Université de Technologie de Compiègne
SY09 : Analyse des données et Apprentissage automatique

SY09 : TP 07 : Éléments de théorie de la décision

Julien Jerphanion

Printemps 2018

Table des matières

1 Questions théoriques

On considérera un problème (simplifié) d'identification de produits chimiques à partir de leur temps de dégradation. Plus particulièrement, on supposera être en présence de $g = 2$ produits, présents en proportions initiales π_1 (classe ω_1), et π_2 (classe ω_2). On suppose pouvoir tester le temps de dégradation des produits selon deux protocoles ; on notera X_1 et X_2 les temps de dégradation selon le premier et le second protocole, respectivement. Pour un même produit, on supposera indépendants ces temps X_1 et X_2 mesurés par chacun des deux protocoles. On suppose en outre que les distributions de temps de dégradation sont modélisés par des lois exponentielles

$$\begin{array}{ll} X_1 \underset{\omega_1}{\sim} \mathcal{E}(\lambda_1) & X_2 \underset{\omega_1}{\sim} \mathcal{E}(\lambda_2) \\ X_1 \underset{\omega_2}{\sim} \mathcal{E}(\theta_1) & X_2 \underset{\omega_2}{\sim} \mathcal{E}(\theta_2) \end{array}$$

1.1 Densité jointe du vecteur $\mathbf{X} = (X_1, X_2)^T$

On suppose ici que l'on ne s'intéresse qu'à un seul type de produit ; la variable latente associée à X_1 X_2 leur est donc commune.

On a par independance de ces deux variables la densité jointe:

$$f : (x_1, x_2) \mapsto \lambda_1\lambda_2 \exp(-\Lambda^T\mathbf{x})\mathbf{1}_{[Z=\omega_1]} + \theta_1\theta_2 \exp(-\Theta^T\mathbf{x})\mathbf{1}_{[Z=\omega_2]}$$

1.2 Frontière de décision linéaire via la stratégie de Neyman-Pearson

Montrons que la frontière de décision est linéaire.

Pour cela, on commence par calculer le ratio des log-vraisemblances :

$$\frac{\mathcal{L}(Z = \omega_1, \mathbf{x})}{\mathcal{L}(Z = \omega_2, \mathbf{x})} = \frac{\lambda_1\lambda_2}{\theta_1\theta_2} \exp(-(\Lambda - \Theta)^T\mathbf{x})$$

On veut :

$$\frac{\mathcal{L}(Z = \omega_1, \mathbf{x})}{\mathcal{L}(Z = \omega_2, \mathbf{x})} \geq c \in \mathbb{R}$$

Ce qui nous donne en déroulant:

$$(\Lambda - \Theta)^T\mathbf{x} \leq -\log\left(c\,\frac{\theta_1\theta_2}{\lambda_1\lambda_2}\right)$$

Dans le cas de l'égalité on obtient l'équation d'un hyperplan:

$$\begin{cases} (\Lambda - \Theta)^T\mathbf{x} & = C \\ C & = -\log\left(c\,\frac{\theta_1\theta_2}{\lambda_1\lambda_2}\right) \in \mathbb{R} \end{cases}$$

1.3 Frontière de décision via la stratégie de Bayes

On se tourne ici vers la stratégie de Bayes:

$$\frac{\mathbb{P}(Z=\omega_1|X=\mathbb{x})}{\mathbb{P}(Z=\omega_2|X=\mathbb{x})} \geq 1$$

On rappelle que l'on a :

$$\mathbb{P}(Z=\omega_l|X=\mathbb{x}) = \frac{\mathbb{P}(X=\mathbf{x}|Z=\omega_l)\ \pi_l}{\sum_k \mathbb{P}(X=\mathbf{x}|Z=\omega_k)\ \pi_k}$$

Ainsi, on obtient :

$$\frac{\mathbb{P}(X=\mathbf{x}|Z=\omega_1)\ \pi_1}{\mathbb{P}(X=\mathbf{x}|Z=\omega_2)\ \pi_2} \geq 1$$

C'est à dire :

$$\frac{\lambda_1\lambda_2 \exp(-\Lambda^T\mathbf{x})}{\theta_1\theta_2 \exp(-\Theta^T\mathbf{x})} \geq \frac{\pi_2}{\pi_1}$$

Soit encore :

$$\exp(-(\Lambda-\Theta)^T\mathbf{x}) \geq \frac{\pi_2}{\pi_1}\frac{\theta_1\theta_2}{\lambda_1\lambda_2}$$

Et donc :

$$(\Lambda-\Theta)^T\mathbf{x} \leq -\log\left(\frac{\pi_2}{\pi_1}\frac{\theta_1\theta_2}{\lambda_1\lambda_2}\right)$$

On en déduit la règle de décision de Bayes :

$$\delta(\mathbf{x}) = \begin{cases}\omega_1 & \text{si} \quad (\Lambda-\Theta)^T\mathbf{x} < C \\ \omega_2 & \text{si} \quad (\Lambda-\Theta)^T\mathbf{x} > C\end{cases} \text{ ; avec } C = -\log\left(\frac{\pi_2}{\pi_1}\frac{\theta_1\theta_2}{\lambda_1\lambda_2}\right)$$

On s'intéresse donc ici aux projetés des $\mathbf{x}$ sur le vecteur $\Lambda-\Theta$ pour prendre une décision de classement.

2 Simulation

2.1 Implémentation du modèle génératif

On cherche à générer un échantillon de n suivant le modèle génératif décrit ci-dessus. Les données peuvent se générer ainsi :

```
generateData = function(n,pi,lambda,theta) {
  # Génération des appartenances aux populations
  n1 = rbinom(n=1,size = n,prob = pi[1])
  n2 = n - n1

  Z = array(0,c(n,1))
```

```
  Z[1:n1] = 1
  Z[(n1+1):n] = 2

  # Génération des vecteurs aléatoires x1,..., xn1
  X = array(data = 0,dim = c(n,2))
  X[1:n1,1] = rexp(n1,lambda[1])
  X[1:n1,2] = rexp(n1,lambda[2])

  # Génération des vecteurs aléatoires xn1+1,..., xn
  X[(n1+1):n,1] = rexp(n2,theta[1])
  X[(n1+1):n,2] = rexp(n2,theta[2])

  data = NULL

  data$X = X
  data$Z = Z
  data$n1 = n1
  data$n2 = n2
  data$n = n

  data
}
```

```
# Paramètres des lois
lambda = c(1,2)
theta = c(2,4)

# Probabilité à priori
pi = c(0.6,0.4)
n = 1000
data = generateData(n,pi,lambda, theta)

X = data$X
Z = data$Z
n1 = data$n1
n2 = data$n2
```

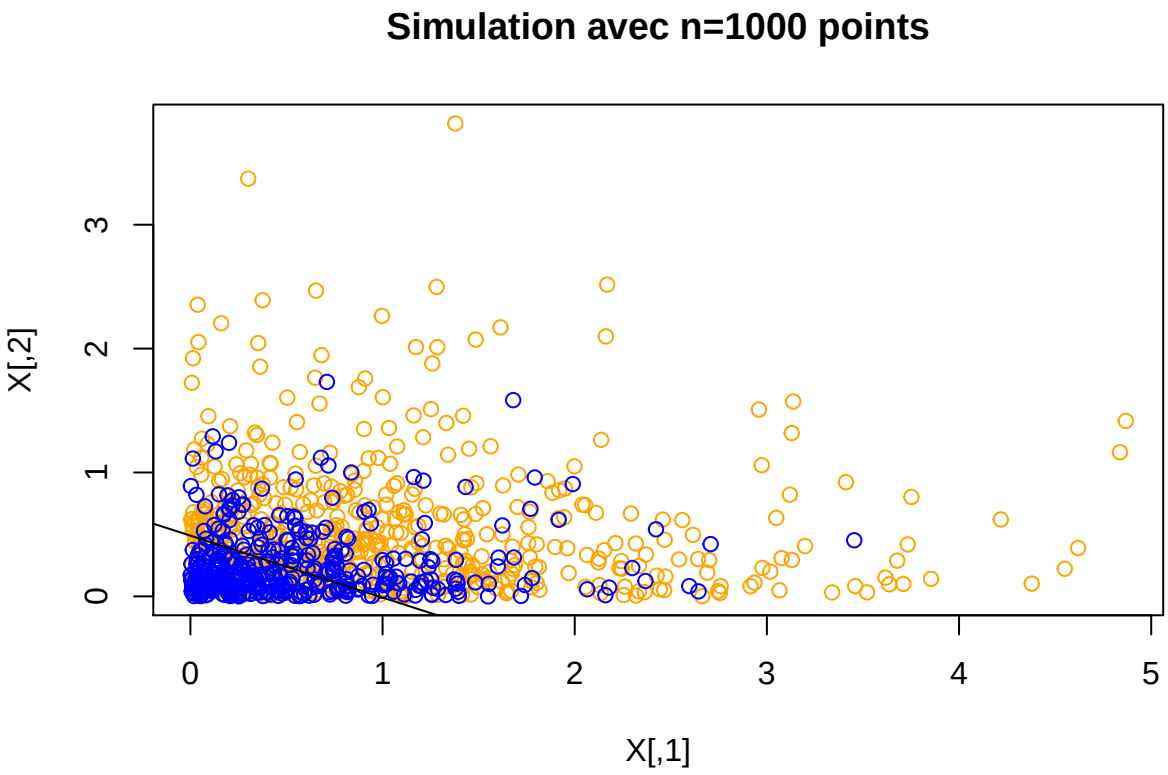
Traçons les individus selon leur classes et la frontière de décision associée au modèle posé.

```
front.bayes = function(X,Z,pi,lambda,theta) {
  C = - log(prod(theta)/prod(lambda) * pi[2]/pi[1])

  # Tracé de la droite
  vect = matrix(lambda - theta)

  a = - vect[1]/vect[2]
  b = C / vect[2]

  plot(X,col=c("orange","blue")[Z],main = paste("Simulation avec n=",n," points",sep = ""))
  abline(b,a)
}
front.bayes(X,Z,pi,lambda,theta)
```



2.2 Éstimation du taux d'erreur de Bayes

On peut maintenant estimer le taux d'erreur de Bayes ainsi :

```
C = - log(prod(theta)/prod(lambda) * pi[2]/pi[1])

vect = matrix(lambda - theta)
res = X %*% vect

# Classe w1 (z=0) si vect^T x =< C
inferedClass = (res > C) +1

# Erreur de classification
bayesError = sum(inferedClass != Z) / n

print(bayesError)
```

```
## [1] 0.328
```

```
tauxError1 = sum((inferedClass != Z)[Z==1]) / n1
tauxError2 = sum((inferedClass != Z)[Z==2]) / n2

print(tauxError1)
```

```
## [1] 0.2449664
```

```
print(tauxError2)
```

```
## [1] 0.450495
```

3 Comparaison avec K-means

On utilise maintenant le classifieur euclidien et le les K plus proches voisins avec le même jeu de données. On charge les différentes fonctions que l'on va utiliser dans la suite ainsi:

```
# Chargement des scripts
source("./fonctions.R")
```

Et, comme précédement, on peut chercher les erreurs de classements pour les deux méthodes.

```
errCeuc = errorsCeuc(X,Z)
errKppv = errorsKppv(X,Z)

print(paste("data","Ceuc","kppv",sep = "|"))
```

```
## [1] "data|Ceuc|kppv"
```

```
print(paste(" - Erreur Xapp", errCeuc$estErrApp,errKppv$estErrApp, sep = "|"))
```

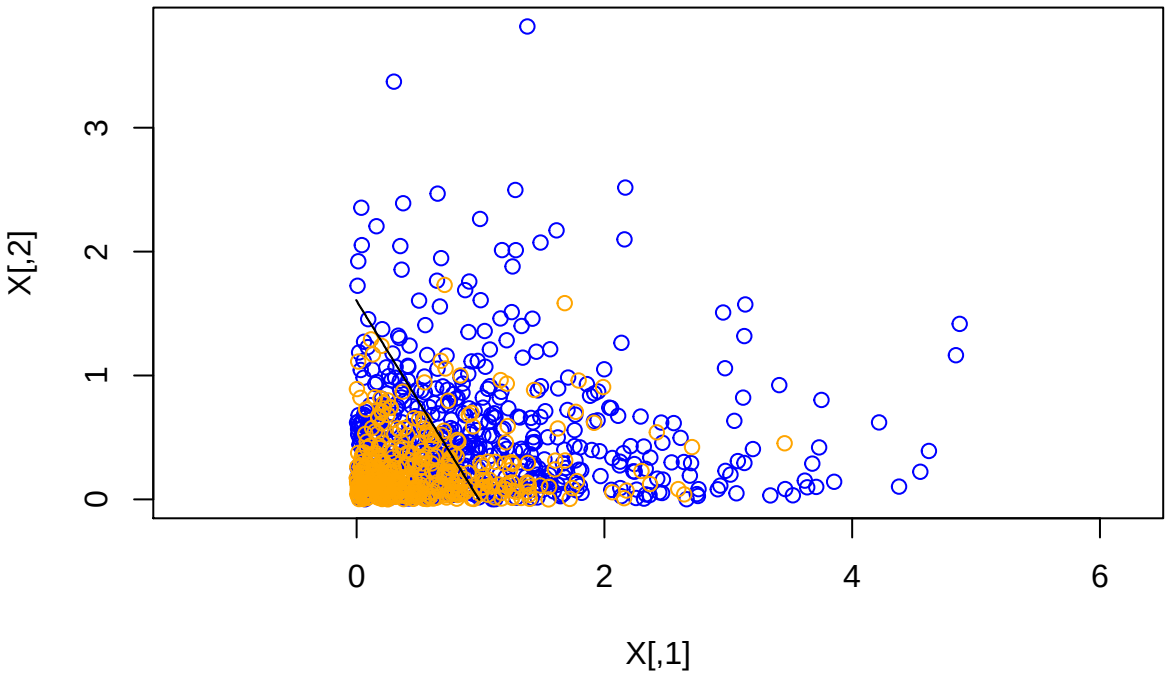
```
## [1] " - Erreur Xapp|0.370195195195195|0.2645"
```

```
print(paste(" - Erreur Xtst", errCeuc$estErrTest,errKppv$estErrTest, sep = "|"))
```

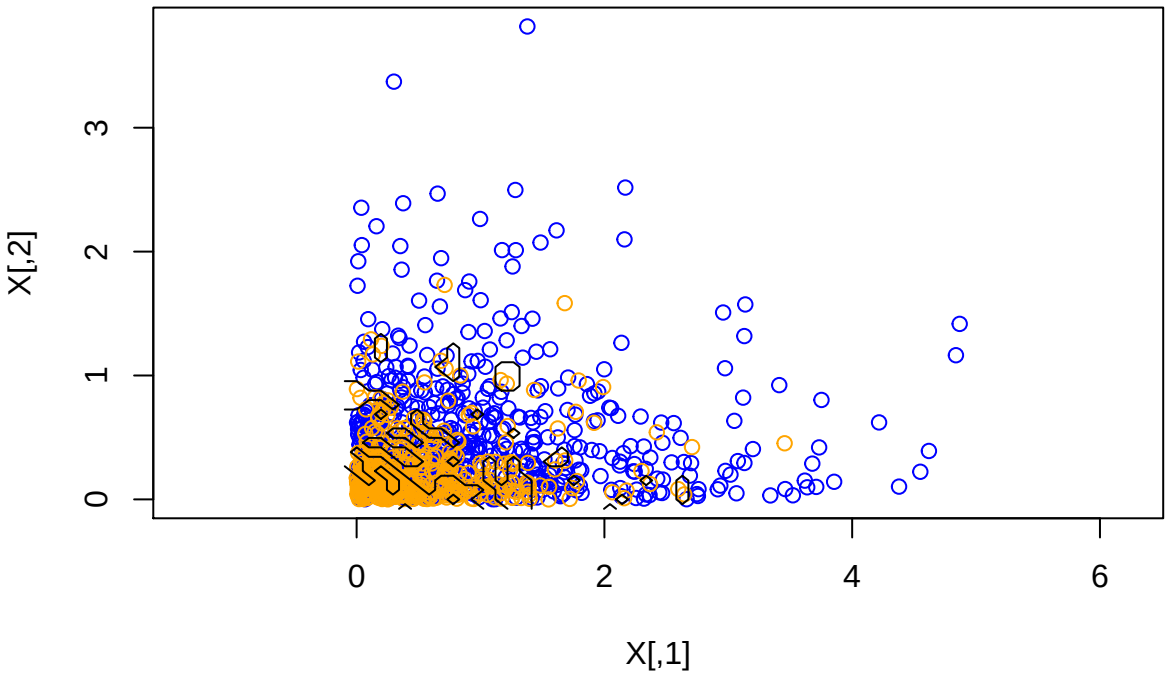
```
## [1] " - Erreur Xtst|0.372604790419162|0.3452"
```

On peut aussi s'amuser à tracer les frontières de décision.

```
mu = ceuc.app(X,Z)
front.ceuc(X,Z,mu,discretisation = 1000)
```




```
front.kppv(X,Z,3)
```



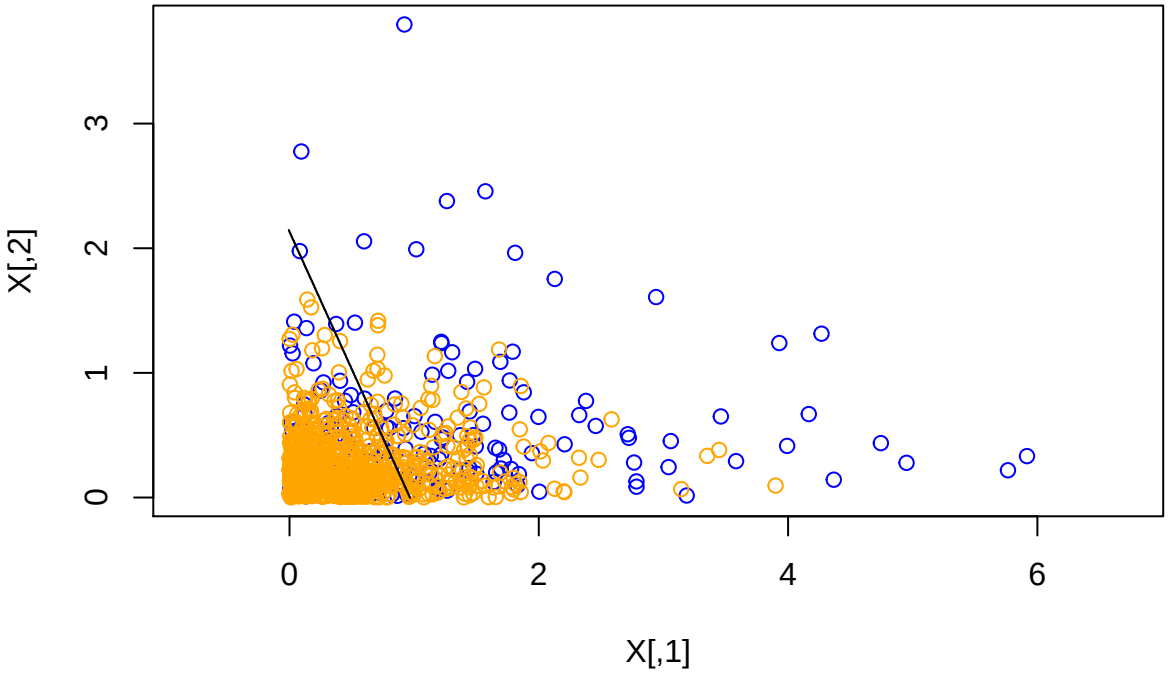
4 Bonus Track : Évolution de la frontière de décision

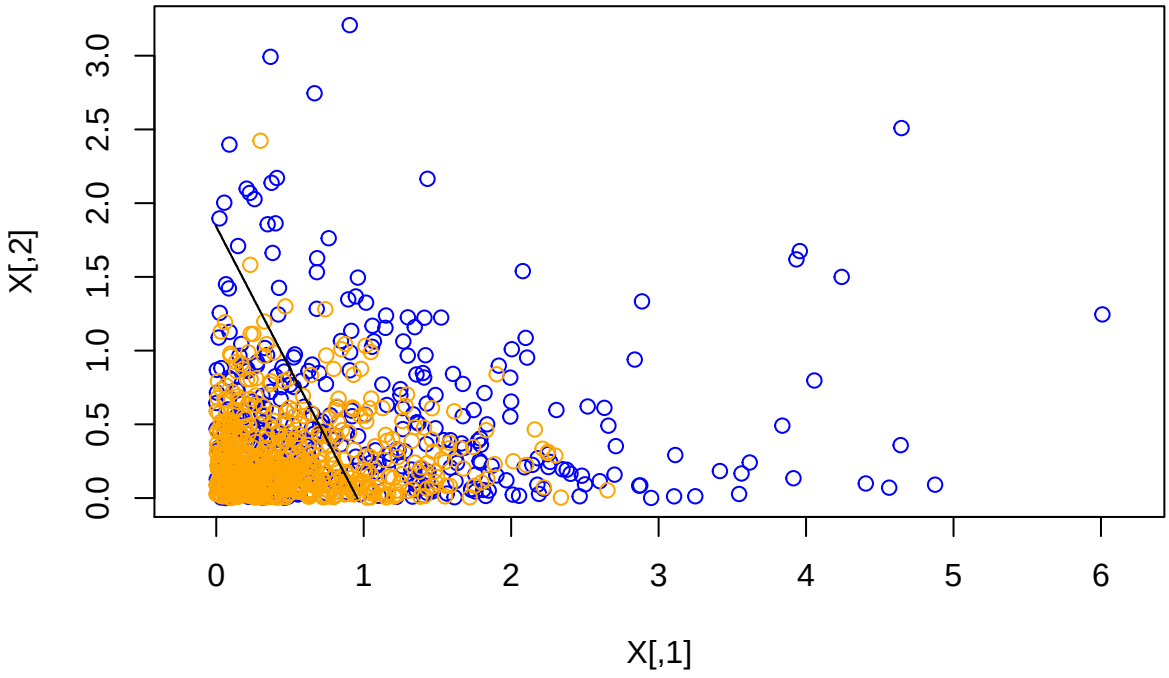
On peut essayer de s'intéresser à la variation des frontières de décisions en fonction de la variation du vecteur de proportion de classes π.

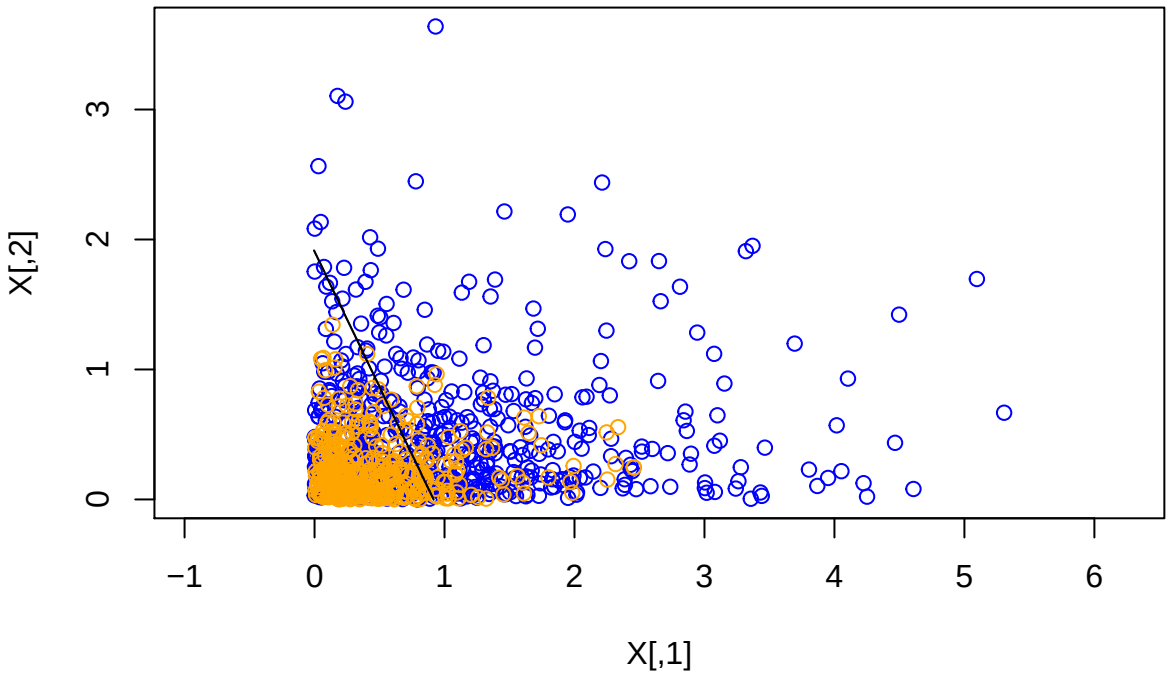
```
front.ceuc.piVariation = function(n,pi1,lambda,theta) {
  n_pi = length(pi1)
  for (i in 1:n_pi) {
    pi = c(pi1[i], 1- pi1[i])
    data = generateData(n,pi,lambda,theta)
    X = data$X
    Z = data$Z
    mu = ceuc.app(X,Z)
    front.ceuc(X,Z,mu,discretisation = 1000)
  }
}
```

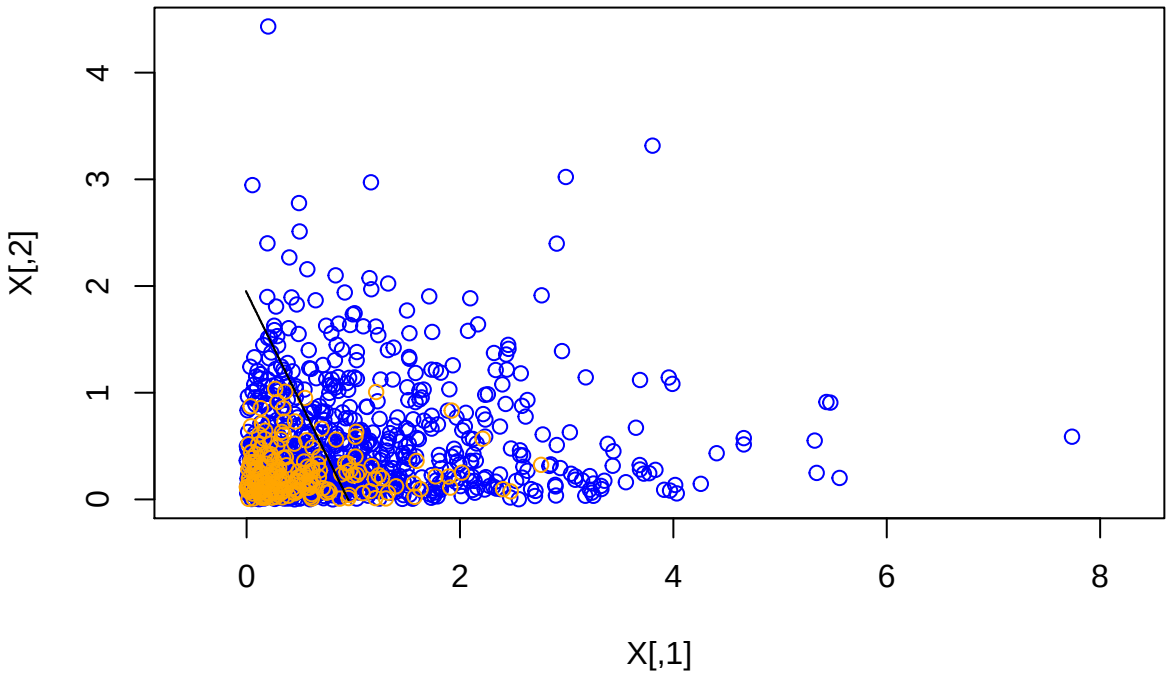
```
# Probabilité à priori
pi1 = c(0.2,0.4,0.6,0.8)
n = 1000

front.ceuc.piVariation(n,pi1,lambda,theta)
```


X[,2]
0
1
2
3
0
2
4
6
X[,1]

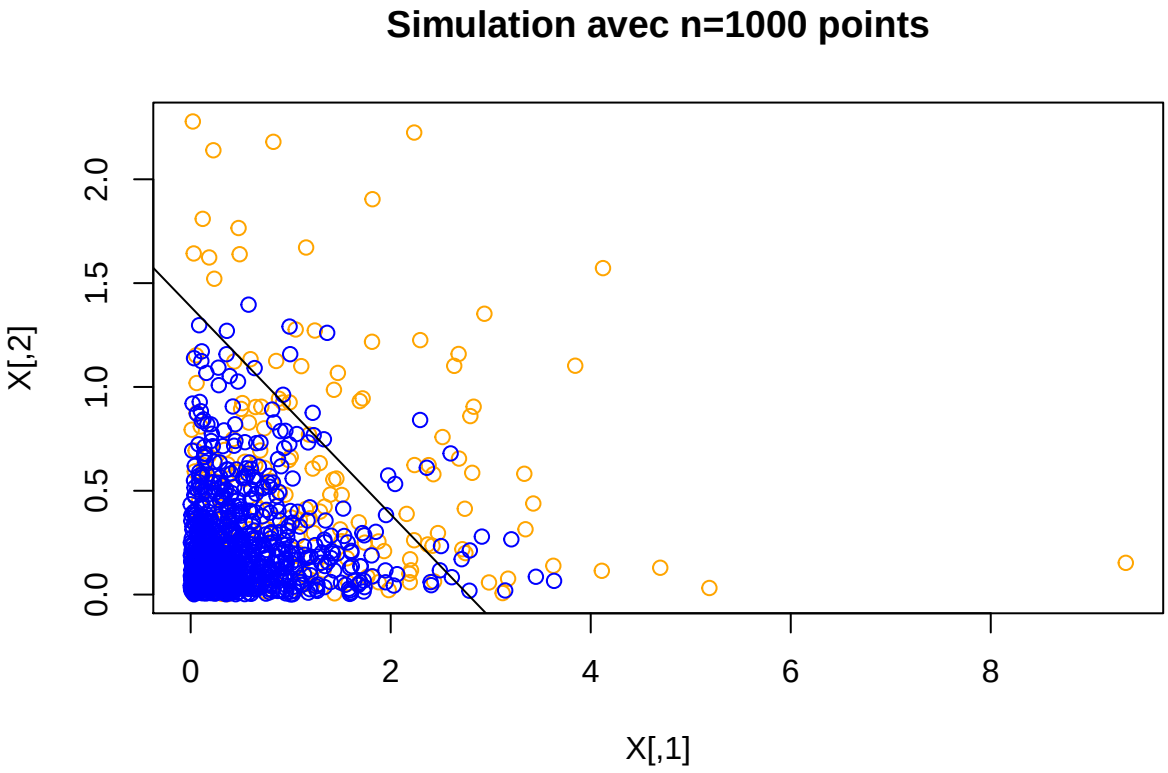

X[,2]
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
1
2
3
4
5
6
X[,1]

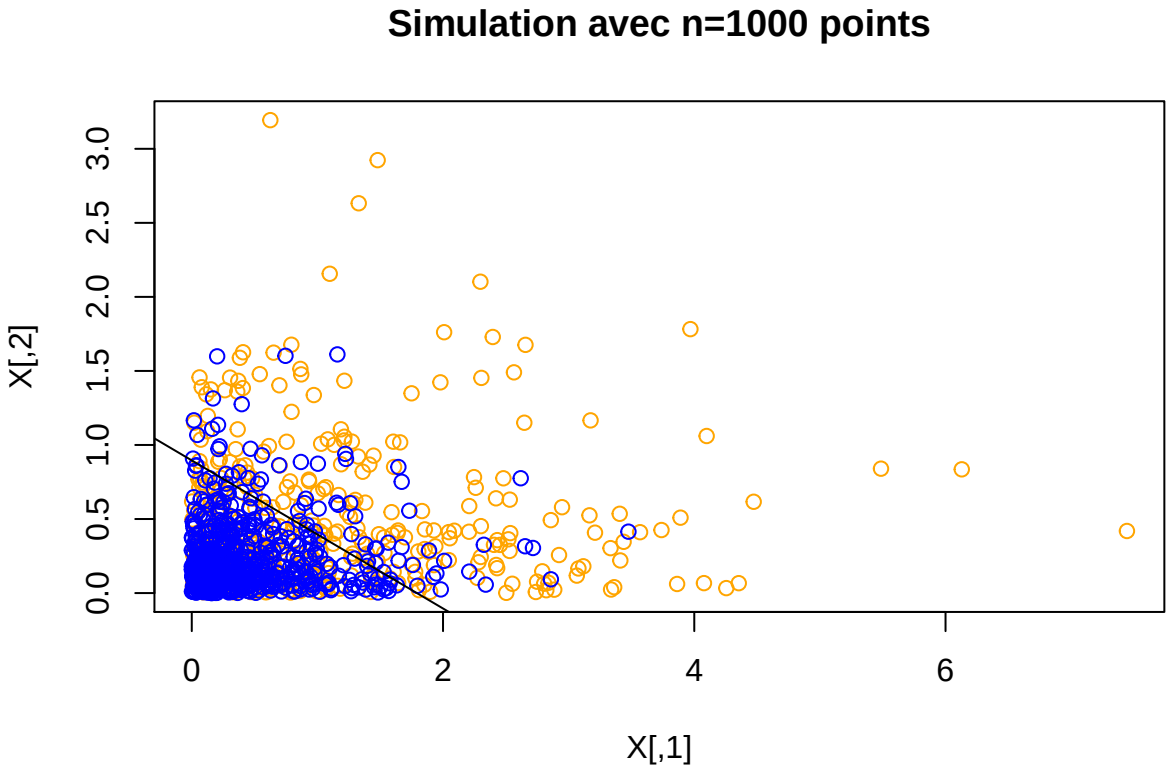

X[,2]
X[,1]
0
1
2
3
−1
0
1
2
3
4
5
6

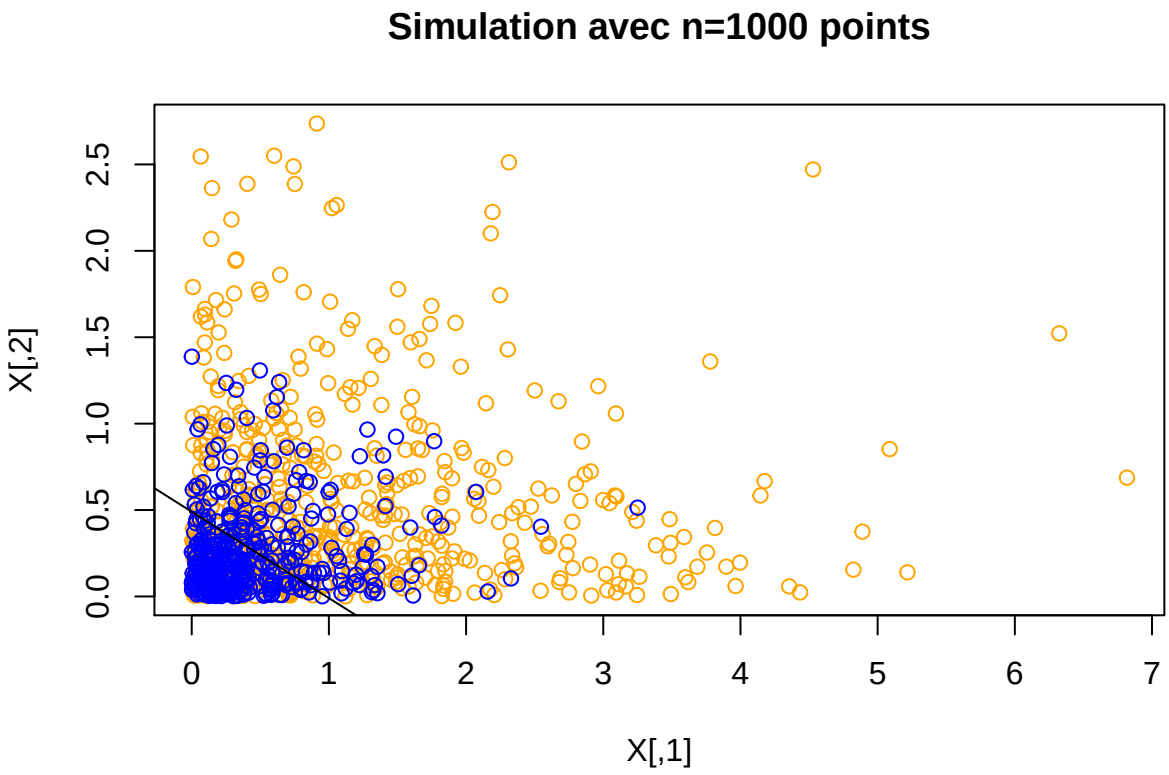


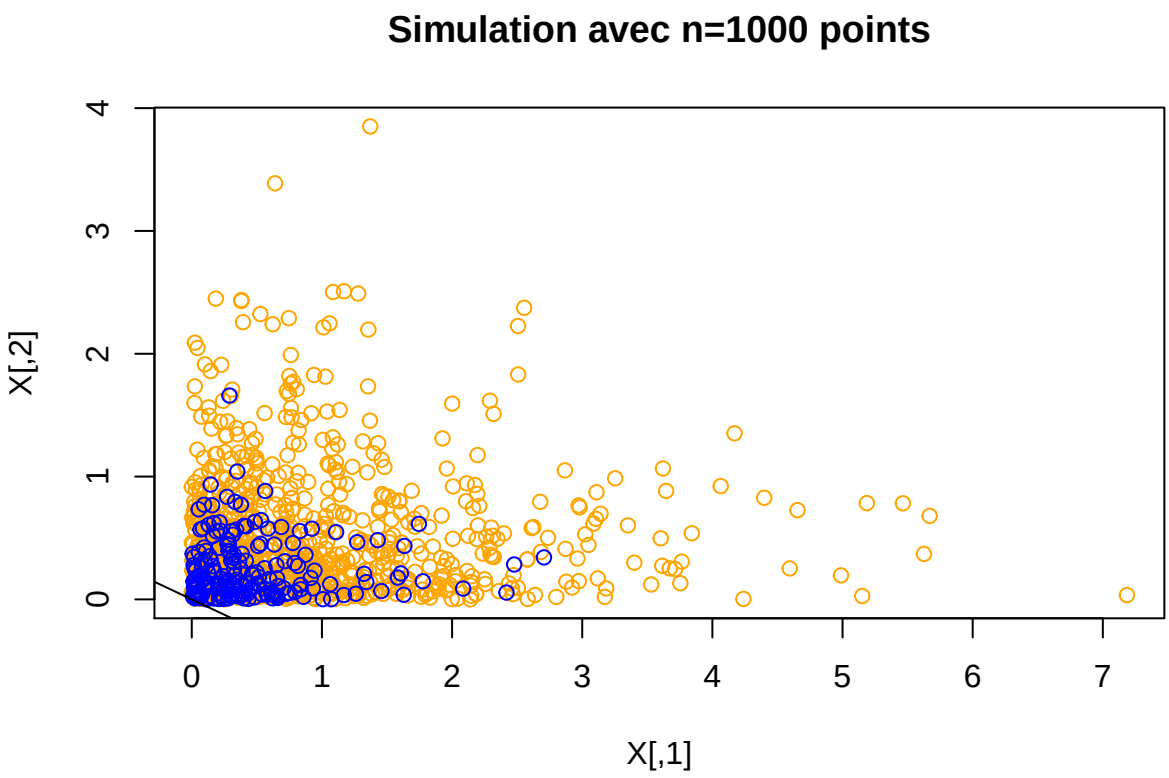

```
front.bayes.piVariation = function(n,pi1,lambda,theta) {
  n_pi = length(pi1)
  for (i in 1:n_pi) {
    pi = c(pi1[i], 1- pi1[i])
    data = generateData(n,pi,lambda,theta)
    X = data$X
    Z = data$Z
    front.bayes(X,Z,pi,lambda,theta)
  }
}
```

```
front.bayes.piVariation(n,pi1,lambda,theta)
```


Simulation avec n=1000 points
X[,2]
0.0
0.5
1.0
1.5
2.0
0
2
4
6
8
X[,1]


Simulation avec n=1000 points
X[,2]
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
2
4
6
X[,1]


Simulation avec n=1000 points
X[,2]
0.0
0.5
1.0
1.5
2.0
2.5
0
1
2
3
4
5
6
7
X[,1]



On peut voir ici que la zone de séparation change d'ordonnée à l'origine ; cela est dû au fait que C change à cause de la variation des proportions de classes.

.