Exercises 2-4

Giovanni Zurlo

12/10/2021

Exercise 1

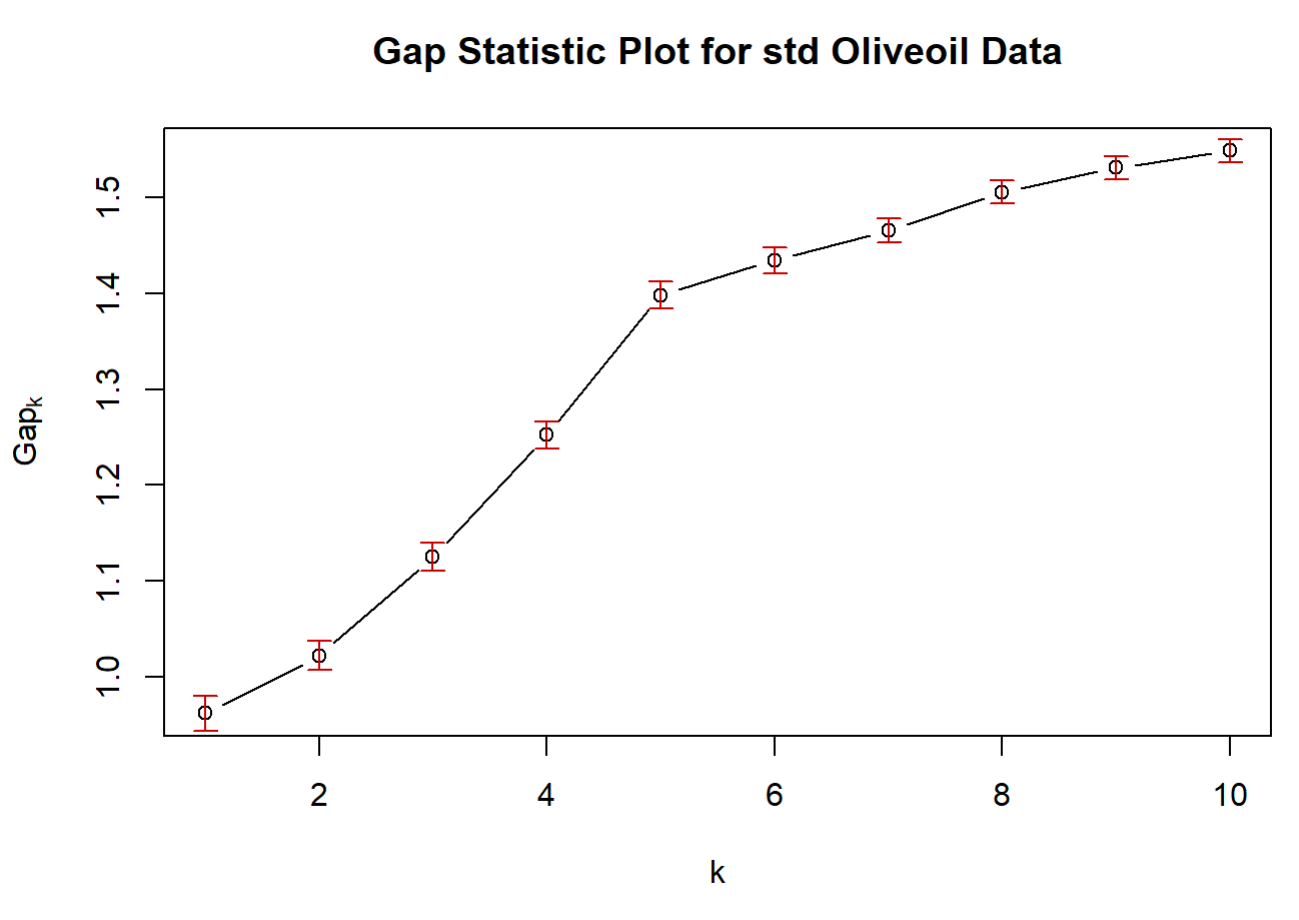
> **1.** *Use the gap statistic method to estimate the number of clusters for the olive oil data, and for Artificial Dataset2. Comment on the solutions.* **2.** *Generate a dataset from a two-dimensional uniform distribution on the rectangle [min x1; max x1] x [min x2; max x2], where x1; x2 are the values of the first and second variable of Artificial Dataset2. Compute K-means clusterings for K from 1 to 10 and show at least three scatterplots of the dataset with clusterings with different K. Compare the values of log Sk from clustering Artificial Dataset2 with those from clustering the uniformly distributed dataset*

Olive Oil Data

```
# Loading and standardizing oliveoil data
library(pdfCluster)
data("oliveoil")
diag(var(oliveoil))
```

```
##   macro.area       region     palmitic palmitoleic      stearic        oleic
##           NA           NA   28423.3515   2755.6583    1350.1903  164681.9365
##     linoleic    linolenic    arachidic   eicosenoic
##   58951.4616     168.1871     485.3319     198.3392
```

```
solive=scale(oliveoil[,3:10])

library(cluster)
set.seed(1234)
cg.solive=clusGap(solive,kmeans, K.max=10, B=150, d.power=2,
                  spaceH0 = 'scaledPCA', nstart=100, iter.max=100)
plot(cg.solive, main="Gap Statistic Plot for std Oliveoil Data")
```




```
# Extracting the best K according to different criteria
maxSE(cg.solive$Tab[,3],cg.solive$Tab[,4],"globalSEmax",SE.factor=2)
```

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

```
maxSE(cg.solive$Tab[,3],cg.solive$Tab[,4],"firstSEmax",SE.factor=2)
```

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

```
maxSE(cg.solive$Tab[,3],cg.solive$Tab[,4],"Tibs2001SEmax",SE.factor=3)
```

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

The gap statistic plot shows the elbow at *k=5*, however the index grows monotonically up to *k=10*. Both the "globalSEmax" and "firstSEmax" criterions suggest choosing *k=9* clusters, for which solution we saw a clear correspondence with the 'region' covariate. The only criterion suggesting the elbow value *k=5* is "Tibs2001SEmax" with 3 as standard error factor. I would choose between *k=5* or *9* depending on the different usage of these results, maybe *5* leads to insights on the data that are more catching and concise.

Artificial Dataset 2

```
# Loading and standardizing the Artificial Dataset 2
artdata=read.table(file.choose())
sartdata=scale(artdata)
library(cluster)
set.seed(1234)
cg.sart=clusGap(sartdata,kmeans, K.max=10, B=130, d.power=2,
                  spaceH0 = 'scaledPCA', nstart=100, iter.max=100)
plot(cg.sart, main="Gap Statistic Plot for std Artificial Data 2")
```


```
# Extracting the best K according to different criteria
maxSE(cg.sart$Tab[,3],cg.sart$Tab[,4],"globalSEmax",SE.factor=2)
```

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

```
maxSE(cg.sart$Tab[,3],cg.sart$Tab[,4],"firstSEmax",SE.factor=2)
```

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

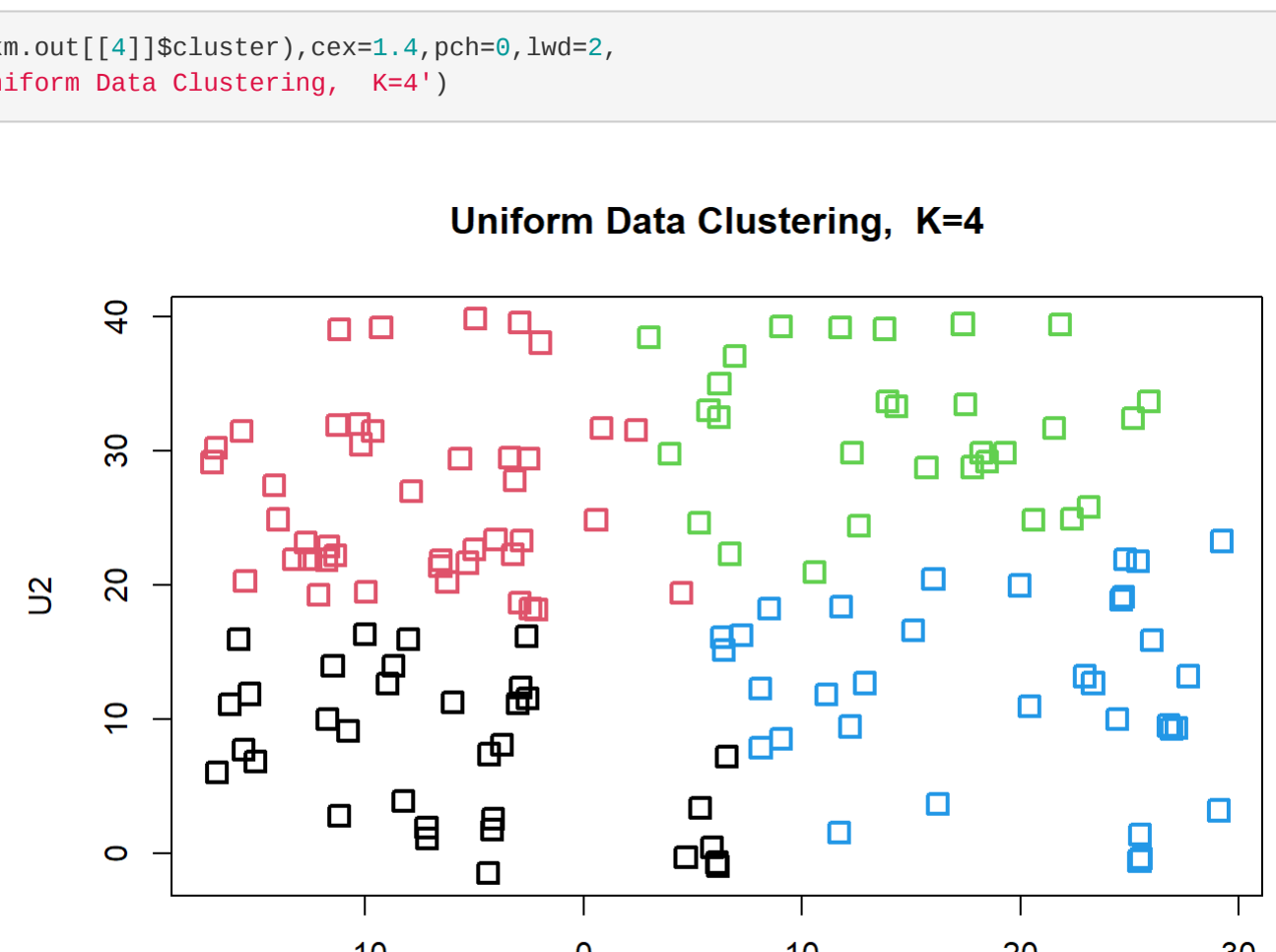
```
maxSE(cg.sart$Tab[,3],cg.sart$Tab[,4],"Tibs2001SEmax",SE.factor=1)
```

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

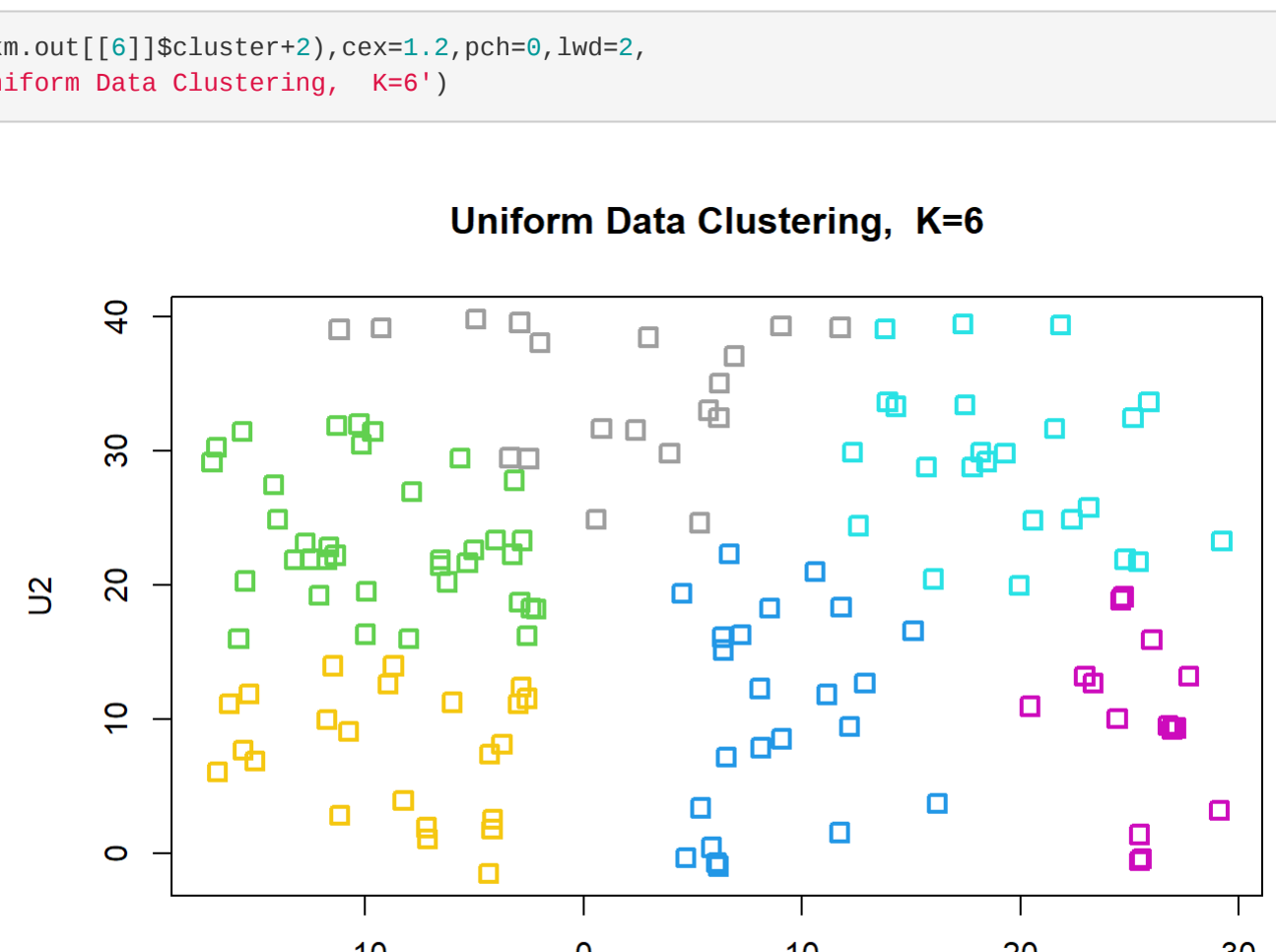
The gap statistic plot shows a local maximum at *k=3*, which we know to be the most natural choice (based on the data generating process). "GlobalSEmax" suggests *9* as the optimal number of groups, but both "Tibs2001SEmax" and "firstSEmax" point towards the local maximum of the graph *k=3*, which seems the most reasonable value.

Comparison with a uniform distribution

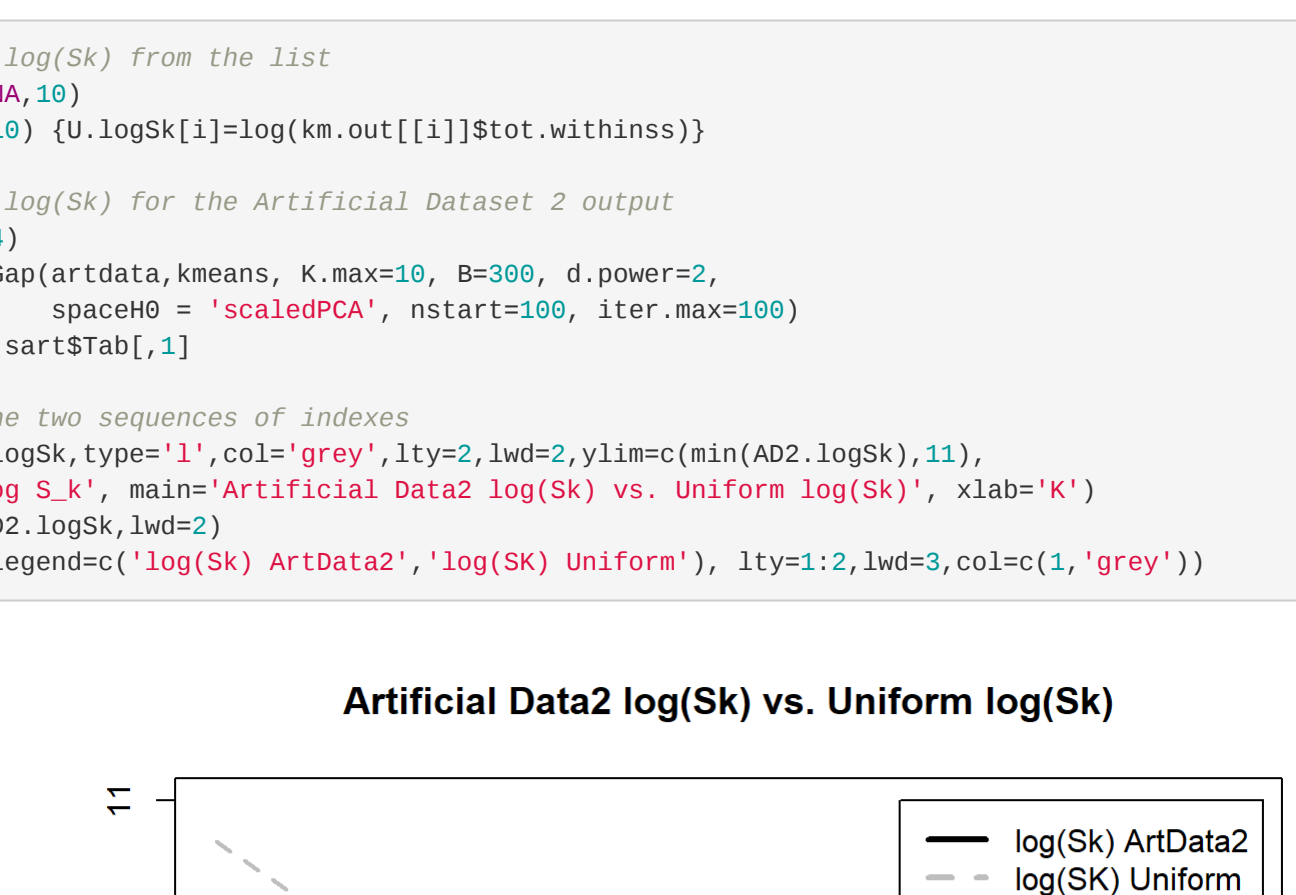
```
set.seed(1234)
# Generating a dataset from a two-dimensional uniform distribution
U1=runif(140, max = max(artdata$V1),min = min(artdata$V1))
U2=runif(140, max = max(artdata$V2),min = min(artdata$V2))
U=data.frame(U1,U2)

# Collecting the outputs of the k-means function in a list, from 1 to K.max
km.out=list()
set.seed(1234)
for (i in 1:10) {km.out[[i]]=kmeans(U,iter.max = 100,nstart =100, centers = i)}

# Three scatterplots of the uniform data with different K
plot(U,col=(km.out[[2]]$cluster+2),cex=1.2,pch=0,lwd=2,
     main='Uniform Data Clustering,  K=2')
```



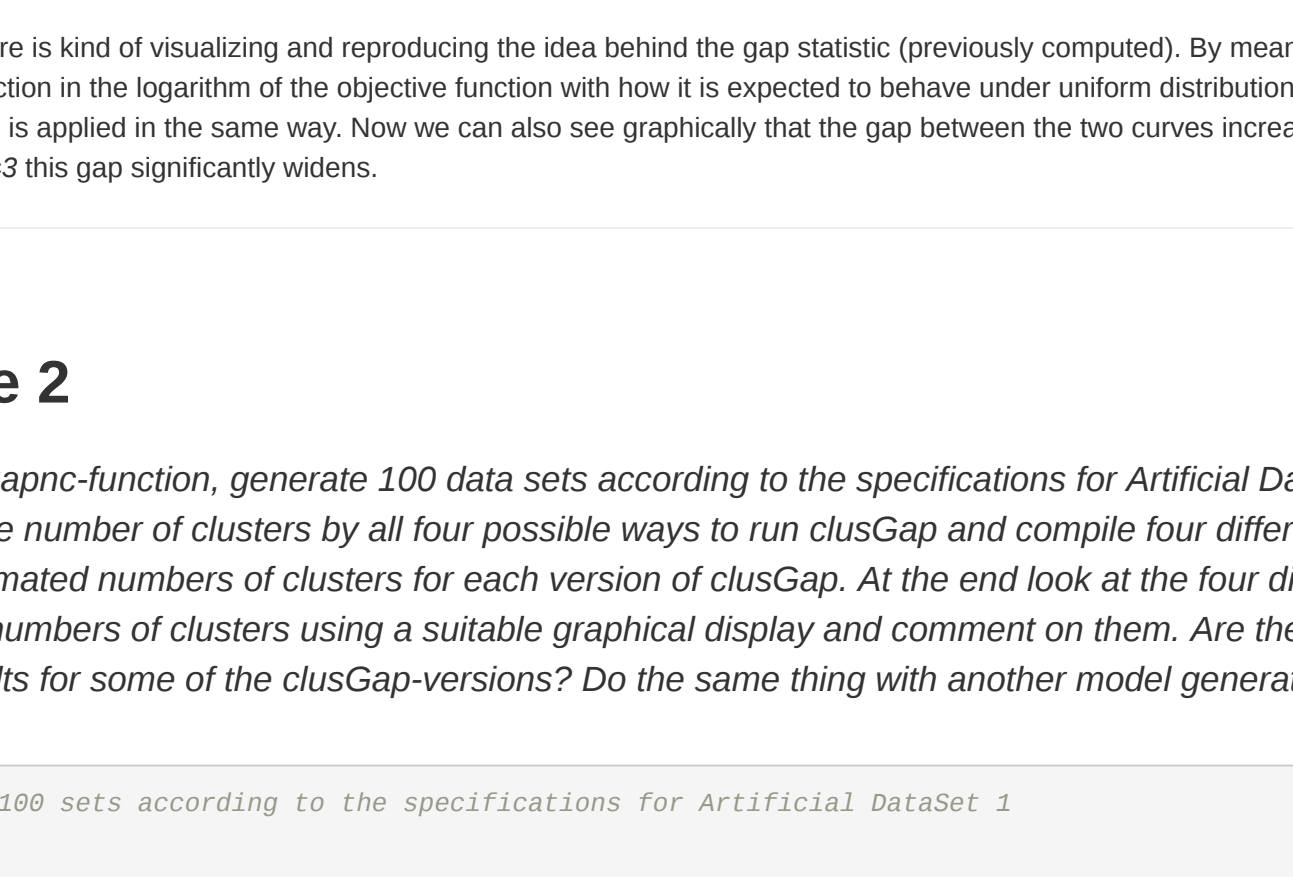

```
plot(U,col=(km.out[[4]]$cluster),cex=1.4,pch=0,lwd=2,
     main='Uniform Data Clustering,  K=4')
```




```
plot(U,col=(km.out[[6]]$cluster+2),cex=1.2,pch=0,lwd=2,
     main='Uniform Data Clustering,  K=6')
```




```
# Extracting log(Sk) from the list
U.logSk=rep(NA,10)
for (i in 1:10) {U.logSk[i]=log(km.out[[i]]$tot.withinss)}

# Extracting log(Sk) for the Artificial Dataset 2 output
set.seed(1234)
cg.sart=clusGap(artdata,kmeans, K.max=10, B=300, d.power=2,
                  spaceH0 = 'scaledPCA', nstart=100, iter.max=100)
AD2.logSk=cg.sart$Tab[,1]

# Plotting the two sequences of indexes
plot(1:10,U.logSk,type='l',col='grey',lty=2,lwd=2,ylim=c(min(AD2.logSk),11),
     ylab='log S_k', main='Artificial Data2 log(Sk) vs. Uniform log(Sk)', xlab='K')
lines(1:10,AD2.logSk,lwd=2)
legend(7,11,legend=c('log(Sk) ArtData2','log(SK) Uniform'), lty=1:2,lwd=3,col=c(1,'grey'))
```



What I'm doing here is kind of visualizing and reproducing the idea behind the gap statistic (previously computed). By means of the last plot I can compare the reduction in the logarithm of the objective function with how it is expected to behave under uniform distribution on the same "domain", for which k-means is applied in the same way. Now we can also see graphically that the gap between the two curves increases with the number of clusters, and at *k=3* this gap significantly widens.

Exercise 2

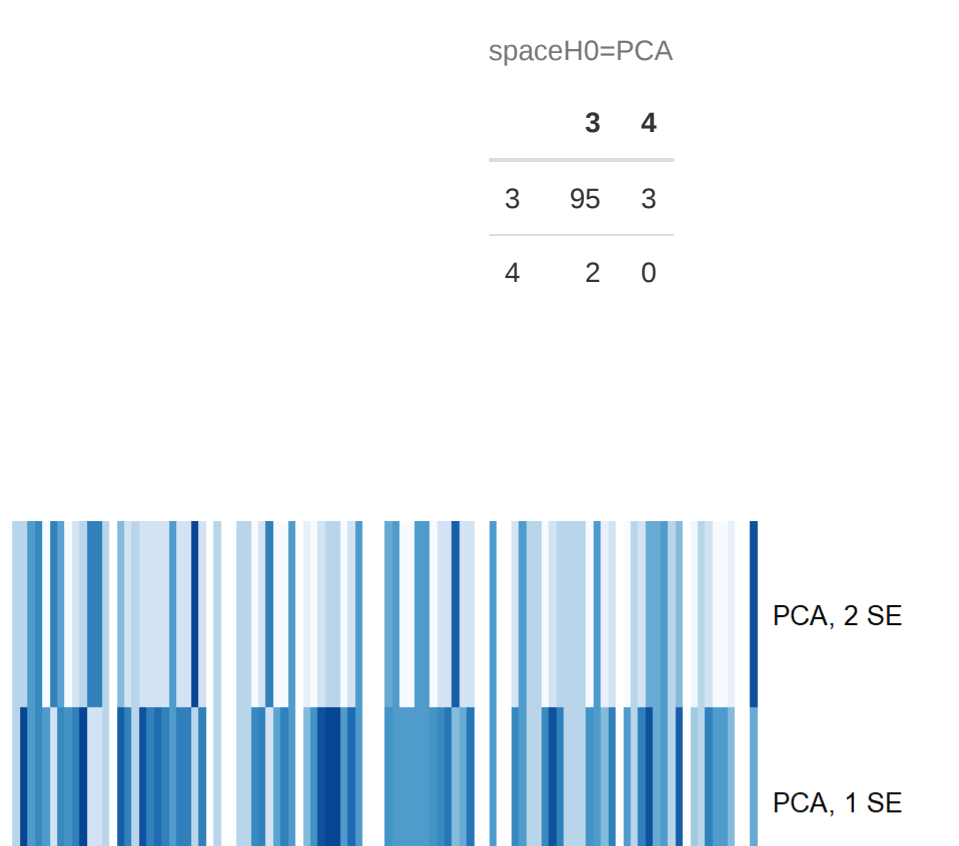
> *Using the gapnc-function, generate 100 data sets according to the specifications for Artificial Data Set 1, estimate the number of clusters by all four possible ways to run clusGap and compile four different vectors with all 100 estimated numbers of clusters for each version of clusGap. At the end 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? Do the same thing with another model generating data*

```
# Generating 100 sets according to the specifications for Artificial DataSet 1
data=list()
simruns=100
kmax=10
library(sn)
set.seed(123)
for (i in 1:simruns) {
  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))
  data[[i]] <- cbind(v1,v2)
}

out.orig1=list()
set.seed(1)
for (i in 1:100) {
  out.orig1[[i]]=gapnc(data[[i]], K.max = kmax ,spaceH0='original',SE.factor=1,
                     iter.max=100, nstart=3)}

## The same FOR loop will be run with different values of 'spaceH0' and 'SE.factor'
```

100 Datasets were generated and stored in a list. The gapnc function is run on each of them (simulating B=100 samples from a uniform distribution) and results are collected in a list; this is performed for each combination of the *spaceH0* and *SE.factor* parameters. Parts of code which are repeated are not shown for the sake of simplicity.

```
# Compile 4 different vectors with all 100 est numbers of clusters for each clusGap run
clu.orig1=rep(NA,simruns)
for (i in 1:simruns) {clu.orig1[i]=out.orig1[[i]]$nc}

clu.orig2=rep(NA,simruns)
for (i in 1:simruns) {clu.orig2[i]=out.orig2[[i]]$nc}

clu.pca1=rep(NA,simruns)
for (i in 1:simruns) {clu.pca1[i]=out.pca1[[i]]$nc}

clu.pca2=rep(NA,simruns)
for (i in 1:simruns) {clu.pca2[i]=out.pca2[[i]]$nc}
```

I then extracted the optimal number of clusters for each simulated dataset and each variant of the clusGap function, obtaining 4 vectors of lenght 100. These have been represented by means of a heatmap in which orange bars represent deviations from the "correct" number of groups. As you can see, methods based on the rotation of principal axes perform slightly worse than those which keep the original space of the sample, but these findings cannot be generalized. Results are also tabulated under here in 2-by-2 tables.



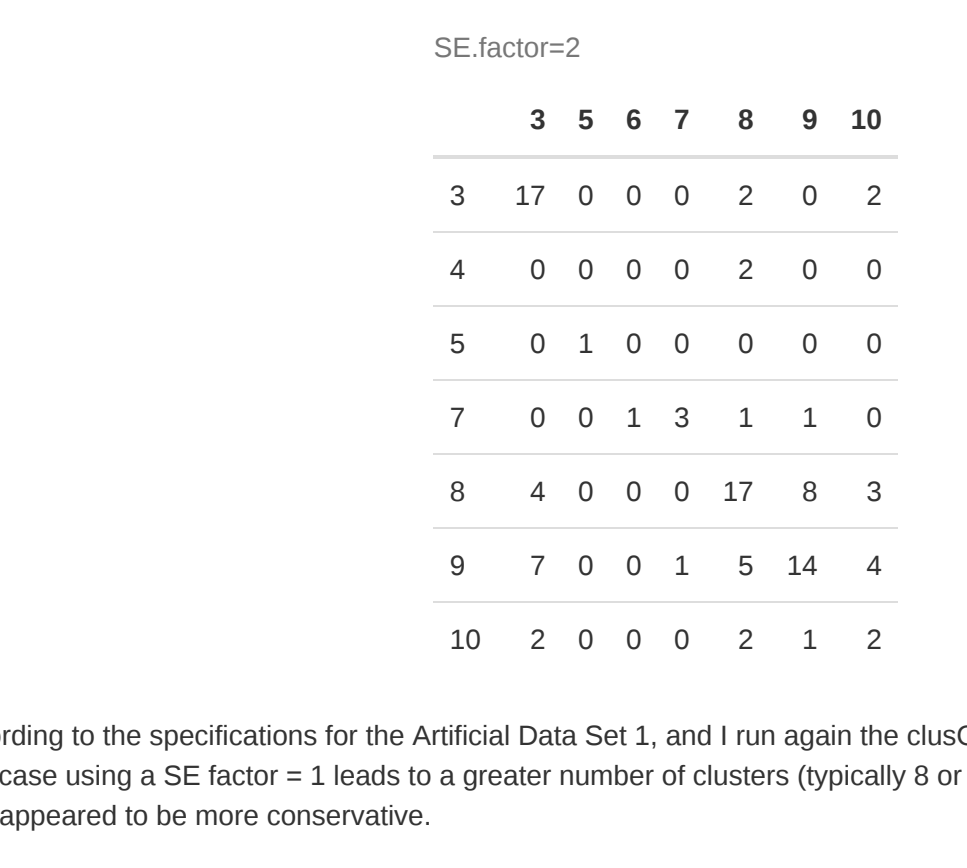
spaceH0=Original

	3	4
3	98	1
4	1	0

spaceH0=PCA

	3	4
3	95	3
4	2	0

Artificial Dataset 2



SE.factor=1

	3	6	7	8	9	10
3	4	0	0	0	0	2
4	0	0	0	0	0	1
6	0	1	0	0	0	0
7	0	0	1	1	0	0
8	1	0	0	8	6	3
9	1	0	0	4	30	12
10	2	0	1	3	12	7

SE.factor=2

	3	5	6	7	8	9	10
3	17	0	0	0	2	0	2
4	0	0	0	0	2	0	0
5	0	1	0	0	0	0	0
7	0	0	1	3	1	1	0
8	4	0	0	0	17	8	3
9	7	0	0	1	5	14	4
10	2	0	0	0	2	1	2

I generated 100 more datasets according to the specifications for the Artificial Data Set 1, and I run again the clusGap with different pairs of parameters. As you can see, in this case using a SE factor = 1 leads to a greater number of clusters (typically 8 or more); on the other hand, optimality criteria with SE factor = 2 appeared to be more conservative.

Exercise 5

> *A political scientist wants to cluster the respondents of a questionnaire using a distance-based method. The questionnaire has preference questions with two response options of the type "I prefer the current situation" is coded 0 and the option that suggests something else is coded 1. Would you prefer the simple matching distance or the Jaccard distance here? Why?*

- If the *"something else"* response options were more or less specific in indicating an alternative view for each question, I would prefer the simple matching distance because it is a better measure in the case of symmetric dummy variables. SMD counts both mutual presences and mutual absence as matches, that is agreement in general, and compares it to the total number of attributes in the universe; it also opens to the possibility of splitting the *"something else"* responses into additional levels, in order to measure dissimilarity in a more precise way. On the other hand, if responses were quite generic, joint absences (preference for something else) could not be correctly read as indicators of similarity. In this case Jaccard distance may represent the most appropriate choice because it only counts mutual presence (preference for the current situation).

> *Geographers want to cluster areas in the Swiss alps according to danger from avalanches in order to produce a map. Their variables are, all for the year 2019: (i) the number of avalanches in the area, (ii) the average percentage of the area covered by an avalanche, (iii) number of persons injured or dead in incidents involving avalanches in the area,(iv) Swiss Francs investment in the security of the ski slopes in the area, (v) Swiss Francs budget for emergency rescue in the area. Would you prefer the Euclidean distance on raw data, the Euclidean distance on scaled data, the Manhattan distance on raw data, the Manhattan distance on scaled data, or the Mahalanobis distance for these data? Why?*

- First, input variables at disposal have all different scale or type (percentages, discrete numeric variables and continuous ones). Researchers should either use a scale-invariant clustering algorithm or a scale invariant distance, otherwise calculations will be influenced by features showing larger differences. Mahalanobis metric is the only scale-invariant candidate we have here, and practically it downweights distances in a way that is proportional to the estimated variances of the corresponding features. So Mahalanobis is surely suited in this case. The euclidean metric applied to scaled data also leads to similar results (they differ by a rotation of axes performed by Mahalanobis), and this represents the most popular strategy in clustering problems. In general, scaling data before the analysis allows using any metric without caring about this issue. This means that researchers could even use the Manhattan distance (on scaled data) whenever they find reasons to do that, like in the case of high-dimensional samples.

EXERCISE 3

The Mahalanobis distance is defined as:

$$d_M(x,y) = \sqrt{(x-y)^T S^{-1}(x-y)}$$

where S is known to be SYMMETRIC and POSITIVE SEMI-DEFINITE.
By the SPECTRAL THEOREM, there exists always a decomposition

$$S = U^T D U = (U^T D^{1/2})(D^{1/2} U) = Q^T Q$$

where $D = \text{diag}(\lambda_1, \dots, \lambda_p)$ is a diagonal $p \times p$ matrix and U is an orthonormal $p \times p$ matrix of eigenvectors.

Then, let $\tilde{x} = D^{-1/2}(U^{-1})^T x$ and $\tilde{y} = D^{-1/2}(U^{-1})^T y$
we can write the Mahalanobis distance between x and y as the euclidean distance between $\tilde{x}$ and $\tilde{y}$

$$d_M(x,y) = \sqrt{(x-y)^T S^{-1}(x-y)} = \sqrt{(\tilde{x}-\tilde{y})^T(\tilde{x}-\tilde{y})} = d_E(\tilde{x},\tilde{y})$$

WHICH IS KNOWN TO SATISFY THE TRIANGULAR INEQUALITY

$$d_E(\tilde{x},\tilde{y}) \le d_E(\tilde{x},w) + d_E(\tilde{x},w) \qquad w \in \mathbb{R}^p$$

since $\|\tilde{x}-\tilde{y}\|_E = \|\tilde{x}-w+w-\tilde{y}\|_E \le \|\tilde{x}-w\|_E + \|w-\tilde{y}\|_E$

The SIMPLE MATCHING DISTANCE is defined as:

$$d_{SM}(x,y) = \frac{1}{p}\sum^{p} 1(x_j \neq y_j)$$

For binary data (*) it can be interpreted as PROPORTIONAL TO THE MANHATTAN DISTANCE:

$$\frac{1}{p}\sum^{p} 1(x_j \neq y_j) = \frac{1}{p}\sum^{p} |x_j - y_j| = \frac{1}{p} d_{L_1}(x,y)$$

Then, $\frac{1}{p} d_{L_1}(x,y) \leq \frac{1}{p} d_{L_1}(x,z) + \frac{1}{p} d_{L_1}(z,y)$

$$d_{L_1}(x,y) \leq d_{L_1}(x,z) + d_{L_1}(z,y)$$

without loss of generality, assume that $z = 0_p$

$$\sum_{j=1}^{p} |x_j - y_j| \leq \sum_{j=1}^{p} |x_j| + \sum_{j=1}^{p} |y_j|$$

which holds using the fact that $|a+b| \leq |a| + |b|$ for any reals a, b.

(*) In general, one could recode any categorical variable by splitting the attributes into a set of dummy variables; in fact, triangular inequality holds in general for the SMD.

EXERCISE 4

Increasing the value of the S.E. factor q may lead to smaller optimal number K_0 of clusters
(Keeping constant the dataset and the collection of B randomly generated uniform samples)

$$\text{so} \quad K_{0,1} \geq K_{0,2} \quad \text{because}$$

$$GAP(K^*) - S_{K^*} > GAP(K^*) - 2S_{K^*}$$

$$S_{K^*} < 2S_{K^*}$$

ALWAYS TRUE SINCE S_{K^*} IS ALWAYS POSITIVE BEING THE ESTIMATED S.E. OF $GAP(K^*)$