SY09 Printemps 2019
TP 6
Classification automatique: K-means avec métrique adaptative

1 Introduction

L'algorithme des K-means avec distance adaptative consiste à utiliser l'algorithme des K-means en utilisant la distance de Mahalanobis à la place de la distance euclidienne.

Définition 1 (Distance de Mahalanobis) *La distance entre deux points $\boldsymbol{x}$ et $\boldsymbol{y}$, au sens d'une métrique induite par une matrice M, est définie par*

$$d_M^2(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^T \; M \; (\boldsymbol{x} - \boldsymbol{y}) \,. \tag{1}$$

La distance de Mahalanobis entre un point $\boldsymbol{x}$ et le centre μ_k de la classe ω_k est la distance de Mahalanobis entre ces points, au sens de la métrique induite par l'inverse $\tilde{V}_k^{-1}$ de la matrice de covariance empirique normalisée de la classe :

$$\begin{aligned} d^2_{V_k^{-1}} &= (\boldsymbol{x} - \mu_k)^T \; V_k^{-1} \; (\boldsymbol{x} - \mu_k) \,, \\ \tilde{V}_k &= (\rho_k \; \det V_k)^{-1/p} \, V_k, \\ V_k &= \frac{1}{n_k} \sum_{i : \boldsymbol{x}_i \in P_k} (\boldsymbol{x}_i - \mu_k) \, (\boldsymbol{x}_i - \mu_k)^T. \end{aligned}$$

L'utilisation de cette métrique permet de tenir compte de la dispersion des points, qui peut varier selon la classe considérée. Ce surcroît de flexibilité a un prix : cela nécessite d'estimer davantage de paramètres (à chaque itération, les matrices de covariance empirique des classes, en plus des centres de gravité). Cette variante est donc plus sensible au manque de données que l'algorithme des K-means classique.

L'objectif de ce TP est d'implémenter puis d'étudier les propriétés (avantages, inconvénients) de l'utilisation d'une telle métrique en classification.

2 Programmation

On cherche à programmer la méthode des K-means avec distance adaptative, décrit dans l'algorithme 1. On prendra en compte les éléments suivants.

Arguments d'entrée La fonction à développer accepte comme arguments d'entrée le tableau de données que l'on cherche à regrouper en classes, et le nombre de classes à rechercher (on pourrait alternativement donner directement à la fonction des coordonnées des centres des classes à partir desquels itérer).

Elle accepte de même un nombre d'itérations maximal $n_{\text{iter.max}}$, un nombre d'essais n_{start}, et une précision ε (par défaut, on peut fixer $n_{\text{iter.max}} = 100$, $n_{\text{start}} = 1$ et $\varepsilon = 10^{-5}$). Le premier argument a pour but d'empêcher l'algorithme d'itérer indéfiniment à la recherche d'un optimum ; le second représente le nombre de fois où l'algorithme sera appliqué au jeu de données X à partir de différentes initialisations, afin d'augmenter les chances de converger vers l'optimum global du critère optimisé ; le dernier vise à déterminer quand l'algorithme a convergé.

Initialisation Les centres des classes μ_k peuvent être initialisés par K points tirés au hasard dans le jeu de données X. Pour cela, on peut s'appuyer sur la fonction `sample`, qui permet de tirer au hasard un certain nombre d'éléments dans un vecteur.

Chaque matrice de covariance normalisée $\tilde{V}_k$ peut être initialisée par $(\rho_k)^{-1/p}\, I_p$, où I_p est la matrice identité de dimension p, et ρ_k est le volume souhaité (imposé) pour la matrice $\tilde{V}_k^{-1}$ (voir ci-dessous). On peut prendre par défaut $\rho_k = 1$, pour tout $k = 1, \ldots, K$.

Mise à jour des paramètres L'algorithme des K-means avec distance adaptative répète les opérations suivantes, jusqu'à convergence :

1. calcul d'une nouvelle partition des données $P = (P_1, \ldots, P_K)$ en K groupes : chaque point est affecté à la classe dont le centre μ_k est le plus proche au sens de la distance de Mahalanobis (calculée avec la matrice de covariance normalisée $\tilde{V}_k$). On peut pour cela s'appuyer sur la fonction `which.min`, combinée avec la fonction `apply`.
2. À partir de cette nouvelle partition, les centres des classes et les matrices de covariance (normalisées) sont mis à jour ; plus particulièrement :

$$\begin{aligned} \mu_k &\longleftarrow \frac{1}{n_k} \sum_{i:\boldsymbol{x}_i \in P_k} \boldsymbol{x}_i; \\ V_k &\longleftarrow \frac{1}{n_k} \sum_{i:\boldsymbol{x}_i \in P_k} (\boldsymbol{x}_i - \mu_k)\,(\boldsymbol{x}_i - \mu_k)^T, \\ \tilde{V}_k &\longleftarrow (\rho_k\, \det V_k)^{-1/p}\, V_k, \end{aligned}$$

   où $n_k = \operatorname{card}\{i : \boldsymbol{x}_i \in P_k\}$.

Pour diverses raisons (qui seront en partie élucidées par la suite), il faut normaliser les matrices de covariance empiriques V_k utilisées dans le calcul de la distance de Mahalanobis. À chaque étape de l'algorithme, la matrice $\tilde{V}_k$ normalisée est ainsi calculée à partir de V_k de telle sorte que $\det \tilde{V}_k^{-1} = \rho_k$.

Convergence À chaque itération de l'algorithme, la convergence peut être testée en calculant la distance δ entre les centres μ_k des classes calculés à l'itération présente et ceux $\mu_{k,\mathrm{prec}}$ calculés à l'itération précédente :

$$\delta = \sum_{k=1}^{K} \|\mu_k - \mu_{k,\mathrm{prec}}\|^2.$$

On supposera que la convergence est atteinte dès lors que $\delta \leqslant \varepsilon$, où ε est une valeur de précision fixée par l'utilisateur.

Essais À chaque essai, une fois que l'algorithme a convergé vers un optimum local — c'est-à-dire une partition définie par K centres $\mu_k^\star$ et matrices $\tilde{V}_k^\star$, on dispose également de la valeur du critère correspondante, c'est-à-dire la somme des distances des points au centre de gravité de leur classe :

$$J^\star = J(\mu_k^\star, \tilde{V}_k^\star) = \sum_{k=1}^{K} \sum_{i:\boldsymbol{x}_i \in P_k} d^2_{(\tilde{V}_k^\star)^{-1}}(\boldsymbol{x}_i, \mu_k).$$

Si la valeur de ce critère est plus faible que celle $J(\mu_k^{\mathrm{prec}}, \tilde{V}_k^{\mathrm{prec}})$ obtenue après une exécution précédente de l'algorithme *pour les mêmes valeurs de* ρ_k, alors l'optimum local considéré est meilleur que l'optimum local précédent.

À chaque nouvelle convergence de l'algorithme, il convient alors de conserver en mémoire les paramètres correspondant à l'optimum local courant, s'ils sont meilleurs que ceux correspondant au meilleur optimum local obtenu précédemment.

Développement L'algorithme à développer est simple et peut être réalisé avec un certain nombre de fonctions usuelles de `R` : `apply`, `sample`, `which.min`, ainsi que la fonction `distXY` fournie. On se basera sur le patron de fonction `adapkm` fourni.

On stockera les coordonnées des centres des classes dans une matrice $K \times p$ et les matrices de covariance dans un tableau $p \times p \times K$. On utilisera une liste pour retourner toutes les sorties.

On évitera autant que possible les boucles (il pourra être nécessaire de boucler sur les classes, mais le code ne devrait pas comporter de boucles sur les individus).

3 Applications

3.1 Données synthétiques

On cherche dans un premier temps à tester l'algorithme développé sur diverses données synthétiques, de manière à appréhender son fonctionnement et ses limitations. On considère pour cela trois jeux de données `Synth1`, `Synth2` et `Synth3`. On pourra charger les données au moyen du code suivant :

```
> X <- read.csv("SynthN.csv", header=T, row.names=1)
> z <- X[,3]
> X <- X[,-3]
```

Pour chacun des jeux de données, effectuer une classification au moyen de l'algorithme des K-means classique, puis de l'algorithme des K-means avec distance adaptative. On comparera les résultats obtenus et on interprétera en fonction des données. On peut utiliser la fonction `adjustedRandIndex` (bibliothèque `mclust`) pour calculer l'adéquation de la partition trouvée à la partition réelle.

Voir table 1 pour les résultats numériques.

TABLE 1 – ARI sur les trois jeux de données synthétiques `Synth1`, `Synth2` et `Synth3`.

données	`Synth1`	`Synth2`	`Synth3`
ARI K-means	0.9799995	0.8456246	0.1328602
ARI K-means adaptatifs	0.9602001	0.8456292	0.3939229

3.2 Données réelles

Iris Déterminer une classification des données `Iris` avec la méthode des K-means classique, puis avec les K-means adaptatifs, pour $K = 2, 3, \ldots, 5$. Calculer la valeur du critère optimisé (inertie dans le premier cas, distance totale dans le second) pour $K = 1$. Afficher les valeurs de critère en fonction de K. Quel semble être le meilleur nombre de classes dans chacun des cas ?

Effectuer une partition des données pour $K = 2$, puis pour $K = 3$. Comparer les résultats obtenus par les deux méthodes, et interpréter.

Il est à présent nécessaire de régulariser les matrices de covariance intra-classes lors des itérations. On obtient (`nstart`=25 toujours) les valeurs d'inertie consignées dans le tableau 2. Les partitions en $K = 2$ et $K = 3$ groupes sont représentées dans la figure 2.

TABLE 2 – Inerties obtenues sur les Iris pour $K = 1$ à $K = 5$.

K	1	2	3	4	5
Inertie K-means	681.37060	152.34795	78.85144	57.22847	46.44618
Inertie K-means adaptatifs	600.00000	60.59326	40.96111	33.71656	29.09319

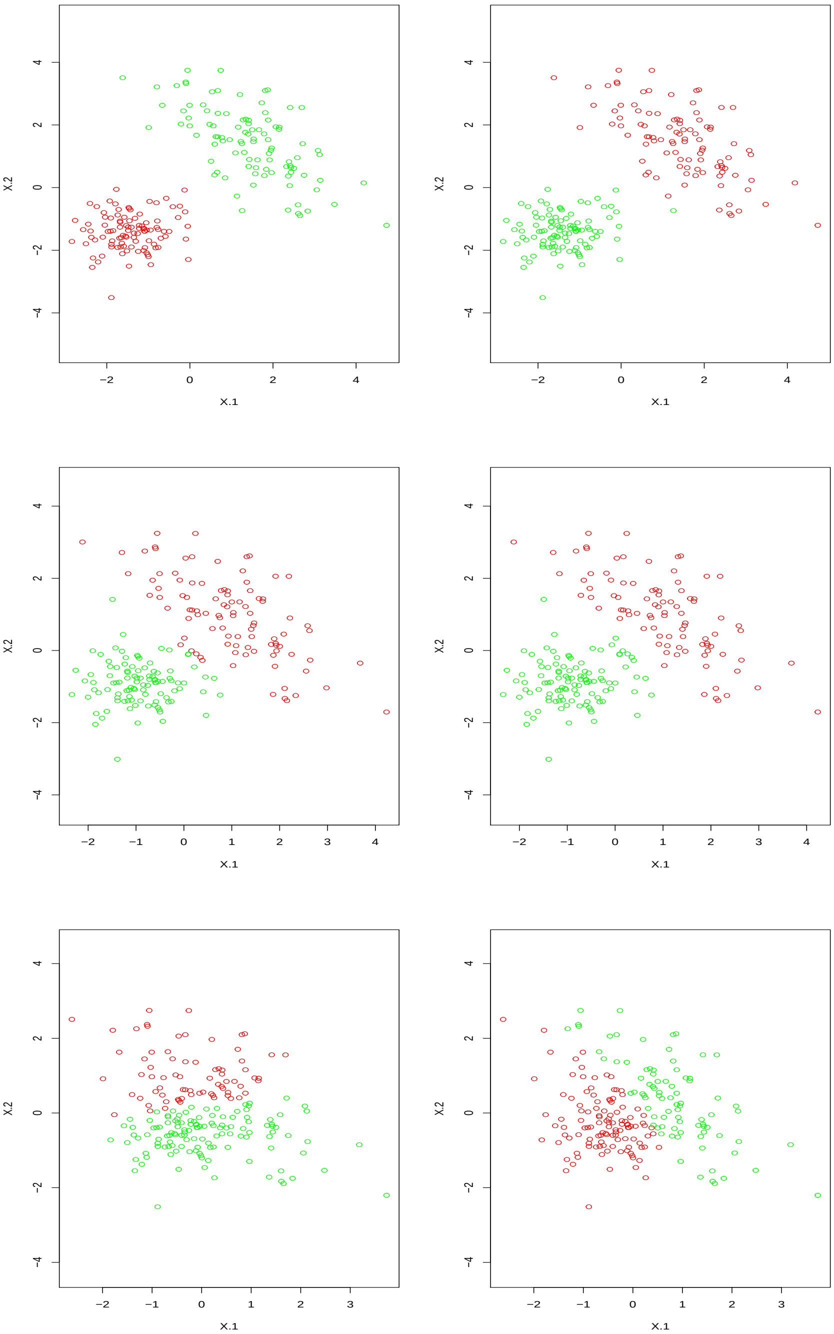


FIGURE 1 – Partitions obtenues sur `Synth1` (haut), `Synth2` (milieu), et `Synth3` (bas).

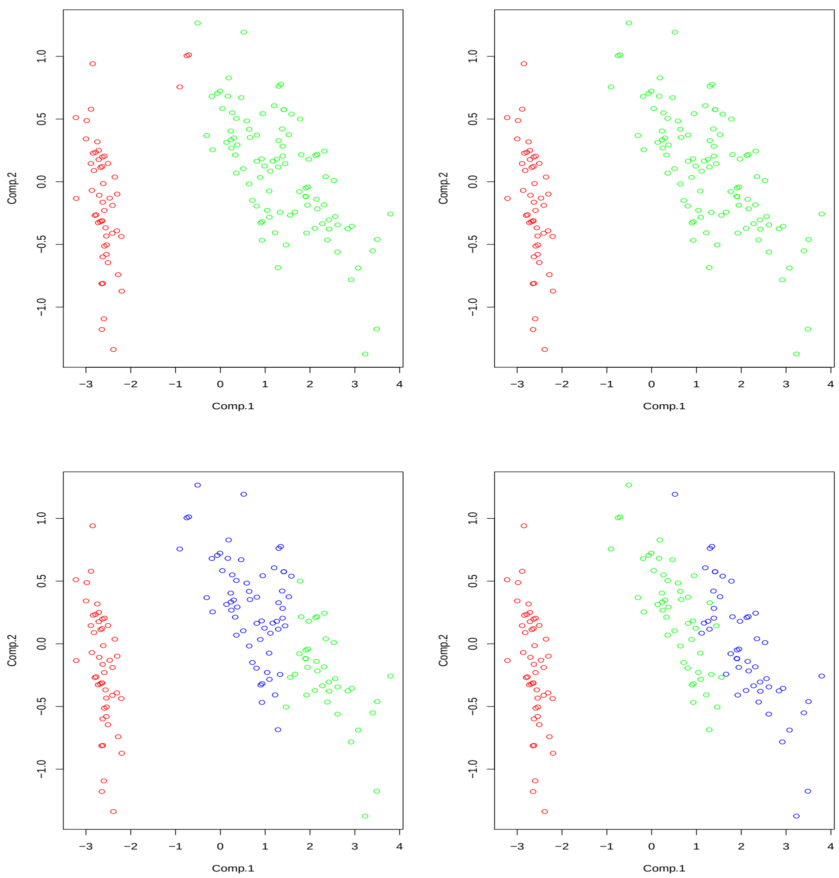


FIGURE 2 – Partitions obtenues sur les Iris avec les K-means (gauche) et les K-means adaptatifs (droite), pour $K = 2$ (haut) et $K = 3$ (bas).

Crabs Déterminer une classification des données `Crabs` avec la méthode des K-means classique, puis avec les K-means adaptatifs, pour $K = 2, 3, \ldots, 8$. Calculer la valeur du critère optimisé (inertie dans le premier cas, distance totale dans le second) pour $K = 1$. Afficher les valeurs de critère en fonction de K. Conclure quant au nombre de classes qui semble le plus raisonnable.

Effectuer une partition des données pour $K = 2$, puis pour $K = 4$. Comparer les résultats obtenus par les deux méthodes ; interpréter.

Classification pixellaire On souhaite effectuer la classification pixellaire [1] d'une image, que l'on pourra charger et afficher au moyen du code suivant :

```
> library(png)
> toucan.im <- readPNG("donnees/toucan.png", native=FALSE)
>
> plot(NA, xlim=c(0, nrow(toucan.im)), ylim=c(0, ncol(toucan.im)), xlab="",
  ↪ ylab="",
>      main="original image")
> rasterImage(toucan.im, 0, 0, nrow(toucan.im), ncol(toucan.im))
```

On transformera l'image en un tableau individus-variables, chaque individu correspondant à un pixel et chaque variable à un niveau de couleur (normalisé entre 0 et 1) :

```
> toucan.mat <- matrix(toucan.im, ncol=dim(toucan.im)[3])
```

On effectuera une classification pixellaire avec les K-means et les K-means adaptatifs, pour un nombre de clusters variant entre $K = 2$ et $K = 5$, en effectuant à chaque fois plusieurs essais (à partir d'initialisations différentes). On prendra soin de comparer les résultats donnés par les deux algorithmes.

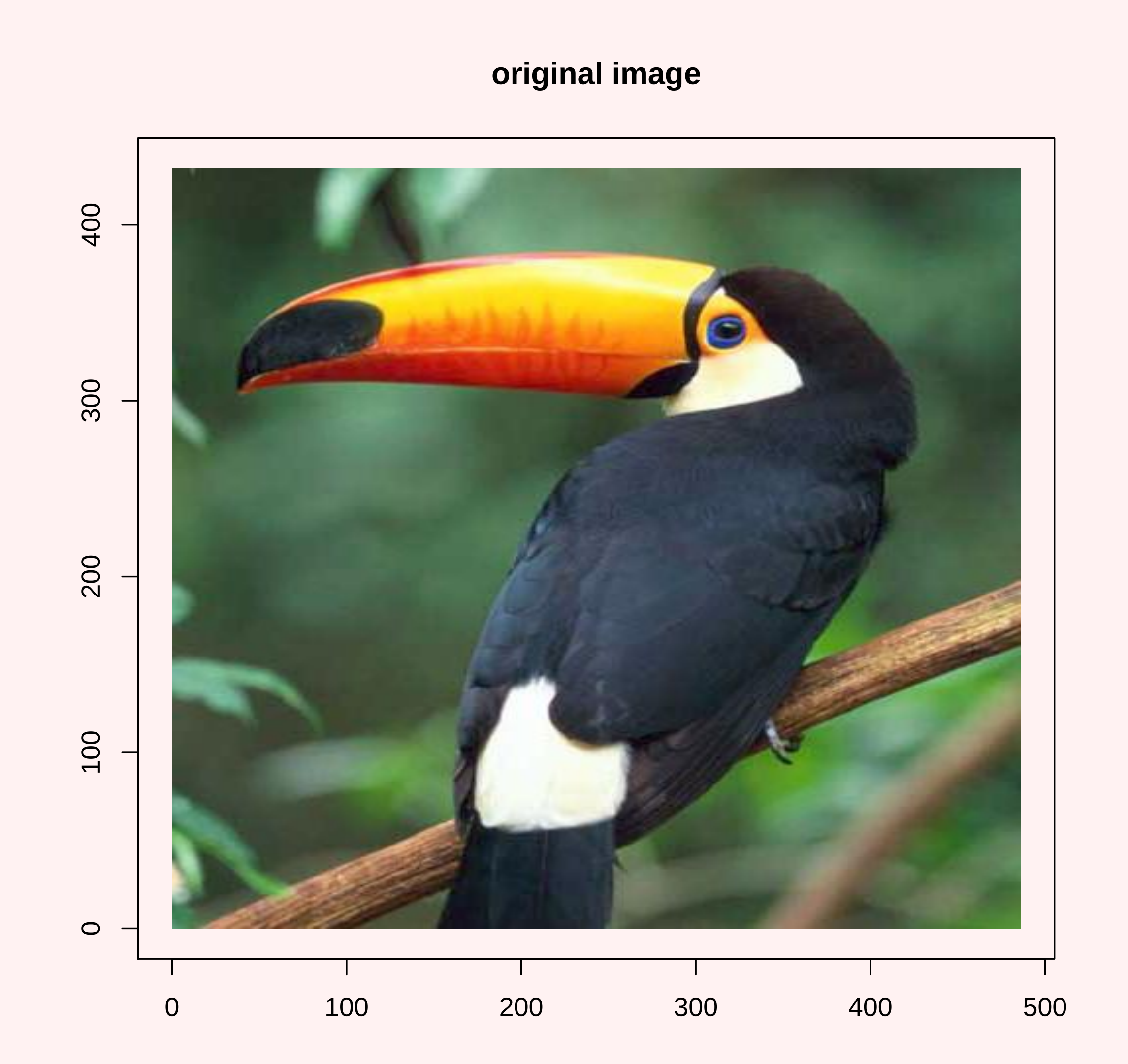
Une fois la classification obtenue, on pourra remplacer chaque pixel par le centre de sa classe et reconstituer ainsi une image simplifiée comme suit (si l'on suppose que les résultats des K-means et des K-means adaptatifs sont respectivement stockés dans les variables `toucan.km` et `toucan.akm`) :

```
> toucan.km.im <- array(toucan.km$centers[toucan.km$cluster],
  ↪ dim=dim(toucan.im))
> toucan.akm.im <- array(toucan.akm$centers[toucan.akm$cluster],
  ↪ dim=dim(toucan.im))
```

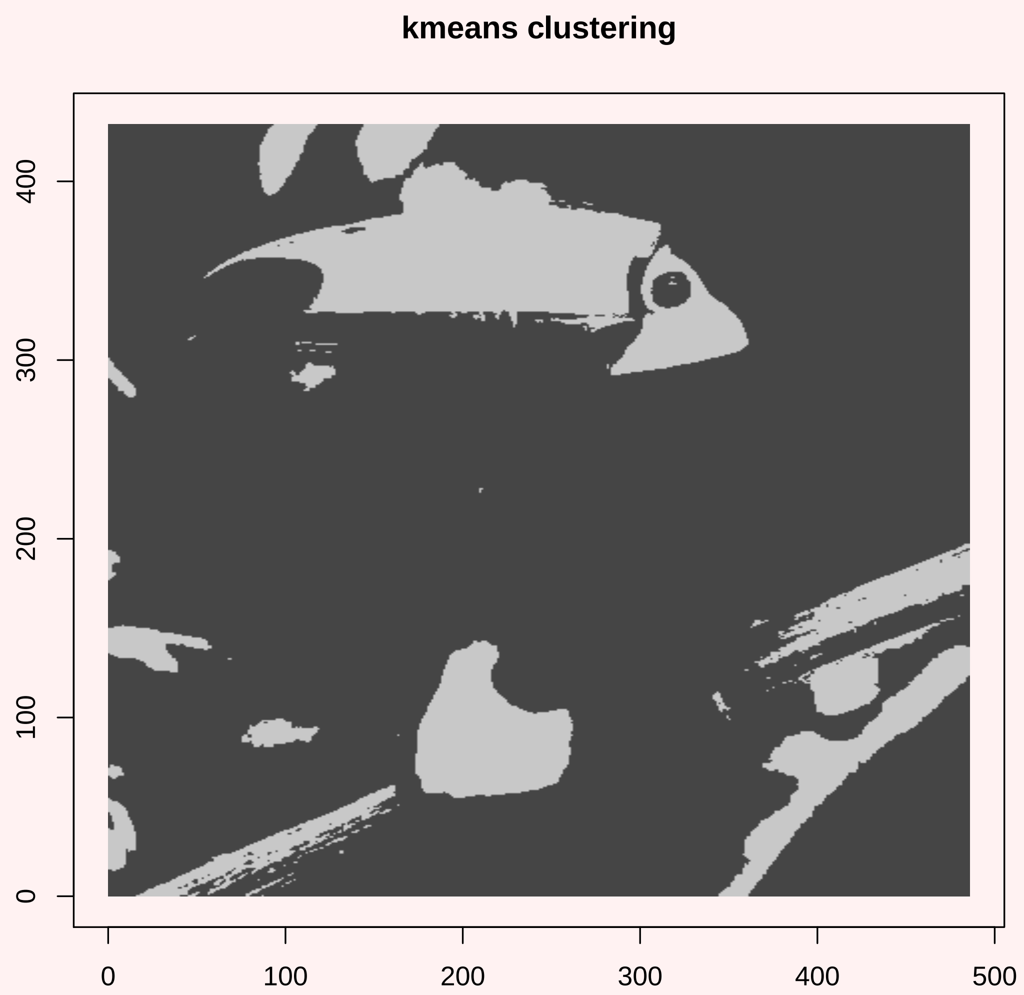
On pourra ensuite afficher ces images comme précédemment.

```
rm(list = ls())
source("adaptivekm.r")
source("fonctions/distXY.r")
library(MASS)
library(png)
nCl <- 2
toucan.im <- readPNG("donnees/toucan.png", native=FALSE)
toucan.mat <- matrix(toucan.im, ncol=dim(toucan.im)[3])
toucan.km <- kmeans(toucan.mat, centers=nCl, nstart=10)
toucan.akm <- adaptivekm(toucan.mat, K=nCl, nstart=10)
toucan.km.im <- array(toucan.km$centers[toucan.km$cluster], dim=dim(toucan.im))
toucan.akm.im <- array(toucan.akm$centers[toucan.akm$cluster], dim=dim(toucan.im))
plot(NA, xlim=c(0, nrow(toucan.im)), ylim=c(0, ncol(toucan.im)), xlab="", ylab="",
        main="original image")
rasterImage(toucan.im, 0, 0, nrow(toucan.im), ncol(toucan.im))
```

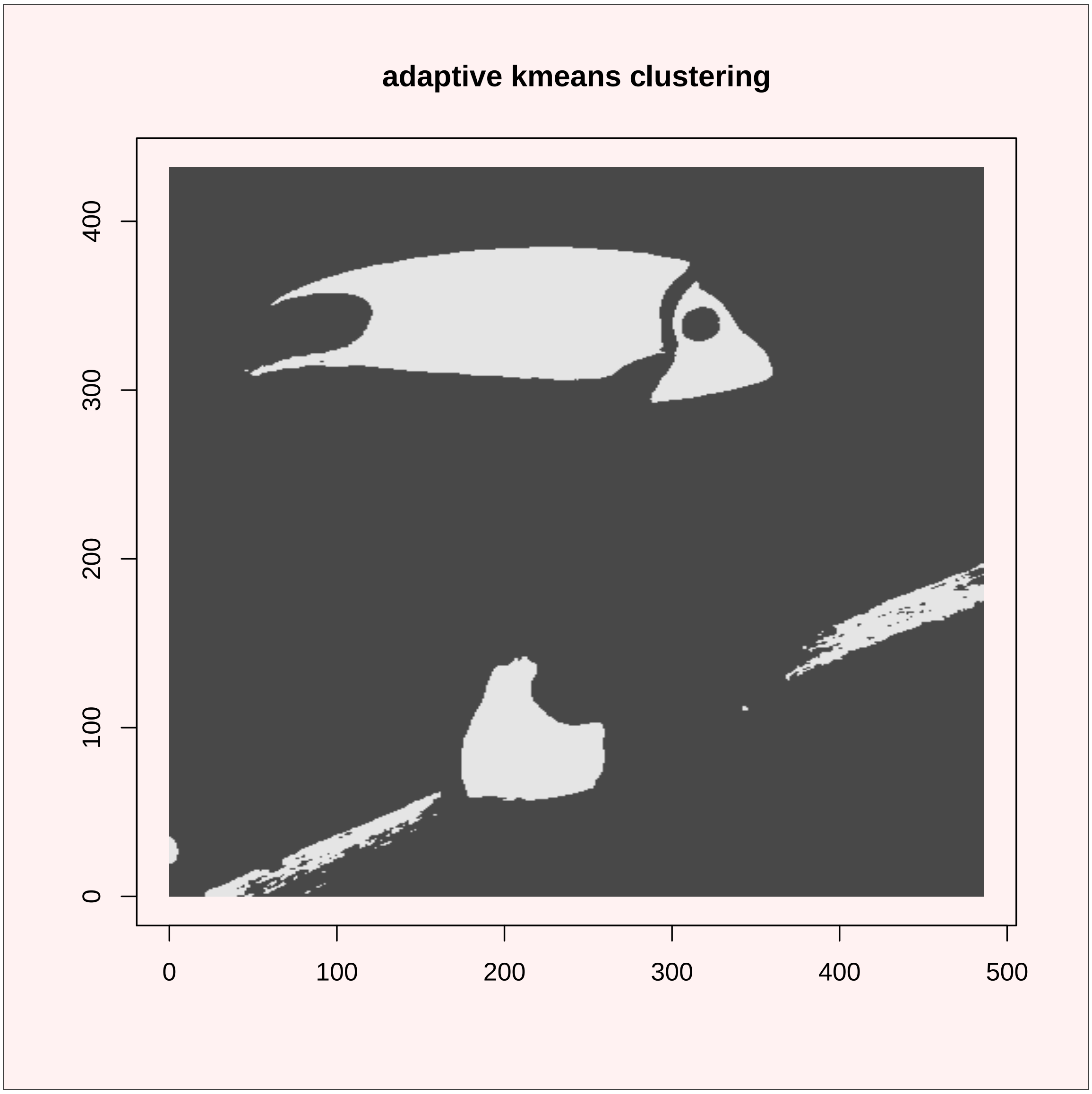
1. La classification pixellaire ne tient pas compte de l'information de voisinage des pixels.




```r
plot(NA, xlim=c(0, nrow(toucan.im)), ylim=c(0, ncol(toucan.im)), xlab="", ylab="",
     main="kmeans clustering")
rasterImage(toucan.km.im, 0, 0, nrow(toucan.im), ncol(toucan.im))
```




```
plot(NA, xlim=c(0, nrow(toucan.im)), ylim=c(0, ncol(toucan.im)), xlab="", ylab="",
        main="adaptive kmeans clustering")
rasterImage(toucan.akm.im, 0, 0, nrow(toucan.im), ncol(toucan.im))
```



4 Preuves de convergence

L'objectif de cette partie est de justifier l'algorithme d'un point de vue théorique.

Plus particulièrement, supposons que l'on dispose d'une partition des données $P = (P_1, \ldots, P_K)$, où la classe de chaque individu est codée par une variable binaire

$$z_{ik} = \begin{cases} 1 & \text{si } \boldsymbol{x}_i \in P_k, \\ 0 & \text{sinon.} \end{cases}$$

On cherche à caractériser chaque classe par un prototype $\boldsymbol{v}_k$ et une métrique définie par une matrice M_k. On souhaite montrer que la minimisation du critère de distance

$$J\left(\{v_k, M_k\}_{k=1,\ldots,K}\right) = \sum_{k=1}^{K} \sum_{i=1}^{n} z_{ik} d_{ik}^2 \tag{2}$$

sous la contrainte $\det M_k = \rho_k$ pour tout $k = 1, \ldots, K$, où la distance $d_{ik}^2 = d_{M_k}^2(\boldsymbol{x}_i, \boldsymbol{v}_k)$ est définie par l'équation (1), donne

$$\boldsymbol{v}_k = \overline{\boldsymbol{x}}_k = \frac{1}{n_k}\sum_{i=1}^{n} z_{ik}\boldsymbol{x}_i, \tag{3}$$

$$M_k^{-1} = (\rho_k \det V_k)^{-1/p} V_k \text{ , avec } V_k = \frac{1}{n_k}\sum_{i=1}^{n} z_{ik}\left(\boldsymbol{x}_i - \boldsymbol{v}_k\right)\left(\boldsymbol{x}_i - \boldsymbol{v}_k\right)^T . \tag{4}$$

À chaque itération, les prototypes des classes sont donc obtenus en calculant les centres de gravité, et les métriques sont définies par les (inverses des) matrices de covariance normalisées.

Remarquons que fixer $\det M_k = \rho_k$ pour tout $k = 1, \ldots, K$ revient à fixer le volume de chaque classe : si l'on n'impose pas cette contrainte, il peut arriver qu'une ou plusieurs classes « absorbent » tous les points au détriment d'une ou plusieurs autres, ce qui cause des problèmes numériques, la matrice de covariance d'une classe d'effectif faible pouvant être de rang incomplet (et donc non inversible).

On admettra que pour minimiser le critère défini par (2)-(1) par rapport à $\boldsymbol{v}_k$ et M_k sous la contrainte $\det M_k = \rho_k$, on peut calculer le lagrangien

$$\mathcal{L}\left(\{v_k, M_k, \lambda_k\}_{k=1,\ldots,K}\right) = J\left(\{v_k, M_k\}_{k=1,\ldots,K}\right) - \sum_{k=1}^{K} \lambda_k(\det M_k - \rho_k), \tag{5}$$

où λ_k est le multiplicateur de Lagrange de la k^{e} contrainte (c'est-à-dire $\det M_k = \rho_k$), et de le minimiser par rapport à λ_k, $\boldsymbol{v}_k$ et M_k : pour cela, il faut vérifier conjointement

$$\frac{\partial \mathcal{L}\left(\{\boldsymbol{v}_k, M_k, \lambda_k\}\right)}{\partial \lambda_k} = 0, \quad \frac{\partial \mathcal{L}\left(\{\boldsymbol{v}_k, M_k, \lambda_k\}\right)}{\partial M_k} = \mathbb{0}, \quad \frac{\partial \mathcal{L}\left(\{\boldsymbol{v}_k, M_k, \lambda_k\}\right)}{\partial \boldsymbol{v}_k} = \mathbb{0}.$$

On définira la dérivée $\partial f(M)/\partial M$ d'une fonction $f(M)$ par rapport à une matrice M de terme général m_{ij} comme la matrice B de terme général $b_{ij} = \partial f(M)/\partial m_{ij}$.

Il faudrait de surcroît montrer que la matrice des dérivées secondes (matrice hessienne) est définie positive pour les valeurs de paramètres qui annulent le vecteur des dérivées premières (vecteur gradient). On pourra admettre ce résultat.

Questions Pour le calcul des dérivées, on pourra s'aider du « Matrix Cookbook », disponible à l'URL : http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3274/pdf/imm3274.pdf.

1. Calculer $\partial \mathcal{L}\left(\{\boldsymbol{v}_k, M_k, \lambda_k\}\right)/\partial \boldsymbol{v}_k$; montrer que la mise à jour des prototypes revient à calculer les centres de gravité des classes définis par l'équation (3).

   La dérivée par rapport à $\boldsymbol{v}_k$ est aisément obtenue :

   $$\begin{aligned}\frac{\partial \mathcal{L}\left(\{\boldsymbol{v}_k, M_k, \lambda_k\}\right)}{\partial \boldsymbol{v}_k} &= \sum_{i=1}^{n} z_{ik}\frac{\partial d_{ik}^2}{\partial \boldsymbol{v}_k}, \\ &= -2M_k^{-1}\sum_{i=1}^{n} z_{ik}(\boldsymbol{x}_i - \boldsymbol{v}_k);\end{aligned}$$

   l'annulation de ce vecteur de dérivées premières donne

   $$\frac{\partial \mathcal{L}\left(\{\boldsymbol{v}_k, M_k, \lambda_k\}\right)}{\partial \boldsymbol{v}_k} = \mathbb{0} \Leftrightarrow \sum_{i=1}^{n} z_{ik}(\boldsymbol{x}_i - \boldsymbol{v}_k) = \mathbb{0} \Leftrightarrow \boldsymbol{v}_k = \frac{1}{n_k}\sum_{i=1}^{n} z_{ik}\boldsymbol{x}_i = \overline{\boldsymbol{x}_k},$$

   où $n_k = \sum_{i=1}^{n} z_{ik}$.

2. Calculer $\partial \mathcal{L}\left(\{v_k, M_k, \lambda_k\}\right)/\partial M_k$, puis $\partial \mathcal{L}\left(\{v_k, M_k, \lambda_k\}\right)/\partial \lambda_k$. Montrer que la mise à jour des matrices M_k définissant la métrique spécifique à chaque classe revient à calculer les inverses des matrices de covariance empiriques normalisées définies par l'équation (4) (plus difficile !).

Commençons par annuler la dérivée par rapport à λ_k :

$$\frac{\partial \mathcal{L}\left(\{\boldsymbol{v}_k, M_k, \lambda_k\}\right)}{\partial \lambda_k} = 0 \Leftrightarrow \det M_k - \rho_k = 0 \Leftrightarrow \det M_k = \rho_k.$$

Le calcul de la matrice des dérivées secondes par rapport aux éléments de M_k donne

$$\begin{aligned}
\frac{\partial \mathcal{L}\left(\{\boldsymbol{v}_k, M_k, \lambda_k\}\right)}{\partial M_k} &= \sum_{i=1}^{n} z_{ik} \frac{\partial d_{ik}^2}{\partial M_k} - \lambda_k \frac{\partial \det M_k}{\partial M_k} \\
&= \sum_{i=1}^{n} z_{ik} (\boldsymbol{x}_i - \boldsymbol{v}_k)(\boldsymbol{x}_i - \boldsymbol{v}_k)^T - \lambda_k \det M_k (M_k^{-1})^T.
\end{aligned}$$

L'annulation conjointe des dérivées premières par rapport à λ_k et M_k donne

$$\begin{aligned}
\left\{\frac{\partial \mathcal{L}}{\partial \lambda_k} = 0, \frac{\partial \mathcal{L}}{\partial M_k} = 0\right\} &\Leftrightarrow \sum_{i=1}^{n} z_{ik} (\boldsymbol{x}_i - \boldsymbol{v}_k)(\boldsymbol{x}_i - \boldsymbol{v}_k)^T = \lambda_k \rho_k (M_k^{-1})^T \\
&\Leftrightarrow M_k^{-1} = \lambda_k^{-1} \rho_k^{-1} \sum_{i=1}^{n} z_{ik} (\boldsymbol{x}_i - \boldsymbol{v}_k)(\boldsymbol{x}_i - \boldsymbol{v}_k)^T \\
&\Leftrightarrow M_k^{-1} = \lambda_k^{-1} \rho_k^{-1} n_k V_k.
\end{aligned}$$

Il reste à déterminer la valeur du multiplicateur de Lagrange λ_k ; on calcule pour cela $\det M_k$:

$$\begin{aligned}
\det M_k &= \det\left((\lambda_k^{-1} \rho_k^{-1} n_k V_k)^{-1}\right) \\
&= \det\left(\lambda_k \rho_k n_k^{-1} V_k^{-1}\right) \\
&= \lambda_k^p \rho_k^p n_k^{-p} \det\left(V_k^{-1}\right);
\end{aligned}$$

cela donne

$$\lambda_k = \left(\rho_k^{-p} \left(\det\left(V_k^{-1}\right)\right)^{-1} \det M_k n_k^p\right)^{1/p} = \left(\rho_k^{1-p} n_k^p \det V_k\right)^{1/p} = \rho_k^{1/p-1} n_k \left(\det V_k\right)^{1/p},$$

ce qui, utilisé dans l'expression précédente de M_k^{-1}, donne

$$\left\{\frac{\partial \mathcal{L}}{\partial \lambda_k} = 0, \frac{\partial \mathcal{L}}{\partial M_k} = 0\right\} \Leftrightarrow M_k^{-1} = \rho_k^{1-1/p} n_k^{-1} \left(\det V_k\right)^{-1/p} \rho_k^{-1} n_k V_k = \rho_k^{-1/p} \left(\det V_k\right)^{-1/p} V_k.$$

Données : Tableau de données X (matrice $n \times p$), nb. de classes K (entier), volumes des classes ρ_k (vecteur de longueur K), nb. max. d'itérations $n_{\text{iter.max}}$ (entier), nb. d'essais n_{start} (entier), tolérance ε pour vérifier la convergence (réel positif)

Résultat : valeur du critère, nb. d'itérations, partition des données, centres des classes, matrices de covariance empiriques normalisées des classes

pour $i = 1, 2, \ldots, n_{\text{start}}$ **faire**

- initialiser μ_k et $\tilde{V}_k$, pour $k = 1, \ldots, K$
- **tant que** la convergence n'est pas atteinte et $n_{\text{iter.max}}$ n'est pas dépassé **faire**
  - mettre à jour la partition des données : affecter chacun des points de X au centre μ_k le plus proche, au sens de la distance de Mahalanobis
  - mémoriser les anciens centres des classes : $\mu_{k,\text{prec}} \longleftarrow \mu_k$
  - mettre à jour les centres des classes μ_k et les matrices de covariance empiriques $\tilde{V}_k$ normalisées au moyen de la partition calculée précédemment
  - mettre à jour le critère de distance d_{tot} (somme des distances aux centres des classes)
  - mettre à jour le critère de convergence
- **fin**
- **si** $J^\star < J^{\text{opt}}$ (avec J^{opt} le meilleur optimum local donné par les essais précédents) **alors**
  - mettre à jour le meilleur optimum local : mémoriser la valeur du critère $J^\star$, le nb. d'itérations, la partition des données, les centres et les matrices de covariance
- **fin**

fin

Algorithme 1 : Algorithme des K-means avec distance adaptative