Statistique computationnelle

Méthodes d'échantillonnage

Vadim BERTRAND, Cheikh-Darou BEYE

14 novembre 2022

Sommaire

1 Exercice 1

1.

$g(x)$ est une densité ssi :

- $g(x)$ est C^0 et positive sur $\mathbb{R}$
- $\int_{-\infty}^{+\infty} g(x)dx = 1$

Or, $exp(u)$ est C^0 et positive sur $\mathbb{R}^-$ (car elle l'est sur $\mathbb{R}$), donc $g(x) = \frac{1}{2}exp(-|x|)$ est C^0 et positive sur $\mathbb{R}$. La figure 1 permet d'illustrer ces propriétés pour $x \in [-5, 5]$.

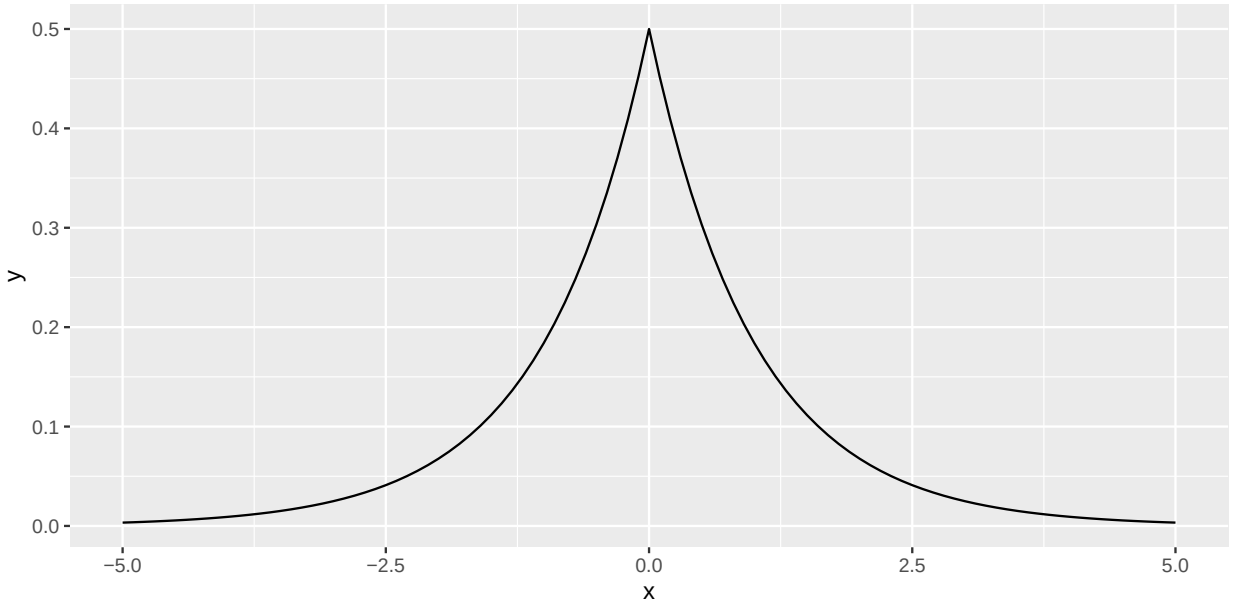

Figure 1: Aperçu de la fonction $g(x) = \frac{1}{2}exp(-|x|)$, pour $x \in [-5, 5]$

Et d'après le développement (1), nous avons bien que $\int_{-\infty}^{+\infty} g(x)dx = 1$:

$$\begin{aligned}\int_{-\infty}^{+\infty} g(x)dx &= \int_{-\infty}^{+\infty} \frac{1}{2}exp(-|x|)dx \\ &= \frac{1}{2}\left(\int_{-\infty}^{0} exp(x)dx + \int_{0}^{+\infty} exp(-x)dx\right) \\ &= \frac{1}{2}\left[[exp(x)]_{-\infty}^{0} + [-exp(-x)]_{0}^{+\infty}\right] \\ &= \frac{1}{2}(1+1) = 1\end{aligned} \tag{1}$$

Ce qui équivaut à vérifier que $g(x)$ est bien une densité.

2.

L'algorithme 1 décrit le fonctionnement de la méthode d'inversion.

Algorithm 1: Méthode d'inversion pour simuler selon g

Data: Q // fonction quantile de la densité g

$u \leftarrow U \sim \mathcal{U}[0,1]$

$x \leftarrow Q(u)$

Soient $G(x)$ la fonction de répartition de $g(x)$ et $Q(y)$ sa réciproque (et donc la fonction quantile de g). Par définition, nous avons : $G(x) = \int_{-\infty}^{x} g(u)du$ et $(Q \circ G)(x) = x$.

Le développement (2) donne l'expression de $G(x)$ pour $x \in \mathbb{R}^+$ et $x \in \mathbb{R}^-$.

$$\begin{aligned} \forall x \in \mathbb{R}^+, \qquad G^+(x) &= \frac{1}{2}\left(\int_{-\infty}^{0} exp(u)du + \int_{0}^{x} exp(-u)du\right) \\ &= \frac{1}{2}\left[[exp(u)]_{-\infty}^{0} + [-exp(-u)]_{0}^{x}\right] \\ &= 1 - \frac{1}{2}exp(-x) \end{aligned} \tag{2a}$$

$$\begin{aligned} \forall x \in \mathbb{R}^-, \qquad G^-(x) &= \frac{1}{2}\int_{-\infty}^{x} exp(u)du \\ &= \frac{1}{2}[exp(u)]_{-\infty}^{x} \\ &= \frac{1}{2}exp(x) \end{aligned} \tag{2b}$$

L'expression des réciproques de G^+ et G^- est obtenu via le développement (3).

$$\begin{aligned} \forall x \in \mathbb{R}^+, \qquad & G^+(x) = 1 - \frac{1}{2}exp(-x) \\ & \Leftrightarrow exp(-x) = 2(1 - G^+(x)) \\ & \Leftrightarrow x = -ln(2(1 - G^+(x))) \\ \Rightarrow \forall y \in [0.5, 1], \qquad & Q^+(y) = -ln(2(1-y)) \end{aligned} \tag{3a}$$

$$\begin{aligned} \forall x \in \mathbb{R}^+, \qquad & G^-(x) = \frac{1}{2}exp(x) \\ & \Leftrightarrow exp(x) = 2G^-(x) \\ & \Leftrightarrow x = ln(2G^-(x)) \\ \Rightarrow \forall y \in [0, 0.5], \qquad & Q^-(y) = ln(2y) \end{aligned} \tag{3b}$$

Munis de ces expressions, nous pouvons implémenter en **R** la fonction de tirage suivante :

```
rg <- function (n) { # retourne les réalisations de g
  Un <- runif(n, min = 0, max = 1) # tirage selon la loi U[0,1]
  Gn <- sapply(Un, function (u) { # tirage selon g via sa fonction quantile
    if (u > 0.5) -log(2*(1-u)) else log(2*u)
  })
}
```

3.

Nous avons utilisé notre procédure d'inversion pour générer un échantillon de taille 1000 selon la densité g. La représentation proposée sur la figure 2 nous permet de valider graphiquement cette procédure, étant donné que la densité empirique de l'échantillon (tracée en bleu) est semblable à la densité théorique (en noir).

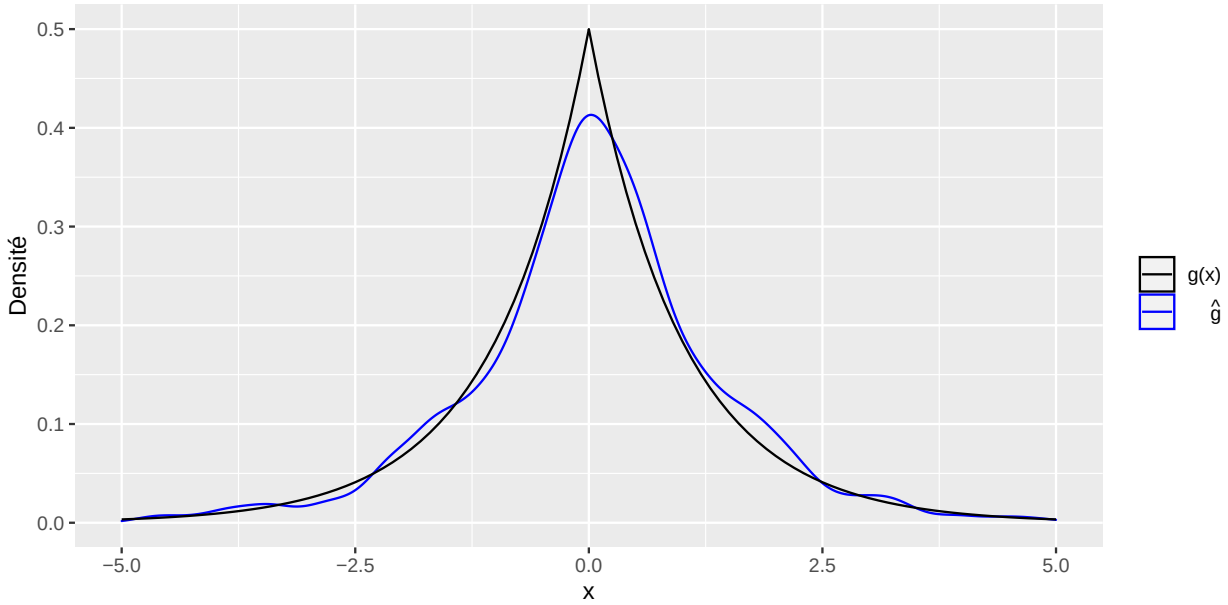

Figure 2: Comparaison de la densité théorique (en noir) et de la densité empirique (en bleu) de g

4.

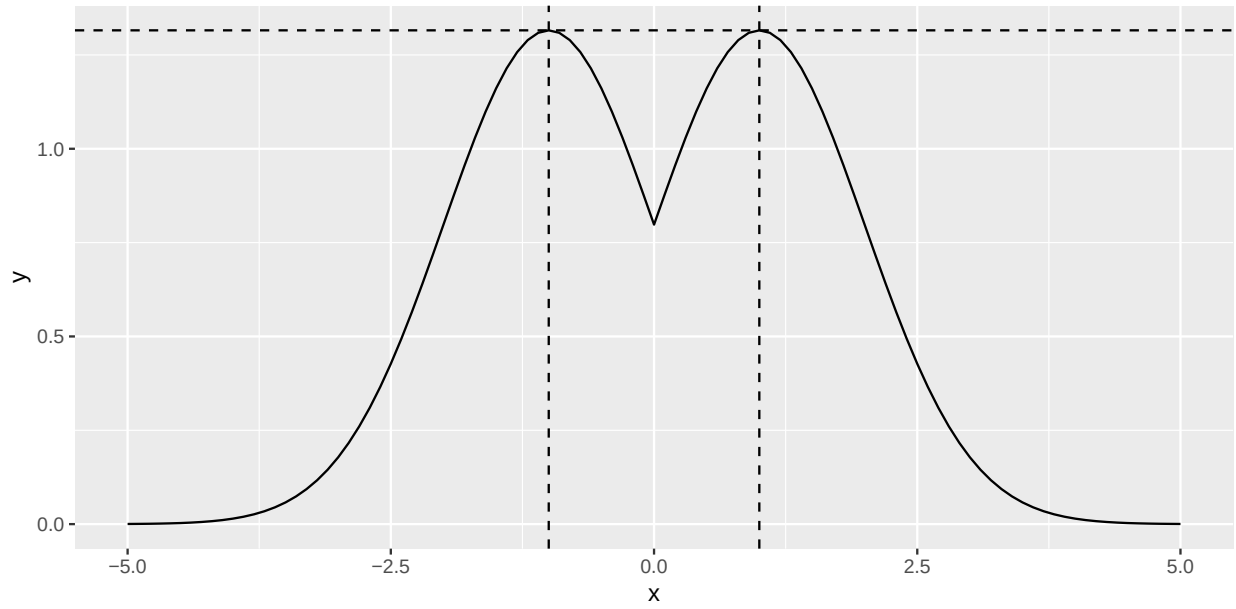

Figure 3: Représentation de $\frac{f(x)}{g(x)}$

La figure 3 illustre bien que $M = \sqrt{\frac{2e}{\pi}}$ est le plus petit majorant tel que $f(x) \leq Mg(x)$, $\forall x$ puisque nous pouvons voir que la courbe noire, représentant la rapport entre $f(x)$ et $g(x)$, est inférieure à M (droite horizontale en pointillés) mais tangente en deux points (-1 et 1, droites verticales en pointillés).

Cette observation graphique est retrouvée par le calcul selon le développement (4).

$$\begin{aligned} \forall x \in \mathbb{R}, \quad & f(x) \leq Mg(x) \\ & \Leftrightarrow M \geq \frac{f(x)}{g(x)} \qquad (g > 0) \end{aligned} \tag{4a}$$

$$\begin{aligned} \forall x \in \mathbb{R}^+, \quad & \frac{f(x)}{g(x)} = \sqrt{\frac{2}{\pi}} exp(\frac{-x^2}{2} + x) \overset{\text{not}}{=} M^+(x) \\ & \Rightarrow M^{+'}(x) = \sqrt{\frac{2}{\pi}} exp(\frac{-x^2}{2} + x)(-x + 1) \\ & donc, M^{+'}(x) = 0 \\ & \Leftrightarrow x = 1 \\ & d'où, \min_{x \in \mathbb{R}^+} M^+(x) = M^+(1) = \sqrt{\frac{2e}{\pi}} \end{aligned} \tag{4b}$$

$$\begin{aligned} \forall x \in \mathbb{R}^-, \quad & \frac{f(x)}{g(x)} = \sqrt{\frac{2}{\pi}} exp(\frac{-x^2}{2} - x) \overset{\text{not}}{=} M^-(x) \\ & \Rightarrow M^{-'}(x) = \sqrt{\frac{2}{\pi}} exp(\frac{-x^2}{2} - x)(-x - 1) \\ & donc, M^{-'}(x) = 0 \\ & \Leftrightarrow x = -1 \\ & d'où, \min_{x \in \mathbb{R}^-} M^-(x) = M^-(-1) = \sqrt{\frac{2e}{\pi}} \end{aligned} \tag{4c}$$

5.

L'idée générale de la procédure de rejet est donnée par l'algorithme 2 et partiellement illustrée sur la figure 4.

Algorithm 2: Méthode de rejet pour simuler selon f

Data: $N = 1000$ // taille du $n-$échantillon simulé selon f
Data: $M = \sqrt{\frac{2e}{\pi}}$ // plus petit majorant tel que $f \leq M * g$
Data: g // densité
$X \leftarrow \emptyset$
$r \leftarrow 0$
while $|X| < N$ **do**
  $x_0 \leftarrow G \sim g$ // voir algorithme 1
  $u_0 \leftarrow U \sim \mathcal{U}[0, M * g(x_0)]$ // entre l'axe des abscisses et la courbe bleu de la figure 4
  if $u_0 \leq f(x_0)$ **then** // entre l'axe des abscisses et la courbe noire de la figure 4
    $X \leftarrow X \cup \{x_0\}$
  else
    $r \leftarrow r + 1$
  end
end

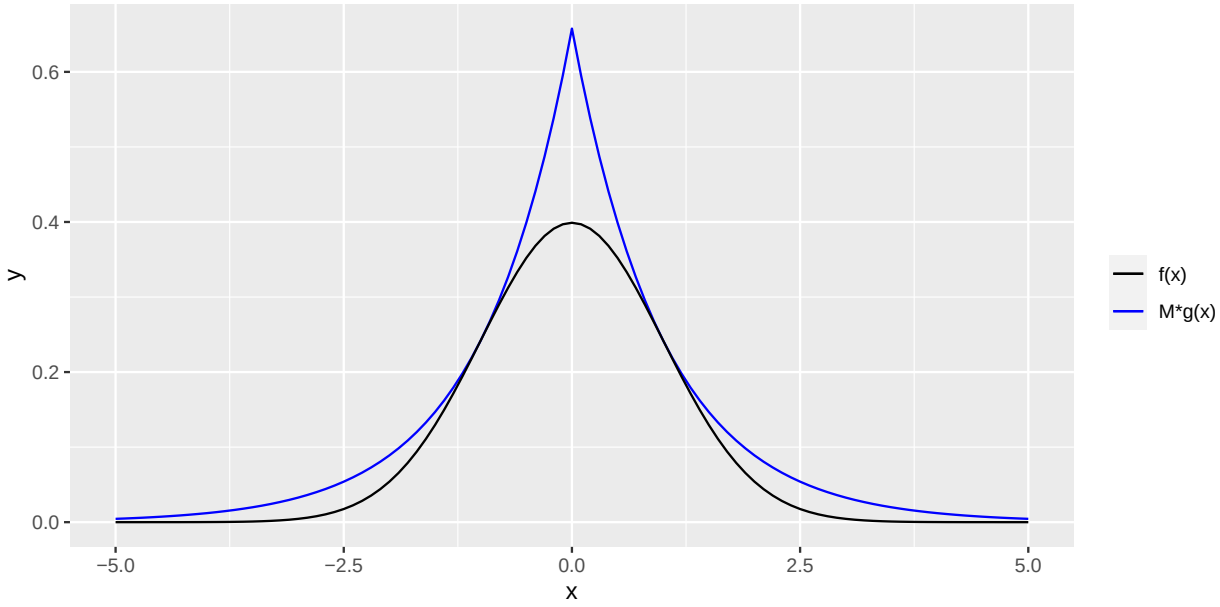

Figure 4: Illustration graphique de la procédure de rejet

Nous avons implémenté cette procédure en **R** via la densité g et le majorant M obtenus précédemment :

```
rf <- function (n) { # retourne les réalisations de f et le taux de rejet
  Xn <- NULL
  rejected <- 0
  while (length(Xn) < n) { # on veut n réalisations
    X0 <- rg(n - length(Xn)) # génération du nombre de réalisations manquantes selon g
    U0 <- sapply(X0, function (x) runif(1, 0, M*g(x))) # tirages dans U[0, g(x0)]
    idx <- U0 <= dnorm(X0) # f=dnorm
    Xn <- c(Xn, X0[idx]) # on conserve les x0 inférieurs ou égaux à f(x0)
    rejected <- rejected + sum(1-idx) # maj du nombre de rejets
  }
  list(Xn = Xn, rate = rejected/(rejected+n))
}
```

De même que pour la procédure d'inversion, nous avons généré un échantillon de taille 1000 selon la densité f via notre méthode de rejet. La figure 5 ci-dessous nous permet de constater que la densité empirique obtenue (tracée en bleu) est très proche de la densité théorique (représentée en noir), et donc d'attester de la pertinence de notre implémentation.

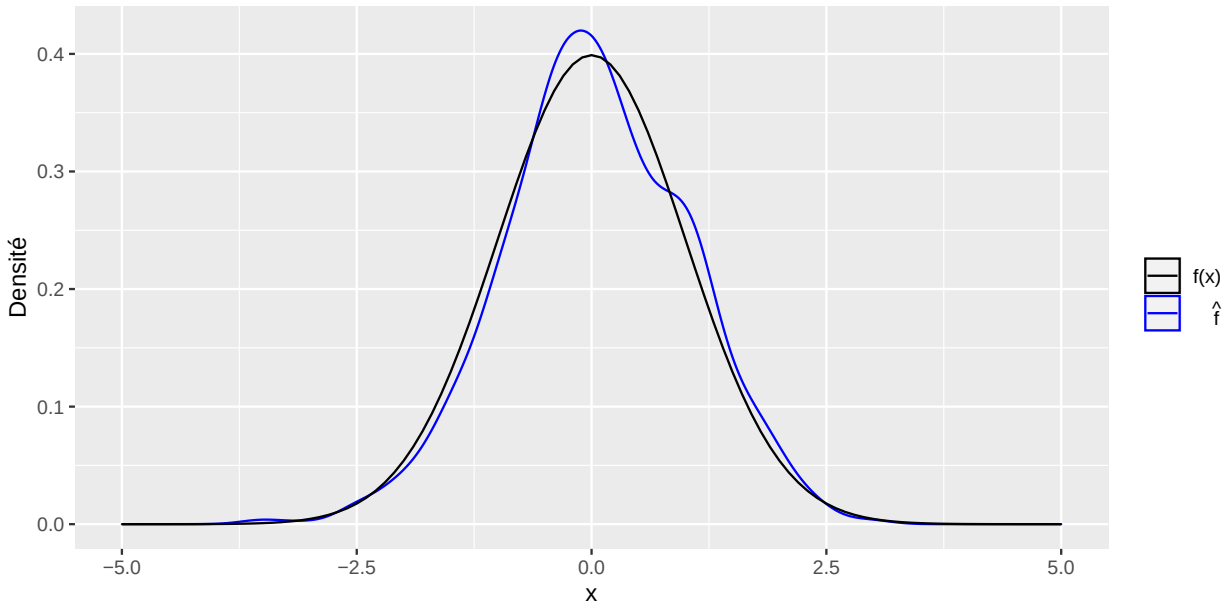

Figure 5: Comparaison de la densité théorique (en noir) et de la densité empirique (en bleu) de f

Le taux de rejet est égale au rapport entre le nombre de tirages rejetés par notre procédure de rejet et le nombre de tirages réalisés total. Au plus ce taux est faible, au plus notre générateur est computationnellement performant.

Nous obtenons un taux de rejet de 0.25, autrement dit : pour 4 tirages, nous en rejetons 1 et nous en acceptons 3. Au premier abord, ce taux peut paraitre étonnamment élevé étant donné que sur la figure 4, la fonction $Mg(x)$ semble assez proche de $f(x)$. Cependant, il faut aussi noter que l'écart entre $Mg(x)$ et $f(x)$ est le plus important pour les valeurs les plus probables de la densité f.
Par ailleurs, si nous nous intéressons aux aires sous les courbes de $f(x)$ et $Mg(x)$ ce taux était même prévisible. Nous savons que $\int_{-\infty}^{+\infty} f(x)dx = \int_{-\infty}^{+\infty} g(x)dx = 1$, donc la "probabilité" (entre guillemets, car ce n'est pas formel) d'accepter un tirage est $\frac{\int_{-\infty}^{+\infty} f(x)dx}{\int_{-\infty}^{+\infty} Mg(x)dx} = \frac{1}{M} \simeq 0.76$ et celle de le rejeter est $1 - \frac{1}{M} \simeq 0.24$, ce qui est proche du taux de rejet obtenu empiriquement.

Si nous souhaitions diminuer le taux de rejet, nous pourrions envisager de :

- considérer l'utilisation d'une densité auxiliaire permettant de mieux approximer f autour de 0, là où sa probabilité est la plus grande,
- utiliser la méthode du rejet adaptatif pour corriger l'approximation de f itérativement.

2 Exercice 2

Dans cet exercice, nous souhaitons nous assurer qu'un taux de bonne classification de 0.95 obtenu par validation-croisée est conservé en confrontant le modèle à de nouvelles données, non utilisées pour son entrainement. Plus précisément, nous souhaitons invalider l'hypothèse que le score de validation est égal à certaines valeurs seuil : $\{0.8, 0.85, 0.9, 0.92, 0.93\}$.
Pour cela il est possible de réaliser des tests statistiques comparant les résultats obtenus sur un nouveau jeu de validation aux différents seuils. Sous l'hypothèse de nulle de ces tests, le taux de validation est supposé égal à $p_0 \in \{0.8, 0.85, 0.9, 0.92, 0.93\}$, l'hypothèse alternative étant qu'il vaut $p_1 = 0.95$.
Sous l'hypothèse nulle, nous savons que :

$$\frac{\bar{X}_n - p_0}{\sqrt{\frac{p_0(1-p_0)}{n}}} \sim \mathcal{N}(0,1)$$

Aussi, nous pouvons utiliser la partie gauche de l'expression comme la statistique de test. Et comme nous avons $p_1 > p_0$, le seuil de rejet serait $q^{\mathcal{N}}_{1-\alpha}$, le quantile $1-\alpha$ de la loi normale centrée-réduite. La figure 6 permet de visualiser la région de rejet de ce test.

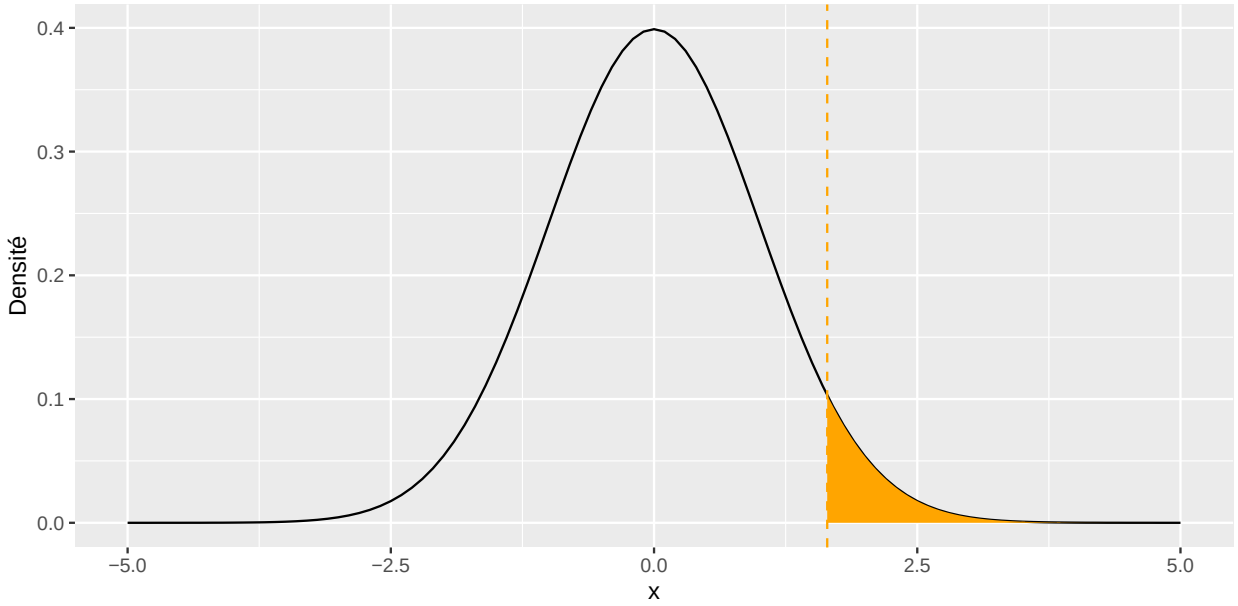

Figure 6: Représentation de la région de rejet du test (en orange)

Afin d'avoir une confiance suffisante dans les résultats des tests, nous aimerions connaître leur puissance statistique (c'est à dire la probabilité de rejeter à raison l'hypothèse nulle, et donc d'accepter un taux de bonne classification de 0.95) en fonction de la taille de l'échantillon de validation.
Cette démarche a été réalisée par l'approche Monte Carlo décrite par l'algorithme 3.

Algorithm 3: Méthode Monte Carlo pour estimer la puissance d'un test

Data: $M = 10000$ // nombre de répétitions
Data: $p_0 \in \{0.8, 0.85, 0.9, 0.92, 0.93\}$ // taux de bonne classification sous H_0
Data: $p_1 = 0.95$ // taux de bonne classification sous H_1
Data: $n \in [50, 500]$ par pas de 50 // taille du jeu de validation
Data: $\alpha = 0.05$ // niveau du test
$r \leftarrow 0$
repeat M **times**
  $X_n \leftarrow B \sim \mathcal{B}(n, p_1)$ // n-échantillon représentant le succès ou non de la classification
  $d \leftarrow \frac{\bar{X_n} - p_0}{\sqrt{\frac{p_0(1-p_0)}{n}}}$ // statistique de test
  if $d > q_{1-\alpha}^{\mathcal{N}}$ **then** // rejet à raison de H_0
    $r \leftarrow r + 1$
  end
end
$p \leftarrow \frac{r}{M}$ // puissance estimée

La figure 7 présente les résultats obtenus. Nous pouvons observer que la puissance du test augmente avec lorsque la taille de l'échantillon de validation augmente. Cela était attendu étant donné que quand n augmente, la statistique de test augmente également.
Le test permettrait de rejeter l'hypothèse d'un taux de bonne classification égale à 0.8 avec une puissance de 1 (donc une certitude de 100%) à partir d'un jeu de validation de taille 100. En revanche, rejeter l'hypothèse d'un taux à 0.93 avec une grande puissance statistique sera impossible, même avec un jeu de validation de taille 500. S'il est primordial d'avoir une grande confiance dans le fait que le taux du classifieur est supérieur à 0.93 il faudra donc beaucoup plus de données. Si un taux de classification à 0.9 est acceptable, alors choisir un jeu de validation de taille 300 permettra d'avoir une bonne confiance dans les performances attendues du classifieur, étant donné que la puissance statistique du test est alors supérieur à 0.9.

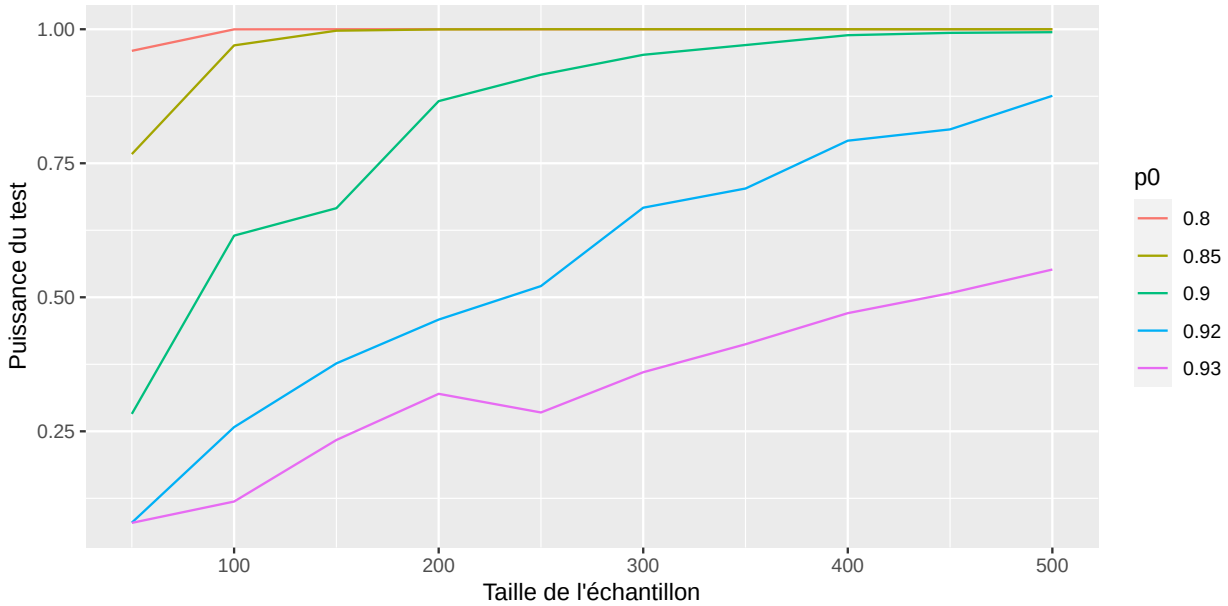

Figure 7: Evolution de la puissance du test selon la taille de l'échantillon, pour différentes valeurs de p_0

3 Exercice 3

1.

Il existe deux mesures centrales en classification binaire : la sensibilité (le taux de positifs effectivement classés comme positifs) et la spécificité (le taux de négatifs effectivement classés négatifs). Un classifieur parfait aurait une sensibilité et une spécificité égales à 1. En pratique, il faut réaliser un compromis entre sensibilité et spécificité en jouant sur la valeur du seuil de discrimination permettant d'affecter les observations à l'une des deux classes selon leur probabilité d'appartenance à la classe des positifs. La courbe ROC permet de visualiser ce compromis pour différents seuils.

L'AUC correspond à l'aire sous la courbe ROC ; plus l'AUC est proche de 1, meilleur est le compromis sensibilité / spécificité, quel que soit le seuil, et donc meilleur est le classifieur. L'AUC peut également être interprété comme la probabilité que le score d'une observation positive soit supérieur au score d'une observation négative, quelles que soient ces observations. Ainsi, un AUC de 1 signifie que le plus petit score parmi les observations positives est supérieur au plus grand score parmi les négatives, et donc qu'il existe un seuil permettant de discriminer parfaitement les deux classes.

2.

Les deux classifieurs ont un AUC élevé, celui du premier étant égal à 0.92 et celui du deuxième à 0.94. Selon ce critère, le classifieur 2 est donc légèrement plus performant que le 1.

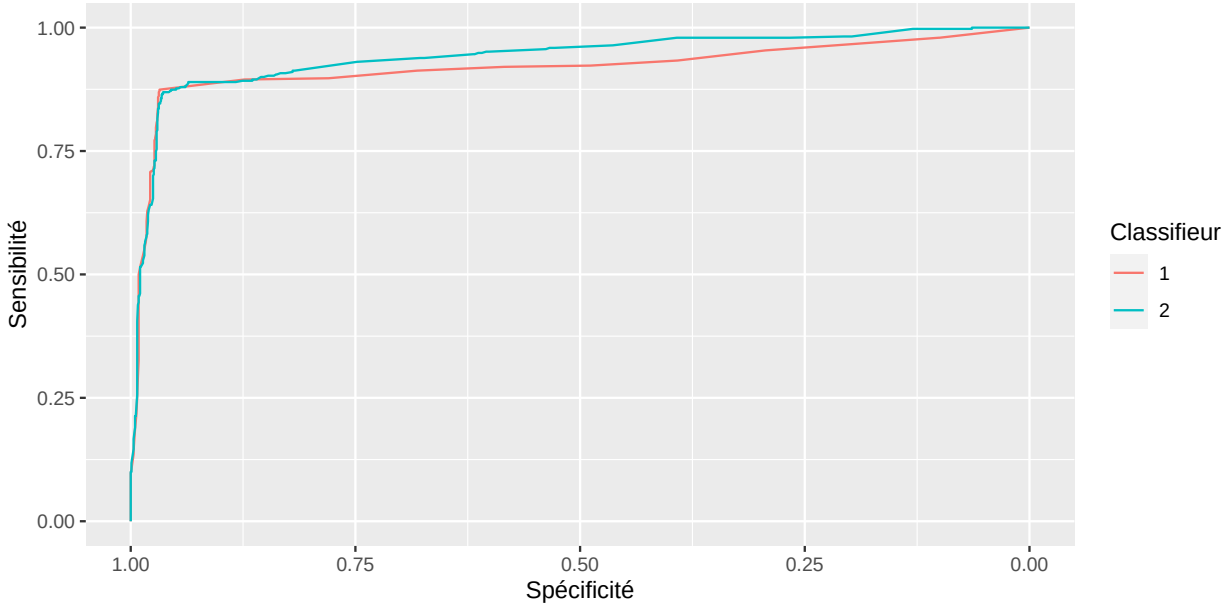

Figure 8: Courbes ROC des deux classifieurs

La figure 8 présentant les courbes ROC des deux classifieurs permet d'arriver à la même conclusion : les deux courbes sont proches des segments ((0,1), (1,1)) et ((1,1), (1,0)), ce qui traduit la bonne qualité des prédicteurs. De plus, la courbe ROC du classifieur 2 (en bleu) est principalement au-dessus de celle du 1 (en rouge), et montre que la sensibilité du classifieur 2 est meilleure que celle du classifieur 1.

3.

Nous voulons à présent obtenir des intervalles de confiance pour ces AUC. Il existe deux procédures bootstrap pour calculer un intervalle de confiance, la méthode des percentiles et celle du pivot. L'approche par les percentiles est la plus intuitive et est énoncée, pour un intervalle à 95% et dans le cas de l'AUC, par l'algorithme 4.

Algorithm 4: Estimation d'un intervalle de confiance par la méthode des percentiles

Data: $B = 1000$ // nombre de répétitions
Data: P_n // scores des cas positifs
Data: N_n // scores des cas négatifs
Data: $sample(échantillon)$ // réalise un tirage avec remise parmi l'échantillon
Data: $auc(positifs,\ négatifs)$ // retourne l'AUC selon les scores des cas positifs et négatifs
Data: $perc(X,\ p)$ // retourne le p-ème percentile de l'échantillon X
$AUC \leftarrow \emptyset$
repeat B **times**
  $P_B \leftarrow sample(P_n)$
  $N_B \leftarrow sample(N_n)$
  $AUC \leftarrow AUC \cup \{auc(P_B,\ N_B)\}$
end
$(perc(AUC,\ \frac{0.05}{2}), perc(AUC,\ 1 - \frac{0.05}{2}))$ // intervalle de confiance

Appliquée aux scores des deux classifieurs étudiés ici, nous obtenons, au niveau 95%, l'intervalle de confiance [0.9, 0.94] pour le classifieur 1 et [0.93, 0.96] pour le 2.

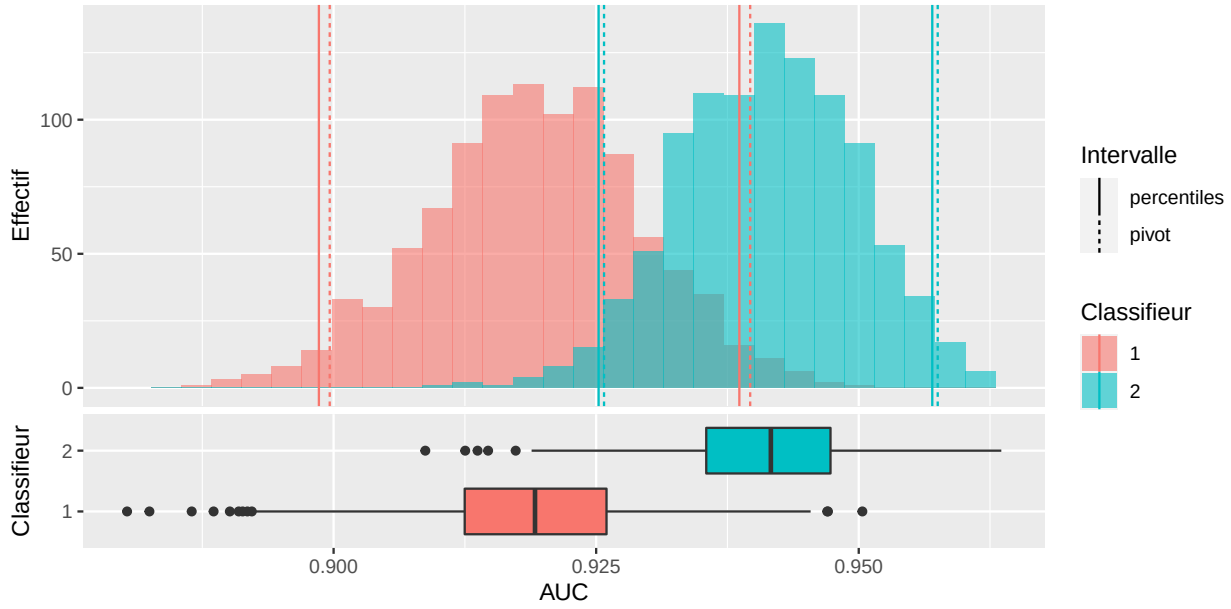

Figure 9: Distribution des AUC obtenus par bootstrap et les intervalles de confiance correspondants

La figure 9 permet de visualiser les distributions des AUC calculées par bootstrap et les intervalles de confiance associés (par la méthode des percentiles décrite précédemment et par celle du pivot). Nous pouvons remarquer que les AUC obtenus pour le classifieur 2 sont légèrement moins étendus que ceux obtenus pour le 1.
Par ailleurs, les boîtes à moustaches laissent penser que la médiane des performances du classifieur 2 sont

nettement supérieurs à celle du classifieur 1, mais les intervalles de confiance n'étant pas disjoints, il semblerait plutôt que les performances des classifieurs ne soient pas statistiquement différentes.

4.

Afin de déterminer si l'un des deux classifieurs est statistiquement plus performant que l'autre, nous pouvons réaliser un test d'égalité de deux AUC via une procédure de permutation en suivant démarche indiquée par l'algorithme 5.

Algorithm 5: Réalisation d'un test d'égalité par permutation

Data: $P = 1000$ // nombre de répétitions
Data: $comp_1$ // comparaisons des scores des cas positifs et négatifs
Data: $comp_2$ // comparaisons des scores des cas positifs et négatifs
Data: $sample(M,\ N)$ // réalise N tirages sans remise dans $1 : M$
Data: $stat(c_1,\ c_2)$ // statistique de test: $abs(mean(c_1),\ mean(c_2))$
$s_0 \leftarrow stat(comp_1,\ comp_2)$
$comp_Z \leftarrow comp_1 \cup comp_2$
$s \leftarrow 1$
repeat P **times**
  $idx \leftarrow sample(|comp_Z|, |comp_1|)$ // tirage sans remise des indices
  $comp_X \leftarrow comp_Z[idx]$ // retourne les valeurs des indices tirés
  $comp_Y \leftarrow comp_Z[-idx]$ // retourne les valeurs des indices non tirés
  if $stat(comp_X,\ comp_Y) \geq s_0$ **then** // comparaison avec la statistique de test initiale
    $s \leftarrow s + 1$
  end
end
$\frac{s}{P+1}$ // p-valeur

La table 1 donne les résultats obtenus via notre procédure de test et la procédure de test par bootstrap proposé par la fonction **roc.test** du package **pROC**. Nous pouvons observer que la p-valeur obtenue par notre procédure par permutation est égale à 0 et permet donc de rejeter l'hypothèse nulle d'égalité des AUC quel que soit le niveau de significativité souhaité. La procédure bootstrap du package **pROC** permet elle de rejeter cette hypothèse à un niveau de significativité de "seulement" 0.05. Ces résultats viennent confirmer l'observation faite sur l'intersection non vide des intervalles de confiance obtenus par bootstrap représentés sur la figure 9 et nous poussent à conclure à la performance supérieure du classifieur 2 sur le 1.

Table 1: p-valeurs des tests d'égalité des AUC

	permutation	bootstrap
p-valeur	0.000999	0.0207208

4 Exercice 4

Nous disposons d'un jeu d'entrainement de 375 observations et d'un jeu de test de 125. Chacune des observations de ces jeux appartient à l'une des catégories 0 ou 1. Dans le jeu d'entrainement, respectivement 188 et 187 observations appartiennent aux catégories 0 et 1 ; dans celui de test, respectivement 62 et 63. Les jeux sont donc équilibrés.

1.

Sur la figure 10 nous pouvons observer les catégories (0 en rouge et 1 en bleu) prises par les observations des jeux d'entrainement (représentées par des points) et de test (représentées par des croix) selon leurs variables $X1$ et $X2$.
Nous avons également représenté les histogrammes des deux variables, pour les jeux de test et d'entrainement.

Nous pouvons remarquer que les deux catégories ne sont pas totalement séparables. Mais il semble que $X1$ est généralement plus faible pour la catégorie 0 que pour la catégorie 1 et à l'inverse, $X2$ est globalement plus élevée pour la catégorie 0 que pour la 1. Cela se vérifie dans les deux jeux de données. Selon les histogrammes, il semblerait que la variable $X2$ permet de mieux discriminer que $X1$ dans le jeu d'entrainement, et que c'est moins le cas dans le jeu de test.

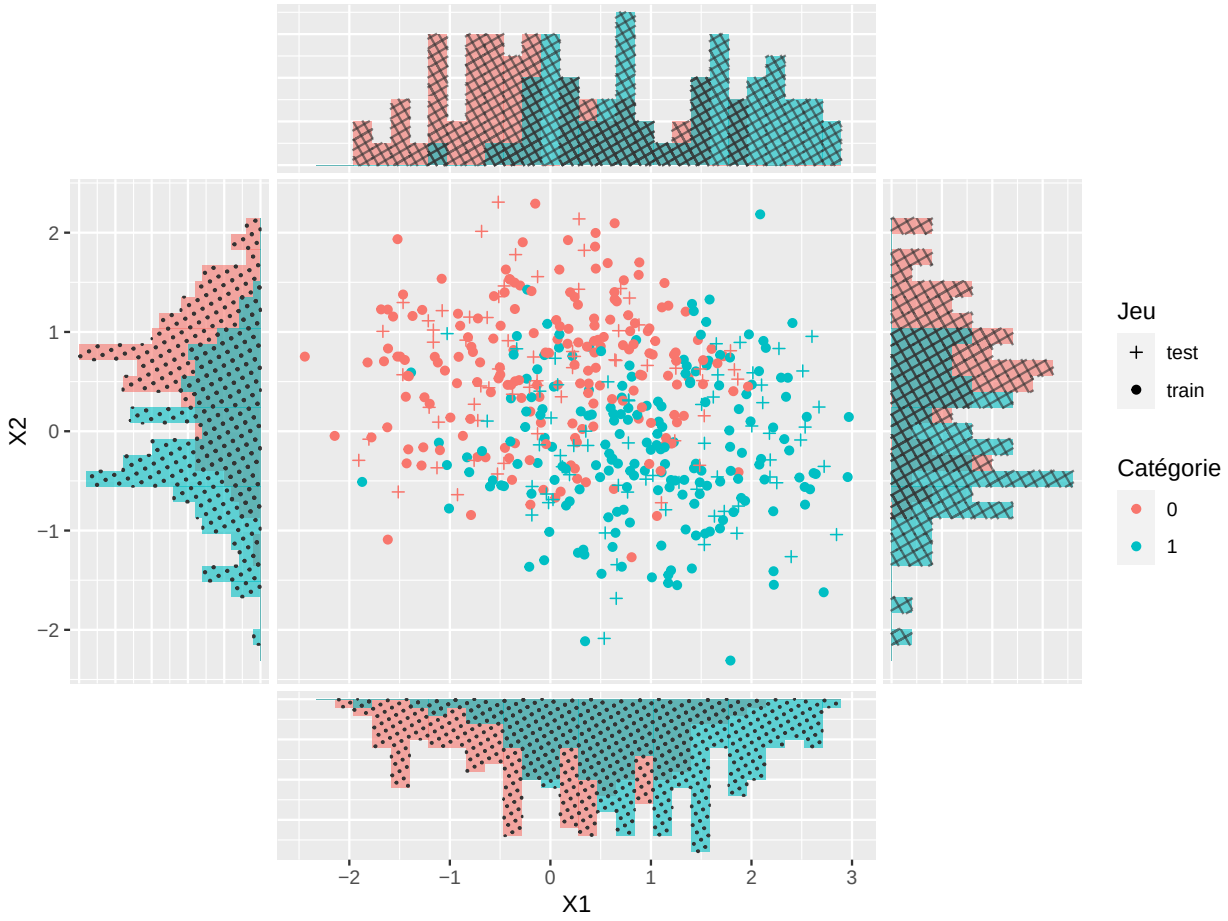

Figure 10: Aperçu des jeux d'entrainement et de test

2.

L'algorithme des k-ppv est très simple à mettre en place : chaque nouvelle observation non labellisée est classée dans la catégorie majoritaire de ses k plus proches voisins labellisés. En théorie, si le nombre d'observations et de voisins sont suffisamment grands et que le nombre d'observations est très grand par rapport au voisinage considéré, ce classifieur est optimal. Mais en pratique, le nombre d'observations étant fini, les k-ppv n'est pas forcément optimal et le choix du voisinage influ sur la performance de l'algorithme.

Nous avons observé cet impact sur la classification des données de test par les k-ppv en considérant $k \in \{1, 3, 5, 7, 9, 11\}$. La table 2 présente les résultats obtenus. Nous pouvons voir qu'en effet le taux de bonne classification fluctue selon k, mais que, pour un voisinage de 1 mis à part, cette variation est assez faible.

Table 2: Taux de bonne classification par k-ppv pour différents k

k	1	3	5	7	9	11
taux	0.680	0.784	0.760	0.776	0.768	0.768

3.

L'un des problèmes souvent rencontrés en apprentissage statistique est le sur-apprentissage, c'est-à-dire l'extraction par le modèle d'informations trop spécifiques aux données d'apprentissage et absentes des données de test.
Il existe de nombreuses approches pour limiter le sur-apprentissage, dont le "bagging" qui, appliqué aux k-ppv, est décrit par l'algorithme 6.

Algorithm 6: Application du "bagging" aux k-ppv

Data: $B = 100$ // nombre de répétitions
repeat B **times**
    tirer avec remise un n-échantillon parmi les observations labellisées
    classer par les k-ppv les observations non-labellisées à partir de ce n-échantillon
end
classer les observations non-labellisées selon la classe majoritaire parmi les B classifications précédentes

Nous avons employé cette méthode en choisissant $B = 100$ sur nos jeux de données. Sur la table 3 nous pouvons observer que les résultats obtenus ne sont pas réellement différents de ceux de l'algorithme classique des k-ppv.

Table 3: Taux de bonne classification par bagging pour différents k

k	1	3	5	7	9	11
taux	0.680	0.784	0.776	0.776	0.760	0.784

4.

Sur la figure 11 nous avons représenté les taux de bonne classification donnés dans les tables précédentes et, en transparence, nous avons ajouté ceux obtenus par les B classifieurs individuels employés pendant la procédure de bagging. Nous pouvons observer que la variance des taux de bonne classification individuels augmente lorsque k augmente, mais cela ne se fait pas au détriment des performances générales de la procédure de bagging.

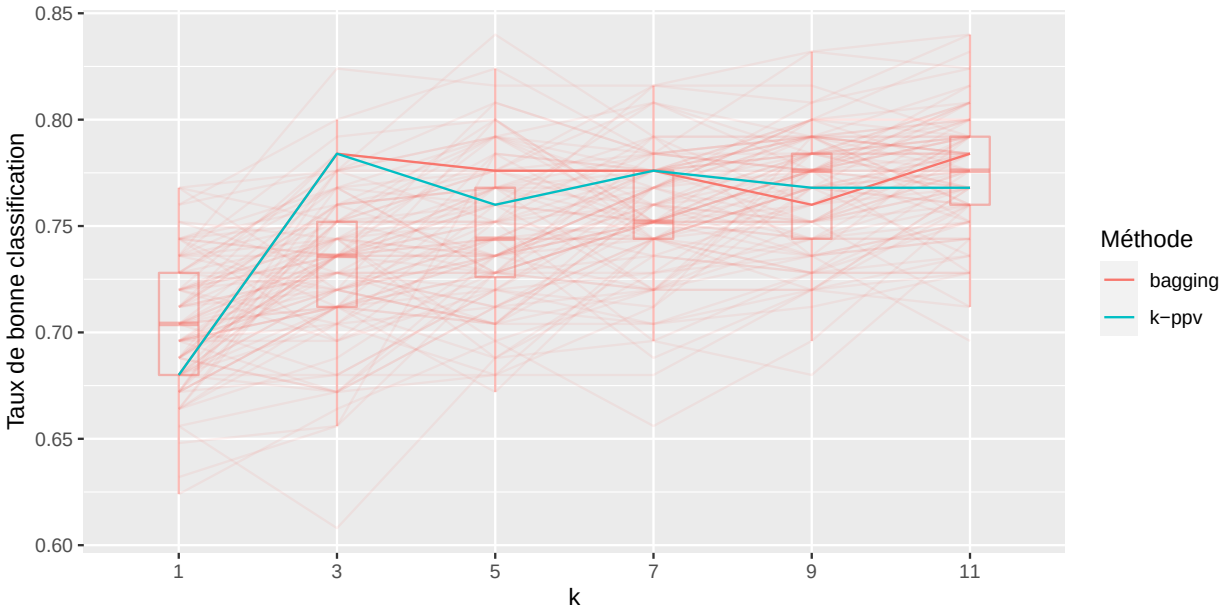

Figure 11: Taux de bonne classification des k-ppv, avec (en rouge) et sans (en bleu) bagging