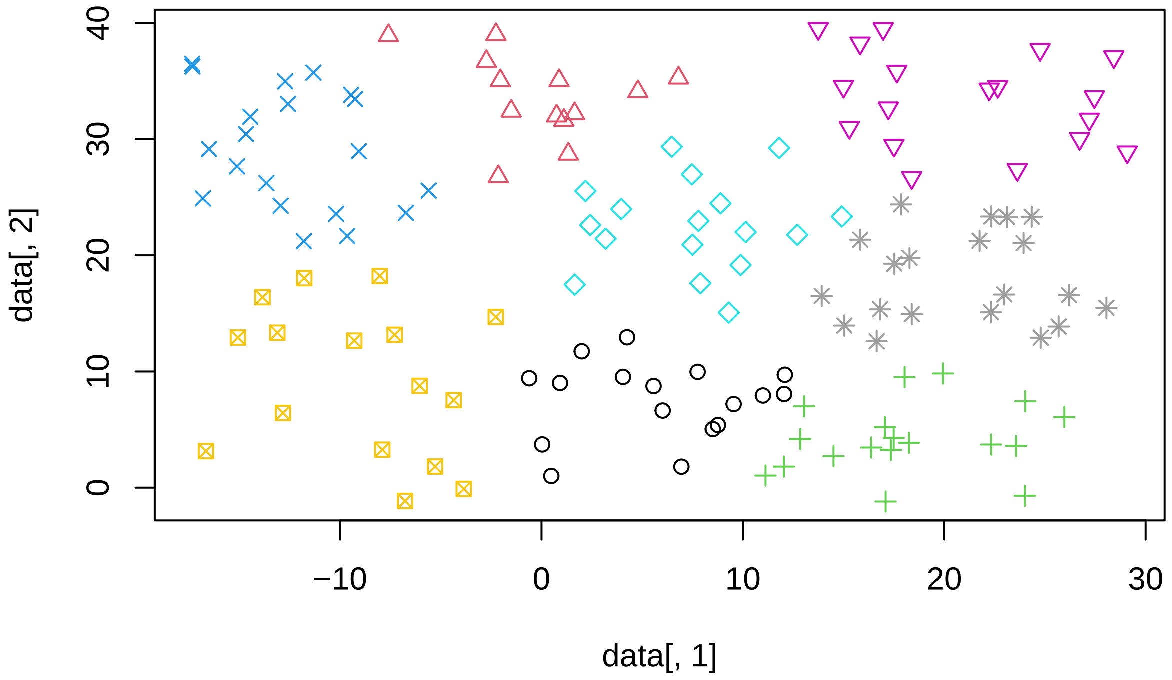
```
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
##
## [[3]]
## K-means clustering with 4 clusters of sizes 38, 36, 24, 41
##
## Cluster means:
##          y1        y2
## 1 17.605178 28.295738
## 2 -6.759935 29.971719
## 3 -5.487989  9.332746
## 4 16.125643  8.344892
##
## Clustering vector:
##   [1] 1 1 2 2 1 2 2 2 2 4 1 1 2 3 4 1 1 4 4 2 4 3 2 2 4 4 3 3 4 4 3 1 1 4 4 3 1
##  [38] 4 3 4 2 2 4 1 4 3 1 4 3 4 1 1 4 4 1 2 2 1 1 4 4 3 3 2 2 4 3 1 1 4 3 2 4 2
##  [75] 1 4 4 4 3 2 2 1 4 1 1 3 2 3 1 4 4 2 1 1 1 1 4 4 2 1 4 3 2 4 4 3 4 2 2 2 1
## [112] 1 3 1 3 2 2 1 3 1 4 2 2 4 3 2 2 1 1 1 4 2 2 2 1 4 3 3 4
##
## Within cluster sum of squares by cluster:
## [1] 3141.248 2779.465 1724.857 2777.043
##  (between_SS / total_SS =  75.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
##
## [[4]]
## K-means clustering with 5 clusters of sizes 24, 24, 40, 23, 28
##
## Cluster means:
##           y1        y2
## 1 -10.913986 30.524423
## 2   4.897116 26.398901
## 3  16.331626  8.113566
## 4  -5.798199  8.978848
## 5  20.699184 29.397519
##
## Clustering vector:
##   [1] 5 5 2 1 5 1 1 1 2 3 5 5 1 4 3 2 5 3 3 1 3 4 1 1 3 3 4 4 2 3 4 5 5 3 3 4 5
##  [38] 3 4 3 2 1 3 2 3 4 5 3 4 3 5 5 3 3 5 1 1 5 2 3 3 4 4 1 1 3 4 2 2 3 4 2 3 2
##  [75] 5 3 3 3 4 2 2 2 2 3 5 5 4 2 4 5 3 3 1 5 5 5 2 3 3 1 2 3 4 2 3 3 4 3 1 1 1 5
## [112] 5 4 2 2 1 2 5 4 2 3 1 1 3 4 2 2 5 5 5 3 1 1 1 5 3 4 4 3
##
## Within cluster sum of squares by cluster:
## [1] 1230.261 1107.994 2619.700 1602.603 1732.312
##  (between_SS / total_SS =  80.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
##
```

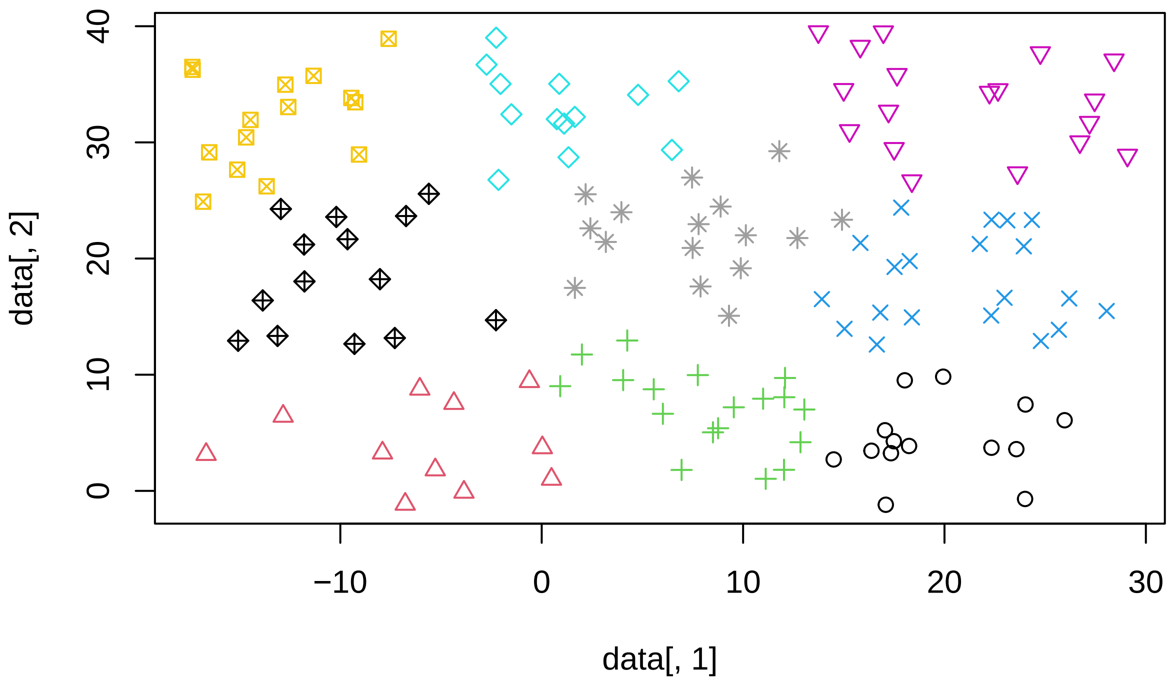
```
## [[5]]
## K-means clustering with 6 clusters of sizes 18, 24, 26, 21, 28, 22
##
## Cluster means:
##          y1        y2
## 1  21.094766 33.486767
## 2  -5.487989  9.332746
## 3  18.483554 18.455868
## 4 -12.137660 29.612160
## 5  14.755972  5.199688
## 6   3.101634 29.377252
##
## Clustering vector:
##   [1] 1 1 6 4 1 4 4 4 6 5 1 1 4 2 5 3 1 5 3 4 5 2 6 4 5 3 2 2 3 3 2 3 1 3 5 2 1
##  [38] 3 2 3 6 4 3 6 5 2 3 5 2 5 1 1 3 5 1 4 4 1 3 5 5 2 2 4 4 5 2 6 3 5 2 6 3 6
##  [75] 1 3 5 5 2 6 6 6 5 1 3 2 6 2 3 5 3 4 3 3 3 6 5 5 4 6 3 2 6 5 5 2 5 4 4 4 1
## [112] 1 2 6 2 6 6 1 2 6 5 4 4 5 2 6 6 3 3 1 5 6 4 4 3 5 2 2 5
##
## Within cluster sum of squares by cluster:
## [1]  745.1064 1724.8569 1173.2416  830.6612 1202.4039  956.1966
##  (between_SS / total_SS =  84.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
##
## [[6]]
## K-means clustering with 7 clusters of sizes 16, 18, 23, 18, 23, 20, 21
##
## Cluster means:
##          y1        y2
## 1  -9.029417  9.320942
## 2  21.094766 33.486767
## 3   3.407469 29.056697
## 4  18.053780  4.171417
## 5  19.706575 18.485975
## 6   5.907171  8.901363
## 7 -12.137660 29.612160
##
## Clustering vector:
##   [1] 2 2 3 7 2 7 7 7 3 6 2 2 7 1 6 5 2 4 5 7 4 6 3 7 6 5 1 1 6 5 1 5 2 5 4 1 2
##  [38] 5 1 6 3 7 5 3 4 6 5 4 1 6 2 2 5 6 2 7 7 2 5 6 4 1 1 7 7 4 6 3 3 4 6 3 5 3
##  [75] 2 5 4 4 1 3 3 3 6 2 5 1 3 1 5 4 5 7 5 5 5 3 4 4 7 3 5 6 3 4 6 1 4 7 7 7 2
## [112] 2 1 3 6 3 3 2 6 3 6 7 7 4 6 3 3 5 5 2 6 3 7 7 5 4 1 1 4
##
## Within cluster sum of squares by cluster:
## [1]  914.6353  745.1064 1055.5198  496.6624  847.7019  711.1935  830.6612
##  (between_SS / total_SS =  87.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
##
## [[7]]
## K-means clustering with 8 clusters of sizes 17, 13, 18, 20, 17, 18, 16, 20
##
## Cluster means:
##             y1        y2
## 1   5.84183788  7.522598
## 2  -0.07715197 33.673778
## 3  18.05378047  4.171417
## 4 -12.36453803 29.146887
## 5   7.53527160 22.580804
## 6  21.09476599 33.486767
## 7  -9.02941676  9.320942
## 8  20.78792268 18.045468
##
## Clustering vector:
##   [1] 6 6 2 4 6 4 4 4 2 1 6 6 4 7 1 5 6 3 8 4 3 1 2 4 1 8 7 7 5 8 7 8 6 8 3 7 6
##  [38] 8 7 5 2 4 8 2 3 1 8 3 7 1 6 6 8 1 6 4 4 6 5 1 3 7 7 4 4 3 1 5 5 3 1 5 8 5
##  [75] 6 8 3 3 7 2 5 5 1 6 8 7 2 7 8 3 8 4 8 8 8 5 3 3 4 5 8 1 5 3 1 7 3 4 2 4 6
## [112] 6 7 5 5 2 2 6 1 5 1 4 4 3 1 2 2 8 5 6 1 2 4 4 8 3 7 7 3
##
## Within cluster sum of squares by cluster:
## [1] 458.0532 312.0182 496.6624 718.1211 487.5609 745.1064 914.6353 617.0732
##  (between_SS / total_SS =  89.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
##
## [[8]]
## K-means clustering with 9 clusters of sizes 14, 11, 18, 20, 13, 18, 15, 16, 14
##
## Cluster means:
##           y1        y2
## 1  19.708854  4.360032
## 2  -5.803545  3.987507
## 3   8.247191  7.099007
## 4  20.787923 18.045468
## 5   1.004788 32.937627
## 6  21.094766 33.486767
## 7 -13.196623 32.127219
## 8   7.602157 22.157876
## 9  -9.833379 18.526190
##
## Clustering vector:
##   [1] 6 6 5 7 6 9 7 9 5 3 6 6 7 9 3 8 6 1 4 9 1 2 5 7 3 4 2 9 8 4 9 4 6 4 1 2 6
##  [38] 4 9 8 5 7 4 5 3 3 4 1 2 3 6 6 4 3 6 7 7 6 8 3 1 2 9 9 7 1 3 8 8 1 3 8 4 8
##  [75] 6 4 1 3 2 5 8 5 3 6 4 2 5 9 4 3 4 7 4 4 4 8 3 1 7 8 4 2 8 1 3 9 1 7 7 9 6
## [112] 6 2 8 8 5 5 6 3 8 3 9 7 1 2 5 5 4 8 6 3 5 7 7 4 1 2 9 1
##
## Within cluster sum of squares by cluster:
## [1] 297.8931 408.3956 439.2507 617.0732 267.1792 745.1064 385.8157 437.6918
## [9] 459.2352
##  (between_SS / total_SS =  90.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
##
## [[9]]
## K-means clustering with 10 clusters of sizes 14, 10, 13, 14, 21, 15, 9, 18, 11, 14
##
## Cluster means:
##           y1        y2
## 1   -9.833379 18.526190
## 2   15.934401 33.672801
## 3    1.004788 32.937627
## 4   19.708854  4.360032
## 5   20.508106 18.297494
## 6  -13.196623 32.127219
## 7   25.795542 32.808446
## 8    8.247191  7.099007
## 9   -5.803545  3.987507
## 10   6.780332 21.567556
##
## Clustering vector:
##   [1]  2  7  3  6  2  1  6  1  3  8  7  7  6  1  8 10  2  4  5  1  4  9  3  6  8
##  [26]  5  9  1 10  5  1  5  2  5  4  9  2  5  1 10  3  6  5  3  8  8  5  4  9  8
##  [51]  2  7  5  8  2  6  6  2 10  8  4  9  1  1  6  4  8 10 10  4  8 10  5 10  2
##  [76]  5  4  8  9  3 10  3  8  7  5  9  3  1  5  8  5  6  5  5  5 10  8  4  6 10
## [101]  5  9 10  4  8  1  4  6  6  1  7  7  9 10 10  3  3  7  8  2  8  1  6  4  9
## [126]  3  3  5  5  7  8  3  6  6  5  4  9  1  4
##
## Within cluster sum of squares by cluster:
##  [1] 459.2352 226.2919 267.1792 297.8931 676.6353 385.8157 153.6024 439.2507
##  [9] 408.3956 300.7406
##  (between_SS / total_SS =  91.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

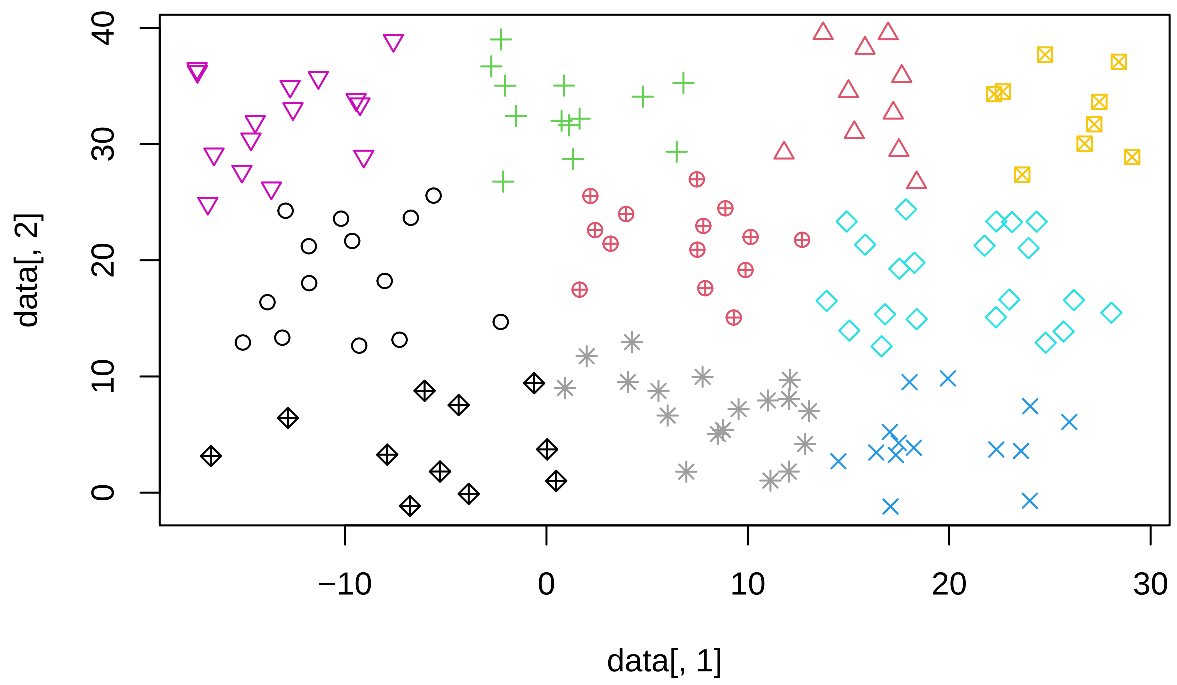
```r
# as examples k=7,8,9,10:

for (i in 7:10){
   plot(data[,1],data[,2], col=kclusterings[[i]]$cluster,  pch=
         kclusterings[[i]]$cluster )
}
```


data[, 1]
data[, 2]
−10
0
10
20
30
0
10
20
30
40


data[, 2]
0
10
20
30
40
−10
0
10
20
30
data[, 1]



data[, 2]
data[, 1]
40
30
20
10
0
−10
0
10
20
30



The cluster analysis seems to work effectively, since the clusters appear to be spherical and thus K-Means is higly suitable.

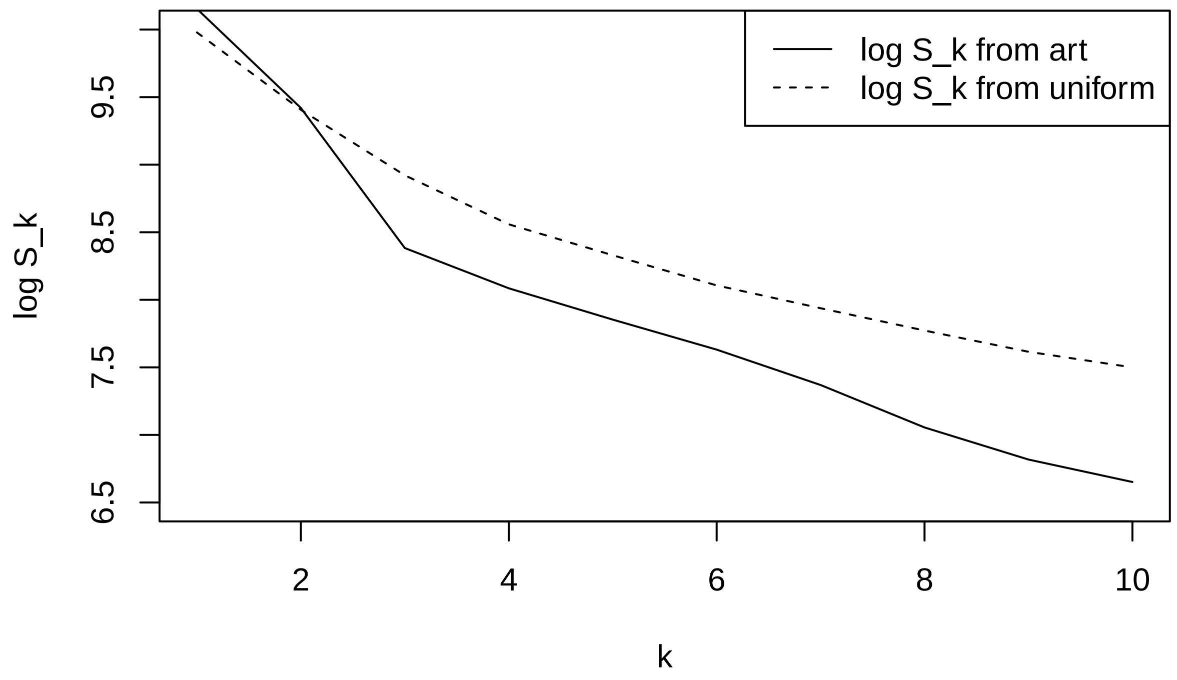
Now we use “clusGap” on our “data” to extract $log(s_k)$ and make a graphical comparison with the $log(s_k)$ of “art”.

Remark: we use spaceH0=“original” instead of “scaledPCA” because we want to use the rectangle along the main axes for the uniform distribution. Anyway, the results are the same.

```
cg3 <- clusGap(data,kmeans,K.max=10,B=100, d.power=2,spaceH0="original",
               nstart=100, iter.max = 50)
print(cg3,method="globalSEmax",SE.factor=2)
```

```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = data, FUNcluster = kmeans, K.max = 10, B = 100, d.power = 2, spaceH0 = "original", nstar
## B=100 simulated reference sets, k = 1..10; spaceH0="original"
##  --> Number of clusters (method 'globalSEmax', SE.factor=2): 1
##           logW   E.logW         gap     SE.sim
##  [1,] 9.978991 9.996364 0.017372987 0.05709351
##  [2,] 9.406393 9.426581 0.020187675 0.06720954
##  [3,] 8.921451 8.971685 0.050234382 0.05216892
##  [4,] 8.558586 8.563917 0.005330706 0.05298978
##  [5,] 8.330004 8.334259 0.004254890 0.05332117
##  [6,] 8.106585 8.130170 0.023585494 0.05125391
##  [7,] 7.937639 7.946842 0.009202823 0.05364136
##  [8,] 7.772591 7.782406 0.009814940 0.05159238
##  [9,] 7.615210 7.633737 0.018527140 0.05216811
## [10,] 7.499711 7.502115 0.002403761 0.05119410
```

```r
plot(1:10,cg2$Tab[,1],xlab="k",ylab="log S_k",type="l", ylim = c(6.5, 10))

points(1:10,cg3$Tab[,1],xlab="k",ylab="log S_k",type="l",lty=2)

legend("topright",c("log S_k from art","log S_k from uniform"),lty=1:2)
```



As we can see, "art" has a steeper decrease in the $log(s_K)$ with respect to "data", revealing that "art" performs better if $K \geq 2$.

2. Using the gapnc-function that automatically estimates the number of clusters for a data set with clusGap with given choices for spaceH0 and SE.factor, generate 100 data sets according to the specifications for Artificial Data Set 1, estimate the number of clusters by all four possible ways to run clusGap and compile four different vectors with all 100 estimated numbers of clusters for each version of clusGap. At the end look at the four distributions of estimated numbers of clusters using a suitable graphical display and comment on them. Are there better or worse results for some of the clusGap-versions ?

Let us create a list of 100 dataframes generated as we did with Artificial Data Set 1. Remark: the performance of the notebook used for this experiment is underwhelming and the time required to compute all the optimal K for the clusters is a lot. Since the principle is the same, let us generate 50 dataframes instead of 100.

```r
library(sn)
```

```
clust <- list(NULL)

for (i in 1:50) {
    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))
    clusterdata <- cbind(v1,v2)
    clust[[i]] <- data.frame(clusterdata)

}
```

Remark: Each clust[[i]] is a dataframe with three underlying distributions. We set 4 different functions to compute “clusGap” with all 4 the options on every dataframe.

```
gapnc_11 <- function(data,FUNcluster=kmeans, K.max=10, B = 100, d.power = 2,
              spaceH0 ="scaledPCA", method ="globalSEmax", SE.factor = 1,...){

          gap1 <- clusGap(data,kmeans,K.max, B, d.power,spaceH0,...)

          nc <- maxSE(gap1$Tab[,3],gap1$Tab[,4],method, SE.factor)

          kmopt <- kmeans(data,nc,...)
          out <- list()
          out$gapout <- gap1
          out$nc <- nc
          out$kmopt <- kmopt
          out
}

gapnc_12 <- function(data,FUNcluster=kmeans, K.max=10, B = 100, d.power = 2,
              spaceH0 ="scaledPCA",method ="globalSEmax", SE.factor = 2,...){

          gap1 <- clusGap(data,kmeans,K.max, B, d.power,spaceH0,...)

          nc <- maxSE(gap1$Tab[,3],gap1$Tab[,4],method, SE.factor)

          kmopt <- kmeans(data,nc,...)
          out <- list()
          out$gapout <- gap1
          out$nc <- nc
          out$kmopt <- kmopt
          out
}

gapnc_21 <- function(data,FUNcluster=kmeans, K.max=10, B = 100, d.power = 2,
                spaceH0 ="original",method ="globalSEmax", SE.factor = 1,...){

          gap1 <- clusGap(data,kmeans,K.max, B, d.power,spaceH0,...)

          nc <- maxSE(gap1$Tab[,3],gap1$Tab[,4],method, SE.factor)

          kmopt <- kmeans(data,nc,...)
          out <- list()
          out$gapout <- gap1
```

```r
          out$nc <- nc
          out$kmopt <- kmopt
          out
}

gapnc_22 <- function(data,FUNcluster=kmeans, K.max=10, B = 100, d.power = 2,
                spaceH0 ="original",method ="globalSEmax", SE.factor = 2,...){

          gap1 <- clusGap(data,kmeans,K.max, B, d.power,spaceH0,...)

          nc <- maxSE(gap1$Tab[,3],gap1$Tab[,4],method, SE.factor)

          kmopt <- kmeans(data,nc,...)
          out <- list()
          out$gapout <- gap1
          out$nc <- nc
          out$kmopt <- kmopt
          out
}
```

We create 4 vectors to allocate the optimal K for all the datasets for each scenario.

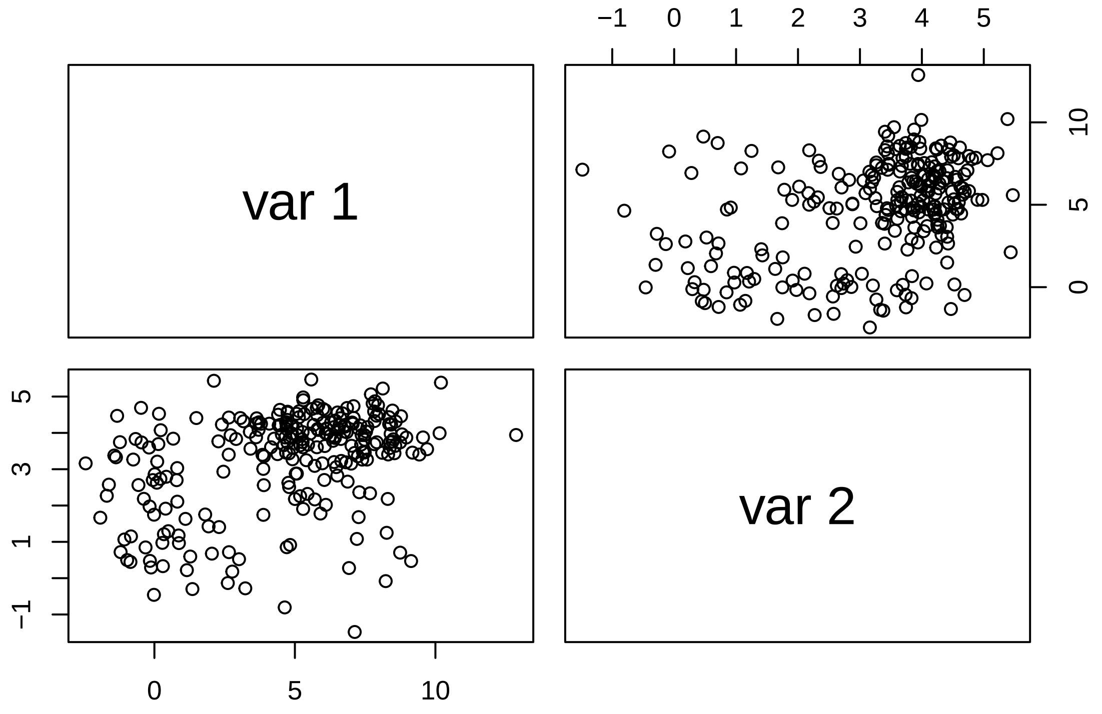
```r
opt_clust_11 <- list(NULL)
opt_clust_12 <- list(NULL)
opt_clust_21 <- list(NULL)
opt_clust_22 <- list(NULL)

for (i in 1:50){
  cgnc11 <- gapnc_11(clust[[i]], nstart=100)
  opt_clust_11[i] <- cgnc11$nc

  cgnc12 <- gapnc_12(clust[[i]], nstart=100)
  opt_clust_12[i] <- cgnc12$nc

  cgnc21 <- gapnc_21(clust[[i]], nstart=100)
  opt_clust_21[i] <- cgnc21$nc

  cgnc22 <- gapnc_22(clust[[i]], nstart=100)
  opt_clust_22[i] <- cgnc22$nc
  }
```

Since we know that the true clusters of each dataset are three as the underlying distributions are exactly three, it is possible to compare it with the optimal K found with our gap statistics and see which options lead to the best performance. Moreover, we could visualize the distributions of K_0 with a barplot. However, there is no need to make a graphical comparison, since all the four methods yield the same optimal number of clusters (i.e. $K_0 = 3$).

```r
# The output of the 4 vectors is not printed because the results are
#all identical:

#opt_clust_11
#opt_clust_12
#opt_clust_21
#opt_clust_22
```

Do the same thing with another model generating data. Here you are asked to generate data from just one further model, with 4 clusters in 4 dimensions, and numbers of observations 50, 100, 150, and 200 for the four clusters. You can use normal distributions for all clusters. These could be generated independently in the four dimensions using rnorm. Before running the simulation, generate a data set from your model and produce a pairs plot of it to check whether the clusters are visible.

```r
set.seed(77665544)

x1 <- rnorm(50,0,1)
x2 <- rnorm(100,2,1.5)
x3 <- rnorm(150,4,0.5)
x4 <- rnorm(200,6,2)

v1 <- cbind(c(x1,x4), c(x2,x3))

pairs(v1)
```



As we can see in the pairs plot, some of the clusters are noticeable:

- in the 1,2 scatterplot, circa 3 clusters can be visually recognised.
- in the 2,1 scatterplot, probably only 2 of them.

We now apply the the "gapnc" functions with the 4 options to evaluate the performance of our cluster analysis performed with the gap statistics. Despite the clusters are all spherical, It might be that the functions will not be able to distinguish all the real clusters.

```
opt_11 <- gapnc_11(v1, nstart=100)
opt_11$nc
```

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

```
opt_12 <- gapnc_12(v1, nstart=100)
opt_12$nc
```

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

```
opt_21 <- gapnc_21(v1, nstart=100)
opt_21$nc
```

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

```
opt_22 <- gapnc_22(v1, nstart=100)
opt_22$nc
```

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

As we predicted, in all the possible scenarios, the "gapnc" functions have estimated only three clusters instead of four.

3.

Regarding the gap statistic, using the notation of Sec. 2.5 of the course, consider choosing $K_{0,q}$ as optimal if it is the smallest K so that:

$$\text{Gap}(K) > \text{Gap}(K^*) - qs_{K^*}, K^* = \arg\max_L \text{Gap}(L)$$

as in "Step 5". Comparing the choices $q = 1$ and $q = 2$, and assuming the same data set to be clustered, and the same B uniform data sets to be generated in "Step 2", which one of these holds: $K_{0,1} \leq K_{0,2}$, or $K_{0,1} \geq K_{0,2}$? Prove your answer (for any data set).

Given the above inequality, if $q = 1$, we get the form:

$$\text{Gap}(K) > \text{Gap}(K^*) - s_{K^*}, K^* = \arg\max_L \text{Gap}(L)$$

As a consequence, for the same K and K^*, it must also be that:

$$\text{Gap}(K) > \text{Gap}(K^*) - 2s_{K^*}, K^* = \arg\max_L \text{Gap}(L)$$

because if $s_{K^*} \geq 0$:

$$\text{Gap}(K^*) - s_{K^*} \geq \text{Gap}(K^*) - 2s_{K^*}$$

Hence:

$$\min(K) : \text{Gap}(K) > \text{Gap}(K^*) - 2s_K{}^* < \min(K) : \text{Gap}(K) > \text{Gap}(K^*) - s_K{}^*$$

Because:

$$\min(K) : \text{Gap}(K) > \text{Gap}(K^*) - s_K{}^* \subseteq \min(K) : \text{Gap}(K) > \text{Gap}(K^*) - 2s_K{}^*$$

Finally:

$$K_{0,1} \geq K_{0,2}$$

Let us try euristically to apply the "gapnc" functions with $q = 1$ and $q = 20$ and check the results on different datasets. Remark: $q = 20$ is to enphasize the difference between the two gaps.

```
gapnc_12 <- function(data,FUNcluster=kmeans, K.max=10, B = 100, d.power = 2,
              spaceH0 ="scaledPCA",method ="globalSEmax", SE.factor = 20,...){

          gap1 <- clusGap(data,kmeans,K.max, B, d.power,spaceH0,...)

          nc <- maxSE(gap1$Tab[,3],gap1$Tab[,4],method, SE.factor)

          kmopt <- kmeans(data,nc,...)
          out <- list()
          out$gapout <- gap1
          out$nc <- nc
          out$kmopt <- kmopt
          out
}
```

```
cgnc1 <- gapnc_11(solive, nstart=100, iter.max=20)
cgnc2 <- gapnc_12(solive, nstart=100, iter.max=20)

cgnc1$nc
```

```
## [1] 10
```

```
cgnc2$nc
```

```
## [1] 4
```

```
geyser <- read.table("geyser.dat",header=TRUE, stringsAsFactors = TRUE)

sgeyser <- scale(geyser)

cgnc3 <- gapnc_11(sgeyser, nstart=100, iter.max=20)
cgnc4 <- gapnc_12(sgeyser, nstart=100, iter.max=20)

cgnc3$nc
```

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

```
cgnc4$nc
```

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

```
bund <- read.table("bundestag.dat",header=TRUE, stringsAsFactors = TRUE)
sbund <- scale(bund[,1:5])

cgnc5 <- gapnc_11(sbund, nstart=100, iter.max=20)
cgnc6 <- gapnc_12(sbund, nstart=100, iter.max=20)

cgnc6$nc
```

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

```
cgnc6$nc
```

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

In practice we confirm what we have already proved mathematically.

4.

Consider the following three observations from $\mathbb{R}^8$, which are visualised on p.109 of the course slides:

$$x_1 = (1, 4, 5, 4, 2, 1, 1, 4),$$
$$x_2 = (2, 3, 2, 2, 3, 3, 3, 3),$$
$$x_3 = (7, 11, 11, 12, 9, 8, 8, 12)$$

If d is the Euclidean or L_1-distance, $d(x_1, x_2) < d(x_1, x_3)$. Invent (or find in the literature) a dissimilarity d^* on $\mathbb{R}^p$, $p > 1$, that expresses the idea that two observations are similar if they tend to be relatively larger or smaller on the same variables, so that in particular $d^*(x_1, x_2) > d^*(x_1, x_3)$, and to show by computation that indeed $d^*(x_1, x_2) > d^*(x_1, x_3)$.

Solution: We want to focus more on the similarity between the trends of the vectors rather than the absolute distance of their elements. It may be useful to create a measure of dissimilarity which computes the squared difference of the differences between variables (i vs $i + 1$) within the two vectors.

Remark: we use the squared differences in order to get only positive values and we generalize the formula by dividing by the number of variables of the vectors -1. Given two vectors X and Y with p elements, the formula is represented as follows:

$$\frac{1}{p-1} \sum_{i=1}^{p-1} \left[(x_i - x_{i+1}) - (y_i - y_{i+1})\right]^2$$

We now use R to proove that $d^*(x_1, x_2) > d^*(x_1, x_3)$:

```r
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)

comp1 <- NULL
comp2 <- NULL
comp3 <- NULL

for (i in 1:7){
  comp1[i] <- ((x1[i]- x1[i+1])-(x2[i]-x2[i+1]))^2
  comp2[i] <- ((x1[i]- x1[i+1])-(x3[i]-x3[i+1]))^2
  comp3[i] <- ((x2[i]- x2[i+1])-(x3[i]-x3[i+1]))^2
}

d12 <- 1/(length(x1)-1)*sqrt(sum(comp1))
d13 <- 1/(length(x1)-1)*sqrt(sum(comp2))
d23 <- 1/(length(x2)-1)*sqrt(sum(comp3))

d12
```

```
## [1] 0.7559289
```

```r
d13
```

```
## [1] 0.404061
```

```r
d23
```

```
## [1] 0.9476071
```

Finally, we can proove that d^* is a dissimilarity because:

- $d^*(X,Y) = d^*(Y,X) \geq 0$ because the subtractions are to the power of 2 hence the commutative property is fulfilled.
- $d^*(X,X) = 0$ (trivially).

In the end we can also confirm that the above dissimilarity fulfills the triangular inequality because:

```r
d12+d23 #> d13
```

```
## [1] 1.703536
```

```r
d13
```

```
## [1] 0.404061
```