Données confidentielles: génération de jeux de données synthétisés par forêts aléatoires pour des variables catégoriques

Mémoire

Maxime Caron

Maîtrise en statistique
Maître ès sciences (M.Sc.)

Québec, Canada

Résumé

La confidentialité des données est devenue primordiale en statistique. Une méthode souvent utilisée pour diminuer le risque de réidentification est la génération de jeux de données partiellement synthétiques. On explique le concept de jeux de données synthétiques, et on décrit une méthode basée sur les forêts aléatoires pour traiter les variables catégoriques.

On s'intéresse à la formule qui permet de faire de l'inférence avec plusieurs jeux synthétiques. On montre que l'ordre des variables à synthétiser a un impact sur l'estimation de la variance des estimateurs.

On propose une variante de l'algorithme inspirée du concept de confidentialité différentielle. On montre que dans ce cas, on ne peut estimer adéquatement ni un coefficient de régression, ni sa variance.

On montre l'impact de l'utilisation de jeux synthétiques sur des modèles d'équations structurelles. On conclut que les jeux synthétiques ne changent pratiquement pas les coefficients entre les variables latentes et les variables mesurées.

Abstract

Confidential data are very common in statistics nowadays. One way to treat them is to create partially synthetic datasets for data sharing. We will present an algorithm based on random forest to generate such datasets for categorical variables.

We are interested by the formula used to make inference from multiple synthetic dataset. We show that the order of the synthesis has an impact on the estimation of the variance with the formula.

We propose a variant of the algorithm inspired by differential privacy, and show that we are then not able to estimate a regression coefficient nor its variance.

We show the impact of synthetic datasets on structural equations modeling. One conclusion is that the synthetic dataset does not really affect the coefficients between latent variables and measured variables.

Table des matières

Liste des tableaux

Liste des figures

Remerciements

C'est sur ce merveilleux mémoire que je termine ma vie universitaire (du moins pour les prochains mois !). Je profite de cette section pour remercier en premier mes collègues qui m'ont accompagné au baccalauréat ainsi qu'à la maîtrise. Je pense entre autres aux gens de ma cohorte en mathématique et en statistique, dont Anick, Vincent, Max, Fred, Anne, Elsa, Jess, Marie, l'autre Max, Virg, Marie-Laure pour ne nommer que vous. Grâce à tous les gens que j'ai côtoyé, j'ai passé une vie sociale et académique extraordinaire. J'aimerais aussi remercier ceux qui m'ont enduré comme colocataire ! ! J'espère que je ne vous ai pas trop nui dans vos études par toutes mes soirées de jeux vidéos ! Je remercie également ceux que j'ai cotoyé dans l'exécutif de l'AESMUL. Vous avez rendu les réunions agréables et tous ensemble, on a rendu la vie universitaire vraiment plaisante ! Merci à ceux qui m'ont supporté pendant l'écriture de ce mémoire comme Val, Évan, Greg (pendant l'été), les deux Séb, Marcus, le beau Luc. Un merci à ceux qui ont été courageux pour lire quelques pages afin de trouver les erreurs dans mon mémoire. Je pense entre autres à Kim et ma mère ! Merci beaucoup ! Je remercie également mes parents pour leur support autant monétaire que moral. Merci également de m'avoir déménagé relativement souvent (environ 5 fois en 6 ans ? ! ? !). Merci d'avoir été dans les moments plus difficiles et d'avoir cru en moi.

J'aimerais également remercier tous les professeurs en statistique, soit Claude Bélisle, Louis-Paul Rivest, Nadia Ghazzali, Anne-Sophie Charest, Thierry Duchesne, Lajmi Lakhal-Chaieb ainsi que les chargés de cours qui m'ont enseigné tout au long de mon cheminement. Votre encadrement et vos exercices/notes de cours m'ont permis d'apprendre dans des circonstances idéales.

Je tiens également à remercier l'Université du Québec à Trois-Rivières, plus spécifiquement, le Laboratoire de recherche sur la performance des entreprises (LaRePE) et Josée St-Pierre pour m'avoir donné accès à des données confidentielles pour réaliser certains tests. Plus précisément, j'ai pu refaire les analyses d'un article publié et tester ensuite si les jeux de données synthétiques modifient les résultats. Sans ces données, cela n'aurait pas été possible.

Finalement, j'aimerais remercier Anne-Sophie Charest pour son support et ses idées qui m'ont permis d'écrire ce mémoire. Merci de m'avoir poussé et soutenu dans les moments plus ardus, cela m'a permis de me surpasser ! Merci de m'avoir encouragé à faire la présentation à Toronto, cela a été un des moments les plus mémorables pendant ma maîtrise (surtout être à Toronto !).

Introduction

Depuis quelques années, le traitement des données confidentielles est devenu primordial en statistique avec l'essor qu'a connu internet. Avant d'expliquer pourquoi cela est important, je vais définir ce qu'on entend par données confidentielles.

En statistique, il y a une étape importante qui est la collecte des données comme lors d'un recensement ou d'une enquête. À cette étape, l'observation qui peut être une personne, un hôpital ou un centre de santé par exemple, est souvent assuré que ses données seront gardées confidentielles. C'est-à-dire que les informations obtenues de l'observation ne seront pas identifiables. Une réidentification nuit à la collecte des données. Intuitivement, une solution serait de retirer les identifiants comme le numéro d'assurance sociale ou la date de naissance, mais cela n'est pas nécessairement suffisant comme il est possible de le constater dans l'exemple suivant :

En 2006, Netflix annonce une compétition dont le but est d'améliorer leur algorithme qui prédit l'évaluation qu'un utilisateur donnera à un film en se basant sur ses précédentes évaluations. Netflix publie une base de données de 17 700 films évalués par un demi-million d'abonnés. Chaque ligne était composée du titre du film, de la date d'évaluation et de l'évaluation. La base avait été nettoyée, c'est-à-dire que tous les identifiants avaient été retirés. Cela n'a pas empêché un adversaire (deux chercheurs, Narayanan et Shmatikov) d'identifier une personne à partir de IMBD (Internet Movie Database), une base de données accessible à tous, en faisant le lien entre les films évalués. Netflix a par la suite modifié certaines politiques en matière de confidentialité (Heffetz et Ligett, 2013).

On entend donc par données confidentielles, les informations privées d'une observation qu'on n'est pas supposé divulguer sans le consentement de l'observation. Pour la suite de ce mémoire, il sera acquis qu'une observation représente l'unité sur laquelle les variables sont mesurées. Il faut que les statisticiens s'assurent de protéger ces données tout en s'assurant également de produire des données de haute qualité. C'est-à-dire que les données aient une grande utilité, qu'elles apportent de l'information qu'il n'était pas possible d'avoir auparavant. Les statisticiens doivent donc répondre à deux objectifs contradictoires, soit de respecter la confidentialité des observations tout en produisant des données de grande qualité, puisque plus les données seront de haute qualité, plus le risque de réidentification d'une observation sera élevé. Il est impératif de trouver une solution pour respecter la confidentialité des données en conservant le maximum d'utilité des données avant que d'autres cas de non-respect de la confidentialité ne viennent compliquer la collecte des données.

Il existe plusieurs méthodes pour diminuer ce risque de réidentification. On a nommé une méthode précédemment, soit le retrait des variables identifiantes, mais on a vu que cette méthode n'était pas efficace dans l'exemple de Netflix. Il est également possible de limiter l'accès aux jeux de données, de perturber les résultats dans les publications ou de perturber les jeux de données originaux. Par exemple, pour limiter l'accès aux jeux de données, il est possible de faire signer des contrats de confidentialité.

Une méthode pour perturber les jeux de données est la génération de plusieurs jeux de données synthétiques. Sous certaines conditions, cette méthode permet de conserver une grande utilité dans les données et permet de diminuer le risque de réidentification tout en permettant l'accès aux jeux de données sans aucune restriction.

On appliquera cette technique en reproduisant les résultats de l'article de Caiola et Reiter (2010) et on caractérisera les conditions dans lesquelles cette technique fonctionne. Le premier objectif du mémoire est de valider la formule permettant de combiner les différents jeux de données synthétisés par forêts aléatoires pour faire de l'inférence.

On s'intéresse ensuite à une propriété très stricte en terme de confidentialité, la confidentialité différentielle. Celle-ci permet de limiter l'influence qu'une observation peut avoir sur l'information publiée et ainsi de contrôler le risque de divulgation. Le second objectif est d'ajouter le principe de la confidentialité différentielle à l'algorithme générant les jeux de données partiellement synthétiques.

Les jeux synthétiques ont souvent été utilisés lorsque les analyses demandées sont des régressions. On explore un domaine différent en étudiant l'impact des jeux synthétiques sur des équations structurelles puisque cela n'a pratiquement pas été étudié. Il s'agit du troisième objectif du mémoire. On se préoccupe particulièrement de tous les coefficients que l'on retrouve dans les équations structurelles, à savoir si ceux-ci sont modifiés par l'utilisation de jeux de données partiellement synthétiques.

Ces différentes préoccupations sont étudiées à travers ce mémoire qui est divisé de la façon suivante : le chapitre 1 introduit la génération de jeux de données partiellement synthétiques à l'aide de forêts aléatoires pour des variables catégoriques ; le chapitre 2 valide des résultats de l'article de Caiola et Reiter (2010) et caractérise les conditions au bon fonctionnement de la méthode ; le chapitre 3 explique la provenance des formules pour combiner des jeux de données imputés et des jeux synthétisés, et répond au premier objectif ; le chapitre 4 présente la confidentialité différentielle et répond au second objectif en étudiant l'effet de l'ajout de la confidentialité différentielle à la génération et à l'analyse de jeux de données partiellement synthétiques ; et le chapitre 5 explore l'impact de la génération de jeux de données partiellement synthétiques sur les coefficients dans un modèle d'équations structurelles des moindres carrés. On terminera avec une discussion des points importants soulevés dans ce mémoire. On note que les éléments présentés au chapitre 1 seront utilisés tout au long du mémoire, et que la théorie pertinente sera ensuite présentée au début de chaque chapitre subséquent.

Chapitre 1

Données synthétiques

Dans ce chapitre, on décrit les jeux de données partiellement synthétiques et on introduit les forêts aléatoires qui nous permettent de générer ces jeux. On décrit ensuite comment mesurer le risque de réidentification et l'utilité dans les jeux de données synthétiques. Ces notions théoriques seront utiles tout au long du mémoire.

1.1 Introduction aux données synthétiques et aux forêts aléatoires

1.1.1 Variables quasi-identifiantes et variables informatives

Les variables d'une base de données publiée peuvent être divisées en deux types : les quasi-identifiantes et les informatives. Ces dernières apportent de l'information sur une observation qu'il n'était pas possible de savoir à la base. Par exemple, le revenu est une variable informative tout comme n'importe quelle information sur un dossier médical d'un patient (atteint d'une maladie par exemple). Les variables quasi-identifiantes sont facilement accessibles via différentes ressources comme internet. Le sexe, la race et l'âge sont des exemples de variables quasi-identifiantes. Elles sont alors considérées comme à grand risque pour la réidentification, c'est-à-dire que c'est avec ces variables qu'il est possible de reconnaître une observation.

Il y a deux approches possibles pour la synthétisation de variables, soit la synthétisation des variables quasi-identifiantes ou la synthétisation des variables informatives. La génération de valeurs synthétiques pour des variables informatives protège la confidentialité car elle ne permet pas à l'adversaire d'apprendre les valeurs exactes pour ces variables (Caiola et Reiter, 2010). Donc, même s'il identifie une observation, l'adversaire ne peut conclure sur la validité des valeurs pour les variables informatives. La génération de valeurs synthétiques pour des variables informatives ne diminue toutefois pas le risque de réidentification, puisqu'il est toujours possible d'identifier une observation sans toutefois apprendre d'information. Dans ce qui suit, l'objectif est de diminuer directement le risque de réidentification. Il est donc préférable de synthétiser les variables quasi-identifiantes.

1.1.2 Données synthétiques

Avant de parler des jeux de données partiellement synthétiques, on s'intéresse à la méthode qui crée des jeux complètement synthétiques. Pour obtenir ces jeux, il faut, à partir d'un certain modèle statistique, par exemple, un modèle de régression, générer de nouvelles valeurs pour l'ensemble du jeu de données, autant les variables quasi-identifiantes que les variables informatives. Ce modèle est basé sur les données originales et permet d'avoir des observations synthétiques et des inférences statistiques valides. L'algorithme pour créer de tel type de jeux de données est composé de trois étapes. La première modélise la distribution des variables mesurées. Pour cette étape, il y a plusieurs méthodes, notamment l'utilisation de modèles de régressions séquentielles. La seconde étape est de simuler de nouvelles observations à partir de cette distribution. La troisième étape est simplement de répéter les étapes précédentes m fois et de publier les m jeux de données synthétiques obtenus (Rubin, 1993)(Reiter, 2004).

Le fait de répéter les étapes m fois permet de faciliter l'estimation de la variance. En effet, les estimateurs des paramètres provenant du jeu de données synthétiques ont deux sources de variabilité, soit lors de la création du modèle (variance associée à un coefficient de régression par exemple) et lors de la génération de valeurs synthétiques. Cette dernière n'étant pas estimable directement avec un seul jeu de données synthétique, il est nécessaire d'en avoir plusieurs. Un choix usuel pour le nombre de jeux de données à créer est 5 ou 10 (Rubin, 1987), mais un choix supérieur à 10 est probablement mieux puisque cela améliore la qualité de l'inférence.

Comme on l'a mentionné précédemment, il existe également une alternative à cette méthode, soit la génération de jeux de données partiellement synthétiques. Il s'agit de synthétiser certaines variables, principalement celles ayant un plus grand risque de réidentification, et de conserver intactes les autres variables. Par exemple, supposons un jeu de données avec les quatre variables suivantes : sexe, age, race et revenu. On doit synthétiser les variables sexe, race et âge puisque ces trois variables ont un grand risque de réidentification ; elles sont des variables quasi-identifiantes telles que définies dans la section 1.1.1. Pour ce faire, on modélise tour à tour chaque variable en fonction de toutes les autres, par exemple la variable sexe en fonction des variables race, âge et revenu. Au final, on obtient un jeu de données synthétiques contenant les valeurs originales de revenu ainsi que les valeurs synthétisées pour les trois autres variables. La procédure à suivre pour générer les données synthétiques est détaillée dans la section 1.1.4.

Il y a évidemment des raisons qui encouragent l'utilisation de jeux de données partiellement synthétiques lorsqu'on compare cette méthode à d'autres. Un avantage de générer ce type de jeux de données est que ceux-ci peuvent en général conserver une plus grande utilité des données puisque certaines variables n'ont pas été modifiées. Également, si les données sont simulées d'une distribution qui représente bien la vraie distribution des données observées, il est possible d'obtenir des inférences valides pour beaucoup d'estimateurs (Reiter, 2004). Il y a cependant des désavantages à utiliser la génération de jeux de données partiellement synthétiques. Par exemple, la génération de plusieurs

jeux de données peut prendre beaucoup de temps pour des gros jeux de données en plus de demander beaucoup de mémoire. Cela peut décourager des instituts et des compagnies à utiliser cette méthode. Également, un second désavantage est que si le modèle utilisé dans la génération des nouvelles valeurs n'est pas approprié, c'est-à-dire que le modèle ne représente pas bien la relation entre les variables indépendantes et la variable dépendante, alors les inférences statistiques obtenues ne seront pas valides (Caiola et Reiter, 2010). Un désavantage propre à la génération de jeux partiellement synthétiques est qu'il y a un prix à payer pour conserver une grande utilité des données, soit un plus grand risque de réidentification (Drechsler *et al.*, 2008).

Le modèle qui permet de générer les nouvelles données est la clé du succès de l'approche des jeux de données partiellement synthétiques. L'idée de base est de générer les nouvelles valeurs d'une variable à synthétiser à partir d'un certain modèle en fonction des autres variables. La relation entre les différentes variables n'est pas nécessairement linéaire, il est alors avantageux de prendre des méthodes non paramétriques qui permettent des relations non-linéaires et des interactions. Il existe quelques méthodes non paramétriques qui ont déjà été utilisées pour la génération de jeux de données partiellement synthétiques : arbre de classification et de régression, forêts aléatoires et machine à vecteurs de support (Drechsler et Reiter, 2011). Dans ce mémoire, la génération de données synthétiques se basera sur les forêts aléatoires.

1.1.3 Introduction aux forêts aléatoires

Une forêt aléatoire est une extension des arbres de classification. Un arbre a pour objectif de partitionner les observations dans $\mathbb{R}^p$ en q catégories. Une feuille représente une extrémité d'une branche qui elle-même représente le résultat du test à un nœud. Pour sa part, le nœud décrit une condition sur une variable. Si cette condition est vérifiée, la branche de gauche est sélectionnée, sinon, c'est la branche de droite. À l'intérieur des feuilles, il y a la proportion associée à chaque classe. Dans l'exemple de la figure 1.1, on souhaite séparer les observations en 2 catégories. Cet arbre a été obtenu à partir d'un jeu de données dans le but de prédire le sexe à partir d'un ensemble de variables. Pour obtenir une prédiction, il faut passer aux travers les nœuds et les branches de l'arbre. Par exemple, si une observation a une valeur inférieure à a pour la variable $Var1$, on doit prendre la branche de gauche. Ensuite, si cette observation a une valeur supérieure à b pour la variable $Var2$, on descend dans la branche de droite pour terminer dans une feuille. Celle-ci contient la proportion d'hommes dans cette feuille, soit 70%. La probabilité que l'observation soit un homme est alors de 70%. Pour la suite de ce mémoire, on réfère à cette action en parlant de descendre dans l'arbre.

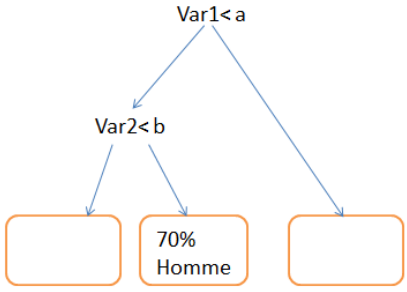

FIGURE 1.1 – Exemple d'arbre de classification.

Pour construire un arbre de classification, l'algorithme utilisé commence d'abord par séparer les observations en deux selon une certaine variable de façon à ce que les groupes soient les plus homogènes possibles par rapport à la variable dépendante. Par exemple, on pourrait être tenté de séparer les hommes des femmes à l'aide de la taille. Pour sélectionner la variable qui permettra d'effectuer cette première partition, on utilise habituellement un des trois critères suivants : l'index de Gini, l'entropie croisée ou le taux d'erreur de classification (James *et al.*, 2013). Il suffit de sélectionner un critère, de calculer ce dernier pour toutes les variables et, finalement, d'effectuer la première partition en sélectionnant la variable obtenant le meilleur résultat au critère. Le critère sélectionné par défaut en R est l'index de Gini. Il s'obtient avec la formule suivante :

$$I_G(f) = \sum_{i=1}^{m} f_i(1 - f_i)$$

où m représente le nombre de classes possibles et f_i désigne la proportion d'observations associée à la classe i. Dans le cas de l'index de Gini, sa valeur minimale est atteinte lorsque les observations se retrouvent dans la même catégorie selon une variable. Le pire cas survient lorsque les observations se retrouvent séparées de manière égales dans les catégories. L'algorithme considère par la suite un des deux groupes et calcule de nouveau le critère sélectionné pour toutes les variables afin de sélectionner celle qui séparera le mieux les observations. L'algorithme fait de même avec l'autre groupe. L'algorithme continue ainsi jusqu'à ce qu'il atteigne un certain critère d'arrêt, par exemple, jusqu'à ce qu'il ne reste que 5 observations par feuille (Hastie *et al.*, 2009). Lorsque l'algorithme est complété, l'arbre doit être élagué pour éviter le sur-ajustement de ce dernier sur les données. Le sur-ajustement a pour conséquence d'avoir une grande variance pour la prédiction, car il y a peu d'observations par feuille. Un nombre insuffisant de feuilles causerait toutefois un biais lors de la prédiction d'une observation. Pour obtenir le bon nombre de feuilles, on compare l'arbre complet à ceux obtenus en enlevant une ou plusieurs branches. On peut utiliser le critère de coût-complexité d'élagage (de l'anglais *cost-complexity pruning*)(Hastie *et al.*, 2009) ou la technique de validation croisée.

Une forêt aléatoire est un regroupement de plusieurs arbres de classification construits à l'aide de

différents sous-ensembles d'observations. Ces arbres n'ont pas à être élagués pour deux raisons. Premièrement, une forêt aléatoire est une extension du *bagging* (ensachage)(Hastie *et al.*, 2009). L'idée du *bagging* est de générer plusieurs échantillons *bootstrap* avec de l'échantillonnage aléatoire et d'entrainer l'arbre de classification avec les différents échantillons. Cela a l'avantage de diminuer la variance.

Deuxièmement, les arbres de classification créés sont décorrélés lorsqu'on effectue une forêt aléatoire, car ces arbres considèrent un sous-ensemble aléatoire de prédicteurs à chaque séparation. Un choix usuel pour un sous-ensemble est $\sqrt{p}$ où p est le nombre de prédicteurs possibles, soit le nombre de variables dans le modèle. Cela signifie qu'à chaque nœud dans l'arbre, l'algorithme ne considérera pas tous les prédicteurs à sa disposition. Sans cette condition, les arbres obtenus seraient corrélés puisqu'ils utiliseraient sensiblement les mêmes variables séparant le mieux les observations dès le départ (James *et al.*, 2013). La différence entre les arbres proviendrait de l'ensachage.

L'utilisation de forêts aléatoires possède plusieurs propriétés avantageuses. On note d'abord qu'il s'agit d'une méthode non paramétrique, c'est-à-dire que la méthode ne fait pas d'hypothèse a priori sur la forme paramétrique du modèle. Cela a l'avantage de représenter les relations les plus complexes en modélisant indirectement des interactions. Un second avantage à utiliser les forêts aléatoires pour la génération de jeux de données partiellement synthétiques est qu'il n'y a pas de réglage à faire. Effectivement, on peut utiliser les paramètres par défaut déjà implémentés dans le logiciel R [1], soit l'utilisation de l'index de Gini pour déterminer les variables à utiliser à chaque nœud, la sélection d'échantillons aléatoires de taille $2n/3$ (le *bagging*), où n est le nombre d'observations, ainsi que le choix de $\sqrt{p}$ pour le nombre de prédicteurs à chaque nœud (Caiola et Reiter, 2010). Également, le nombre d'arbres par défaut est 500.

1.1.4 Génération de jeux partiellement synthétiques par forêts aléatoires

L'algorithme permettant de créer des jeux partiellement synthétiques est spécialement conçu pour la synthétisation de variables catégoriques. Soit la matrice Y_0 contenant toutes les variables qui ne sont pas à synthétiser, c'est-à-dire une matrice où chaque colonne représente une variable qui ne sera pas altérée et où les lignes représentent les observations. Soit également les vecteurs Y_i où $i = 1, ..., s$ et Y_i représente une variable à synthétiser. Le vecteur Y_i^* est composé des nouvelles valeurs de la variable i. L'algorithme qui suit permet de créer un jeu de données partiellement synthétiques. Il suffit de répliquer cet algorithme m fois pour obtenir l'ensemble désiré. Dans les prochains chapitres, lorsqu'on réfère à la méthode de Caiola et Reiter, il s'agit de l'algorithme 1 [2] ci-dessous qu'il est question.

1. La version de la librairie *randomForest* est 4.6-7.
2. Cet algorithme est dans l'annexe B sous le nom GenSyn2.

Algorithme 1 pour la génération de jeux partiellement synthétiques (Caiola et Reiter, 2010)

Pour tout $i = 1, ..., s$:

1. ***Ajustement du modèle*** : on crée la forêt aléatoire avec la variable dépendante catégorique Y_i en fonction des variables Y_0 et $Y_1, ..., Y_{i-1}$.

$$mod = FA(Y_i \sim \{Y_0, Y_1, ..., Y_{i-1}\})$$

2. ***Obtention de la distribution des probabilités*** : on trouve la feuille associée à chaque observation dans chaque arbre en utilisant les valeurs des variables Y_0 et $Y_1^*, ..., Y_{i-1}^*$, soit les variables déjà synthétisées, et on note la classe ayant la plus grande proportion dans chaque feuille, soit le maximum des p_{jk} sur les j où k représente le k^e arbre et j représente une des classes. On trouve le maximum pour chaque arbre (k) et on compte le nombre de fois que chaque classe est retenue pour une observation, soit n_j. À partir de cela, on obtient la probabilité associée à chaque classe en calculant la fréquence relative de chaque classe sur le nombre d'arbres, ce qui équivaut à n_j/n où n représente le nombre d'arbres.

$$dist = pred(mod, (Y_0, Y_1^*, ..., Y_{i-1}^*))$$

3. ***Tirage aléatoire des nouvelles valeurs (Y_i^*)*** : on effectue un tirage pour chaque observation en fonction de leur distribution obtenue à l'étape précédente pour obtenir une valeur synthétique d'une observation.

$$Y_i^* = Tirage(Dist)$$

Recommencer pour le prochain i.

Pour bien saisir l'algorithme précédent, prenons l'exemple où la variable sexe et la variable race doivent être synthétisées. Pour ce faire, on ajuste un modèle en fonction des variables qui ne sont pas à synthétiser, soit les variables revenu et âge, pour la première variable à synthétiser, soit le sexe des observations. Ensuite, il faut tout d'abord vérifier pour chaque observation et pour chaque arbre dans quelle feuille cette observation termine sa descente et ensuite obtenir la prédiction de cette feuille. Après être descendu dans les 500 arbres comme décrit dans la section 1.1.3, on se retrouve avec une distribution de 500 résultats obtenus que l'on peut transformer en probabilité. Par exemple, supposons que 300 arbres ont voté que l'observation était un homme et 200 que celle-ci était une femme, les probabilités sont 0.6 (soit $300/500$) et 0.4. Finalement, il ne manque qu'à effectuer un tirage aléatoire pour cette observation avec les probabilités obtenues en utilisant une distribution multinomiale si le nombre de classes est supérieur à 2 ou une distribution binomiale si la variable à synthétiser est dichotomique. Dans le cas où la variable sexe est maintenant synthétisée et que la variable race doit

être synthétisée, on utilise les valeurs des variables revenu, âge et sexe (les valeurs originales) pour ajuster le modèle. Lorsque vient le temps de faire descendre chaque observation dans l'arbre pour obtenir les probabilités de chaque classe, il faut prendre les variables revenu et âge ainsi que la variable sexe synthétisée. Une fois les probabilités calculées, il ne reste qu'à effectuer un tirage aléatoire pour obtenir les nouvelles valeurs pour la variable race.

1.2 Risque de divulgation et mesure de l'utilité

1.2.1 Risque de divulgation

Il faut évaluer si la génération de jeux de données partiellement synthétiques diminue le risque de réidentification. L'objectif est de calculer la probabilité de réidentification pour chaque observation provenant du jeu initial. L'idée est de savoir combien d'observations seraient réidentifiées si l'adversaire connaissait les informations quasi-identifiantes pour chaque observation. On veut également savoir combien d'observations seraient identifiées à tort, c'est-à-dire que l'adversaire connait une observation, mais l'attribue a une autre observation dans le jeu de données. Dans ce cas, il fait une erreur. Pour ce faire, la méthode proposée par Reiter et Mitra (2008) sera appliquée, mais d'abord, voici quelques notations provenant de leur article.

Supposons un jeu de données contenant n unités et p variables. Soit y_{jk} la valeur de l'observation j à la variable k, pour $j = 1, ..., n$ et $k = 1, ..., p$. L'échantillon peut contenir une colonne supplémentaire pour représenter l'identifiant de chaque observation, par exemple le numéro d'assurance sociale d'une personne ou un index. La première étape pour évaluer le risque de réidentification consiste à séparer les variables en deux, soit celles étant disponibles via une base de données externe, dénotées y_{jA}, qui permettent la réidentification, donc les quasi-identifiantes et celles que l'on suppose seulement disponibles dans ce jeu de données, dénotées y_{jU}, soit les variables informatives. Également, on note que m représente le nombre de jeux de données synthétiques publiés. Les nouvelles valeurs pour l'observation j de la variable k sont notées $z_{jk}^{(l)}$ pour $l = 1, ..., m$, pour tout j et k. On suppose que seulement les variables contenues dans A sont sujettes à être synthétisées pour prévenir la réidentification. On sépare $z_{jA}^{(l)}$ en deux groupes de variables $(z_{jS}^{(l)}, z_{jD})$ où $z_{jS}^{(l)}$ représente les variables provenant de A qui ont été synthétisées et où z_{jD} représente les autres variables de A qui n'ont pas été synthétisées, c'est-à-dire que $z_{jk} = y_{jk}$ pour un k provenant de D. Finalement, il y a $\mathbf{Z}_S$ qui représente la collection des z_{jS} où $z_{jS} = (z_{jS}^{(1)}, ..., z_{jS}^{(m)})$, $\mathbf{Z}_D$ qui est définie de la même façon, $\mathbf{Z}$ qui représente tous les jeux de données partiellement synthétiques publiés et $\mathbf{Y}_S$ représentant les valeurs originales des variables synthétisées (Reiter et Mitra, 2008).

On suppose également qu'un adversaire s'attaquant à notre jeu de données connait un vecteur d'information t et qu'il veut obtenir une véritable identification, soit lorsque $z_{j0} = t_0$ où z_{j0} est l'identifiant de l'observation j (cette variable n'est évidemment pas publiée dans les jeux de données synthétiques) et t_0 est l'identifiant de la cible de l'adversaire. On note que j peut prendre différentes valeurs entre 1 et n et représente le numéro d'une observation. Pour ce qui est du vecteur t, il contient des infor-

mations sur des variables présentes dans $\mathbf{Z}$. Soit J une variable aléatoire qui vaut j si $z_{j0} = t_0$ où j est dans le jeu de données original et qui vaut $n + 1$ si j n'est pas dans le jeu initial avec $z_{j0} = t_0$. La variable J peut donc prendre une valeur entre 1 et $n + 1$. Comme mentionné dans l'article de Reiter et Mitra, il est plus simple de considérer que l'adversaire connait une observation présente dans le jeu de données ce qui veut dire que j est nécessairement dans le jeu de données original. Cela enlève donc la possibilité que J prenne comme valeur $n + 1$. Cela permet d'obtenir une mesure du risque d'identification plus conservatrice. Il est néanmoins possible de calculer cette mesure du risque lorsqu'on veut considérer que l'observation n'est pas nécessairement présente dans le jeu de données. Cela est plus complexe puisqu'il faut obtenir une estimation de la taille de la population. Les détails concernant les modifications à apporter au calcul qui suit sont fournis dans l'article de Reiter et Mitra (2008). Il faut calculer la probabilité $P(J = j|t, \mathbf{Z}, K)$ où K est l'information fournie à propos du modèle utilisé pour synthétiser les variables et où j est dans le jeu de données initial. Pour évaluer le risque dans l'exemple des pages suivantes, on considère que l'adversaire n'a pas d'information sur la synthétisation. Comme le mentionne leur article, l'adversaire ne connait pas les vraies valeurs dans $\mathbf{Y}_S$, il doit calculer la probabilité selon les différentes valeurs possibles que peut prendre $\mathbf{Y}_S$ comme suit :

$$P(J = j|t, \mathbf{Z}, K) = \int P(J = j|t, \mathbf{Z}, \mathbf{Y}_S, K)P(\mathbf{Y}_S|t, \mathbf{Z}, K)d\mathbf{Y}_S$$

Pour évaluer la probabilité de gauche dans l'intégrale, Reiter et Mitra ont supposé que les variables non disponibles, soit celles dans U, n'aident pas à la réidentification sachant $(\mathbf{Z}_D, \mathbf{Y}_S)$, c'est-à-dire que $P(J = j|t, \mathbf{Z}, \mathbf{Y}_S, K) = P(J = j|t, \mathbf{Z}_D, \mathbf{Y}_S)$. Pour les variables qui sont dans z_{jD}, c'est-à-dire celles qui proviennent de A, mais qui n'ont pas été synthétisées, la probabilité précédente vaut 0 lorsque les valeurs de ces variables ne sont pas identiques aux valeurs provenant de t_D (soit les valeurs des variables non-synthétisées que l'adversaire connait). Dans le cas contraire, il faut évaluer combien il y a de combinaisons de ($\mathbf{Z}_D$,$\mathbf{Y}_S$) parmi les n observations d'un jeu de données partiellement synthétique tel que le vecteur z_{jA} est identique au vecteur t. Ce total sera dénoté n_t. La probabilité désirée pour les j tel que z_{jA} est identique à t est $1/n_t$ sauf si n_t est zéro. Dans ce cas, il faut évaluer combien d'observations ont le vecteur z_{jD} pareil au vecteur t_D et on note ce total par n_t^*. La probabilité devient donc $1/n_t^*$. La probabilité vaut zéro pour les autres j. On répète pour chaque jeu de données synthétiques afin d'obtenir une matrice de probabilité de dimension n par m. Bref, le calcul est plutôt simple : on prend une observation du jeu original (avec les variables quasi-identifiantes seulement) ; on regarde combien de fois cette observation se répète dans un jeu synthétique et on note ce total n_t. Si l'observation n'est présente aucune fois, ce qui arrive lorsqu'une combinaison (un vecteur t) est rare et qu'après la synthétisation, on ne retrouve plus cette combinaison parmi les jeux synthétiques, on fait un autre calcul. On compte le nombre de fois que cette observation se répète sur les variables quasi-identifiantes non-synthétisées dans un jeu de données synthétique.

Pour le terme de droite dans l'intégrale, en considérant l'hypothèse que K est vide, la probabilité est

simplement $P(\mathbf{Y}_S|t, \mathbf{Z}, K) = 1/m$. Une seconde possibilité pour évaluer cette probabilité dans le cas où K est vide est de considérer seulement la valeur la plus plausible. Par exemple, si sur cinq jeux de données, trois ont conclu que la valeur synthétique était 2 et les deux autres ont obtenu 3 et 5 pour une variable à 5 catégories, il faut alors calculer la probabilité de gauche dans l'intégrale avec $\mathbf{Y}_S = 2$ sans avoir à calculer la probabilité pour les autres valeurs possibles. La probabilité vaut maintenant $P(\mathbf{Y}_S|t, \mathbf{Z}, K) = 1$. Si plusieurs valeurs ont la même fréquence, il faut supposer que l'adversaire tire au hasard parmi celles-ci. Pour évaluer cette probabilité dans ce mémoire, c'est la moyenne des m jeux partiellement synthétiques qui est utilisée.

On peut combiner les deux termes de l'intégrale pour ensuite obtenir la probabilité désirée. On obtient alors une probabilité moyenne associée à chaque observation. Ensuite, on doit recommencer pour différent vecteur t, soit en posant t comme étant une observation du jeu de données original y. On utilise un total de n valeurs différentes de t puisque nous avons une base de données composée de n observations. Un exemple de calcul est fait dans la sous-section 2.1.2.

Une fois les probabilités estimées, on s'intéresse au pourcentage de véritable identification ainsi qu'au pourcentage de fausse identification. Une vraie identification survient lorsqu'une observation aura une seule observation synthétique avec la plus grande probabilité et que celle-ci a le même identifiant. On pose k_i qui vaut 1 s'il s'agit d'une véritable identification [3] et 0 sinon. Alors, le pourcentage de véritable identification est : $\sum_{i=1}^{n} k_i/n$. Pour obtenir une fausse identification, il faut qu'une seule observation synthétique ait la plus grande probabilité, mais que l'identifiant ne soit pas le bon. Lorsqu'on a une fausse identification [4], f_i vaut 1. S'il ne s'agit pas d'une fausse identification, f_i vaut alors 0. Le pourcentage est calculé sur un dénominateur différent, soit le nombre de cas qui n'ont qu'une seule observation avec la plus grande probabilité que l'on dénote g. Ainsi, le pourcentage de fausse identification est $\sum_{i=1}^{g} f_i/g$ (Caiola et Reiter, 2010).

Exemple de calcul du risque de divulgation

Soit un jeu de données contenant 3 variables devant être synthétisées avec 5 observations. Les jeux synthétiques et le jeu original sont présentés dans le tableau 1.1. Jeu1 représente le premier jeu de données synthétiques, Jeu2 représente le second, etc. Les trois variables sont discrètes et peuvent prendre comme valeur soit 1 ou 2.

3. C'est-à-dire quand $c_i = 1$ et que $d_i = 1$ où c_i représente le nombre d'observations avec la plus grande probabilité obtenue et d_i indique si la véritable observation figure parmi celles ayant la plus grande probabilité. Alors $k_i = 1$ si $c_i d_i = 1$, sinon $k_i = 0$.

4. Il suffit de poser $f_i = 1$ lorsque $c_i(1 - d_i) = 1$, zéro dans le cas contraire et ensuite, on compte le nombre d'observation tel que $c_i = 1$, ce total est noté g.

Jeu original				Jeu1			Jeu2		
ID	V1	V2	V3	V1	V2	V3	V1	V2	V3
1	1	1	1	1	1	2	2	1	1
2	1	1	2	1	1	1	1	2	2
3	2	1	1	2	1	1	1	1	1
4	2	2	2	2	2	1	2	1	2
5	2	2	2	2	2	2	1	1	2
ID	Jeu3			Jeu4			Jeu5		
1	1	1	1	1	1	2	1	1	1
2	2	1	1	1	1	2	1	1	1
3	1	1	2	2	1	1	1	1	1
4	1	1	2	2	2	2	2	2	2
5	2	2	1	2	1	1	1	2	2

Tableau 1.1 – Matrice présentant les différents jeux de données (original et synthétiques) pour l'exemple sur le calcul du risque de divulgation.

Pour évaluer le risque de réidentifiation, on suppose d'abord que le vecteur t est $(1, 1, 1)$. On regarde combien de fois ce vecteur est présent dans chaque jeu synthétique. Dans ce cas-ci, il est présent une fois dans les trois premiers jeux synthétiques, il n'est pas présent dans le quatrième et est présent trois fois dans le dernier jeu. La matrice de probabilité associée à ce vecteur t est le suivante :

ID	Jeu1	Jeu2	Jeu3	Jeu4	Jeu5	Moyenne
1	0	0	1	0.20	0.33	0.31
2	1	0	0	0.20	0.33	0.31
3	0	1	0	0.20	0.33	0.31
4	0	0	0	0.20	0	0.04
5	0	0	0	0.20	0	0.04

Tableau 1.2 – Matrice des probabilités associées au vecteur $t = (1, 1, 1)$.

Selon la notation définie au chapitre précédent, on a $c_1 = 3$ et $d_1 = 1$, puisque l'observation l'identifiant du vecteur t (soit t_0=1) figure parmi les trois identifiants retenus avec la plus grande probabilité. Puisque $c_1 * d_1 \neq 1$ et $c_1 * (1 - d_1) \neq 1$, il n'y a pas de conclusion avec ce vecteur. En prenant le vecteur $t = (1, 1, 2)$, la matrice de probabilité est la suivante :

ID	Jeu1	Jeu2	Jeu3	Jeu4	Jeu5	Moyenne
1	1	0	0	0.50	0.20	0.34
2	0	0	0	0.50	0.20	0.14
3	0	0	0.50	0	0.20	0.14
4	0	0	0.50	0	0.20	0.14
5	0	1	0	0	0.20	0.24

Tableau 1.3 – Matrice des probabilités associées au vecteur $t = (1, 1, 2)$.

Ici, on se retrouve avec $c_2 = 1$ et $d_2 = 0$, cela implique que $c_2 * (1 - d_2) = 1$ et qu'il s'agit d'une

fausse réidentification. En effet, l'observation 1 n'est pas le vecteur $(1, 1, 2)$ tel que prédit par les jeux de données synthétiques, mais plutôt l'observation $(1, 1, 1)$.

1.2.2 Mesure de l'utilité

La méthode proposée par Karr *et al.* (2006) pour calculer l'utilité des jeux de données synthétiques mesure le chevauchement des intervalles de confiance. Pour obtenir ces intervalles, on doit d'abord être en mesure de combiner les jeux partiellement synthétiques pour obtenir une estimation du coefficient désiré ainsi qu'une estimation de la variance associée à ce coefficient. Pour évaluer la variance, il faut considérer deux sources de variabilité, soit celle associée aux coefficients du modèle (dénotée u_l) et celle associée au fait qu'on utilise seulement un nombre fini d'imputations (dénotée b_m) (Reiter, 2003).

Proposition 1 *(Caiola et Reiter, 2010)*

Pour obtenir une estimation Q d'un paramètre d'un modèle sous quelques hypothèses lorsqu'on est en présence de m jeux partiellement synthétiques, il faut utiliser la statistique suivante :

$$\overline{q}_m = \sum_{l=1}^{m} q_l/m$$

où q_l représente l'estimation du coefficient du modèle avec le l^e jeu de données synthétiques.

Un estimé de la variance du coefficient est obtenu ainsi :

$$T_P = \overline{u}_m + b_m/m \tag{1.1}$$

où $\overline{u}_m = \sum_{l=1}^{m} u_l/m$, $b_m = \sum_{l=1}^{m}(q_l - \overline{q}_m)^2/(m-1)$ et u_l représentant la variance du coefficient dans le jeu synthétique l.

L'intervalle de confiance au seuil α pour l'estimation Q est :

$$\overline{q}_m \pm t_{\nu_m,\alpha/2}\sqrt{T_P}$$

où ν_m représente les degrés de liberté d'une loi de Student et vaut $(m-1)[1 + m\overline{u}_m/b_m]^2$.

Nous reviendrons au chapitre 3 sur la justification et la validité de l'estimateur T_P.

Il est intéressant de comparer les intervalles de confiance pour mesurer l'utilité, car si les intervalles obtenus sont similaires, l'utilité sera élevée malgré le fait d'avoir modifié certaines valeurs. La méthode proposée pour comparer les intervalles de confiance mesure le chevauchement de ceux-ci (Karr

et al., 2006). Soit l'intervalle de confiance pour un coefficient du jeu de données originales : [L_o ; U_o] et soit l'intervalle de confiance pour un coefficient obtenu d'un ensemble de jeux synthétiques : [L_s ; U_s], la méthode pose L_i=max(L_o, L_s) et U_i=min(U_o, U_s).

Le chevauchement est obtenu ainsi :

$$IO_k = \frac{U_i - L_i}{2(U_o - L_o)} + \frac{U_i - L_i}{2(U_s - L_s)}.$$

Cela donne le chevauchement des intervalles de confiance pour le coefficient k. Pour obtenir l'utilité finale, il suffit de calculer $\sum_{k=1}^{p} \frac{IO_k}{p} = IO$ où p représente le nombre de coefficients, c'est-à-dire que l'utilité finale est la moyenne des chevauchements en considérant tous les coefficients. Si la statistique IO obtenue est près de 1, alors l'utilité est élevée. La valeur 1 est le maximum que peut atteindre cette statistique, il s'agit du cas où les intervalles de confiance sont identiques. Le pire des cas survient lorsque tous les intervalles de confiance n'ont aucun chevauchement, la statistique IO vaut alors zéro.

Un avantage d'utiliser cette formule survient lorsqu'on compare deux intervalles de confiance avec l'original et que les deux englobent ce dernier, la mesure IO favorise le plus petit des deux (Drechsler, 2011).

Chapitre 2

Validation et étude de sensibilité de l'algorithme 1

L'objectif de ce chapitre est de valider les résultats de l'article de Caiola et Reiter (2010) en reproduisant les exemples qu'ils ont faits. Ensuite, On étudie certaines propositions faites par cette article concernant l'algorithme 1. Plus précisément, l'ordre des variables à synthétiser ainsi que le choix du nombre optimal de jeux à créer lors de la synthétisation sont les deux caractéristiques qui nous intéressent.

2.1 Validation des résultats de Caiola et Reiter

Les calculs pour obtenir l'utilité ainsi que le risque de réidentification tels que définis au chapitre 1 seront détaillés en reproduisant l'exemple d'application provenant de l'article de Caiola et Reiter (2010). L'exemple porte sur des données provenant d'une enquête des ménages aux États-Unis, soit le *March* 2000 *U.S. Current Population Survey*. Le jeu de données contient 10 000 observations et 12 variables. Sur ces variables, il y a en quatre qui ont été considérées comme des variables à haut risque de réidentification, soit le sexe, la race, l'état marital ainsi que l'âge. La variable âge étant continue, on synthétise seulement les trois autres variables pour diminuer le risque de réidentification. Au total, il y a 484 observations uniques sur les quatre variables précédentes.

On utilise l'algorithme 1 avec Y_1=état marital, Y_2= race, Y_3=sexe et Y_0= les autres variables. On crée également deux ensembles de 5 jeux de données partiellement synthétiques sur lesquels 3 régressions différentes ayant chacune des particularités ont été effectuées. Par exemple, une des particularités est qu'une des régressions comporte une interaction entre deux variables synthétisées. On parle des particularités de chaque régression dans la section 2.1.3.

2.1.1 Quelques statistiques descriptives

Sur le jeu de données original

La principale statistique pour décrire les variables catégoriques est la proportion des observations qu'on retrouve dans chaque classe. Ces proportions obtenues avec les $10\ 000$ observations sont présentées dans le tableau 2.1. On note que la variable éducation est également une variable catégorique. Elle est composée de 16 catégories débutant à 31 (secondaire non-terminé) et finissant à 46 (étude graduée terminée). La majorité des observations se retrouve dans les catégories $39-40$ (environ 49%) et $43-44$ (22%).

Sexe	Proportion	**État marital**	Proportion
Homme	54.1%	Marié au civil	51.31%
Femme	45.9%	Marié-forces armées	0.22%
Race	Proportion	Marié-Époux absent	1.22%
Blanc	87.4%	Veuf	13.87%
Noir	9.8%	Divorcé	14.03%
Amérindien	0.75%	Séparé	2.63%
Asiatique	2.1%	Jamais marié	16.72%

Tableau 2.1 – Statistiques descriptives des variables catégoriques pour le jeu original.

Pour les variables numériques, les statistiques suivantes ont été obtenues pour chaque variable : la moyenne, l'écart-type, le minimum, le maximum et la médiane. Ces statistiques sont présentées dans le tableau 2.2 qui a été également obtenu avec les $10\ 000$ observations. Note au lecteur que les variables nombre de personnes dans le foyer et nombre de personnes mineures ont été considérés comme numériques malgré le fait que ce sont des variables discrètes.

Variable	Abrév.	Moy.	Écart-type	Min.	Max.	Méd.
Age	Age	51.28	17.62	15	90	49
Pension alimentaire	ALMVAL	39.24	902.35	0	54008	0
Pension alimentaire enfant	CSP	199.27	1373.49	0	23917	0
Montant des taxes propriété	PROTAX	1079.30	2376.25	0	99997	500
Revenu	INC	51779.14	49446.72	-21011	768742	38596.5
Nombre de pers. dans le foyer	HNUM	2.47	1.43	1	16	2
Nombre de pers. mineures	NUML18	0.66	1.08	0	10	0
Prestation de sécurité sociale	SS	2759.15	4871.51	0	50000	0

Tableau 2.2 – Statistiques descriptives des variables numériques pour le jeu original.

Sur un ensemble de cinq jeux de données synthétiques

Il est intéressant de regarder ce qui advient des variables synthétisées : est-ce que les proportions sont les mêmes qu'initialement ? Le tableau 2.3 montre la moyenne des proportions qu'on retrouve parmi

les cinq jeux de données de $10\,000$ observations. Les résultats sont sensiblement les mêmes que ceux obtenus dans le tableau 2.1 avec une variation se limitant à maximum 3.2 points de pourcentage.

Sexe	Proportion	**État marital**	Proportion
Homme	57.3%	Marié au civil	53.33%
Femme	42.7%	Marié-forces armées	0.18%
Race	Proportion	Marié-Époux absent	0.90%
Blanc	89.0%	Veuf	14.15%
Noir	8.7%	Divorcé	13.33%
Amérindien	0.6%	Séparé	1.96%
Asiatique	1.7%	Jamais marié	16.16%

Tableau 2.3 – Statistiques descriptives des variables catégoriques pour un ensemble de 5 jeux de données partiellement synthétiques.

2.1.2 Évaluation du risque de divulgation

D'abord, on suppose que l'adversaire connait les valeurs des quatre variables ayant un grand risque de divulgation, soit l'âge, le sexe, la race et l'état marital d'une observation qui est dans le jeu de données. Cette hypothèse simplifie quelque peu les calculs à faire pour évaluer les probabilités de chaque observation puisque $P(J = n + 1|t, \mathbf{Z}, K) = 0$ pour toutes les observations qui ne sont pas dans Z_0. L'ensemble K est vide pour évaluer le risque, puisqu'on suppose que l'adversaire ne connait pas la transformation effectuée aux données.

Pour bien comprendre comment évaluer le risque, prenons t comme étant la première observation de la base originale [1], alors t est le vecteur suivant pour l'état marital, la race, le sexe et l'âge : $(1, 1, 1, 20)$ et son numéro d'identification est 8248 (il s'agit de la première observation du jeu lorsqu'il est ordonné en ordre croissant en fonction des 4 variables mentionnées précédemment). Il est présent 2 fois dans le premier jeu synthétique, soit pour les observations ayant comme numéro d'identification 3170 et 6068. Celles-ci ont donc une probabilité d'être une identification de $1/2$ et toutes les autres observations de ce jeu synthétique ont une probabilité de 0. Pour les quatre autres jeux synthétiques, le vecteur t est présent respectivement 4 fois, 3 fois, 4 fois et 3 fois. Toutes ces informations ayant été mises dans une matrice de dimensions $10\,000$ par 5, on calcule pour chaque ligne la moyenne. On obtient alors une probabilité moyenne pour chaque observation d'être celle que l'adversaire cherche. On trouve la plus haute probabilité qui est 26.67% pour notre exemple. Cette probabilité moyenne est associée à une seule observation, soit l'observation ayant pour identifiant 3170. Puisque l'identifiant obtenu dans ce cas n'est pas le même que l'identifiant de l'observation de départ, il s'agit d'une fausse identification. En effet, on assigne à tort cette observation, c'est-à-dire que $c_1 = 1$ et $d_1 = 0$ en utilisant la notation de la section 1.2.1.

1. En annexe B, j'ai fourni le code R et C++ pour calculer le risque.

Le pourcentage de véritable identification est d'environ 6.9% et le pourcentage de fausse identification est de 82.2%. Comme la technique qui a été utilisée pour générer des jeux de données synthétiques est celle qui synthétise toutes les valeurs d'une variable, on compare les résultats obtenus à la technique 3.1a-3.3a de l'article de Caiola et Reiter. Ils avaient alors obtenu un taux de véritables identifications de 2.8% et un taux de fausses identifications de 91%. Il n'a pas été possible d'obtenir des résultats identiques.

2.1.3 Mesure de l'utilité

Suivant Caiola et Reiter (2010), les coefficients de trois différentes régressions appliquées sur le jeu de données original ainsi que sur deux ensembles de jeux de données synthétiques seront d'abord comparés. Puis, on va comparer les intervalles de confiance en utilisant la mesure d'utilité décrite au chapitre 1, soit en utilisant la mesure de chevauchement des intervalles de confiance des coefficients tel que décrit dans la section 1.2.2. La première régression est effectuée sur une variable dépendante transformée et considère également une interaction entre deux variables synthétisées. La seconde régression n'aura pas d'interaction, mais la variable dépendante sera également transformée. Enfin, la dernière régression sera faite sur un petit échantillon avec peu de prédicteurs.

Première régression

La régression a comme variable dépendante le log du revenu et les variables indépendantes sont les suivantes : sexe, race, Éducation, HNUM, PROTAX, age, age^2, état marital. On ajoute également l'interaction entre les variables sexe et état marital. Puisque la variable dépendante est le log du revenu, seulement les observations ayant un revenu supérieur à 0 sont conservées. La régression est donc effectuée sur 9938 observations. Les coefficients sont présentés dans le tableau 2.4 où Ensemble 1 et 2 représentent chacun un ensemble de 5 jeux de données synthétiques. Il faut noter que la génération des jeux synthétiques a été effectuée sur les 9938 observations contrairement à la section $2.1.1$ pour les statistiques descriptives où les $10\,000$ observations ont été utilisées pour générer les jeux partiellement synthétiques.

Il est facile de comparer les coefficients obtenus et de dire s'ils ressemblent à ceux du jeu de données initial. Pour s'aider, on a calculé l'erreur relative [2] pour chaque coefficient et on a obtenu une moyenne d'erreur relative de 25.7% avec un maximum de 155.5% pour la variable état marital=4 (veuf) pour le premier ensemble de jeux de données synthétiques. Pour le second, on obtient une erreur relative moyenne de 39.76% avec certaines variables ayant 200% d'erreur relative, soit l'état marital=4 (veuf) et l'état marital=3 (marié-époux absent).

2. Erreur relative=(valeur observée - valeur espérée)/valeur espérée

	Jeu original	Ensemble 1	Ensemble 2
Ordonnée	4.86	4.96	4.98
Noir	-0.19	-0.19	-0.18
Amérindien	-0.36	-0.27	-0.37
Asiatique	-0.06	-0.09	-0.12
Éducation	0.11	0.11	0.11
HNUM > 1	0.49	0.47	0.46
PROTAX*10 000	0.4	0.4	0.4
Age	0.04	0.04	0.04
Age^2*1 000	-0.42	-0.41	-0.41
Femme	-0.02	-0.06	-0.05
Marié-forces armées	0.46	0.44	0.25
Marié-époux absent	0.05	0.02	-0.06
Veuf	0.04	-0.02	-0.04
Divorcé	-0.19	-0.21	-0.23
Séparé	-0.32	-0.27	-0.38
Jamais marié	-0.16	-0.17	-0.19
Femme, mariée-forces armées	-0.56	-0.36	-0.39
Femme, mariée-époux absent	-0.62	-0.49	-0.47
Femme veuve	-0.35	-0.30	-0.27
Femme divorcée	-0.29	-0.28	-0.28
Femme séparée	-0.38	-0.44	-0.31
Femme jamais mariée	-0.28	-0.27	-0.27

Tableau 2.4 – Coefficients obtenus pour la première régression.

Puisque nous nous intéressons au chevauchement des intervalles de confiance des coefficients, on doit avoir ces intervalles. Ils sont présentés dans le tableau 2.5. Ces intervalles sont similaires à ceux obtenus dans l'article de Caiola et Reiter (2010). En fait, ils sont identiques comme prévu pour la régression sur les données originales puisqu'on a le même échantillon.

	Jeu original		Ensemble 1		Ensemble 2	
	[0.025%	0.975%]	[0.025%	0.975%]	[0.025%	0.975%]
Ordonnée	4.60	5.12	4.69	5.22	4.72	5.24
Noir	-0.24	-0.14	-0.25	-0.14	-0.25	-0.11
Amérindien	-0.53	-0.19	-0.46	-0.07	-0.57	-0.18
Asiatique	-0.17	0.04	-0.22	0.05	-0.25	0.02
Éducation	0.11	0.12	0.11	0.12	0.11	0.12
HNUM > 1	0.45	0.54	0.42	0.52	0.41	0.51
PROTAX*100	0.003	0.005	0.004	0.005	0.004	0.005
Age	0.04	0.05	0.03	0.05	0.03	0.04
Age^2*100	-0.047	-0.037	-0.046	-0.036	-0.046	-0.036
Femme	-0.07	0.02	-0.11	-0.00	-0.10	0.00
Marié-forces armées	-0.39	1.31	-0.51	1.39	-0.43	0.92
Marié-époux absent	-0.14	0.23	-0.26	0.30	-0.31	0.19
Veuf	-0.07	0.15	-0.14	0.09	-0.15	0.07
Divorcé	-0.26	-0.11	-0.29	-0.13	-0.32	-0.14
Séparé	-0.49	-0.16	-0.51	-0.04	-0.61	-0.16
Jamais marié	-0.23	-0.10	-0.25	-0.10	-0.26	-0.12
Femme, mariée-forces armées	-1.47	0.35	-1.39	0.67	-1.16	0.39
Femme, mariée-époux absent	-0.89	-0.34	-0.85	-0.13	-0.83	-0.11
Femme veuve	-0.46	-0.23	-0.42	-0.17	-0.39	-0.14
Femme divorcée	-0.39	-0.20	-0.38	-0.19	-0.39	-0.16
Femme séparée	-0.59	-0.18	-0.74	-0.14	-0.58	-0.04
Femme jamais mariée	-0.37	-0.20	-0.36	-0.18	-0.36	-0.18

Tableau 2.5 – Intervalles de confiance des coefficients obtenus pour la première régression.

Il n'est pas évident de comparer visuellement les intervalles obtenus, on mesure donc le chevauchement. Les résultats du chevauchement pour chaque coefficient sont dans le tableau 2.6. Certaines variables synthétisées ont conservé une grande utilité comme lorsqu'on a une femme mariée dans les forces armées avec un chevauchement supérieur à 90% dans les deux ensembles de jeux synthétiques. La mesure d'utilité moyenne (soit la moyenne pour chaque colonne) est 84.8% et 81.5% respectivement pour le premier et le second ensemble des jeux de données synthétiques. Ces résultats sont bons et ce même pour l'interaction entre le sexe et l'état marital, deux variables qui ont été synthétisées. Le chevauchement de cette interaction varie entre 66% et 95%.

	Ens. 1	Ens. 2
Ordonnée	0.82	0.77
Noir	0.94	0.88
Amérindien	0.75	0.94
Asiatique	0.90	0.80
Éducation	0.87	0.90
HNUM > 1	0.78	0.68
PROTAX	0.75	0.75
Age	0.91	0.85
Age^2	0.90	0.90
Femme	0.68	0.72
Marié-forces armées	0.95	0.87
Marié-époux absent	0.84	0.78
Veuf	0.72	0.63
Divorcé	0.86	0.72
Séparé	0.85	0.86
Jamais marié	0.93	0.82
Femme, mariée-forces armées	0.90	0.90
Femme, mariée-époux absent	0.82	0.79
Femme veuve	0.80	0.66
Femme divorcée	0.95	0.92
Femme séparée	0.83	0.85
Femme jamais mariée	0.91	0.93

Tableau 2.6 – Chevauchement des intervalles de confiance pour chaque coefficient.

Deuxième régression

Pour cette régression, la variable dépendante est la racine carrée de la prestation de sécurité sociale. Cela nous oblige à sélectionner les observations qui ont une prestation de sécurité sociale supérieure à 0. Suivant Caiola et Reiter (2010), on sélectionne également les observations ayant un âge supérieur à 54. La taille de l'échantillon est alors de $2\ 725$ observations. Les variables indépendantes dans le modèle sont les suivantes : sexe, race, état marital catégorisé, éducation catégorisée et âge. Cependant, la construction des variables indépendantes catégorisées n'est possiblement pas la même. En effet, les informations nécessaires à la création de ces variables sont manquantes dans l'article (Caiola et Reiter, 2010). On construit les variables dichotomiques suivantes : veuf (1 si état marital=4, 0 si non), divorcé (état marital=5) et célibataire (état marital=7). On réduit ensuite le nombre de classes pour la variable éducation comme suit : école secondaire (valeur 32 à 39 de la variable éducation), quelques études au collège (40), diplôme d'études collégiales (41-42) et études post-collégiales(43 à 46).

Les coefficients obtenus pour cette régression sont présentés dans le tableau 2.7. Les coefficients des variables non-synthétisées ne varient presque pas. Il est cependant plus difficile de conclure sur les coefficients pour les variables race et état marital qui ont plusieurs catégories (ou variables dichotomiques pour représenter la variable initiale). En effet, certaines variables dichotomiques semblent

variées davantage. Quant à la variable sexe, le coefficient demeure relativement constant. Pour appuyer que les coefficients sont sensiblement pareils, on a obtenu les erreurs relatives pour tous les coefficients et on a obtenu une erreur relative moyenne de 8.8% avec un maximum atteint de 33.76% pour la variable dichotomique divorcé dans le premier ensemble. Dans le second ensemble, on a obtenu une erreur relative moyenne de 8.4% avec un maximum de 16.9% pour la variable dichotomique célibataire.

	Jeu original	Ensemble 1	Ensemble 2
Ordonnée	73.44	73.07	70.84
Femme	-12.86	-12.73	-12.39
Noir	-5.88	-7.13	-6.48
Amérindien	-20.46	-24.50	-19.09
Asiatique	-22.27	-19.85	-22.89
Veuf	7.69	7.84	7.32
Divorcé	-1.15	-1.54	-1.30
Célibataire	-2.79	-2.84	-2.31
Secondaire	10.38	11.08	11.70
Collégial	14.60	15.22	15.52
DEC	11.07	11.88	12.62
Universitaire	16.66	17.12	17.70
Age	0.20	0.20	0.22

Tableau 2.7 – Coefficients obtenus pour la seconde régression.

Les intervalles de confiance sont obtenus de la même façon que précédemment tout comme le chevauchement pour la mesure d'utilité. On note que le chevauchement moyen est de 92.9% et de 92.1% pour les deux ensembles de jeux de données partiellement synthétiques. Ces résultats sont plus élevés dans cette régression que dans la précédente. Un explication plausible est qu'il y a moins de variables initialement et qu'une des variables catégoriques est passée de 7 à 4 catégories. Si une catégorie n'avait que quelques observations dans la première régression après la synthétisation, il est normal de mal estimer son coefficient (qu'il soit plus variable). En regroupant ainsi, on s'assure d'avoir un nombre respectable d'observations par catégorie.

Troisième régression

Cette régression est effectuée avec la racine carrée du paiement de pension alimentaire comme variable dépendante et requiert donc un paiement supérieur à 0. La majorité des observations n'ayant pas de paiement de pension alimentaire, l'échantillon est constitué de 460 observations. La création de la variable dichotomique non-blanc est faite à partir de la variable race. Les valeurs de 2, 3 et 4 sont classées comme étant non-blanc. Les coefficients obtenus sont présentés dans le tableau 2.8. Il est facile de constater dans cette table que les coefficients des variables synthétisées ne sont pas trop affectés par la synthétisation. En fait, l'erreur relative moyenne est de 13.0% et 6.6% pour les deux ensembles de jeux de données.

	Jeu original	Ensemble 1	Ensemble 2
Ordonnée	-63.01	-62.50	-63.03
Femme	2.41	1.08	1.97
Non-blanc	-8.70	-8.02	-7.64
Éducation	2.95	2.96	2.96
NUML18	1.56	1.54	1.52

Tableau 2.8 – Coefficients obtenus pour la troisième régression.

L'utilité moyenne obtenue est de 96.9% pour le premier ensemble de jeux de données synthétiques et de 97.9% pour le second. Ces résultats sont supérieurs à ceux obtenus précédemment. Encore une fois, le nombre de variables utilisées dans le modèle influence sur l'utilité. De plus, on a moins de variables synthétisées dans le modèle de régression. Les autres variables ne changeant pas, les coefficients sont plus facilement estimables.

Conclusion sur l'utilité

L'utilité des données est relativement bonne pour les trois régressions avec une utilité moyenne supérieure à 80%. Malgré la synthétisation des variables sexe et état marital, nous avons obtenu une bonne utilité pour l'interaction entre les deux atteignant même 90% à quelques occasions. On a remarqué une amélioration de l'utilité lorsqu'on utilise moins de variables dans la régression comme dans la dernière alors que 2 des 4 variables utilisées proviennent de variables synthétisées.

2.2 Étude de sensibilité de l'algorithme 1

On souhaite caractériser quand la fonction qui génère les jeux de données synthétiques fonctionne bien. Selon certaines suggestions de Caiola et Reiter (2010), on s'est penché sur deux caractéristiques en particulier. On souhaite d'abord voir s'il existe un choix plus approprié pour le nombre de jeux synthétiques à créer. On s'intéresse également à l'impact de l'ordre de synthétisation des variables sur l'utilité et sur le risque puisque Caiola et Reiter (2010) suggère que l'ordre peut avoir un impact.

2.2.1 Choix du nombre de jeux synthétiques à créer

Le choix usuel de $M = 5$ ou $M = 10$ a été suggéré dans un premier temps dans le livre de Rubin (1987) pour l'imputation multiple et ce choix a été transféré à la génération de jeux synthétiques pour la confidentialité. Les ordinateurs n'étant pas très performants à l'époque, il y a eu peu de démarches (simulations) pour valider ce choix. Est-ce qu'il y aurait un meilleur choix ?

On a décidé de tester différents choix pour le nombre de jeux synthétiques, soit $M = 2$, 3, 4, 5, 10, 20 et 50. On effectue seulement une simulation pour avoir une idée des valeurs intéressantes à tester. On obtient la figure 2.1 à partir du jeu de données provenant de l'article de Caiola et Reiter (2010) où la mesure utilisée pour l'utilité est le chevauchement des intervalles de confiance de la seconde régression (section 2.1.3) et la mesure de risque est le pourcentage de véritable identification. Pour lire

ce graphique qui a été suggéré par Duncan *et al.* (2001), on cherche les points qui maximisent l'utilité et qui minimise le risque de réidentification. On souhaite donc être près du coin inférieur droit. Sur cette figure où le risque de réidentification est en fonction de l'utilité des données, le meilleur choix est définitivement dans l'intervalle allant de 2 à 10 jeux synthétiques. En effet, plus M augmente, plus le risque de réidentification augmente. On se concentre sur cet intervalle (plus précisément sur les valeurs de $M = 2, 3, 4, 5, 6, 8$ et 10) pour faire plusieurs simulations.

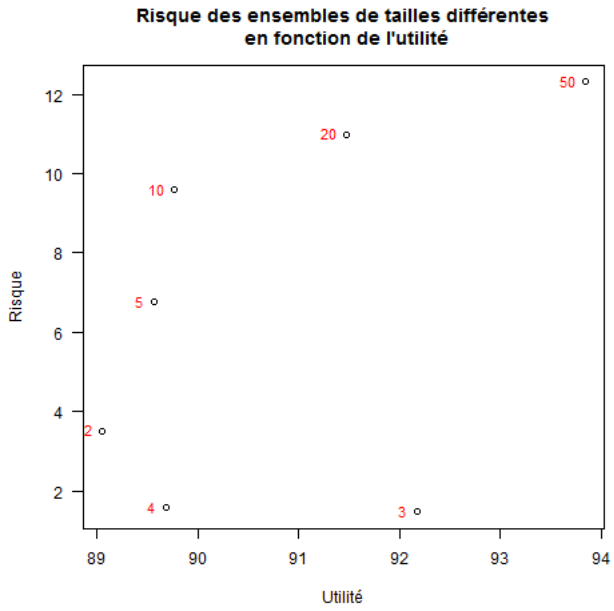

FIGURE 2.1 – Mesure du risque pour différentes tailles des ensembles de jeux synthétiques en fonction de l'utilité des données.

On répète maintenant l'expérience 40 fois pour obtenir la figure 2.2. Puisque le risque est plutôt stable lors des simulations, on a inversé les axes pour pouvoir se concentrer sur la variance de l'utilité. On a construit les barres d'erreur en utilisant la moyenne plus ou moins un écart-type dans le but d'illustrer la variabilité pour la moyenne de l'utilité. La mesure du risque utilisée est la moyenne observée des risques pour les 40 répétitions. Le résultat obtenu précédemment où le choix de $M = 3$ semblait le meilleur n'est pas validé par la figure 2.2. Le meilleur choix dépend de ce que l'on veut. Par exemple, si on souhaite conserver le plus d'utilité possible, un choix proche de 10 serait approprié. Pour ce qui est du choix $M = 2$, malgré un faible risque de réidentification, il n'assure pas une grande utilité selon cet exemple et est très variable en terme d'utilité contrairement aux autres choix possibles.

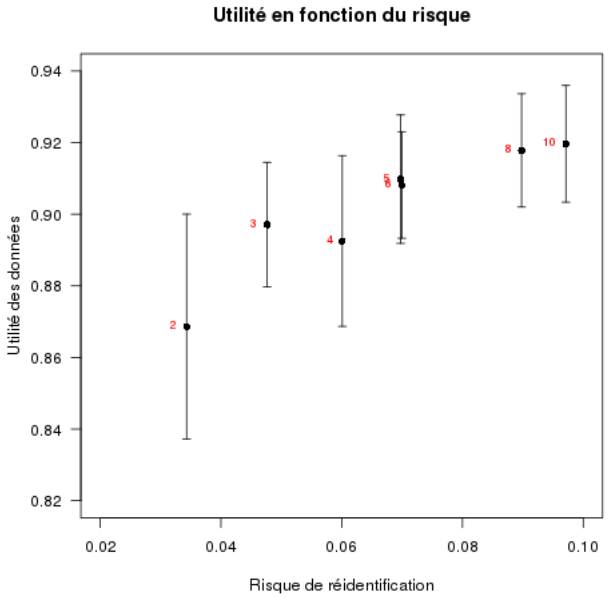

FIGURE 2.2 – Utilité des données pour différentes tailles d'ensembles de jeux synthétiques en fonction du risque de réidentification.

2.2.2 Ordre des variables à synthétiser

Il est mentionné dans l'article de Caiola et Reiter (2010) que l'ordre des variables à synthétiser a une certaine importance lors de la génération des jeux de données partiellement synthétiques. En fait, il considère trois approches pour l'ordre : ordonnée de façon aléatoire, ordonnée en fonction du nombre de catégories pour les variables à synthétiser ou ordonnée en testant toutes les différentes combinaisons pour obtenir le résultat optimal. On veut vérifier si l'ordre influence effectivement les résultats concernant le risque et l'utilité.

On décide de synthétiser le jeu de données de Caiola et Reiter (2010) selon les différentes combinaisons possibles pour comparer le risque et l'utilité dans chaque cas. Les variables à synthétiser sont les mêmes, soit le sexe (S), la race (R) et l'état marital (M) et on génère 5 jeux de données partiellement synthétiques en utilisant les 10 000 observations. Les résultats qui sont présentés dans le tableau 2.9 ont été obtenus après 20 simulations en mesurant l'utilité et le risque tout comme l'écart-type de chaque mesure.

Combinaisons	Utilité moyenne	Écart-type utilité (Point de %)	Risque moyen	Écart-type risque (Point de %)
S-R-M	90.66%	1.99	**6.52%**	0.19
S-M-R	89.91%	1.70	6.54%	0.20
R-S-M	91.12%	1.54	6.80%	0.15
R-M-S	**91.16**%	1.80	7.01%	0.17
M-S-R	90.39%	1.29	6.92%	0.20
M-R-S	90.52%	1.79	7.06%	0.17

Tableau 2.9 – Comparaison de l'utilité et du risque en fonction des différentes combinaisons pour la génération des jeux de données partiellement synthétiques.

On peut conclure que l'ordre n'a pas beaucoup d'impact sur l'utilité ainsi que sur le risque de ré-identification pour cet exemple. La différence la plus importante pour l'utilité est de 1.25 point de pourcentage et pour le risque, elle est de 0.54 point de pourcentage. Cependant, il est possible que l'ordre ait un certain impact sur l'utilité et le risque dans certaines circonstances. Voici un exemple où l'utilité et le risque sont influencés par l'ordre des variables alors qu'une de ces variables dépend des autres.

Simulation démontrant que l'ordre peut jouer un rôle dans la génération des jeux de données partiellement synthétiques

En reprenant le jeu de données de Caiola et Reiter (2010), on modifie une variable, soit la race des observations de la façon suivante :

$$RACE1 = 7 * SEXE + MARITAL.$$

Cette variable prenant des valeurs entre 8 et 22, on la catégorise de la façon suivante afin d'obtenir des catégories de taille similaire :

$$RACE\_cat = \begin{cases} 1 & si \quad RACE1 < 11 \\ 2 & si \quad 11 \leq RACE1 < 15 \\ 3 & si \quad 15 \leq RACE1 < 19 \\ 4 & sinon \end{cases}$$

Puisque la corrélation est vraiment forte avec la variable sexe ($\rho > 0.90$), on ajoute un peu de bruit sur notre variable $RACE\_cat$ à l'aide d'une distribution binomiale :

$$RACE = RACE\_cat + binom(2, 0.5) - 1.$$

Les valeurs possibles de RACE sont dans l'intervalle $[0, 5]$ avec 6 catégories. Les corrélations entre cette variable, le sexe et l'état marital sont présentées dans le tableau 2.10. La corrélation est vraiment élevée entre la nouvelle variable et les deux autres avec une corrélation supérieure à 0.50. La corrélation entre le sexe et l'état marital n'est pas aussi forte.

	Sexe	Race	État marital
Sexe	1		
Race	0.7783	1	
État marital	0.2973	0.5296	1

Tableau 2.10 – Corrélation entre les trois variables à synthétiser.

On teste avec 50 simulations les différents ordres pour la génération des jeux synthétiques avec cette variable remplaçant l'originale. On synthétise toujours ces 3 variables et on crée également 5 jeux synthétiques à chaque réplication. Les graphiques 2.3 et 2.4 obtenus avec les simulations permettent de visualiser l'impact de l'ordre des variables à synthétiser. En effet, le premier graphique met en relation l'utilité avec les différents ordres et le second permet de voir le risque en fonction des ordres. Encore une fois, il s'agit de la moyenne plus ou moins un écart-type pour créer les intervalles.

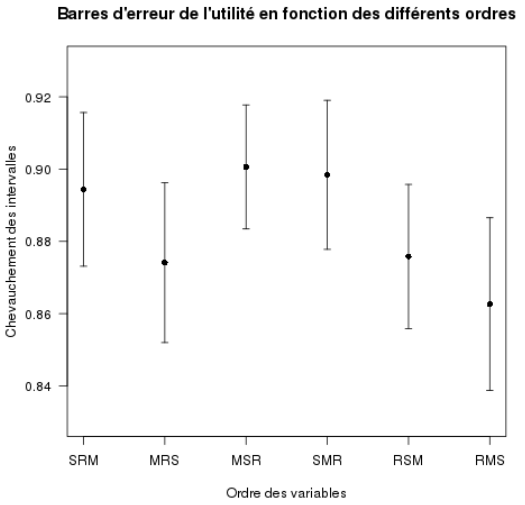

FIGURE 2.3 – Graphique des barres d'erreur de l'utilité en fonction des différents ordres.

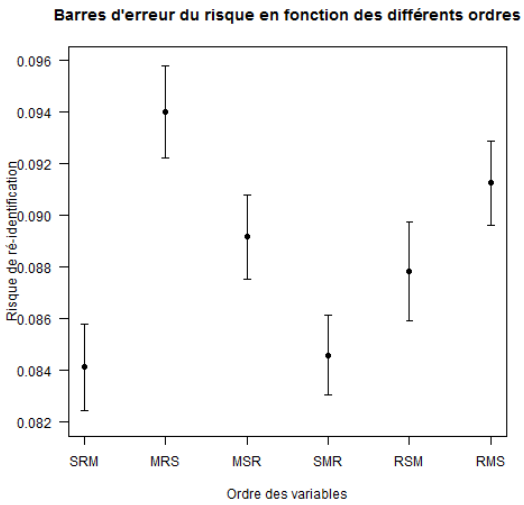

FIGURE 2.4 – Graphique des barres d'erreur du risque en fonction des différents ordres.

Les graphiques 2.3 et 2.4 montrent que l'ordre influence sur l'utilité des données de manière prévisible et ensuite, que l'ordre influence également le risque de réidentification. Lorsqu'on dit de manière prévisible, cela signifie qu'a priori, on s'attendait à ce résultat, soit que l'utilité augmente lorsque la variable race est dans les dernières à synthétiser puisqu'elle dépend des deux autres variables. En fait, si la variable Race vient avant le sexe, l'utilité est nettement inférieure. Évidemment, cela provoque l'effet inverse pour le risque qui, lui, est supérieur.

Dans le tableau 2.11, on constate que la plus grande différence d'utilité se chiffre à 3.8 points de pourcentage et pour le risque, il varie de 1 point de pourcentage. L'ordre optimal est définitivement le suivant : sexe-état marital-race. Celui-ci maximise l'utilité tout en minimisant le risque.

Combinaisons	Utilité moyenne	Écart-type utilité (Point de %)	Risque moyen	Écart-type risque (Point de %)
S-R-M	89.44%	2.13	**8.41%**	0.17
M-R-S	87.41%	2.21	9.40%	0.18
M-S-R	**90.06%**	1.72	8.91%	0.16
S-M-R	89.84%	2.06	8.46%	0.16
R-S-M	87.58%	1.99	8.78%	0.19
R-M-S	86.27%	2.39	9.13%	0.16

Tableau 2.11 – Comparaison de l'utilité et du risque en fonction des différentes combinaisons pour la génération des jeux de données partiellement synthétiques.

Cet exemple permet de conclure qu'il est possible que l'ordre ait un impact sur l'utilité ainsi que le risque et ce, même si dans ce cas-ci, la différence n'est pas très grande.

Dans ce chapitre, on a appliqué la théorie vue au chapitre 1 à un exemple avec des données réelles. On a également testé s'il existait un choix optimal pour le nombre de jeux synthétiques à créer. On a obtenu que ce choix se situe entre 2 et 10 et que cela dépend de ce qu'on cherche à maximiser, soit l'utilité ou le risque. Les simulations concernant l'ordre des variables à synthétiser a permis de conclure qu'effectivement l'ordre pouvait influencer à la fois l'utilité et le risque. Cela vaut la peine de regarder les différentes combinaisons possibles lors de la synthétisation des jeux de données.

Chapitre 3

Test des formules combinant les jeux de données partiellement synthétiques

Dans ce chapitre, on s'intéresse à la validité des formules proposées par Reiter (2003) pour faire de l'inférence à partir de jeux de données partiellement synthétiques créés suivant l'algorithme 1. Plus précisément, on souhaite tester la validité de l'estimateur T_P défini à la proposition 1 du chapitre 1. Bien que Caiola et Reiter (2010) utilise les formules de Reiter (2003) pour mesurer l'utilité de leur mécanisme de génération de jeux de données synthétiques, la validité de T_P dans ce cas n'a jamais été démontrée. On peut en outre douter de la validité de cet estimateur car certaines hypothèses nécessaires pour sa dérivation ne sont pas vérifiées par la méthode de génération de jeux de données synthétiques basée sur les forêts aléatoires. Ce problème a été ignoré dans la littérature jusqu'à présent.

On note que les formules proposées par Reiter (2003) sont basées sur des formules similaires proposées par Rubin (1987) pour faire de l'inférence en combinant plusieurs jeux de données après l'imputation de valeurs manquantes. Comprendre ces formules originales aide à comprendre celles de Reiter (2003).

Ce chapitre comporte donc deux sections. La première porte sur les formules de Rubin (1987) pour l'inférence à partir de jeux de données créés par imputation multiple. On présente les formules, et on illustre leurs limites à l'aide de simulations. Dans la deuxième section, on revient au problème de l'inférence pour des jeux de données synthétiques créés avec des forêts aléatoires pour la protection de la confidentialité, et on teste empiriquement la validité de l'estimateur T_P.

3.1 Formules pour des données manquantes

Lorsqu'on est en présence de plusieurs données manquantes, il est fortement conseillé de remplacer ces valeurs manquantes, soit en utilisant des régressions ou la moyenne par exemple. Il est préférable de faire de l'imputation multiple, car cela permet d'obtenir une meilleure estimation du paramètre d'intérêt ainsi qu'une estimation appropriée de la variance du paramètre. Il y a cependant une

condition, les données manquantes doivent être considérées de type MCAR (*Missing Completely At Random*) ou MAR (*Missing At Random*).

En fait, il existe trois types de données manquantes, nous en avons nommés deux précédemment (MCAR et MAR), la troisième possibilité est que les données soient considérées de type manquant non-aléatoirement (MNAR, *Missing Not At Random*). Définissons chaque type de données manquantes. Le type manquant complètement aléatoire (MCAR) survient si la probabilité qu'une valeur soit manquante n'est pas reliée aux données observées et non-observées de cette observation. Par exemple, si quelqu'un oublie involontairement de répondre à une question, il s'agit d'une donnée manquante de type MCAR. On dit que les données sont manquantes aléatoirement (MAR) si une valeur manquante conditionnelle aux valeurs observées est indépendante des valeurs non-observées. Dans le dernier cas, les valeurs ne sont pas aléatoirement manquantes, car la probabilité qu'une valeur d'une observation soit manquante dépend de la valeur sous-jacente et cette dépendance demeure même en ayant les données observées (Carpenter et Kenward, 2012). Par exemple, si un individu ne répond pas à une question portant sur son revenu annuel et qu'aucune autre question corrélée à cette question ne fait partie du questionnaire, il s'agit alors d'une donnée manquante de type MNAR. Cependant, si une variable corrélée est présente et que celle-ci explique la probabilité de non-réponse, disons le montant payé en impôt, alors la valeur manquante est de type MAR. Un analyste doit évaluer de quel type sont les données manquantes.

Pour faire de l'imputation multiple, on doit créer plusieurs jeux de données imputés pour considérer toute l'incertitude entourant le modèle d'imputation. Voici un aperçu de l'algorithme pour l'imputation de données :

Aperçu de l'algorithme pour la génération de jeux de données imputés (Carpenter et Kenward, 2012)

1. Pour $k = 1...m$: imputation des valeurs manquantes en utilisant la distribution des valeurs manquantes sachant les valeurs observées pour obtenir m jeux de données imputés.
2. Ajustement du modèle pour obtenir les paramètres d'intérêt (par exemple, ajustement d'une régression pour obtenir les coefficients) pour tous les jeux imputés. On obtient m estimés pour chaque coefficient d'intérêt, soit β_k, ainsi que m estimés de la variance de chaque coefficient, $Var(\beta_k)$.
3. On combine les différents résultats en utilisant les règles de Rubin (voir proposition 2 ci-dessous).

3.1.1 Formules combinant les jeux de données imputés

La proposition 1 du chapitre 1 permet de combiner les résultats pour des jeux de données synthétisés pour la confidentialité. Similairement à cette proposition, voici les formules pour combiner les différents jeux de données imputés en raison de données manquantes où la principale différence est

lorsqu'on évalue la variance.

Proposition 2 *(Rubin, 1987)*

Soit β, le paramètre que l'on souhaite estimer et soit m, le nombre d'imputations effectuées, l'estimation de β est :

$$\widehat{\beta}_{MI} = \sum_{k=1}^{m} \widehat{\beta}_k / m$$

où $\widehat{\beta}_k$ représente l'estimation du paramètre avec le k^e jeu de données imputé.

Un estimateur de la variance du coefficient est obtenu ainsi :

$$T_M = \widehat{W} + \left(1 + \frac{1}{m}\right)\widehat{B} \tag{3.1}$$

où $\widehat{W}$ est la moyenne de la variance intra jeu de données (within), $\widehat{W} = \frac{\sum_{k=1}^{m} \widehat{\sigma}_k^2}{m}$ et où $\widehat{\sigma}_k^2$ est la variance associée au paramètre dans le k^e jeu de données imputé. $\widehat{B}$ est la variance entre les paramètres des différents jeux imputés (between), $\widehat{B} = \frac{1}{m-1}\sum_{k=1}^{m}(\widehat{\beta}_k - \widehat{\beta}_{MI})^2$.

Des exemples de simulation qui illustrent ces formules, se trouvent dans l'annexe A tout comme la démonstration de cette proposition afin de ne pas alourdir inutilement le mémoire. Ces exemples permettent d'affirmer que la formule 3.1 fonctionne bel et bien pour évaluer la variance lorsqu'on impute des valeurs manquantes.

3.1.2 Discussion des limites

La procédure d'imputation est à la fois bayésienne et fréquentiste. Elle se base sur la distribution bayésienne prédictive pour effectuer les imputations. De plus, elle utilise la loi des grands nombres pour obtenir les intervalles de confiance des paramètres d'intérêt. L'imputation multiple utilise deux approximations importantes (Carpenter et Kenward, 2012). Premièrement, on utilise une distribution normale multivariée asymptotique comme distribution a posteriori. Deuxièmement, on estime la moyenne et la variance de cette distribution a posteriori par l'estimé et la matrice de variance-covariance obtenus à partir des équations estimantes du jeu complet. En d'autres mots, on suppose que l'espérance et la variance de la distribution a posteriori du paramètre d'intérêt sont suffisamment bien approximées par l'ajustement d'un modèle directement aux données imputées.

Si une des approximations n'est pas valide, les formules combinant des jeux imputés pour évaluer la variance ne le seront pas également. En effet, si les données ne proviennent pas d'une distribution bayésienne prédictive, la seconde approximation n'est pas nécessairement valide. Cela implique par la suite que les estimations obtenues ne le seront pas également tout comme les intervalles de confiance.

Une hypothèse nécessaire pour obtenir des estimations valides des variances fréquentistes est que le modèle d'imputation et le vrai modèle soient compatibles. C'est-à-dire que la distribution obtenue serait la même pour une distribution prédictive bayésienne que pour une procédure bayésienne complète si celle-ci existait (Carpenter et Kenward, 2012).

Description des paramètres modifiables

Cette section permet de suivre différents exemples qui seront traités dans les prochaines pages, autant pour les données imputées que pour les données synthétisées.

D'abord, on génère un jeu de données de la façon suivante à moins d'avis contraire :

$$X \sim N_2\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}10 & 5\\5 & 10\end{pmatrix}\right)$$

$$y = X_1 + X_2 + \epsilon$$

où $\epsilon \sim N(0, 100)$. X représente alors deux variables explicatives.

Dans cette section, on est en présence de données manquantes, on retire des observations en fonction du résultat de l'expérience de Bernoulli associée à chaque observation ou selon un tirage aléatoire d'un certain nombre d'observations pour avoir des données manquantes dans le jeu de données simulé. Dans la section sur la confidentialité, lorsqu'on synthétise, deux choix sont possibles, soit on modifie chaque observation comme au chapitre 1, soit on en sélectionne certaines au hasard.

La méthode d'imputation ou de synthétisation sélectionnée varie d'un exemple à l'autre, cela peut être avec la distribution prédictive a posteriori ou par forêts aléatoires. La distribution prédictive a posteriori est utilisée lorsqu'on synthétise ou qu'on impute avec des régressions. En utilisant les forêts aléatoires pour obtenir les nouvelles valeurs, il y a deux options, soit on effectue un tirage pour obtenir une nouvelle valeur comme dans l'algorithme 1, soit on prend tout simplement la valeur prédite par les arbres de la forêt aléatoire.

Le nombre de variables à synthétiser ou à imputer est évidemment un paramètre modifiable. Pour les données manquantes, seulement le cas simple d'une variable à imputer sera étudié. Dans le cas de données synthétisées, on s'intéressera d'abord au cas simple où une seule variable est altérée puis à un cas plus complexe avec deux variables synthétisées.

Le dernier paramètre important est le choix du paramètre à estimer. Il y a deux choix, soit on estime la moyenne de la variable synthétisée, soit on s'intéresse aux coefficients d'une régression impliquant la variable synthétisée.

Ci-dessous se trouve un exemple où les formules ne sont pas en mesure d'estimer la variance d'une variable imputée à travers les jeux obtenus par imputation.

Exemple 1 *Exemple où les formules d'imputation de la proposition 2 ne s'appliquent pas*

On utilise un jeu de données simulé selon la section 3.1.2. On est en présence de données manquantes pour une variable, y *et on veut imputer celle-ci avec des forêts aléatoires. Celles-ci s'ajustent en posant* y *comme variable dépendante et les autres variables disponibles comme variables indépendantes.*

Pour les valeurs considérées manquantes, on supprime de façon aléatoire certaines valeurs du jeu simulé en fonction du résultat d'une expérience de Bernoulli avec paramètre $p = 0.387$.

Puisque y *est une variable continue, il est difficile d'obtenir une distribution pour chaque observation manquante à partir de forêt aléatoire puisqu'il n'y a pas de matrice de votes. Il n'y a alors pas de tirage effectué pour obtenir une nouvelle valeur. Celle-ci s'obtient en prenant la valeur prédite par la forêt aléatoire.*

On calcule ensuite la moyenne de y *ainsi que la variance associée à celle-ci pour chaque jeu imputé (*$m = 100$ *jeux imputés au total). On réplique ce processus* $3\ 000$ *fois.*

En fait, il existe déjà une fonction en R qui impute les valeurs manquantes avec des forêts aléatoires, soit mice.impute.rf. On compare les résultats obtenus par cette fonction avec l'imputation faite par la fonction randomForest (Forêts aléatoires) tout en utilisant les règles de Rubin énoncées dans la proposition 2. Les résultats sont présentés dans le tableau 3.1 après $3\ 000$ *réplications. On retrouve dans ce tableau la moyenne des réplications pour la variance, la valeur espérée de celle-ci ainsi que l'erreur relative. La valeur espérée est simplement la variance du coefficient à travers les réplications, ce que l'on cherche à estimer.*

Paramètre	Méthode d'imputation	Moyenne	Espérée	Erreur relative
T_M	Forêts aléatoires (500 arbres)	0.9546	2.1293	55.2%
T_M	*mice.impute.rf* (10 arbres)	1.6166	2.1218	23.8%
T_M	Forêts aléatoires (10 arbres)	1.0410	2.0665	49.6%

Tableau 3.1 – Tableau résumant les résultats obtenus lors de la simulation de $3\ 000$ réplications de 100 jeux imputés par forêts aléatoires.

On sous-estime la véritable variance lorsqu'on utilise une forêt aléatoire pour imputer des valeurs manquantes puisqu'on observe une variance de 2.1293 *comparativement à l'estimation qui est de* 0.9543*, soit une erreur relative de* 55.2% [1]*. Cette erreur est causée par le fait que l'approximation bayésienne n'est probablement pas appropriée. Pourtant, on a également testé la fonction qui sert à imputer les données manquantes à partir de forêts aléatoires, mice.impute.rf avec cet exemple. La variance moyenne obtenue est* 1.6166 *comparativement à la variance des moyennes des jeux imputés qui est* 2.1218*. Ce résultat est un peu plus près de la vraie variance, mais on la sous-estime encore*

1. On a expérimenté avec différentes valeurs de K, n, m et avec un jeu de données différent qui concluent tous qu'on sous-estime la variance. Voir le code en annexe C

avec une erreur relative de 23.8%*. Puisque la différence est relativement grande entre les deux méthodes d'imputation, la différence peut être causée par les valeurs par défaut de ces fonctions, par exemple le nombre d'arbres créés qui diffère dans les deux méthodes. On teste la méthode des forêts aléatoires avec* 10 *arbres. Le résultat n'est pas mieux puisqu'on sous-estime toujours la vraie variance avec une erreur relative de près de* 50% *et l'estimation de la variance ne se rapproche pas de celle obtenue avec mice.impute.rf. La différence principale se trouve dans la prédiction (la création des distributions) où la fonction mice.impute.rf prédit d'abord les nœuds des observations qui ne sont pas manquantes avec un arbre, puis la fonction obtient les nœuds pour les observations manquantes avec cet arbre. Elle cherche ensuite la valeur de l'observation manquante*[2]*. Ce n'est pas cela qui est effectué lorsqu'on impute avec les forêts aléatoires. Il serait cependant possible de le faire en utilisant un code similaire à la fonction mice.impute.rf.*

Exemple 2 *Suite Exemple 1 : cas discret avec et sans tirage*

Dans cet exemple, on catégorise la variable y *de la façon suivante :*

$$y = (X_1 + X_2 + \epsilon)/2$$

où $\epsilon \sim N(0, 100)$*. Ensuite,* y *est arrondi à l'entier le plus près afin qu'il soit catégorique.*

Les données manquantes sont retirées de la même manière que dans l'exemple 1, soit en supprimant des observations en fonction de l'expérience de Bernoulli. On teste alors deux méthodes pour obtenir les valeurs imputées. La première méthode remplace les données manquantes par la valeur prédite par les arbres de la forêt aléatoire. Pour ce qui est de la seconde méthode, on utilise le principe vu dans le chapitre 1 pour obtenir la distribution de probabilité de chaque observation et faire un tirage (comme l'algorithme 1), cela nous donne alors une valeur qui en remplace une manquante.

On obtient les résultats du tableau 3.2 après $3\ 000$ *réplications (toujours avec* $m = 100$*). Dans ce tableau, on a les résultats pour les deux méthodes d'imputation, soit avec et sans tirage. On retrouve la moyenne des réplications, la valeur espérée ainsi que l'erreur relative.*

Paramètre	Méthode d'imputation (param. modifiés)	Moyenne des rép.	Espérée	ER
T_M	Sans tirage	0.1043	0.1904	52.2%
T_M	Avec tirage	0.1265	0.1440	12.2%
T_M	Avec tirage, $var(X_i) = 12$,$cov(X_1, X_2) = 7$	0.1365	0.1625	16%

Tableau 3.2 – Résultats des diverses simulations pour l'imputation avec et sans tirage en utilisant les forêts aléatoires.

2. Pour plus de détails, la fonction est visible dans R dans la librairie *mice* version 2.22.

Encore une fois, on sous-estime la variance de la moyenne avec une erreur relative de 52.2%*. La formule* T_M *ne semble pas fonctionner lors de l'imputation par forêts aléatoires sans effectuer de tirage.*

Le résultat est mieux lorsqu'on effectue un tirage d'une distribution puisqu'on obtient une erreur relative de 12.2%*. On a effectué une seconde simulation en modifiant les paramètres suivants :* $var(X_i) = 12$ *pour* $i = 1, 2$ *et* $cov(X_1, X_2) = 7$*. Cette simulation n'obtient pas une erreur relative désastreuse, elle est de* 16%*. Cependant, on sous-estime encore la véritable variance.*

Le problème avec l'imputation par forêts aléatoires peut se situer à différents endroits. Il peut être causé par le fait que l'estimateur ne possède pas une distribution normale asymptotique pour les données complètes ni pour les données incomplètes ou encore, les valeurs manquantes ne sont peut-être pas générées de la vraie distribution prédictive a posteriori. Un récent article a par ailleurs montré à l'aide de plusieurs simulations que l'utilisation des forêts aléatoires pour imputer les valeurs manquantes conduisait à une estimation biaisée de la variance (Bartlett, 2014).

Une possibilité pour améliorer les résultats est l'utilisation de mélanges de loi normale qui permettent entre autres de stabiliser les inférences (Steele *et al.*, 2010). Cette méthode a été appliquée, mais cela n'a pas changé les résultats dans notre cas.

3.2 Test des formules utilisées pour des jeux de données partiellement synthétiques

Dans cette deuxième section, on revient au problème de l'inférence pour des jeux de données synthétiques créés avec des forêts aléatoires pour la protection de la confidentialité. On teste empiriquement la validité de l'estimateur T_P provenant de la proposition 1.

3.2.1 Comparaison des formules : cas des données imputées et des données synthétisées

Il y a une distinction dans le calcul de la variance pour les cas suivants : l'imputation des données manquantes et la génération de jeux de données partiellement synthétiques. En effet, les formules sont respectivement les suivantes :

$$T_M = \overline{v}_k + \left(1 + \frac{1}{m}\right) b_k \text{ (équation 3.1)}$$

$$T_P = \overline{v}_k + \frac{b_k}{m} \text{ (équation 1.1).}$$

T_M est composé de trois termes, soit $\overline{v}_k$ qui estime la $Var(q_{obs})$ (q_{obs} représente le coefficient observé), b_k/m qui estime la variance additionnelle causée par le fait qu'on utilise un nombre fini d'imputations et b_k pour tenir compte de la variabilité causée par la non-réponse (Reiter, 2003). On n'a

justement pas besoin de ce dernier terme lors de l'évaluation de la variance sur des jeux de données partiellement synthétiques, car ce sont toujours les mêmes observations qui sont synthétisées : il n'y a pas de hasard dans le choix des observations alors que, pour l'imputation des données manquantes, on suppose que les observations manquantes sont aléatoires. Par exemple, si on synthétise les valeurs de y pour ceux qui ont une valeur supérieure à 5, ce sera toujours les mêmes qui seront synthétisées avec le jeu de données original. Il existe également une formule pour calculer la variance lorsqu'un jeu de données est complètement synthétique, soit $T_C = \left(1 + \frac{1}{m}\right) b_k - \overline{v}_k$. Celle-ci ne sera pas utilisée dans ce mémoire.

Exemple 3 *Simulation pour obtenir la variance T_P*

Soit les données simulées selon l'exemple 2 où y est catégorique et deux variables indépendantes X.

Les observations qui doivent être synthétisées sont choisies selon l'expérience de Bernoulli avec $p = 0.387$.

On utilise des régressions pour obtenir de nouvelles valeurs. On calcule les coefficients avec toutes les informations disponibles ainsi que la variance associée à ces coefficients avec les formules suivantes :

$$\widehat{\beta}_{ori} = (X'X)^{-1}X'y$$
$$\boldsymbol{V}_{ori} = Var(\widehat{\beta}_{ori}) = \sigma^2(X'X)^{-1}$$

où ori signifie original puisqu'on utilise toutes les données disponibles pour obtenir ces estimations. On calcule de nouvelles valeurs de y pour les observations à synthétiser avec l'équation suivante :

$$\tilde{Y}_{M,k} = X_M(\widehat{\beta}_{ori} + bi_k) + e_k$$

où $bi_k \sim N_2(\boldsymbol{0}, \boldsymbol{V}_{ori})$ ($\boldsymbol{0}$ représente un vecteur de 0) et $e_k \sim N(0, \tilde{\sigma}^2/n_{obs})$ avec $\tilde{\sigma}^2 = \boldsymbol{V}_{ori}/(X'X)^{-1}$.

On s'intéresse à la moyenne de la variable y après la synthétisation. On obtient la variance des moyennes de y après les réplications :

$$Var(\widehat{\mu}) = 0.9094.$$

L'objectif est d'obtenir une estimation de cette variance. En regardant la variance obtenue avec la formule d'imputation de la proposition 2 ainsi qu'avec la formule T_P, soit l'équation 1.1, on obtient les résultats présentés dans le tableau 3.3.

	Résultats	Erreur relative
$var(\widehat{\mu})$	0.9094	-
T_M	1.1973	31.7%
T_P	0.9361	2.9%

Tableau 3.3 – Tableau résumant les différentes valeurs obtenues pour l'estimation de la variance.

Clairement, on surestime la variance associée à notre estimation de la moyenne après la génération des jeux synthétiques avec la formule T_M (3.1) puisque l'erreur relative est 31.7%. En appliquant la formule T_P (1.1), on obtient un meilleur résultat, soit une variance moyenne de 0.9361 avec une erreur relative de 2.9%. Il s'agit effectivement du terme b_k qui est en trop lorsqu'on synthétise les données pour conserver la confidentialité des observations.

3.2.2 Limites des formules de la proposition 1 combinant des jeux de données partiellement synthétiques

On se questionne sur les limites de la formule T_P. Dans l'exemple précédent, on a créé des jeux synthétiques sans utiliser les forêts aléatoires en générant de nouvelles valeurs pour quelques observations d'une variable. Est-ce que les formules fonctionnent en utilisant les forêts aléatoires ? Fonctionnent-elles lorsqu'on synthétise certaines observations d'une variable avec une forêt aléatoire ? Que se passe-t-il lorsqu'il y a plus d'une variable à synthétiser ? On s'intéresse également à la couverture des intervalles de confiance des jeux synthétiques. Contiennent-ils le paramètre original ?

Vérification de la formule 1.1 pour calculer la variance dans différents cas

Tout d'abord, il est possible de montrer que la formule T_P de la proposition 1 s'applique lorsqu'on synthétise toutes les valeurs d'une variable discrète en utilisant l'algorithme 1 vu au chapitre 1 ou simplement en utilisant les valeurs prédites comme valeurs synthétiques. La méthode du chapitre 1 modélise la variable à synthétiser en fonction des autres variables. Puis après avoir obtenu la distribution, la méthode tire de nouvelles valeurs de celle-ci. Voici un exemple utilisant ces deux méthodes pour obtenir les valeurs synthétiques et qui utilise la formule T_P pour évaluer la variance.

Exemple 4 *Exemples de génération de jeux synthétiques par forêts aléatoires avec et sans tirage*

Soit un jeu de données simulé de la façon suivante :

$$X \sim N_2\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 0.6 \\ 0.6 & 1 \end{pmatrix} \right)$$

$$Y = 3 \cdot X_1 - 1 \cdot X_2 + \epsilon$$

où $\epsilon \sim N(0, 9)$. *Puisqu'on veut traiter* Y *dans un cas discret, on arrondit* Y *à l'entier le plus près. Cette variable peut prendre différentes valeurs tout dépendant des valeurs de* X. *Lors d'une simulation, les valeurs de* Y *sont situées entre* -11 *et* 11. *Pour augmenter les variances au final, on a également considéré un jeu de données simulé où les variables X provenaient d'une loi normale multivariée ayant une moyenne* 0 *pour les deux variables, une variance de* 3 *également pour les deux variables et une covariance de* 1.6 *entre les deux variables.*

On synthétise entièrement la variable Y*, il n'y a pas de sélection à faire. On utilise les forêts aléatoires pour synthétiser cette variable sans effectuer de tirage dans un premier cas, puis l'algorithme 1 vu au chapitre 1 pour obtenir les jeux de données partiellement synthétiques*[3]*. Ainsi, on peut comparer les deux méthodes.*

On s'intéresse aux coefficients de la régression de Y *en fonction de* X *ainsi qu'à la variance des coefficients comme dans les exemples vus précédemment.*

On obtient les résultats du tableau 3.4 après $3\,000$ *réplications avec* 10 *ou* 50 *jeux synthétiques et* 300 *observations. Dans ce tableau, on a les variances observées, la variance espérée et l'erreur relative pour différentes méthodes.*

Méthode			Observé (T_P)		Espérée		ER	
	$V(X_i)$	Cov	$\widehat{\beta}_1$	$\widehat{\beta}_2$	$\widehat{\beta}_1$	$\widehat{\beta}_2$	$\widehat{\beta}_1$	$\widehat{\beta}_2$
Forêts aléatoires sans tirage	1	0.6	0.7397	0.7425	0.7492	0.7283	1.3%	2%
Algorithme 1	1	0.6	0.0494	0.0492	0.0470	0.0462	5.1%	6.5%
Algorithme 1	3	1.6	0.1417	0.1414	0.1340	0.1272	5.7%	11.2%
Algorithme 1, $m = 50$	3	1.6	0.1350	0.1350	0.1252	0.1220	7.8%	10.7%

Tableau 3.4 – Résumé des résultats obtenus pour la synthétisation complète d'une variable dans un cas simple.

Les résultats du tableau 3.4 permettent de constater que l'erreur relative ne diminue pas lorsqu'on augmente le nombre de jeux de données synthétiques à 50. *En effet, l'erreur relative ne diminue que de* 0.5% *dans un cas et augmente de* 2.1% *dans l'autre. Également, la formule 1.1 semble mieux fonctionner lorsque la matrice de variance-covariance des variables indépendantes est petite. C'est-à-dire que plus la variance de ces variables augmente, plus cela varie lorsqu'on tente de prédire avec les forêts aléatoires. On peut conclure que la formule* T_P *fonctionne pour évaluer la variance lorsqu'on génère des jeux de données partiellement synthétiques quand on synthétise une seule variable.*

Cependant, la formule T_P ne s'applique pas dans une situation où l'on synthétise seulement quelques valeurs d'une variable à l'aide de forêts aléatoires sans effectuer de tirage. L'exemple qui suit met en évidence une limite de la formule T_P. En effet, on sous-évalue la véritable variance lorsqu'on s'intéresse à la variance de la moyenne de la variable synthétisée. La formule T_P ne fonctionne pas

3. Voir dans l'annexe B pour regarder la fonction GenSyn2.

également lorsqu'on s'intéresse à la variance des coefficients de la régression peu importe que la variable synthétisée soit la variable dépendante ou indépendante.

Exemple 5 *Exemple de synthétisation de certaines observations sur une variable au lieu de synthétiser entièrement celle-ci*

Soit le jeu de données simulé de façon identique à l'exemple 4. Pour certains cas, on génère les données en modifiant la matrice de covariances pour avoir une variance supérieure à la fin. Les détails sont fournis dans le tableau 3.5.

On synthétise certaines observations sélectionnées au hasard pour une variable en particulier, soit 100 *des* 300 *observations pour la variable* Y *lors de chaque réplication (N.B. Pas lors de la création de chaque jeu synthétique, il faut que ce soit toujours les mêmes observations sélectionnées pour chaque jeu).*

Les valeurs de ces observations sont alors remplacées par les prédictions d'une forêt aléatoire (sans tirage). On teste également la fonction GenSynPAR[4] *qui permet de faire comme au chapitre 1 à un détail près. Ce détail est qu'on peut choisir les observations qui seront synthétisées au lieu d'altérer toutes les observations.*

On souhaite calculer la moyenne de la variable discrète Y*, on utilise* 10 *ou* 50 *jeux synthétiques ainsi que* $3\ 000$ *réplications pour obtenir les résultats du tableau 3.5.*

Paramètres	Var obtenue (T_P)	Espérée	ER
$var = 1$, $cov = 0.6$, $m = 10$	0.0495	0.0678	27%
$var = 3$, $cov = 1.6$, $m = 50$	0.0951	0.1137	16.4%
$var = 1$, $cov = 0.53$, $m = 50$	0.0507	0.0671	24.4%
GenSynPAR $var = 1$, $cov = 0.53$, $m = 50$	0.0553	0.0559	0.9%

Tableau 3.5 – Résultats de la formule T_P des différents modèles testés lorsqu'on synthétise certaines observations d'une variable.

Les résultats semblent similaires pour le premier cas du tableau 3.5, mais lorsqu'on calcule l'erreur relative, on obtient une erreur de 27%*. De plus, on sous-estime la vraie variance ce qui n'est pas souhaitable. Ces résultats ont été obtenus en utilisant seulement* 10 *jeux de données partiellement synthétiques et* $3\ 000$ *réplications. En augmentant le nombre de jeux de données synthétiques à* 50 *ainsi que les variances-covariances, on sous-évalue la variance bien que l'erreur relative a diminué à* 16.4%*. En reprenant des variances similaires à notre jeu de départ et avec* $m = 50$*, on note une légère amélioration comparée à l'erreur relative du départ avec un pourcentage de* 24.4%*. Si on compare les résultats avec ceux obtenus par la fonction GenSynPAR, le fait de tirer au hasard améliore grandement l'estimation de la variance. En effet, on obtient une erreur relative de seulement* 0.9%*.*

4. Voir la fonction en annexe C

On s'intéresse également à la variance associée aux coefficients de régression. On utilise le même principe pour générer des jeux de données, soit l'utilisation de forêts aléatoires et on synthétise toujours Y en choisissant au hasard 100 observations parmi les 300 lors de chaque réplication. L'unique différence est que X_1 est arrondi à l'entier le plus près puisque cette variable sera synthétisée dans une des simulations. Cela va permettre de vérifier si la formule 1.1 fonctionne lorsqu'une variable indépendante dans une régression est synthétisée. Après $3\ 000$ réplications, on obtient les résultats présentés dans le tableau 3.6 avec $m = 50$. On a noté les coefficients moyens des variables ainsi que la variance observée, la variance espérée et l'erreur relative. On a également testé la fonction GenSynPAR en synthétisant partiellement Y ou X_1.

Variable Synt.	Coefficient moyen		Var observée (T_P)		Espérée		ER	
	X_1	X_2	X_1	X_2	X_1	X_2	X_1	X_2
Y	2.9294	-0.8807	0.0299	0.0325	0.0555	0.0538	46.0%	39.5%
Y (GenSynPAR)	3.0007	-0.9998	0.0382	0.0413	0.0369	0.0418	3.3%	1.3%
X_1	3.2276	-1.0852	0.0398	0.0398	0.0486	0.0503	18.2%	20.87%
X_1 (GenSynPAR)	3.0026	-1.0041	0.0383	0.0415	0.0381	0.0428	0.5%	2.9%

Tableau 3.6 – Résultats des régressions après la synthétisation intra-variable.

L'estimation de la variance par la formule T_P ne fonctionne pas lorsque les observations synthétiques ne proviennent pas d'un tirage. En effet, peu importe que la variable synthétisée soit une variable dépendante ou indépendante du modèle de régression, les erreurs relatives sont élevées dans les deux cas atteignant 20.87% et 46% comme maximum. De plus, on sous-estime encore la véritable variance. L'unique point positif est que les coefficients moyens obtenus ressemblent aux coefficients originaux. En utilisant la fonction GenSynPAR, l'estimation de la variance fonctionne parfaitement dans les deux cas, l'erreur relative demeure inférieure à 5%.

Quand on synthétise avec des forêts aléatoires sans effectuer de tirage, la formule T_P ne s'applique pas lorsqu'on synthétise quelques valeurs d'une variable avec peu de jeux de données synthétiques. Dans ce cas, il faut éviter de synthétiser seulement certaines valeurs au lieu de synthétiser l'ensemble des valeurs d'une variable lors de la génération des jeux de données partiellement synthétiques avec les forêts aléatoires. Si on génère plusieurs jeux de données synthétiques, l'erreur relative devient plus petite, mais on sous-estime toujours la véritable variance. Il faut cependant disposer de beaucoup de temps puisque la fonction *randomForest* créant les forêts aléatoires demande du temps à exécuter. Tout dépendant de la taille du jeu de données, la fonction peut prendre entre quelques heures à plus d'une journée. Dans un autre cas, on a également remarqué que la formule T_P fonctionne très bien lorsque les jeux sont créés à l'aide d'un tirage. Il est important d'effectuer les tirages pour obtenir les nouvelles valeurs lorsqu'on synthétise quelques observations d'une variable.

Les exemples vus jusqu'à présent ne considèrent que la simulation d'une seule variable. Qu'advient-il lorsqu'on synthétise plus d'une variable ? Est-ce que la formule T_P fonctionne toujours ? Il est possible de constater que la formule T_P fonctionne dans certains cas seulement lorsqu'on synthétise

deux variables parmi 5 et ce, que ce soit pour calculer la variance de la moyenne ou la variance des coefficients de régression.

Exemple 6 *Simulations sur le comportement de T_P lors de la génération de deux variables synthétiques*

Le jeu de données est généré de la façon suivante :

$$X \sim N_4\left(\boldsymbol{0}, \begin{pmatrix} 1 & 0.6 & 0.3 & 0.7 \\ 0.6 & 1 & 0.4 & 0.6 \\ 0.3 & 0.4 & 1 & 0.7 \\ 0.7 & 0.6 & 0.7 & 1 \end{pmatrix}\right)$$

où $\boldsymbol{0}$ représente un vecteur de 0 de dimension 4. On arrondit les valeurs obtenues pour que les variables soient catégoriques variant dans la majorité des cas entre -3 et 3 et on calcule ensuite Y :

$$Y = 3 \cdot X_1 - 1 \cdot X_2 + 2 \cdot X_3 - 3 \cdot X_4 + \epsilon$$

où $\epsilon \sim N(0, 9)$. On arrondit Y pour qu'il soit également catégorique.

On synthétise entièrement deux variables. On teste deux combinaisons que l'on retrouve dans le tableau 3.7. On utilise ensuite l'algorithme 1 pour synthétiser les deux variables, Y et X_1 dans un cas et X_1 et X_2 dans l'autre. On réplique $5\,000$ fois et on crée 10 ou 20 jeux synthétiques lors de chaque réplication. On calcule pour chaque jeu synthétique les coefficients de la régression de Y en fonction de X et la variance associée à ces coefficients. Les résultats concernant les valeurs des coefficients, les variances observées et les variances espérées sont présentés dans le tableau 3.7.

Dans l'ordre Y-X_1, la plus grande erreur relative vient définitivement du coefficient de la dernière variable qui est X_4. Cette erreur est de 13.1%. Dans l'ensemble, les résultats sont bons et ne surestiment pas la variance avec 10 jeux synthétiques lorsque deux variables sont synthétisées dont la variable dépendante dans la régression. Dans l'ordre X_1-Y, les résultats obtenus avec 20 jeux synthétiques ne sont pas concluants, les erreurs relatives atteignent 54.4%. On surestime clairement la véritable variance. À titre comparatif, on a ajouté ce qu'on observe lorsqu'on ne synthétise aucune variable (c'est-à-dire qu'on copie le jeu de données initial $m = 50$ fois pour créer les jeux synthétiques). Il y a une petite erreur relative ce qui est normal puisqu'on ne s'attend pas à obtenir une estimation exacte de la variance avec un petit nombre de jeux de données.

Dans le tableau 3.8, on a les résultats pour d'autres combinaisons de variables à synthétiser. Ces résultats ont été obtenus avec 20 jeux synthétiques et $5\,000$ réplications. Il est évident selon ce tableau qu'on surestime la variance obtenue lors des réplications. La formule T_P ne semble pas s'appliquer dans ces exemples avec moins de 20 jeux de données synthétiques.

Variables synthétisées	Coefficient	T_P	Espérée	ER
Aucune ($m = 50$)	β_1	0.0578	0.0571	1.2%
	β_2	0.0458	0.0459	0.3%
	β_3	0.0523	0.0516	1.4%
	β_4	0.0849	0.0856	0.8%
Y et X_1 ($m = 10$)	β_1	0.0565	0.0521	8.4%
	β_2	0.0394	0.0378	4.2%
	β_3	0.0440	0.0427	3.0%
	β_4	0.0724	0.0640	13.1%
X_1 et X_2 ($m = 20$)	β_1	0.0903	0.0585	54.4%
	β_2	0.0650	0.0530	22.7%
	β_3	0.0613	0.0505	21.4%
	β_4	0.0965	0.0677	42.5%

Tableau 3.7 – Résultats obtenus de la formule 1.1 pour chaque coefficient de régression pour deux ordres différents et lorsqu'aucune variable n'est synthétisée.

Variables synthétisées		Paramètre 1	Paramètre 2	Paramètre 3	Paramètre 4
X_1 et X_4	$var(\widehat{\beta})$	0.0566	0.0445	0.0548	0.0723
	T_P	0.0931	0.0573	0.0688	0.0991
	ER	64.5%	28.8%	25.5%	37.1%
Y et X_3	$var(\widehat{\beta})$	0.0459	0.0358	0.0364	0.0457
	T_P	0.0513	0.0453	0.0456	0.0733
	ER	11.7%	26.5%	25.3%	60.4%
Y et X_4	$var(\widehat{\beta})$	0.0450	0.0346	0.0375	0.0618
	T_P	0.0475	0.0380	0.0484	0.0706
	ER	5.6%	9.8%	29.1%	14.2%
Y et X_5	$var(\widehat{\beta})$	0.0474	0.0324	0.0395	0.0544
	T_P	0.0494	0.0388	0.0435	0.0761
	ER	4.2%	19.8%	10.1%	39.9%

Tableau 3.8 – Variances obtenues des coefficients de régression pour différentes combinaisons de variables synthétisées.

Dans un autre cas, on a vérifié si l'estimation est meilleure avec $m = 50$ *jeux synthétiques. Les résultats obtenus après* $5\,000$ *réplications en synthétisant dans l'ordre* X_1 *et* Y *sont dans le tableau 3.9. On souhaite également comparer adéquatement l'ordre de synthétisation en même temps que le nombre de jeux synthétiques utilisés. Dans ce tableau, on peut voir que l'ordre des variables à synthétiser influence l'erreur relative. En effet, la plus grande erreur relative dans un ordre est* 32.6% *et dans l'autre, elle vaut* 14.3%*. On voit également à l'aide de ce tableau que l'augmentation du nombre de jeux synthétiques n'améliore pas l'estimation de la variance puisque l'erreur relative augmente légèrement lorsqu'on passe de* 10 *à* 50 *jeux synthétiques. En effet, la pire erreur vaut* 13.1% *avec* 10 *jeux*

synthétiques et augmente à 14.3% *avec* 50*. Cette petite augmentation n'est pas significative, mais elle montre que l'augmentation du nombre de jeux n'améliore pas la précision de l'estimation dans cet exemple.*

Variables synthétisées		Paramètre 1	Paramètre 2	Paramètre 3	Paramètre 4
X_1 et Y (m=50)	$\widehat{\beta}$	2.22	-0.74	1.37	-2.05
	$var(\widehat{\beta})$	0.0751	0.0445	0.0478	0.0731
	T_P	0.0996	0.0497	0.0517	0.0872
	ER	32.6%	11.7%	8.2%	19.3%
Y et X_1 (m=50)	$\widehat{\beta}$	1.70	-0.63	1.13	-1.64
	$var(\widehat{\beta})$	0.0511	0.0353	0.0374	0.0582
	T_P	0.0518	0.0361	0.0404	0.0665
	ER	1.4%	2.3%	7.9%	14.3%
Y et X_1 (m=10)	$\widehat{\beta}$	1.70	-0.62	1.12	-1.64
	$var(\widehat{\beta})$	0.0521	0.0378	0.0427	0.0640
	T_P	0.0565	0.0394	0.0440	0.0724
	ER	8.4%	4.2%	3.0%	13.1%

Tableau 3.9 – Tableau récapitulatif permettant de comparer l'erreur relative en fonction de l'ordre des variables synthétisées ainsi qu'en fonction du nombre de jeux synthétiques utilisés.

Puisque ce sont les coefficients qui intéressent davantage les chercheurs, on a également les valeurs moyennes obtenues pour les quatre coefficients. Ils sont relativement loin des coefficients originaux qui étaient 3*,* -1*,* 2 *et* -3*. Cette grande différence peut provenir du fait qu'on utilise un jeu de données simulé contenant très peu de variables contrairement aux chapitres précédents qui utilisaient un vrai jeu de données avec davantage de variables. Dans cet exemple, on peut tout de même conclure que l'estimation des coefficients n'a pas fonctionné.*

Couverture des jeux de données synthétiques

Concernant la couverture qu'offre les intervalles de confiance des jeux synthétiques, on utilise le jeu de données de Caiola et Reiter (2010) dans un premier temps puis un jeu de données simulé. À l'aide du vrai jeu de données, on veut voir si les coefficients obtenus dans la seconde régression du chapitre 2 sont présents dans chaque jeu synthétique. C'est-à-dire qu'on construit un intervalle de confiance pour chaque jeu synthétique et on valide si cet intervalle de confiance contient le coefficient du jeu de données original. Il y a une nuance importante ici. Il existe trois types de coefficients : le véritable coefficient, le coefficient obtenu du jeu original et le coefficient obtenu des jeux de données partiellement synthétiques. L'objectif de la couverture des intervalles de confiance obtenus du jeu de données original est d'inclure le véritable coefficient dans environ 95% des cas lorsqu'on connait ce coefficient. Pour ce qui est des intervalles de confiance obtenus pour chaque jeu de données partiellement

synthétique, on souhaite qu'ils ressemblent à ceux du jeu original et qu'ils contiennent le coefficient de ce dernier.

Parmi les 12 coefficients de la régression 2 du chapitre 2, on note que trois de ceux-ci ont une couverture inférieure à 95%, soit lorsque la race est amérindien ou asiatique (Race=3 ou 4) et lorsque l'état marital est célibataire. En effet, ces trois variables qui ont été synthétisées ne contiennent pas le coefficient du jeu original dans plus de 5% des cas (200 jeux synthétiques créés). La plus petite couverture est de 82.5%. Cependant, lorsqu'on combine les résultats pour construire les intervalles de confiance comme il a été vu au chapitre 1, la couverture est parfaite pour tous les ensembles de jeux de données synthétiques créés (40 au total). Cela signifie que l'utilisation de la proposition 1 assure qu'on retrouve le coefficient du jeu original dans l'intervalle de confiance obtenu des jeux synthétiques.

Dans cet exemple, on ne considère que les coefficients provenant d'un jeu de données et ceux-ci ne sont pas nécessairement le reflet des vrais coefficients. On simule un jeu de données avec des coefficients dès le départ pour évaluer si les intervalles de confiance des données synthétiques offrent une bonne couverture sur les véritables coefficients. Il est possible de constater que les intervalles de confiance des jeux synthétiques ne couvrent pas bien les coefficients originaux à l'aide de la simulation suivante.

Exemple 7 *Simulations étudiant la couverture des intervalles de confiance pour des coefficients originaux*

Pour les simulations suivantes, on a utilisé deux jeux de données dont un qui est généré selon l'exemple 4 et l'autre qu'on génère ainsi :

$$X \sim N_3\left(\begin{pmatrix}0\\0\\0\end{pmatrix}, \begin{pmatrix}1 & 0.6 & 0.7\\0.6 & 1 & 0.5\\0.7 & 0.5 & 1\end{pmatrix}\right)$$

$$Y = 3 \cdot X_1 - 1 \cdot X_2 + 2 \cdot X_3 + \epsilon$$

où $\epsilon \sim N(0, 9)$. *Les coefficients originaux sont* 3 *et* -1 *dans un cas et* 3, -1 *et* 2 *dans le second cas. On arrondit ensuite* Y *à l'entier le plus près pour qu'il soit une variable discrète. On synthétise la variable* Y *selon l'algorithme 1. En tout, on crée* 300 *jeux de données synthétiques pour le premier jeu généré et on calcule l'intervalle de confiance de la régression associé à chacun des jeux. On obtient une couverture de* 67% *et de* 98% *pour ce premier jeu de données.*

Pour le second jeu généré, on calcule la couverture de chaque jeu de données partiellement synthétique et celle offerte par un ensemble de 5 *jeux de données en utilisant les formules de la proposition 1 du chapitre 1. On crée un total de* 500 *jeux synthétiques en utilisant la même méthode que précédemment. On a un total de* 100 *ensembles de* 5 *jeux de données synthétiques. En calculant la couverture*

de chacun des jeux, on obtient 79%*,* 93% *et* 84.6% *respectivement pour les trois coefficients. La couverture des* 100 *ensembles de* 5 *jeux synthétiques est parfaite avec* 100% *pour les trois coefficients.*

3.2.3 Conclusion

Dans le chapitre, il y a quelques conclusions intéressantes à tirer. On a entre autres montré que la formule 3.1 de la proposition 2 pour combiner les variances après l'imputation de données manquantes ne fonctionne pas lorsqu'on impute avec des forêts aléatoires. Cependant, ce qui nous intéressait davantage était de montrer que la formule T_P pour combiner des jeux de données synthétisés par forêts aléatoires fonctionne. Celle-ci ne s'applique pas lorsqu'on synthétise seulement certaines observations d'une variable pour évaluer la variance de la moyenne de celle-ci lorsque les jeux sont créés sans tirage avec les forêts aléatoires. La formule T_P ne fonctionne pas plus quand on évalue la variance des coefficients de régression lorsque la variable synthétisée est dépendante ou indépendante toujours sans tirage. Cependant, lorsqu'on effectue les tirages avec l'algorithme 1, la formule T_P fonctionne pour combiner les différents jeux de données synthétiques après avoir synthétisé quelques observations d'une variable. Il est primordial d'effectuer les tirages pour obtenir des estimations valides. Quand vient le temps de synthétiser plus d'une variable, la formule T_P ne fonctionne pas toujours et ce, peu importe le nombre de jeux synthétiques utilisés. En effet, cela dépend de l'ordre et des variables synthétisées pour que la formule T_P fonctionne.

Pour ce qui est de la couverture, on obtient des résultats moyens. Les pourcentages de couverture sont supérieurs à 79% mais inférieurs à 93% dans un exemple simulé avec 3 variables indépendantes. Ces pourcentages valent 67% et 98% avec deux variables. Dans un vrai jeu de données, la couverture est bonne pour la majorité des variables, principalement celles n'ayant pas été synthétisées. Cependant, la couverture diminue sous la barre des 95% pour certaines variables synthétisées.

Chapitre 4

Test d'une extension à l'algorithme 1 dans une optique de confidentialité différentielle

L'objectif de ce chapitre est de tester une extension à la méthode de Caiola et Reiter (2010), soit l'algorithme 1 présentée au chapitre 1. Cette extension permettrait d'améliorer la protection de la confidentialité de l'algorithme 1, en s'inspirant de l'idée de la confidentialité différentielle. On définira d'abord la confidentialité différentielle, une mesure plus rigoureuse de protection de confidentialité d'un mécanisme de protection. Puis, on présentera l'extension proposée et on testera son comportement à l'aide de simulations.

4.1 Introduction à la confidentialité différentielle

La méthode proposée dans le premier chapitre repose sur l'hypothèse qu'aucune source externe supplémentaire ne devient disponible après la publication des jeux de données partiellement synthétiques. En effet, l'information auxiliaire que possède l'adversaire a un impact sur les variables qu'il faut synthétiser. On fait l'hypothèse qu'il a seulement les informations sur certaines variables (le vecteur d'information) lorsqu'on évalue la probabilité de réidentification. Cette hypothèse peut devenir fausse à la suite de la publication d'une nouvelle base de données. Il faut proposer une solution qui sera valide dans le pire des cas, c'est-à-dire qu'on doit être en mesure de garantir la confidentialité à un certain niveau sans avoir d'hypothèse à propos de l'adversaire ni à propos de l'information à sa disposition (Heffetz et Ligett, 2013).

Il existe déjà une solution qui assure un certain niveau de confidentialité, il s'agit de la confidentialité différentielle. Celle-ci cherche à garantir à un potentiel participant que peu importe sa décision de participer ou non à l'étude, *presque* la même information peut être apprise d'une sortie publiée des observations déjà dans le jeu de données ou de ses propres données (Heffetz et Ligett, 2013).

L'emphase sur le mot presque est importante, car sans celle-ci, cela implique que les résultats sont identiques avec et sans une observation. La conséquence est qu'on n'a simplement pas d'utilité dans les données puisque l'apport d'une observation est nul.

En d'autres mots, la confidentialité différentielle limite l'influence qu'une observation peut avoir sur l'information publiée. Cette définition intuitive permet aux répondants et aux non-répondants la possibilité de nier leur participation ou leur non-participation à l'étude et cela ne peut pas être contredit par un adversaire (Heffetz et Ligett, 2013).

Si nous sommes en mesure d'ajouter le principe de confidentialité différentielle lors de la génération des jeux de données partiellement synthétiques tout en conservant l'utilité élevée, on sera alors en mesure de garantir à un certain niveau la confidentialité des données des répondants. Cela revient à se demander s'il est possible d'améliorer la méthode proposée dans le chapitre 1 pour augmenter la confidentialité en s'inspirant de l'idée de la confidentialité différentielle.

4.1.1 Définition de confidentialité différentielle

Pour assurer la confidentialité d'un répondant, on doit regarder comment réagit notre fonction lors de l'ajout de cette observation. On a deux bases de données voisines, c'est-à-dire un jeu de données identique à l'autre, mais dont un des deux jeux de données contient une observation de plus (par observation, on entend une ligne et les colonnes représentent les différentes variables du jeu de données).

Définition 1 *Confidentialité différentielle (Dwork* et al.*, 2006)*

Une fonction aléatoire K, telle une fonction générant des jeux de données synthétiques, assure une confidentialité différentielle de niveau ϵ si pour tout $S \in$ domaine(K) et pour toutes bases de données voisines D et D',

$$P[K(D) = S] \leq e^{\epsilon}\, P[K(D') = S],$$

pour $\epsilon \geq 0$ et où l'espace des probabilités pour chaque cas ne dépend que de l'aléatoire de K (c'est-à-dire que les bases de données D et D' sont considérées fixes).

Malheureusement, le meilleur choix d'ϵ n'est pas défini, on sait cependant qu'il doit être petit pour que *presque* soit valide. En fait, si ϵ vaut 0, cela implique que le résultat est identique dans les deux bases de données, mais comme mentionné précédemment, on n'a pas d'utilité dans nos données si cela survient. Dans l'autre cas extrême, soit lorsque $\epsilon = \infty$, il y a une sortie qui obtient une probabilité non-nulle avec un jeu de données, mais pas avec le jeu voisin. Le choix de ϵ est relativement important puisque c'est cette valeur qui contrôle le niveau de confidentialité garanti au répondant.

4.1.2 Applications et extensions

La confidentialité différentielle amène plusieurs applications intéressantes pour le traitement des données. Parmi celles-ci, il y a l'application à des groupes, à des jeux synthétiques et à des analyses postérieures.

La propriété de confidentialité différentielle s'applique à un groupe de répondants. Une fonction qui est différentiellement confidentielle de niveau ϵ est différentiellement confidentielle de niveau $k\epsilon$ pour un groupe de k individus ou inversement ; si l'on souhaite garantir à un groupe une garantie ϵ, on applique une fonction différentielle de niveau ϵ/k.

À partir de la dernière application découle la suivante : appliquer l fonction différentiellement confidentielle de niveau ϵ donne une confidentialité différentielle de niveau $l\epsilon$. Il s'agit de la propriété de composition séquentielle de la confidentialité différentielle (Heffetz et Ligett, 2013). Celle-ci peut s'interpréter pour les jeux de données synthétiques : si on veut publier M jeux de données synthétiques avec une confidentialité différentielle de niveau ϵ, il faut générer les jeux indépendamment avec confidentialité différentielle de niveau ϵ/M (Charest, 2012).

Une propriété fort intéressante est que si un jeu de données assure une confidentialité différentielle de niveau ϵ, alors les analyses postérieures possèdent également la propriété de confidentialité différentielle. Par exemple, en appliquant un calcul différentiel de niveau ϵ, une analyse effectuée sur le calcul précédent conserve également la propriété de confidentialité différentielle au même niveau (Heffetz et Ligett, 2013).

La confidentialité différentielle est à ce jour, la mesure la plus stricte en confidentialité. Elle est tellement stricte que plusieurs chercheurs ont modifié la définition originale pour l'affaiblir. Les différentes définitions moins strictes ne seront pas étudiées dans ce mémoire.

4.1.3 Introduction au synthétiseur Dirichlet-Multinomiale

Il existe plusieurs possibilités pour atteindre la confidentialité différentielle. La méthode la plus populaire, tout dépendant du type de sortie désirée, est l'ajout de bruit de Laplace. Celle-ci est une des plus simples. La méthode est la suivante :

Méthode de Laplace (Dwork *et al.* (2006))

Soit un jeu de données $X = (X_1, ..., X_n)$ et soit $f(X)$, la sortie désirée (par exemple, un tableau de fréquences). Pour diminuer le risque de réidentification et assurer une confidentialité différentielle de niveau ϵ, on va publier la sortie provenant de l'équation suivante :

$$\kappa(X) = f(X) + e$$

où $e \sim Laplace\big(0, \frac{\epsilon}{\Delta f}\big)$, soit une distribution centrée et de variance $2\big(\frac{\epsilon}{\Delta f}\big)^2$. Δf représente la sensibilité de la fonction f et est la différence maximale entre la fonction appliquée à deux jeux de données voisins. En termes mathématiques, $\Delta f = max_{x_1,x_2 voisines}|f(x_1) - f(x_2)|$.

Cependant, cette méthode est utile lorsque les données sont des variables continues. Puisque nous nous concentrons sur la synthétisation seulement des variables catégoriques, on utilise le Dirichlet-Multinomiale qui permet de générer des entiers. La méthode est décrite dans la sous-section suivante. On explique ensuite comment ajouter cette méthode à notre algorithme générant les jeux de données partiellement synthétiques.

4.1.4 Synthétiseur Dirichlet-Multinomiale

Algorithme pour synthétiser avec le Dirichlet-Multinomiale (Machanavajjhala *et al.*, 2008)

Soit des données entières non-négatives $X = (x_1, ..., x_k)$.

1. On initialise $\alpha = (\alpha_1, ..., \alpha_k)$ en fonction du théorème 1 ci-dessous.
2. On effectue un tirage de la distribution a posteriori suivante :

$$\tilde{\theta} \sim Dirichlet(\alpha + X)$$

3. On crée des données synthétiques en tirant des valeurs de la distribution suivante :

$$\tilde{X} \sim Multinomiale(\tilde{n}, \tilde{\theta})$$

Le jeu de données obtenu est de taille $\tilde{n}$.

Il a déjà été démontré que l'algorithme pour synthétiser avec le Dirichlet-Multinomiale est différentiellement confidentielle sous certaine condition. En voici le théorème.

Théorème 1 *(Machanavajjhala* et al.*, 2008)(Charest, 2012)*

Le synthétiseur de Dirichlet-Multinomiale est différentiellement confidentiel de niveau ϵ si et seulement si :

$$\alpha_i \geq \frac{\tilde{n}}{e^{\epsilon} - 1} \quad i \in 1, ..., k.$$

4.2 Modifications aux forêts aléatoires

On veut modifier l'algorithme 1 présenté au chapitre 1 afin d'utiliser l'idée de la confidentialité différentielle. On veut un algorithme qui utilisera sensiblement le même principe que le synthétiseur Dirichlet-Multinomiale. Pour ce faire, on obtient d'abord les arbres ainsi que le nombre de votes associées à chaque classe pour toutes les observations. Par exemple, lors de la génération du sexe d'une observation en particulier, on obtient sur les 500 arbres, 400 votes pour la classe homme et 100 votes pour la classe femme. Ces résultats sont obtenus en utilisant les variables déjà synthétisées et celles qui ne sont pas à synthétiser pour descendre dans les arbres. Les votes sont ensuite assignés au vecteur X.

On initialise α selon le nombre de classes. On regardera en détail le choix de la valeur de α dans la prochaine section. Ensuite, on effectue un tirage aléatoire d'une distribution de Dirichlet ayant comme paramètre le vecteur $(X+\alpha)$. Le résultat de ce tirage constitue les nouvelles probabilités pour effectuer le tirage d'une distribution multinomiale. Finalement, la valeur obtenue lors de ce tirage est la nouvelle valeur de la variable pour une observation. On fait de même pour toutes les observations à synthétiser et on répète les étapes m fois pour obtenir les m jeux de données partiellement synthétiques.

En fait, si on reprend l'algorithme 1 de la section 1, seulement l'étape 3 change. On peut le constater dans l'algorithme 2 présenté à la page suivante.

Algorithme 2 pour la génération de jeux partiellement synthétiques avec l'ajout du principe du Dirichlet-Multinomiale

Pour tout $i = 1, ..., s$:

1. ***Ajustement du modèle*** : on crée la forêt aléatoire avec la variable dépendante catégorique Y_i en fonction des variables Y_0 et $Y_1, ..., Y_{i-1}$.

$$mod = FA(Y_i \sim \{Y_0, Y_1, ..., Y_{i-1}\})$$

2. ***Obtention des votes*** : on trouve la feuille associée à chaque observation dans chaque arbre en utilisant les valeurs des variables Y_0 et $Y_1^*, ..., Y_{i-1}^*$, soit les variables déjà synthétisées, et on note la classe ayant la plus grande proportion dans chaque feuille. Pour chaque observation, on a le résultat de tous les arbres. Par exemple, une observation peut obtenir pour une variable à deux catégories 300 arbres votant pour une et 200 autres votant pour la seconde catégorie.
3. ***Tirage aléatoire d'une nouvelle distribution de probabilité en considérant les votes (vecteur X) et α*** :

$$Dist^* = Dirichlet(X + \alpha)$$

4. ***Tirage aléatoire des nouvelles valeurs (Y_i^*)*** : on effectue un tirage pour chaque observation en fonction de la distribution $Dist^*$ obtenue à l'étape précédente.

$$Y_i^* = Multinomiale(1, Dist^*)$$

Recommencer pour le prochain i.

Le jeu de données partiellement synthétique obtenu n'est pas exactement différentiellement confidentiel. En effet, on ne fait que s'inspirer de l'idée de la confidentialité différentielle.

4.3 Choix du paramètre α et utilité des données

L'ajout de la propriété de confidentialité différentielle n'est pas gage de succès. Il faut également trouver une façon de sélectionner α de manière à obtenir une utilité élevée des données tout en conservant le risque de réidentification faible. Pour obtenir le meilleur α, la sélection se fera graphiquement en opposant le risque de réidentification et l'utilité des données tel que proposé par Duncan *et al.* (2001).

Pour mesurer le risque, on utilise la même mesure que dans les chapitres précédents, soit le pourcentage de véritable réidentification.

Les intervalles de confiance ne sont cependant pas valides. On illustre ci-dessous que la formule 1.1

proposé dans le chapitre 1 pour combiner les jeux de données partiellement synthétiques n'est pas valide avec différentes valeurs de α. Pour ce faire, on utilise la même technique qu'au chapitre précédent, c'est-à-dire qu'on effectue plusieurs réplications d'une procédure et on regarde si la variance observée est la même que la variance calculée avec la formule T_P.

On propose l'alternative suivante pour trouver le meilleur α : on fait deux graphiques afin de comparer l'erreur relative des coefficients et l'erreur relative de la variance. Dans les deux cas, on évaluera le risque avec le pourcentage de véritable réidentification.

4.3.1 Vérification de la formule T_P avec des α différents

L'ajout du paramètre α lors de la génération des jeux de données partiellement synthétiques augmente la variabilité dans les valeurs synthétiques obtenues. En effet, cette variation est causée par le fait que plus α est grand, plus la distribution pour effectuer le tirage ressemble à une distribution uniforme. Le choix de la valeur va varier d'un jeu synthétique à l'autre sans que les arbres obtenus n'aient d'impact. L'hypothèse de départ est que plus le paramètre α augmente, plus l'estimation de la variance sera biaisée. On vérifie comment réagit la variance de la moyenne de Y lorsque cette variable est synthétisée dans le jeu de données simulé suivant :

$$X \sim N_2\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 10 & 7 \\ 7 & 10 \end{pmatrix} \right)$$

$$Y = X_1 + X_2 + \epsilon$$

où $\epsilon \sim N(0, 9)$ et où la variable Y est arrondie à l'entier le plus près. On fait un total de $3\ 000$ réplications au cours desquelles on crée 5 jeux synthétiques de taille $n = 300$. Les jeux synthétiques sont créés à partir d'un certain α fixe lors des réplications. Les résultats obtenus sont présentés dans le tableau 4.1. Dans ce tableau, on a la variance espérée, la variance obtenue avec la formule T_P et l'erreur relative pour les différentes valeurs de α. On remarque que plus α augmente, plus l'erreur relative augmente. On obtient des estimations biaisées de la variance avec la formule T_P. On a également testé avec une autre matrice de covariances pour voir l'impact de celle-ci dans le calcul de la formule. Les résultats sont également dans le tableau 4.1. L'erreur relative est plus petite initialement, mais plus α augmente, plus l'erreur relative augmente. Cette erreur atteint 63.3% avec $\alpha = 100$.

Dans le cas où le nombre d'observations est de $10\ 000$ et toujours avec 5 jeux synthétiques, on obtient les résultats moyens des $5\ 000$ réplications dans le tableau 4.2. Ce tableau contient encore une fois la variance espérée, la variance calculée (T_P) et l'erreur relative.

Les résultats ne sont guère mieux puisque les erreurs relatives grimpent à 96%. La formule ne s'applique pas pour évaluer la variance de la moyenne d'une variable synthétisée lorsqu'on utilise 5 jeux synthétiques. Cette fois, on teste avec 25 jeux synthétiques en espérant que l'erreur relative soit plus

Matrice cov.	Valeur de α	Variance espérée	T_P	Erreur relative
var=10, cov=7	1	0.1546	0.1652	6.9%
var=10, cov=7	2	0.1667	0.1821	9.2%
var=10, cov=7	10	0.3151	0.2638	16.3%
var=10, cov=7	100	0.7659	0.3884	49.3%
var=1, cov=0.7	1	0.0485	0.0487	0.4%
var=1, cov=0.7	10	0.0846	0.0737	12.9%
var=1, cov=0.7	100	0.3319	0.1219	63.3%

Tableau 4.1 – Tableau montrant le biais lorsqu'on synthétise avec des valeurs de α élevées pour différentes matrices de covariances.

Valeur de α	Variance espérée	T_P	Erreur relative
1	0.0097	0.0062	36.1%
2	0.0219	0.0078	64.4%
10	0.1612	0.0145	91.0%
100	0.5375	0.0225	95.8%

Tableau 4.2 – Tableau montrant le biais avec un grand jeu de données (n=10 000) lorsqu'on synthétise avec des valeurs de α élevées.

petite. Le nombre d'observations est diminué à 300 puisque l'utilisation du Dirichlet-Multinomiale ralentie considérablement la génération des jeux synthétiques et ce, malgré l'utilisation de calcul parallèle lors des réplications. Les résultats obtenus après $2\ 000$ réplications sont dans le tableau 4.3 et permettent de conclure que malgré un grand nombre de jeux de données synthétiques, l'erreur relative reste élevée lorsque α est grand puisque celle-ci vaut 55.1%.

Valeur de α	Variance observée	T_P	Erreur relative
1	0.1538	0.1557	1.2%
10	0.2616	0.2281	12.8%
100	0.7234	0.3249	55.1%

Tableau 4.3 – Tableau montrant le biais avec un grand nombre de jeux de données synthétiques et avec des valeurs de α élevées.

Dans un dernier cas, on a regardé les variances associées aux coefficients de régression comme dans l'article de Caiola et Reiter (2010) au chapitre 2. On reprend les mêmes simulations que précédemment, avec $n = 10\,000$ et $k = 5$. Les résultats obtenus après les $3\,000$ réplications sont présentés dans le tableau 4.4 où T_{Pi} représente l'estimation de la variance obtenue pour la variable i et variance de X_i représente la variance lors des réplications.

Les résultats permettent de conclure que la formule ne s'applique pas, car on surévalue la variance de beaucoup. Cette conclusion était prévisible puisque la formule ne fonctionne pas toujours dans le cas où α vaut 0 (résultat du chapitre 3) et on augmente la variabilité lorsque α augmente.

Valeur de α	Variance X_1	T_{P1}	ER_1	Variance X_2	T_{P2}	ER_2
1	0.00026	0.00068	161.5%	0.00026	0.00068	161.5%
10	0.00054	0.00268	396.3%	0.00055	0.00268	387.3%
100	0.00078	0.00443	467.9%	0.00076	0.00444	484.2%

Tableau 4.4 – Tableau montrant les variances des coefficients de régressions obtenues avec différentes valeurs de α.

4.4 Effet de α dans un exemple spécifique

On utilise le jeu de données de Caiola et Reiter (2010) encore une fois, mais on l'utilise pour faire des simulations. On sélectionne au hasard $4\,000$ observations parmi les $10\,000$ avec remise. On synthétise entièrement 3 variables comme dans le chapitre 2. On synthétise les 3 mêmes, soit le sexe, la race et l'état marital. On crée 5 jeux de données synthétiques et on réplique $3\,000$ fois. Lors de chaque réplication, on note les coefficients de la régression énoncée ci-dessous, ainsi que le résultat de la formule T_p associé à chaque coefficient, le risque associé aux 5 jeux synthétiques et l'erreur relative des coefficients par rapport aux coefficients originaux. La régression est la suivante : la variable dépendante est la racine carrée de la prestation de sécurité sociale perçue et les variables indépendantes sont dans l'ordre : sexe, race, état marital, age et revenu.

Au final, on calcule l'erreur relative moyenne des coefficients et l'erreur relative de la variance pour les mettre sur les deux graphiques décrits précédemment. De plus, le risque moyen associé à chaque ensemble de jeux synthétiques est calculé. On calcule ces valeurs pour différentes valeurs de α.

Avec la première valeur de α, soit 0, on note que certaines erreurs relatives atteignent des valeurs extrêmes supérieures à $1\,000\%$ que ce soit avec l'algorithme 1 (du chapitre 1) ou que ce soit avec la fonction GenSyn_DM (algorithme 2) [1] qui permet de générer des jeux de données avec différentes valeurs de α. Les résultats sont similaires entre les deux méthodes puisqu'elles utilisent sensiblement la même technique. On utilise la médiane au lieu d'utiliser la moyenne dans le premier graphique pour représenter l'erreur relative des coefficients. Les boîtes à moustaches ont été réalisées en considérant seulement 95% des répétitions pour éviter les valeurs trop extrêmes. Sur la figure, on a retiré toutes les valeurs extrêmes pour mieux visualiser les erreurs relatives puisque les distributions sont nettement asymétriques. Cela a pour effet de modifier l'axe de l'utilité, celle-ci varie entre 0 et $1\,000$ en présence des valeurs extrêmes et ne varie qu'entre 0 et 200 sans ces valeurs. Pour le second graphique, on a calculé l'erreur relative de la variance pour chaque réplication. On a également retiré les valeurs extrêmes pour mieux visualiser les boîtes à moustaches. Les résultats sont présentés dans les figures 4.1 et 4.2.

Le premier point à remarquer est le risque de réidentification qui est plus élevé comparativement à ce qu'on a observé au chapitre 2. Dans les faits, on avait obtenu un risque au alentour de 7% et avec un α valant 0, on note un risque d'environ 11.7%. Le risque augmente puisqu'on utilise moins

1. Voir le code en Annexe D

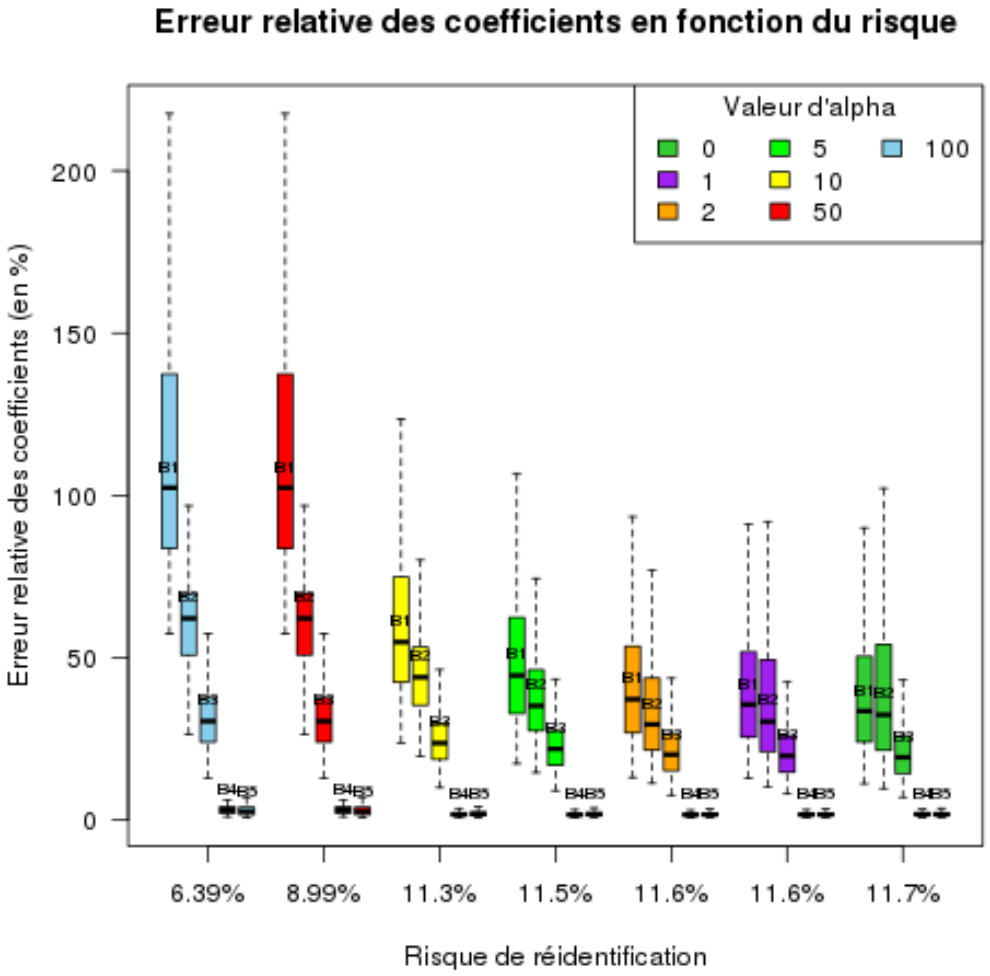

FIGURE 4.1 – Boîtes à moustaches des erreurs relatives pour chaque coefficient selon différentes valeurs de α en fonction du risque de réidentification.

d'observations et autant de variables, il y a davantage de combinaisons uniques. En effet, on utilise seulement $4\,000$ observations et dans le chapitre 2, on en utilise $10\,000$. En analysant spécifiquement une réplication où $\alpha = 1$, il y a environ 480 observations qui sont réidentifiées parmi les $4\,000$ sélectionnées pour la régression. Parmi ces 480 observations, environ 20 observations représentent des vrais cas uniques dans l'ensemble des $10\,000$ de départ. On note également que le risque est similaire pour des valeurs de α petites tout en notant une légère diminution lorsque α augmente doucement. Avec un α valant 100, le risque diminue pratiquement de moitié, il vaut 6.39%, un risque semblable au risque avec les $10\,000$ observations.

Pour ce qui est de l'utilité des coefficients, on constate une certaine croissance dans l'erreur relative lorsque α augmente. En effet, la croissance se voit relativement bien en regardant la figure 4.1, plus précisément le premier coefficient ($B1$), où l'erreur relative augmente surtout à partir de $\alpha = 5$. On

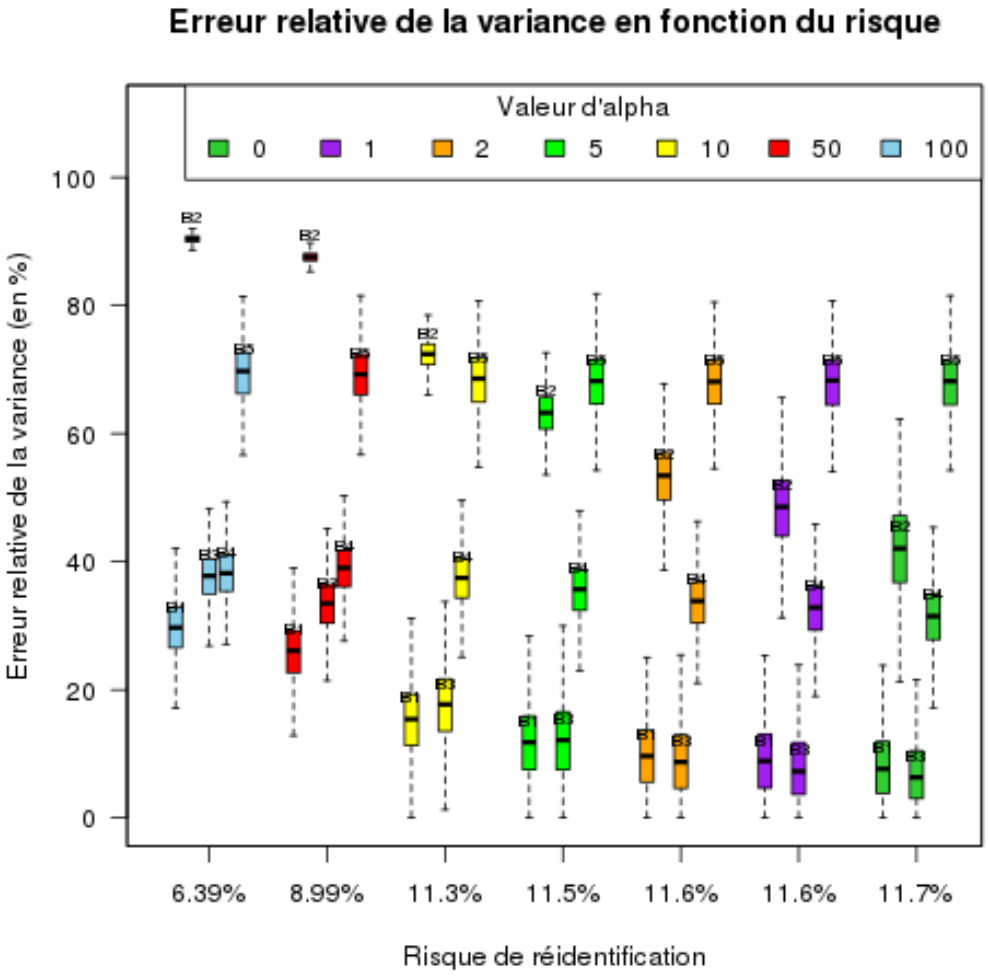

FIGURE 4.2 – Boîtes à moustaches des erreurs relatives de la variance pour chaque coefficient selon différentes valeurs de α en fonction du risque de réidentification.

peut conclure que plus α augmente, moins on est en mesure d'obtenir des estimations valides des coefficients. On note également que l'utilité est très similaire pour les deux valeurs de α suivantes : 50 et 100, mais qu'un écart considérable est observé pour le risque de réidentification.

Dans la figure 4.2, on remarque encore une fois que l'erreur relative de la variance augmente lorsque α augmente. Avec les deux figures, il est possible de voir que les erreurs relatives des deux derniers coefficients ($B4$ et $B5$) sont toujours faibles et les erreurs relatives de la variance associées à ces coefficients sont élevées. En fait, les véritables variances sont vraiment petites ce qui fait que la moindre sur-estimation cause une grande erreur relative pour ces variables.

Pour en revenir sur la confidentialité différentielle, on utilise le théorème 1 pour obtenir la valeur de ϵ. Normalement, on sélectionne α en conséquence du niveau de confidentialité différentielle désiré. Nous avons fait l'inverse dans ce chapitre pour montrer comment se comporte l'utilité et le risque

lorsque α augmente. En supposant que le meilleur choix de α serait lorsque ce dernier vaut 100, on observe alors une confidentialité différentielle de niveau 3.66. Comme on souhaite obtenir un ϵ petit, il faudrait un α beaucoup plus grand pour avoir une valeur de 1 par exemple. En fait, il vaudrait $2\ 327.9$ pour obtenir un tel ϵ.

Comme il a été mentionné précédemment, on veut s'inspirer de la confidentialité différentielle pour améliorer la confidentialité lors de la génération de jeux partiellement synthétiques. On a seulement ajouté l'idée générale à l'algorithme. On a vu dans les résultats que l'ajout du Dirichlet-Multinomiale ne permet pas d'obtenir de bonnes estimations des coefficients et de la variance de ceux-ci.

Chapitre 5

Équations structurelles et confidentialité

Alors qu'on s'est concentré sur la moyenne d'une variable synthétisée et sur les coefficients de régression dans les chapitres précédents, on explore un domaine peu étudié à ce jour dans ce chapitre. On analyse l'impact qu'a la synthétisation de variables quasi-identifiantes dans un modèle d'équations structurelles. Plus précisément, les coefficients des chemins après la synthétisation de variables seront comparés avec les coefficients originaux d'un modèle provenant d'un article de St-Pierre et Audet (2011). On veut ainsi valider si les conclusions seraient les mêmes avant et après la synthétisation d'un vrai jeu de données.

5.1 Article de St-Pierre et Audet (2011)

L'article de St-Pierre et Audet (2011) ainsi que les données pour réaliser l'article nous ont été fournis pour pouvoir étudier l'effet de la synthétisation sur les équations structurelles. Voici une brève description du jeu de données ainsi que des principaux résultats et modèles utilisés dans l'article.

5.1.1 Description du jeu de données

Les données ont été récoltées via un questionnaire entre 2000 et 2007. Le but de cette collecte était de répondre à certaines hypothèses comme celle-ci : la stratégie influence la relation entre le capital d'innovation et la performance des petites et moyennes entreprises (PMEs). Celles-ci sont la population cible dans cette étude. Un total de 267 PMEs ont répondu au questionnaire. Ces compagnies sont séparées en trois groupes distincts, soit celles utilisant une stratégie prospective (116), celles ayant une stratégie hybride (69) et celles ayant une stratégie défensive (82).

Le jeu de données original contient plus de 850 variables permettant de mesurer les pratiques de l'entreprise, les ressources disponibles, etc. Pour les modèles d'équations structurelles, on conserve 33 variables et c'est avec cette base de données qu'on créera les jeux de données synthétiques.

Parmi les variables retenues pour les équations structurelles, il y a 22 variables continues et 11 variables discrètes. Dans ce mémoire, on a seulement vu comment synthétiser des variables catégo-

riques, ce sont ces variables qui nous intéressent pour la génération des jeux synthétiques. On regarde ce à quoi réfère chaque variable catégorique pour voir si l'information liée à celle-ci serait accessible via internet. Par exemple, il est possible de savoir sur le site de l'entreprise elle-même ou sur le site du centre de recherche industrielle du Québec (CRIQ) si une entreprise possède ou non un produit de marque maison. Cela est une des variables retenues dans le jeu de données.

Les données ayant été fournies sous l'entente d'un contrat de confidentialité (on parlait de cette technique dans l'introduction, soit de limiter l'accès aux jeux de données avec des ententes de confidentialité), il n'y a pas d'information révélatrice sur une quelconque PME dans ce chapitre ni de statistiques autre que les coefficients reliés aux équations structurelles.

5.1.2 Introduction aux équations structurelles

Avant d'énoncer les résultats obtenus ainsi que la méthodologie de l'étude, on doit définir ce qu'est un construit ainsi que ce qu'est une équation structurelle. D'abord, un construit latent représente une variable non-mesurable par des questions. Prenons l'exemple suivant où l'on souhaite mesurer l'intelligence d'une personne, il n'est pas possible de mesurer cela directement. Cependant, on peut mesurer l'intelligence indirectement à l'aide de différents tests d'aptitude (Mueller, 1996).

Dans un cadre théorique, les variables latentes ont déjà des variables assignées tandis que dans un cadre exploratoire, on utilise l'analyse factorielle exploratoire pour assigner les variables observées à des variables latentes. Il est ensuite nécessaire d'interpréter les construits obtenus. Pour valider le choix des variables assignées, on utilise le coefficient associé (le *loading*) pour déterminer si une variable est importante au construit ou non. La norme est de prendre les valeurs supérieures à 0.7 afin d'obtenir un AVE (*average variance extracted*) supérieur à 0.5. Parfois, on doit admettre d'autres variables si seulement une variable est retenue parmi plusieurs afin d'enrichir le construit (St-Pierre et Audet, 2011).

Le sens des flèches reflète la relation entre une variable latente et les variables mesurées. Si elles partent de la variable latente vers les variables mesurées, alors la variable latente cause les variables mesurées (lien réflexif)(Mueller, 1996). Inversement, on dit que les variables mesurées sont les causes du construit (lien formatif)(Sanchez, 2013). Par exemple, on veut mesurer l'attaque des Canadiens de Montréal. Le nombre de buts, le nombre de tirs et le temps de possession sont causés par l'attaque, mais celle-ci est causée par le temps mis à l'entrainement sur l'attaque et le maniement de la rondelle. Le sens des flèches serait celui sur la figure 5.1 [1].

1. Exemple fortement inspiré de celui de Sanchez (2013)

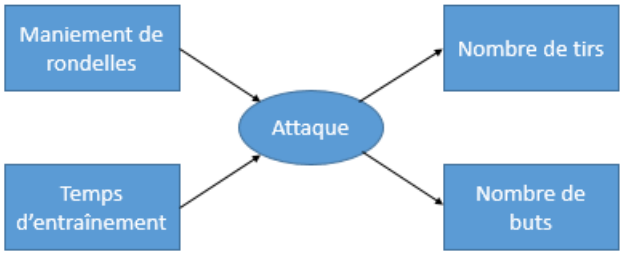

FIGURE 5.1 – Exemple de modèle avec les deux types de relation entre les variables.

Sur la figure 5.1, il est possible de constater que la forme n'est pas identique pour les types de variables. Suivant la convention, les variables latentes sont représentées par une forme elliptique et les variables mesurées sont de formes rectangulaires.

Ce type de construit est une équation structurelle à une variable latente. Lorsque plusieurs variables sont impliquées dans le modèle, il existe quelques types particuliers d'équations structurelles, soit les équations structurelles linéaires (LISREL) et les équations structurelles des moindres carrés (PLS).

5.1.3 Différence entre deux types d'équations structurelles

On note d'abord que LISREL utilise l'approche de l'analyse de structure de covariance par maximum de vraisemblance. Les données recueillies en marketing ne satisfont pas souvent les hypothèses pour utiliser le maximum de vraisemblance (Fornell et Bookstein, 1982). Une des hypothèses est que les données doivent suivre une loi normale multivariée.

L'approche PLS évite justement ces hypothèses et s'assure de ne pas obtenir de solutions inappropriées (des solutions impossibles) qui est un sérieux problème pour la méthode LISREL. Par solutions inappropriées, c'est qu'il est possible d'obtenir des coefficients à l'extérieur des valeurs possibles qu'ils peuvent prendre. PLS se base sur les composantes principales pour minimiser la variance des résidus (Fornell et Bookstein, 1982).

Également, l'utilisation de PLS est favorable lorsque l'objectif est l'application et la prédiction. Les autres types tels que LISREL sont mieux pour tester et développer lorsque la théorie a priori est forte (Chin, 1997). Enfin, PLS sera en mesure d'expliquer mieux les relations complexes.

Pour plus de détails sur ces deux méthodes, l'article de Fornell et Bookstein (1982) compare les deux méthodes à l'aide d'exemples et explique de manière plus détaillée la méthode PLS.

5.1.4 Méthodologie et résultats

Le modèle utilisé dans l'article est présenté dans la figure 5.2. Les liens avec les variables mesurées (ainsi que ces dernières) ne sont pas présentés pour alléger la figure. Ces liens sont tous de types réflexifs et les variables mesurées associées à chaque variable latente sont décrites dans le tableau 5.2.

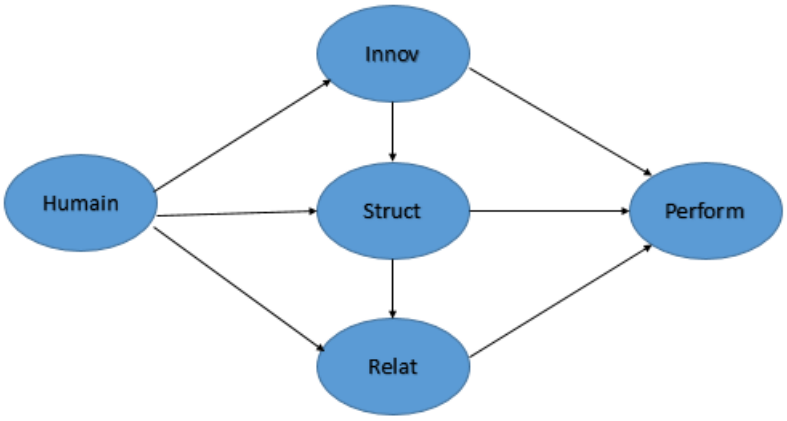

FIGURE 5.2 – Modèle d'équations structurelles d'intérêt.

Ce modèle est appliqué à 3 ensembles de données, soit les 267 PMEs, les PMEs ayant une stratégie prospective ou celles ayant une stratégie défensive. Le type d'équations structurelles qui a été utilisé pour réaliser ce modèle est le suivant : *partial least square* (PLS).

Pour l'ensemble comptant toutes les PMEs, les relations qui se sont avérées significatives entre les variables latentes sont les suivantes : Humain et Innov, Humain et Struct, Innov et Struct, Struct et Relat.

Pour l'ensemble où les PMEs ont une stratégie défensive, les liens significatifs sont les suivants : Humain et Innov, Humain et Struct, Innov et Struct, Relat et Perform.

Ce sont ces deux ensembles qui nous intéressent particulièrement puisque un des ensembles comprend toutes les observations et l'autre est le plus petit sur lequel le modèle a été testé dans l'article.

5.2 Résultats

5.2.1 Risque de réidentification

On évalue le risque de réidentification avec des jeux de données synthétique. On a évalué le risque avant et après la génération de ces jeux. Puisque le risque est initialement à 0% (prévisible puisqu'on utilise deux variables dichotomiques $(0-1)$ et qu'il demeure à 0 après, on ne peut conclure sur l'amélioration de la confidentialité. On décide d'ajouter la variable âge de l'entreprise qui est disponible dans le jeu de données original pour évaluer le risque de réidentification avec trois variables au lieu

de deux.

On refait les jeux synthétiques avec cette variable supplémentaire dans le jeu de données original. Cela ajoute de l'information lors de la génération des forêts aléatoires. Cet ajout a pour effet de changer le risque de réidentification en utilisant toujours les deux variables synthétisées (sans l'âge de l'entreprise). En effet, le risque augmente à 0.4%. Cette légère hausse est causée par le fait qu'une observation synthétique est réidentifiée. On note également que le nombre de fausses identifications augmente aussi. On obtient 85.7% de fausses identifications.

Lorsqu'on ajoute la variable continue pour évaluer le risque, on obtient les résultats du tableau 5.1 où l'on note les risque de réidentification et de fausse identification ainsi que les observations uniques réidentifiées. Dans le tableau, on remarque qu'on identifie seulement 28 uniques sur les 114 qui sont supposément réidentifiés (42.3% de 267 PMEs). Initialement, on identifiait 62 uniques sur les 267 observations. Les probabilités de vraiment réidentifier les non-uniques varient d'un cas à l'autre, alors on ne peut réidentifier avec certitude les 86 observations non-uniques parmi les 114.

	Avant	Après
Vraies identifications	23.2%	42.3%
Fausses identifications	0%	45.4%
Uniques réidentifiées	62	28

Tableau 5.1 – Tableau résumant les différents risques obtenus avec les jeux synthétiques.

5.2.2 Équations structurelles obtenues

Deux équations structurelles ont été retenues dans ce mémoire pour la comparaison entre les coefficients originaux et ceux obtenus après la synthétisation. En premier, on considère l'ensemble des PMEs alors qu'en second, on considère seulement les entreprises ayant une stratégie défensive. Ce choix a été fait en fonction de la taille de ce groupe qui est plus petit que le groupe des entreprises ayant une stratégie prospective. Également, le choix a été influencé par l'article de St-Pierre et Audet (2011), car celui-ci ne s'intéresse pas aux résultats du groupe ayant une stratégie hybride.

Il est important de noter également que le logiciel utilisé pour refaire les analyses n'est pas le même que pour les analyses initiales. En effet, nous utilisons le logiciel R avec la librairie *plspm* [2] alors que le logiciel SPSS avait été initialement utilisé pour la réalisation de l'article. Par conséquent, les résultats varient légèrement lors de la sélection des variables en fonction des coefficients et on obtient alors des conclusions différentes avec le jeu de données original. Dans les modèles d'équations structurelles obtenus, il est également impossible de déterminer si un coefficient entre deux variables latentes est significatif puisque nous n'avons pas d'estimé de la variance. On ne peut pas conclure sur l'influence des variables latentes.

2. Version 0.4.1

Dans le premier cas, on a l'ensemble des PMEs et on synthétise deux variables en utilisant l'algorithme 1 du chapitre 1 parmi les 33 variables retenues pour le modèle. Les deux variables en question sont les suivantes : MAIMA qui permet de savoir si l'entreprise possède un produit maison et q42estprot qui permet de savoir si les innovations de l'entreprise sont protégées (on est également en mesure de trouver cette information sur internet). On note ici que ces deux variables sont liées au construit Innov. On a créé 5 jeux synthétiques pour obtenir les résultats du tableau 5.2. Ce tableau est composé de 8 colonnes, une pour représenter le coefficient obtenu avec le jeu original, 5 pour les coefficients des 5 jeux de données synthétiques et une pour représenter la moyenne des 5 coefficients des jeux synthétiques. La première colonne est composée des variables mesurées.

On obtient également les coefficients des chemins entre les variables latentes pour chaque jeu de données, soit le jeu original et les 5 jeux synthétiques. Les coefficients sont présentés dans le tableau 5.3. La moyenne des 5 coefficients des jeux synthétiques est également ajoutée au tableau comme précédemment.

À l'aide des tableaux 5.2 et 5.3, on voit que la différence n'est pas énorme entre les coefficients originaux et les coefficients obtenus après la synthétisation. La différence est vraiment minime également entre les coefficients originaux et les coefficients moyens obtenus. Les différences les plus importantes sont observées lorsque la variable latente impliquée est Innov, celle-ci contient les deux variables synthétisées. En effet, que la variable latente soit impliquée directement ou indirectement, cela affecte les coefficients des chemins entre les variables latentes. Prenons par exemple le chemin entre Innov et Perform, le coefficient original est de -0.039 et en moyenne, il est de -0.069 pour les 5 jeux de données partiellement synthétiques.

Comme on s'intéresse à l'impact de la synthétisation de variables dans des modèles d'équations structurelles, on suppose qu'on souhaite conserver les variables ayant des coefficients supérieurs à 0.5 pour chaque construit. On peut voir si certains construits sont représentés par différentes variables qu'avec le jeu de données original. Le tableau 5.4 résume les variables associées au construit Innov pour chaque jeu synthétique. Les autres construits ne sont pas considérés puisque c'est toujours les mêmes variables qui sont conservées.

En conservant que les variables ayant un coefficient supérieur à 0.5 pour reproduire les équations structurelles, on obtient les résultats présentés dans le tableau 5.5. Dans ce tableau, les coefficients des chemins sont similaires. La synthétisation n'a pas affecté les résultats outre que le choix des variables. Dans certains construits, on sélectionnait une variable de moins. En regardant les coefficients obtenus, il est possible de croire que les mêmes liens significatifs ressortiraient entre les différents jeux puisque les coefficients sont similaires. Cependant, comme il a été mentionné précédemment, on n'a pas d'estimé de la variance pour faire des tests sur les coefficients.

Variable	Coef. orig.	Coef. jeu synt					Coef. Moy.
Humain							
empcolb	0.47	0.47	0.46	0.48	0.47	0.47	0.47
q23emp	-0.25	-0.23	-0.24	-0.24	-0.25	-0.24	-0.24
q28perf	0.41	0.40	0.41	0.41	0.41	0.41	0.41
q58int3c	0.46	0.46	0.46	0.46	0.46	0.46	0.46
FORSAPCT	0.19	0.20	0.19	0.19	0.19	0.19	0.19
q29str2	0.28	0.29	0.28	0.28	0.28	0.29	0.28
q47sicfepm	0.46	0.45	0.47	0.46	0.46	0.45	0.46
q60form	0.77	0.78	0.77	0.77	0.77	0.78	0.77
Innov							
rdemp	0.48	0.49	0.49	0.49	0.49	0.45	0.48
REDREDES	0.79	0.82	0.79	0.80	0.81	0.80	0.80
q9rd	0.46	0.46	0.49	0.48	0.47	0.49	0.48
q58tectot	0.25	0.24	0.28	0.23	0.24	0.26	0.25
MAIMA	0.54	0.46	0.43	0.55	0.44	0.51	0.48
q42estprot	0.61	0.59	0.57	0.56	0.55	0.52	0.56
Struct							
frvadven	0.56	0.56	0.55	0.56	0.56	0.56	0.56
q60org	0.77	0.77	0.77	0.77	0.76	0.77	0.77
RECIN	0.48	0.49	0.49	0.49	0.48	0.48	0.49
RECEX	0.35	0.34	0.35	0.35	0.35	0.35	0.35
q47iman	0.49	0.48	0.49	0.49	0.49	0.48	0.49
q40nqd2	0.33	0.34	0.33	0.32	0.33	0.33	0.33
Relat							
TROIC	-0.30	-0.30	-0.29	-0.30	-0.30	-0.31	-0.30
TROIF	-0.45	-0.45	-0.44	-0.45	-0.45	-0.45	-0.45
QUALIMOB	0.06	0.06	0.06	0.06	0.06	0.06	0.06
q14cliap	0.53	0.53	0.54	0.53	0.53	0.53	0.53
q58protot	0.51	0.50	0.51	0.51	0.50	0.50	0.50
CLIESATI	0.17	0.17	0.17	0.17	0.17	0.17	0.17
q14repr	0.58	0.58	0.58	0.58	0.58	0.58	0.58
q9sansRD	0.30	0.30	0.31	0.30	0.30	0.30	0.30
q58ext3c	0.47	0.47	0.48	0.47	0.47	0.47	0.47
CONAD	0.51	0.52	0.52	0.51	0.51	0.51	0.51
Perform							
benacta0	0.52	0.54	0.51	0.53	0.53	0.53	0.53
ventev2	0.31	0.32	0.27	0.31	0.32	0.34	0.31
benebpro0	-0.89	-0.88	-0.90	-0.89	-0.89	-0.88	-0.89

Tableau 5.2 – Tableau des coefficients pour les variables mesurées du modèle d'équations structurelles avec l'ensemble des PMEs.

Dans le second cas, il y a seulement 82 observations, il est possible qu'il y ait davantage de mouvement dans les coefficients. En effet, puisqu'il y a moins d'observations pour créer la forêt aléatoire, cette dernière aura plus de difficulté à prédire convenablement qu'en ayant les 267 PMEs. En synthétisant

Départ du lien	arrivé	Coef. orig.	Coef. jeu synt.					Coef. moy.
Humain	Innov	0.623	0.609	0.621	0.618	0.605	0.633	0.617
Humain	Struct	0.633	0.620	0.649	0.649	0.629	0.646	0.639
Humain	Relat	0.462	0.463	0.462	0.465	0.459	0.459	0.462
Innov	Struct	0.205	0.232	0.182	0.178	0.217	0.181	0.198
Innov	Perform	-0.039	-0.050	-0.054	-0.073	-0.077	-0.092	-0.069
Struct	Relat	0.175	0.172	0.172	0.173	0.177	0.178	0.174
Struct	Perform	-0.273	-0.262	-0.261	-0.259	-0.256	-0.249	-0.257
Relat	Perform	-0.123	-0.122	-0.122	-0.111	-0.112	-0.106	-0.115

Tableau 5.3 – Tableau des coefficients des chemins entre les variables latentes du modèle d'équations structurelles avec l'ensemble des PMEs.

Innov					
Jeu Orig	Jeu Synt.1	Jeu Synt.2	Jeu Synt.3	Jeu Synt.4	Jeu Synt.5
REDRESES	REDRESES	REDRESES	REDRESES	REDRESES	REDRESES
q42estprot	q42estprot	q42estprot	q42estprot	q42estprot	q42estprot
MAIMA			MAIMA		MAIMA

Tableau 5.4 – Variables ayant un coefficient supérieur à 0.5 dans le construit Innov.

Départ du lien	arrivé	Coef. orig.	Coef. jeu synt.					Coef. moyen
Humain	Innov	0.530	0.556	0.562	0.522	0.532	0.552	0.545
Humain	Struct	0.520	0.514	0.521	0.540	0.520	0.528	0.525
Humain	Relat	0.221	0.219	0.216	0.220	0.218	0.220	0.219
Innov	Struct	0.384	0.384	0.374	0.359	0.390	0.358	0.373
Innov	Perform	0.025	-0.005	0.007	-0.021	-0.019	-0.039	-0.015
Struct	Relat	0.301	0.304	0.306	0.303	0.304	0.302	0.304
Struct	Perform	-0.344	-0.321	-0.324	-0.314	-0.312	-0.303	-0.315
Relat	Perform	-0.102	-0.100	-0.103	-0.097	-0.099	-0.096	-0.099

Tableau 5.5 – Tableau des coefficients des chemins entre les variables latentes du modèle d'équations structurelles en utilisant seulement les variables avec un coefficient supérieur à 0.5.

deux variables comme précédemment, on obtient les tableaux 5.6 et 5.7. Dans ces tableaux, il n'y a pas beaucoup de variabilité dans les coefficients. De plus, les coefficients moyens sont semblables aux coefficients originaux.

Cependant, on constate à l'aide du tableau 5.8 qu'un construit n'a pas toujours les mêmes variables significatives (celles avec un coefficient supérieur à 0.5). Il s'agit du construit Perform qui pourtant n'a pas de variables synthétisées.

Dans le tableau 5.8, il y a un peu plus de variabilité dans la sélection des variables puisqu'un seul construit a parfois des variables sélectionnées différentes. En utilisant ces variables pour faire les équations structurelles, on obtient les résultats présentés dans le tableau 5.9 où il est possible de constater que la sélection des variables a un impact. Entre autres, le dernier chemin, soit celui allant de

Variable	Coef. orig.	Coef. jeu synt					Coef. moy.
Humain							
empcolb	0.429	0.430	0.441	0.428	0.406	0.429	0.427
q23emp	-0.002	-0.002	0.011	-0.004	-0.001	-0.002	0.000
q28perf	0.301	0.313	0.296	0.303	0.335	0.301	0.310
q58int3c	0.374	0.376	0.392	0.376	0.384	0.374	0.380
FORSAPCT	0.192	0.201	0.200	0.203	0.228	0.192	0.205
q29str2	0.411	0.419	0.402	0.408	0.419	0.411	0.412
q47sicfepm	0.556	0.563	0.552	0.562	0.559	0.556	0.559
q60form	0.783	0.769	0.775	0.777	0.771	0.783	0.775
Innov							
rdemp	0.767	0.738	0.736	0.724	0.709	0.767	0.734
REDREDES	0.806	0.786	0.794	0.808	0.844	0.806	0.807
q9rd	0.513	0.541	0.565	0.527	0.517	0.513	0.533
q58tectot	0.181	0.245	0.247	0.239	0.093	0.181	0.201
MAIMA	0.443	0.396	0.421	0.454	0.483	0.443	0.439
q42estprot	0.032	0.107	0.085	0.109	0.066	0.032	0.080
struct							
frvadven	0.244	0.242	0.260	0.239	0.176	0.244	0.232
q60org	0.857	0.847	0.853	0.854	0.871	0.857	0.857
RECIN	0.336	0.356	0.336	0.349	0.339	0.336	0.344
RECEX	0.182	0.182	0.173	0.181	0.162	0.182	0.176
q47iman	0.698	0.709	0.699	0.703	0.699	0.698	0.702
q40nqd2	0.388	0.385	0.387	0.381	0.422	0.388	0.392
Relat							
TROIC	-0.484	-0.473	-0.477	-0.478	-0.363	-0.484	-0.455
TROIF	-0.447	-0.432	-0.435	-0.440	-0.254	-0.447	-0.402
QUALIMOB	0.329	0.320	0.322	0.330	0.231	0.329	0.306
q14cliap	0.388	0.398	0.394	0.392	0.365	0.388	0.387
q58protot	0.306	0.334	0.328	0.313	0.610	0.306	0.378
CLIESATI	-0.089	-0.085	-0.086	-0.085	0.034	-0.089	-0.062
q14repr	0.648	0.650	0.650	0.649	0.561	0.648	0.632
q9sansRD	0.518	0.519	0.518	0.521	0.552	0.518	0.525
q58ext3c	0.299	0.329	0.322	0.307	0.603	0.299	0.372
CONAD	0.482	0.465	0.471	0.480	0.321	0.482	0.444
Perform							
benacta0	0.604	0.539	0.584	0.623	0.788	0.604	0.627
ventev2	0.521	0.450	0.465	0.490	0.816	0.521	0.548
benebpro0	-0.739	-0.808	-0.774	-0.738	0.036	-0.739	-0.605

Tableau 5.6 – Tableau des coefficients associés aux variables mesurées pour les 82 PMEs.

Relat à Perform est parfois positif et parfois négatif. La conclusion serait différente selon les différents jeux synthétiques obtenus ici. De même, le coefficient moyen des 5 jeux partiellement synthétiques est positif alors qu'avec le jeu de données original, il est négatif. Cependant, la valeur associée au chemin n'est probablement pas significative. Donc, la conclusion ne changerait surement pas en ce qui a trait

Départ du lien	arrivé	Coef. orig.	Coef. jeu synt.					Coef. moy.
Humain	Innov	0.671	0.678	0.662	0.683	0.645	0.671	0.668
Humain	Struct	0.692	0.698	0.705	0.694	0.705	0.692	0.699
Humain	Relat	0.472	0.481	0.493	0.473	0.565	0.472	0.497
Innov	Struct	0.179	0.164	0.153	0.174	0.165	0.179	0.167
Innov	Perform	-0.120	-0.146	-0.113	-0.125	-0.174	-0.120	-0.136
Struct	Relat	0.163	0.152	0.147	0.161	0.037	0.163	0.132
Struct	Perform	-0.063	-0.032	-0.063	-0.058	-0.204	-0.063	-0.084
Relat	Perform	-0.158	-0.173	-0.170	-0.159	0.279	-0.158	-0.076

Tableau 5.7 – Tableau des coefficients entre les variables latentes pour chaque jeu.

Perform					
Jeu Orig	Jeu Synt.1	Jeu Synt.2	Jeu Synt.3	Jeu Synt.4	Jeu Synt.5
benacta0	benacta0	benacta0	benacta0	benacta0	benacta0
benebpro0	benebpro0	benebpro0	benebpro0		benebpro0
				ventev2	ventev2

Tableau 5.8 – Variables ayant un coefficient supérieur à 0.5 dans le construit Perform.

aux liens significatifs entre les variables latentes.

Départ du lien	arrivé	Coef. orig.	Coef. jeu synt.					Coef. moy.
Humain	Innov	0.526	0.526	0.526	0.526	0.537	0.536	0.530
Humain	Struct	0.789	0.789	0.789	0.789	0.783	0.783	0.787
Humain	Relat	-0.051	-0.051	-0.051	-0.051	-0.042	-0.041	-0.047
Innov	Struct	0.177	0.177	0.177	0.177	0.185	0.184	0.180
Innov	Perform	-0.112	-0.112	-0.112	-0.112	-0.149	-0.155	-0.128
Struct	Relat	0.412	0.412	0.412	0.412	0.404	0.402	0.408
Struct	Perform	-0.083	-0.083	-0.083	-0.083	-0.186	-0.183	-0.124
Relat	Perform	-0.099	-0.099	-0.099	-0.099	0.182	0.172	0.011

Tableau 5.9 – Tableau des coefficients entre les variables latentes pour chaque jeu en considérant seulement les variables significatives.

Les résultats laissent entrevoir que l'application d'équations structurelles à des jeux de données partiellement synthétiques est possible. Il faut cependant s'assurer d'avoir un bon nombre d'observations, car, comme on l'a vu précédemment, un petit nombre d'observations peut causer des changements dans les conclusions des modèles. Nous avons conservé dans nos équations des coefficients à signe contraire ce que normalement il ne faut pas faire pour conserver un construit unidimensionnel. Plus précisément, il s'agit du construit Perform où on a pratiquement toujours conservé deux variables ayant un coefficient supérieur à 0.5. Cela peut influencer l'allure du construit.

5.2.3 Équations structurelles avec confidentialité différentielle

Pour faire suite au chapitre 4, on a testé la fonction $GenSyn\_DM$ de l'algorithme 2 avec $\alpha = 10$ lors de la génération des jeux de données partiellement synthétiques avec les données sur les PMEs. On a ensuite appliqué le modèle d'équations structurelles vu précédemment. Les résultats sont présentés dans les tableaux 5.10 et 5.11. On a seulement inclus les variables mesurées pour le construit Innov et pour le construit Perform puisque seulement ces construits ont été affectés par la synthétisation. Les variables significatives varient d'un jeu à l'autre (valeur ou variable en gras). Cependant, cela ne semble pas affecter outre mesure les coefficients des chemins entre les variables latentes puisque les coefficients moyens ressemblent aux coefficients originaux comme il est possible de le constater dans le tableau 5.11.

Variable	Coef. orig.	Coef. jeu synt					Coef. moy.
Innov							
rdemp	0.478	0.495	0.449	0.443	**0.512**	0.466	0.473
REDREDES	0.792	0.815	0.823	0.802	0.809	0.800	0.810
q9rd	0.464	0.478	0.499	0.457	0.489	0.468	0.478
q58tectot	0.248	0.267	0.230	0.257	0.293	0.249	0.259
MAIMA	**0.539**	0.431	**0.500**	**0.526**	0.333	**0.604**	0.479
q42estprot	**0.613**	**0.558**	**0.573**	**0.560**	**0.554**	0.453	**0.540**
Perform							
benacta0	**0.524**	**0.517**	**0.525**	**0.541**	0.473	**0.531**	**0.517**
ventev2	0.308	0.305	0.335	0.341	0.252	0.346	0.316
benebpro0	-0.891	-0.895	-0.887	-0.878	-0.920	-0.882	-0.892

Tableau 5.10 – Coefficients obtenus pour les variables mesurées pour les construits Innov et Perform avec les jeux synthétisés à l'aide du Dirichlet-Multinomiale où $\alpha = 10$.

Départ du lien	arrivé	Coef. orig.	Coef. jeu synt.					Coef. moy.
Humain	Innov	0.623	0.614	0.618	0.623	0.632	0.629	0.623
Humain	Struct	0.633	0.653	0.621	0.624	0.670	0.641	0.642
Humain	Relat	0.462	0.467	0.460	0.466	0.471	0.468	0.466
Innov	Struct	0.205	0.175	0.225	0.220	0.145	0.195	0.192
Innov	Perform	-0.039	-0.034	-0.010	-0.059	-0.050	-0.062	-0.043
Struct	Relat	0.175	0.171	0.175	0.172	0.166	0.168	0.170
Struct	Perform	-0.273	-0.275	-0.279	-0.259	-0.261	-0.253	-0.266
Relat	Perform	-0.123	-0.124	-0.134	-0.119	-0.124	-0.118	-0.124

Tableau 5.11 – Coefficients obtenus entre les variables latentes avec les jeux synthétisés à l'aide du Dirichlet-Multinomiale où $\alpha = 10$.

En utilisant seulement les variables ayant un coefficient supérieur à 0.5, on obtient les coefficients du modèle d'équation structurelle. Ils sont présentés dans le tableau 5.12 où on remarque qu'un jeu synthétique est particulièrement affecté par la sélection des variables. Les résultats pour le jeu synthétique 4 montrent parfois des résultats positifs au lieu d'être négatifs. Cela est probablement causé

par le dernier construit, soit Perform. Dans ce dernier, on prend deux variables ayant des coefficients de signes opposés, mais étant supérieurs à 0.5 en valeur absolue pour la majorité des jeux sauf le quatrième. Il y a certains cas où la dimension du construit Perform n'est pas unidimensionnelle. Pour ce qui est des autres jeux synthétiques, les résultats sont similaires aux résultats obtenus avec le jeu original.

Départ du lien	arrivé	Coef. orig.	Coef. jeu synt.					Coef. moy.
Humain	Innov	0.530	0.555	0.541	0.526	0.565	0.579	0.553
Humain	Struct	0.520	0.528	0.519	0.519	0.564	0.517	0.529
Humain	Relat	0.221	0.213	0.220	0.221	0.208	0.215	0.215
Innov	Struct	0.384	0.371	0.383	0.391	0.305	0.372	0.365
Innov	Perform	0.025	0.073	0.060	0.006	0.054	0.049	0.048
Struct	Relat	0.301	0.308	0.303	0.302	0.312	0.307	0.306
Struct	Perform	-0.344	-0.359	-0.362	-0.331	0.253	-0.348	-0.229
Relat	Perform	-0.102	-0.113	-0.108	-0.099	0.108	-0.109	-0.064

Tableau 5.12 – Coefficients obtenus entre les variables latentes avec les jeux synthétisés à l'aide du Dirichlet-Multinomiale où $\alpha = 10$ en conservant seulement les variables significatives.

On a testé la fonction $GenSyn\_DM$ avec une petite valeur de α, il est hâtif de conclure sur la performance du Dirichlet-Multinomiale sur les équations structurelles. On peut tout de même conclure qu'à première vue, les résultats ne semblent pas affectés par l'ajout de variation avec $\alpha = 10$.

On décide de tester avec une autre valeur, soit $\alpha = 50$. Avec celle-ci, les résultats changent déjà beaucoup. Certains coefficients significatifs avec le jeu de données original ne le sont que peu souvent dans les jeux partiellement synthétiques. Par exemple, le coefficient de la variable *q42estprot* est supérieur à 0.5 dans le jeu de données original, mais il n'est supérieur à cette valeur qu'une seule dans les jeux de données synthétiques, soit dans le jeu synthétique 3.

Variable	Coef. orig.	Coef. jeu synt					Coef. moy.
Innov							
rdemp	0.478	**0.515**	0.481	0.488	**0.511**	0.477	0.494
REDREDES	0.792	0.823	0.825	0.808	0.810	0.821	0.817
q9rd	0.464	0.495	**0.500**	0.465	**0.505**	**0.505**	0.494
q58tectot	0.248	0.280	0.294	0.262	0.299	0.282	0.283
MAIMA	**0.539**	0.459	0.413	**0.515**	0.423	0.480	0.458
q42estprot	**0.613**	0.381	0.379	**0.511**	0.389	0.340	0.400
Perform							
benacta0	**0.524**	**0.501**	**0.523**	**0.529**	**0.527**	0.483	**0.512**
ventev2	0.308	0.307	0.295	0.340	0.303	0.264	0.302
benebpro0	-0.891	-0.902	-0.893	-0.884	-0.890	-0.915	-0.897

Tableau 5.13 – Coefficients obtenus pour les variables mesurées pour les construits Innov et Perform avec les jeux synthétisés à l'aide du Dirichlet-Multinomiale où $\alpha = 50$.

En utilisant les variables ayant un coefficient supérieur à 0.5, on reproduit le modèle d'équations

structurelles. On s'intéresse particulièrement aux coefficients entre les variables latentes. Les résultats présentés dans le tableau 5.14 montrent que les coefficients ne sont pas trop affectés. Ils sont similaires malgré les nombreuses sélections différentes pour la création des variables latentes.

Départ du lien	arrivé	Coef. orig.	Coef. jeu synt.					Coef. moy.
Humain	Innov	0.530	0.571	0.570	0.541	0.556	0.570	0.561
Humain	Struct	0.520	0.538	0.539	0.495	0.566	0.539	0.535
Humain	Relat	0.221	0.214	0.210	0.221	0.206	0.210	0.212
Innov	Struct	0.384	0.342	0.352	0.423	0.310	0.352	0.356
Innov	Perform	0.025	0.000	0.063	0.055	-0.005	0.063	0.035
Struct	Relat	0.301	0.308	0.312	0.302	0.313	0.312	0.309
Struct	Perform	-0.344	-0.316	-0.345	-0.363	0.282	-0.345	-0.218
Relat	Perform	-0.102	-0.105	-0.116	-0.106	0.118	-0.116	-0.065

Tableau 5.14 – Coefficients obtenus entre les variables latentes avec les jeux synthétisés à l'aide du Dirichlet-Multinomiale où $\alpha = 50$ en conservant seulement les variables significatives.

Dans cet exemple, on a testé deux valeurs différentes, dont une qui a encouragé la sélection de différentes variables pour la création des variables latentes. Malgré cela, les résultats des coefficients entre les variables latentes sont restés similaires. Selon cet exemple, la synthétisation des variables en utilisant le principe du Dirichlet-Multinomiale ne semble pas avoir d'impact majeur sur les coefficients des modèles d'équations structurelles autre que pour la sélection des variables significatives. Cependant, la valeur maximale testée, soit 50, n'est pas très élevée pour conclure que la synthétisation n'affecte jamais les coefficients entre les variables latentes.

Conclusion

Dans ce mémoire, plusieurs points ont été abordés. Notamment, il a été question de données manquantes, de jeux de données partiellement synthétiques et d'équations structurelles. Les différents points vus avaient tous un objectif commun, soit de trouver une solution au problème de confidentialité. Ce chapitre se veut être un récapitulatif des principaux résultats obtenus tout au long du mémoire.

D'abord, on a vu dans le chapitre 1 comment synthétiser des variables et comment évaluer le risque et l'utilité auprès des jeux synthétiques créés. Ce chapitre a mis la table pour le chapitre 2 puisqu'on a appliqué ces techniques pour reproduire les résultats de l'article de Caiola et Reiter (2010). Dans ce second chapitre, on a également étudié brièvement l'impact qu'a l'ordre des variables à synthétiser sur l'utilité et le risque ainsi que l'impact du nombre de jeux synthétiques à créer pour obtenir des résultats optimaux. On a obtenu que l'ordre avait effectivement un impact à l'aide d'un jeu de données simulé et observé que certains ordres étaient plus efficaces que d'autres. On a également conclu sur le choix optimal du nombre de jeux synthétiques à utiliser. En fait, cela varie entre 3 et 10 tout dépendant de ce que l'on cherche à optimiser.

On a ensuite testé la formule 1.1 permettant de combiner les différents jeux synthétiques dans le chapitre 3 ce qui permet de répondre au premier objectif, soit de valider cette formule. À l'aide de simulations, on a montré que celle-ci fonctionne lorsqu'on synthétise avec une simple régression. Cependant, on a également montré que la formule ne s'applique pas toujours avec l'utilisation de forêts aléatoires, car on a obtenu à de multiples occasions des estimations biaisées de la véritable variance. Par exemple, lorsqu'on synthétise complètement plusieurs variables, la formule T_P ne fonctionnait pas pour évaluer la variance associée aux différents coefficients d'une régression. On a remarqué l'importance d'effectuer les tirages des valeurs synthétiques à partir d'une distribution. Sans cela, la formule T_P ne fonctionne pas pour évaluer la variance lorsqu'on synthétise quelques observations d'une variable.

Un des points principaux mentionné au début du mémoire était d'ajouter une extension s'inspirant du principe de la confidentialité différentielle à la génération de jeux de données partiellement synthétiques. En fait, cela représentait le second objectif. Dans le chapitre 4, on a montré que la formule 1.1 pour combiner ces jeux synthétiques ne fonctionne vraiment pas lorsque α augmente alors qu'on observait des biais évidents ainsi que des erreurs relatives supérieures à 90% dans certains cas ! On a alors utilisé une alternative pour vérifier si la confidentialité différentielle permettait de diminuer le

risque tout en gardant l'utilité des données élevée. Il s'agissait de faire deux graphiques de boîtes à moustaches. Le premier utilise l'erreur relative des coefficients en fonction du risque de réidentification alors que le second considère l'erreur relative de la variance des coefficients en fonction du risque. Les résultats ont montré que l'utilité des données n'est pas conservée puisque ni les coefficients ni la variance de ceux-ci sont bien estimés. On observe de très grandes erreurs relatives dans la majorité des cas. En fait, on a des valeurs extrêmes atteignant près de $1\ 000\%$ d'erreur que l'on retire pour faire les graphiques.

Dans le dernier chapitre, on a exploré un domaine peu ou pas étudié jusqu'à présent pour répondre au dernier objectif, soit l'impact de la synthétisation de variables sur des équations structurelles. On a utilisé un modèle provenant d'un article (St-Pierre et Audet, 2011) et on a également utilisé un véritable jeu de données. On a pu remarquer que les coefficients originaux ainsi que les coefficients moyens obtenus après la synthétisation sont dans l'ensemble très similaires. Ce résultat est vrai pour les coefficients entre les variables latentes et également, pour les coefficients entre les variables mesurées et une variable latente. Il faut cependant s'assurer de sélectionner les coefficients ayant une valeur supérieure à 0.5 et que le construit soit unidimensionnel pour obtenir des résultats similaires.

Il reste évidemment beaucoup de travail à accomplir pour obtenir à la fois une grande utilité dans les données tout comme un risque négligeable de réidentification des données. Par exemple, il serait pertinent de trouver une formule pour combiner les jeux synthétiques lorsque plusieurs variables sont synthétisées selon l'algorithme 1 présenté dans le chapitre 1. Il serait également intéressant de regarder comment se comporte la formule 1.1 qui combine les résultats des jeux synthétiques lorsqu'on synthétise selon un différent algorithme. C'est-à-dire qu'on génère une variable à synthétiser en considérant toutes les autres variables disponibles incluant les variables qui doivent être synthétisées (elles sont à synthétiser et ne l'ont pas été encore). Cette méthode diffère de l'algorithme 1 au sens où seules les variables déjà synthétisées et les variables qui ne sont pas à synthétiser étaient utilisées pour générer la forêt aléatoire pour une variable à synthétiser.

Bibliographie

Jonathan BARTLETT : Methodology for multiple imputation for missing data in electronic healt record data. International Biometric Society, 2014.

Gregory CAIOLA et Jerome P REITER : Random forests for generating partially synthetic, categorical data. *Transactions on Data Privacy*, 3(1):27–42, 2010.

James CARPENTER et Michael KENWARD : *Multiple imputation and its application*. John Wiley & Sons, 2012.

Anne Sophie CHAREST : *Creation and Analysis of Differentially-Private Synthetic Datasets*. Thèse de doctorat, PhD Thesis, Carnegie Mellon University, 2012.

Wynne W CHIN : Overview of the pls method. *University of Houston*, 1997.

Jörg DRECHSLER : *Synthetic datasets for statistical disclosure control : theory and implementation*, volume 201. Springer, 2011.

Jörg DRECHSLER, Stefan BENDER et Susanne RÄSSLER : Comparing fully and partially synthetic datasets for statistical disclosure control in the german iab establishment panel. *Transactions on Data Privacy*, 1(3):105–130, 2008.

Jörg DRECHSLER et Jerome P REITER : An empirical evaluation of easily implemented, nonparametric methods for generating synthetic datasets. *Computational Statistics & Data Analysis*, 55 (12):3232–3243, 2011.

George T DUNCAN, Sallie A KELLER-MCNULTY et S Lynne STOKES : Disclosure risk vs. data utility : The ru confidentiality map. *In Chance*. Citeseer, 2001.

Cynthia DWORK, Frank MCSHERRY, Kobbi NISSIM et Adam SMITH : Calibrating noise to sensitivity in private data analysis. *In Theory of Cryptography*, pages 265–284. Springer, 2006.

Claes FORNELL et Fred L BOOKSTEIN : Two structural equation models : Lisrel and pls applied to consumer exit-voice theory. *Journal of Marketing research*, pages 440–452, 1982.

Trevor HASTIE, Robert TIBSHIRANI, Jerome FRIEDMAN, T HASTIE, J FRIEDMAN et R TIBSHIRANI : *The elements of statistical learning*, volume 2. Springer, 2009.

Ori HEFFETZ et Katrina LIGETT : Privacy and data-based research. Rapport technique, National Bureau of Economic Research, 2013.

Gareth JAMES, Daniela WITTEN, Trevor HASTIE et Robert TIBSHIRANI : *An introduction to statistical learning*. Springer, 2013.

Alan F KARR, Christine N KOHNEN, Anna OGANIAN, Jerome P REITER et Ashish P SANIL : A framework for evaluating the utility of data altered to protect confidentiality. *The American Statistician*, 60(3):224–232, 2006.

Ashwin MACHANAVAJJHALA, Daniel KIFER, John ABOWD, Johannes GEHRKE et Lars VILHUBER : Privacy : Theory meets practice on the map. *In Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 277–286. IEEE, 2008.

Ralph O MUELLER : *Basic principles of structural equation modeling : An introduction to LISREL and EQS*. Springer, 1996.

Jerome P REITER : Inference for partially synthetic, public use microdata sets. *Survey Methodology*, 29(2):181–188, 2003.

Jerome P REITER : New approaches to data dissemination : A glimpse into the future (?). *Chance*, 17(3):11–15, 2004.

Jerome P REITER et Robin MITRA : Estimating risks of identification disclosure in partially synthetic data. 2008.

Donald B RUBIN : Multiple imputation for nonresponse in surveys. new york : J, 1987.

Donald B RUBIN : Statistical disclosure limitation. *Journal of Official Statistics*, 9(2):461–468, 1993.

Gaston SANCHEZ : Pls path modeling with r. *Online, January*, 2013.

Josée ST-PIERRE et Josée AUDET : Intangible assets and performance : Analysis on manufacturing smes. *Journal of Intellectual Capital*, 12(2):202–223, 2011.

Russell J STEELE, Naisyin WANG et Adrian E RAFTERY : Inference from multiple imputation for missing data using mixtures of normals. *Statistical methodology*, 7(3):351–365, 2010.

Annexe A

Données manquantes : démonstration et exemples

Proposition de la formule T_M (version complète)

Proposition 3 *(Rubin, 1987)*

Soit β, le paramètre que l'on souhaite estimer et soit m, le nombre d'imputations effectuées, l'estimation de β est :

$$\widehat{\beta}_{MI} = \sum_{k=1}^{m} \widehat{\beta}_k / m$$

où $\widehat{\beta}_k$ représente l'estimation du paramètre avec le k^e jeu de données imputé.

Un estimateur de la variance du coefficient est obtenu ainsi :

$$T_M = \widehat{W} + \left(1 + \frac{1}{m}\right)\widehat{B}$$

où $\widehat{W}$ est la moyenne de la variance intra jeu de données (within), $\widehat{W} = \frac{\sum_{k=1}^{m} \widehat{\sigma}_k^2}{m}$ et où $\widehat{\sigma}_k^2$ est la variance associée au paramètre dans le k^e jeu de données imputé. $\widehat{B}$ est la variance entre les paramètres des différents jeux imputés (between), $\widehat{B} = \frac{1}{m-1}\sum_{k=1}^{m}(\widehat{\beta}_k - \widehat{\beta}_{MI})^2$. Finalement, la distribution de β est approximativement la suivante :

$$\beta \sim t_v(\widehat{\beta}_{MI}, T_M)$$

où $v = (m-1)(1 + r_m^{-1})^2$ et où $r_m = (1 + 1/m)\frac{B_m}{\widehat{W}}$.

Démonstration de la formule T_M

Cette démonstration est inspirée de celle présentée par Carpenter et Kenward (2012) qui se base elle-même sur celle de Rubin (1987). J'ai fait quelques ajouts pour améliorer la compréhension de celle-ci.

D'abord, on note qu'on utilise un modèle bayésien afin de prédire les valeurs manquantes. On met une distribution dite a priori sur les paramètres de la distribution des observations. Puis, on calcule une distribution a posteriori pour tenir compte de l'information fournie par les données en utilisant également la distribution a priori. La distribution obtenue est une distribution des paramètres conditionnelle aux données et permet toute inférence liée aux paramètres.

On suppose d'abord qu'on tire m valeurs de la distribution a posteriori de Y_{miss}, soit $P(Y_{miss}|X,Y_{obs})$ où X représente les variables explicatives complètes, Y_{obs} sont les valeurs observées de Y. On pose $S_m = (\widehat{Q}_{*l}, U_{*l}, l = 1...m)$ pour représenter l'ensemble des statistiques des jeux de données complets par imputation, soit $\widehat{Q}$ et U qui sont évalués pour chaque jeux complets où $\widehat{Q} = E(Q|X, Y_{inc})$ et $U = V(Q|X, Y_{inc})$. On veut obtenir une estimation de Q étant donné S_m. On sait que s'il était possible d'observer S_∞, il serait possible d'approximer la distribution a posteriori de Q comme suit :

$$(Q|\overline{Q}_\infty, \overline{U}_\infty, B_\infty) \sim N(\overline{Q}_\infty, \overline{U}_\infty + B_\infty)$$

où $\overline{Q}_\infty = E[\widehat{Q}|X, Y_{obs}]$, $\overline{U}_\infty = E[U|X, Y_{obs}]$ et $B_\infty = V[\widehat{Q}|X, Y_{obs}]$. À l'aide de la théorie asymptotique, la distribution d'échantillonnage de la moyenne a posteriori suit une normale et celle de la variance a posteriori tend à avoir un ordre de variance plus petite que la moyenne a posteriori. En fait, S_m est constitué à l'aide de m tirages indépendants et identiquement distribués comme suit :

$$(\widehat{Q}_{*l}|X, Y_{obs}) \sim N(\overline{Q}_\infty, B_\infty) \tag{A.1}$$

$$(U_{*l}|X, Y_{obs}) \sim f(\overline{U}_\infty, << B_\infty) \tag{A.2}$$

où le symbole $<<$ représente que la variabilité est d'un ordre inférieur à B_∞ et f est quelconque. En acceptant les deux approximations précédentes, on a pour n'importe quelle distribution f a priori non-informative sur U_∞ :

$$(U_\infty|S_m, B_\infty) \sim f\left(\overline{U}_m, << \frac{B_\infty}{m}\right)$$

où $\overline{U}_m = \sum_{l=1}^{m} \frac{U_{*l}}{m}$.

Si la distribution a priori de $\overline{Q}_\infty|B_\infty$ est proportionnelle à une constante et qu'on utilise la distribution normale pour obtenir $\widehat{Q}_{*l}$, la distribution conditionnelle de $(\overline{Q}_\infty|S_m, B_\infty)$ est normale :

$$(\overline{Q}_\infty|S_m, B_\infty) \sim N\left(\overline{Q}_m, \frac{B_\infty}{m}\right)$$

où $\overline{Q}_m = \sum_{l=1}^{m} \frac{\widehat{Q}_{*l}}{m}$.

On veut ensuite la distribution de $(Q|S_m, B_\infty)$, on a les distributions conditionnelles suivantes : $(Q|\overline{Q}_\infty, \overline{U}_\infty, B_\infty)$, $(\overline{U}_\infty|S_m, B_\infty)$ et $(\overline{Q}_\infty|S_m, B_\infty)$. Puisque la variabilité est négligeable dans la seconde distribution conditionnelle, on peut obtenir la distribution suivante :

$$(Q|S_m, \overline{Q}_\infty, B_\infty) \sim N(\overline{Q}_\infty, \overline{U}_m + B_\infty). \tag{A.3}$$

On doit maintenant intégrer sur toutes les valeurs possibles de $\overline{Q}_\infty$ en combinant l'équation A.3 et la distribution conditionnelle de $(\overline{Q}_\infty|S_m, B_\infty)$. Pour simplifier la lecture, on pose $A = Q$, $B = S_m$, $C = \overline{Q}_\infty$, $D = \overline{U}_m + B_\infty$, $E = \overline{Q}_m$, $F = \frac{B_\infty}{m}$ et $G = B_\infty$. On a donc :

$$(A|B, C, G) \sim N(C, D)$$
$$(C|B, G) \sim N(E, F).$$

On veut la distribution de $(A|B, G)$. On utilise le théorème des probabilités totales pour un cas continu comme ci-dessous. Le calcul pour obtenir la distribution de $(A|B, G)$ ne figurait pas dans la preuve, mais il aide à comprendre le résultat final.

$$\begin{aligned} P(A|B, G) &= \int_C P(A|B, C, G)P(C|B, G)dC \\ &= \int \frac{1}{\sqrt{2\pi D}} e^{\frac{-(A-C)^2}{2D}} \frac{1}{\sqrt{2\pi F}} e^{\frac{-(C-E)^2}{2F}} dC \\ &= \frac{1}{\sqrt{2\pi D}\sqrt{2\pi F}} e^{-\frac{A^2}{2D} - \frac{E^2}{2F}} \int e^{\frac{-1}{2}\left[\frac{C^2}{D} - \frac{2AC}{D} + \frac{C^2}{F} - \frac{CE}{F}\right]} dC \end{aligned} \tag{A.4}$$

On peut réécrire l'expression dans l'exposant à l'intérieur de l'intégrale de cette façon :

$$\begin{aligned} \frac{-1}{2}\left[\frac{C^2}{D} - \frac{2AC}{D} + \frac{C^2}{F} - \frac{CE}{F}\right] &= \frac{-1}{2DF}[FC^2 - 2FAC + DC^2 - 2DCE] \\ &= \frac{-1}{2DF}[F + D]\left[C^2 - \frac{2CFA}{F + D} - \frac{2CDE}{F + D}\right] \\ &= \frac{-1}{2DF}[F + D]\left[C^2 - 2C\left[\frac{FA + DE}{F + D}\right]\right] \\ &= \frac{-1}{DF}[F + D]\left[\left(C - \frac{FA + DE}{F + D}\right)^2 - \left(\frac{FA + DE}{F + D}\right)^2\right] \end{aligned}$$

En mettant la dernière équation simplifiée obtenue dans l'équation A.4, on retrouve aisément une distribution normale. Il suffit d'ajouter un terme dans l'intégrale afin d'avoir une loi normale de moyenne $\mu = \frac{FA+DE}{F+D}$ et de variance $\sigma^2 = \frac{DF}{F+D}$:

$$P(A|B,G) = \frac{1}{\sqrt{2\pi D}\sqrt{2\pi F}} e^{(-\frac{A^2}{2D} - \frac{E^2}{2F})} e^{\left(\left[\frac{F+D}{2DF}\right]\left[\frac{FA+DE}{F+D}\right]^2\right)} \int e^{\frac{-1}{DF}[F+D]\left[\left(C - \frac{FA+DE}{F+D}\right)^2\right]} dC$$

$$P(A|B,G) = \frac{\sqrt{2\pi\frac{DF}{F+D}}}{\sqrt{2\pi D}\sqrt{2\pi F}} e^{(-\frac{A^2}{2D} - \frac{E^2}{2F})} e^{\left(\left[\frac{F+D}{2DF}\right]\left[\frac{FA+DE}{F+D}\right]^2\right)} \int \frac{1}{\sqrt{2\pi\frac{DF}{F+D}}} e^{\frac{-1}{DF}[F+D]\left[\left(C - \frac{FA+DE}{F+D}\right)^2\right]} dC.$$

L'intégrale sur le domaine de C donne évidemment 1. Après simplification, on a la densité suivante :

$$P(A|B,G) = \frac{\sqrt{\frac{1}{F+D}}}{\sqrt{2\pi}} e^{\frac{-1}{2}\left[\frac{A^2}{D} + \frac{E^2}{F} - \frac{1}{DF(F+D)}(FA+DE)^2\right]}.$$

Le terme entre crochet de l'exposant peut se réécrire ainsi :

$$\begin{aligned}
\frac{A^2}{D} + \frac{E^2}{F} - \frac{1}{DF(F+D)}(FA+DE)^2 =& \frac{1}{DF(F+D)}((F+D)FA^2 + E^2D(F+D) - F^2A^2 \\
& - 2FADE - D^2E^2) \\
=& \frac{A^2 + E^2 - 2AE}{F+D} \\
=& \frac{(A-E)^2}{F+D}.
\end{aligned}$$

On obtient donc :

$$P(A|B,G) = \frac{1}{\sqrt{2\pi(F+D)}} e^{\frac{-1}{2(F+D)}(A-E)^2}.$$

On reconnait la distribution d'une loi normale.

$$(A|B,G) \sim N(E, F+D).$$

En remplaçant par les termes originaux, on trouve que :

$$(Q|S_m, B_\infty) \sim N\left(\overline{Q}_m, \overline{U}_m + B_\infty + \frac{B_\infty}{m}\right).$$

Il ne reste qu'à trouver la distribution conditionnelle de $(B_\infty|S_m)$ puisqu'on cherche $(Q|S_m)$. On suppose que la distribution a priori de $log(B_\infty)$ est proportionnelle à une constante. Sous les distributions asymptotiques des équations A.1 et A.2, la distribution conditionnelle de $(B_\infty|S_m)$ est proportionnelle à une χ^2 inverse avec $m-1$ degrés de liberté :

$$(m-1)\frac{B_m}{B_\infty}|S_m \sim \chi^2_{m-1}$$

où $B_m = \sum_{l=1}^{m} \frac{(\widehat{Q}_{*l}-\overline{Q}_m)^2}{m-1}$. Si la distribution conditionnelle de $(\overline{U}_m + (1+\frac{1}{m})B_\infty|S_m)$ était également une χ^2, on aurait que $(Q|S_m)$ suit une loi de Student. Cependant, il est faux de conclure qu'une variable suivant une χ^2 inverse plus une constante soit distribuée selon une χ^2 inverse. On peut cependant l'approximer par cette distribution. On a besoin de l'approximation suivante pour ce faire :

Si yx^{-1} suit une loi du χ^2 avec y degrés de liberté, alors $y(1+a^{-1})^2(1+a)/(1+ax)$ suit également une loi du χ^2 avec $y(1+a^{-1})^2$ degrés de liberté. L'idée général pour obtenir ces équations est d'utiliser les deux premiers moments d'une χ^2.

Dans notre cas, on a $y = m-1$, $x = B_\infty/B_m$, $a = r_m = (1+1/m)\frac{B_m}{\overline{U}_m}$. Les degrés de liberté sont $v = (m-1)(1+r_m^{-1})^2$. On pose $T_m = \overline{U}_m + (1+1/m)B_m$. L'approximation est donc :

$$vT_m[\overline{U}_m + (1+1/m)B_\infty]|S_m \sim \chi^2_v.$$

On peut maintenant trouver la distribution de $Q|S_m$:

$$(Q|S_m) \sim t_v(\overline{Q}_m, T_m).$$

Cette démonstration a permis de comprendre la provenance de cette formule utilisée afin de combiner différents jeux de données imputés.

Deux exemples utilisant la formule T_M pour calculer la variance lorsqu'on utilise des jeux de données imputés

Exemple 8 *Calcul de la moyenne d'une variable imputée dont la variance est connue*

On veut estimer la moyenne d'un échantillon ainsi que la variance associée à cette moyenne. Normalement, il s'agit simplement de $\overline{Y}$ et de σ^2/n, mais on a des données manquantes dans l'échantillon et on ne veut pas perdre d'information. Comme les données non-observées sont manquantes complètement aléatoires (MCAR) dans l'exemple suivant, il est possible d'utiliser l'imputation multiple. Pour ce faire, on a besoin d'un modèle adéquat pour obtenir les valeurs imputées. On suppose que la taille

d'échantillon est suffisamment grande pour que la distribution de μ, qui est considéré aléatoire, soit approximativement une normale avec moyenne $\overline{Y}_{obs}$ (la moyenne des Y observés non-manquants) et de variance σ^2/n_{obs} (n_{obs} est la taille d'échantillon sans les données manquantes). En combinant cette distribution et la distribution des Y observés, on obtient la loi a posteriori suivante :

$$Y_M|Y_{obs} \sim N\left[\overline{\boldsymbol{Y}}_{obs}, \sigma^2_{mat}\left(1+\frac{1}{n_{obs}}\right)\right]$$

où $\overline{\boldsymbol{Y}}_{obs}$ représente un vecteur de dimension n_{miss} (nombre de valeurs manquantes) composé de $\overline{Y}_{obs}$. σ^2_{mat} est une matrice diagonale avec σ^2 sur celle-ci dont la dimension est n_{miss} par n_{miss}.

Plus simplement, afin d'obtenir une nouvelle valeur pour la k^e imputation et un i parmi les données manquantes, on utilise la formule suivante :

$$\tilde{Y}_{i,k} = \overline{Y}_{obs} + e_{i,k} + s_k$$

où $e_{i,k} \sim N(0, \sigma^2)$ et $s_k \sim N(0, \sigma^2/n_{obs})$.

Puisqu'il s'agit d'un exemple simple, on peut évaluer théoriquement la variance de la moyenne de Y auprès des jeux de données imputés comme le démontre Carpenter et Kenward (2012). La variance théorique est $\frac{\sigma^2}{n_{obs}}(1+\frac{\pi_M}{m})$ où π_M est la proportion de données manquantes et m représente le nombre de jeux imputés créé.

Pour la simulation, on sélectionne un échantillon de taille $n = 100$ d'une loi normale centrée et de variance $\sigma^2 = 100$. On fixe le nombre de valeurs qui seront manquantes à 35 afin d'avoir une variance théorique constante (la même pour chaque réplication). On sélectionne 35 unités parmi les 100 et ces unités sont considérées comme données manquantes. On impute ensuite les données manquantes avec la méthode décrite précédemment, on note la moyenne des y et la variance de ceux-ci. On crée en tout 100 jeux de données imputés. On calcule ensuite la moyenne globale des y ainsi que la variance selon la formule 3.1 proposée par Rubin (1987). Pour valider la formule, on réplique $10\ 000$ fois la procédure décrite. On retrouve dans le tableau A.1 les différents résultats dont la moyenne de y des réplications ainsi que sa variance. On y retrouve également la variance estimée avec la formule et la variance théorique tout comme les erreurs relatives.

On obtient environ la moyenne initiale qui était $\mu = 0$ avec une moyenne de -0.0146 et on trouve une estimation de la variance similaire à la véritable variance après $10\ 000$ réplications avec une erreur relative de moins de 0.1%. La variance des réplications est pratiquement égale à la variance théorique avec une erreur relative de 0.7%.

Exemple 9 *Calcul de la variance des coefficients de régression*

Résultats des réplications			
Moyenne de y	-0.0146		
	Moyenne	Écart-type	Erreur relative
Variance des réplications	1.5325	-	0.7%
T_M	1.5441	0.1401	0.02%
V_{theo}	1.5438	0	-

Tableau A.1 – Résultats des 10 000 réplications de 100 jeux imputés pour la variance estimée, pour la variance théorique et pour notre variable y.

Pour ce deuxième exemple, on veut estimer les coefficients d'une régression linéaire en présence de données manquantes ainsi que la variance associée aux coefficients. À partir des données observées, on estime les coefficients de régression ainsi que la variance associée avec les formules suivantes :

$$\widehat{\beta}_{obs} = (X'_{obs}X_{obs})^{-1}X'_{obs}Y_{obs}$$
$$\boldsymbol{V}_{obs} = Var(\widehat{\beta}_{obs}) = \sigma^2(X'_{obs}X_{obs})^{-1}$$

où X_{obs} est une matrice dont les colonnes sont les vecteurs des variables indépendantes et où les lignes sont les valeurs de ces variables pour lesquelles Y est observé. On suppose que l'échantillon est relativement large pour que $\tilde{\beta}|Y_{obs} \sim N(\widehat{\beta}_{obs}; \boldsymbol{V}_{obs})$. On peut alors imputer avec le modèle suivant :

$$\tilde{Y}_{M,k} = X_M\tilde{\beta} + e_k$$

où X_M représente la matrice des données manquantes (les lignes où Y est manquant) et $e_k \sim N(0, \sigma^2)$. Puisque c'est le coefficient de la régression qui nous intéresse, on l'estime ainsi pour le k^e jeu de données imputé :

$$\tilde{\beta}_k = \widehat{\beta}_{obs} + (X'X)^{-1}X'_M(X_M bi_k + e_k)$$

où $bi_k \sim N(0, V_{obs})$. En calculant la moyenne des $\tilde{\beta}_k$, on obtient $\widehat{\beta}_{MI}$. La variance théorique (Carpenter et Kenward, 2012) associée à cet estimateur est :

$$V_{theo} = Var(\widehat{\beta}_{MI}) = \boldsymbol{V}_{obs} + \frac{1}{m}\{\boldsymbol{V}_{obs} - \sigma^2(X'X)^{-1}\}.$$

L'estimation de la variance est :

$$T_M = \frac{\sum_{k=1}^{m} v_k}{m} + \left(1 + \frac{1}{k}\right)m_k$$

où v_k est la variance du paramètre dans le jeu synthétique k et B_k est la variance entre les paramètres des différents jeux synthétiques.

Dans notre simulation, on a généré un jeu de données de $n = 100$ observations comme suit :

$$x_1 \sim Binom(10, 0.463)$$

$$x_2 \sim \begin{cases} Binom(5, 0.73) & si \quad x_1 > 5 \\ Binom(5, 0.41) & si \quad x_1 \leq 5 \end{cases}$$

On crée la variable dépendante en fonction de x_1, de x_2 et d'un terme d'erreur :

$$y = 3 \cdot x_1 + 12 \cdot x_2 + \epsilon$$

où $\epsilon \sim N(0, \sigma^2 = 100)$.

On imite l'exemple précédent pour la sélection des données manquantes en sélectionnant au hasard 35 unités. On impute selon le modèle mentionné. Les résultats obtenus après $5\ 000$ réplications sont les suivants où les nombres entre parenthèses représentent l'écart-type :

$$Var(\widehat{\beta}_{rep}) = \begin{pmatrix} 0.3803 & -0.5966 \\ -0.5966 & 1.1387 \end{pmatrix}$$

$$Moyenne(V_{theo}) = \begin{pmatrix} 0.3782\ (0.0747) & -0.5957\ (0.1240) \\ -0.5957\ (0.1240) & 1.1376\ (0.2244) \end{pmatrix}$$

$$Moyenne(T_M) = \begin{pmatrix} 0.3771\ (0.0818) & -0.5941\ (0.1351) \\ -0.5941\ (0.1351) & 1.1350\ (0.2463) \end{pmatrix}$$

L'estimation de la variance de notre coefficient fonctionne et donc, que les formules proposées fonctionnent bien dans ces exemples.

Annexe B

Code Chapitre 2

Code pour générer les jeux synthétiques

```
GenSyn2= function(n=1,MatIni,ColSyn,nArbre=500,sortie=list())
{
  library("randomForest")
  MatFin=matrix(NA, nrow(MatIni),ncol(MatIni))
  VarExp=NA
  toChange=1:ncol(MatIni) %in% ColSyn
  MatFin[,toChange==F]=as.matrix(MatIni[,toChange==F])
  for(i in 1:length(ColSyn))
  {
    VarSyn=NA
    VarSyn<-as.factor(MatIni[,ColSyn[i]])
    VarExp<-as.matrix(MatIni[,c(which(toChange==F), which(toChange
            ==T)[1:i-1])])
    RF.VarSyn=randomForest(VarExp,VarSyn,ntree=nArbre,xtest=as.matrix
              (MatFin[,c(which(toChange==F), which(toChange==T)[1:i-1
              ])]))
    for(j in 1:nrow(MatIni))
    {
      proba=RF.VarSyn$test$votes[j,]
      MatFin[j,ColSyn[i]]=as.numeric(row.names(which(rmultinom(1,1,
         prob=proba)!=0,arr.ind=T)))
    }
  }
  colnames(MatFin)=colnames(MatIni)
  sortie[[n]]=MatFin
```

```
  ifelse(n==1,return(sortie),sortie[[n-1]]<-return(GenSyn2(n-1,MatIni,
   ColSyn,nArbre,sortie)))
}
```

Code Utilité

```
# Karr et al. 2006 A framework...

Inter=function(IC)
{
  Intersection=matrix(NA,dim(IC)[1],((dim(IC)[2]/2)-1))
  for(i in 1:dim(IC)[1])
  {
    for(j in 1:((dim(IC)[2]/2)-1))
    {
      Li=max(IC[i,1],IC[i,2*j+1])
      Ui=min(IC[i,2],IC[i,2*j+2])
      Intersection[i,j]<-ifelse(Li>Ui, 0, (1/2)*(Ui-Li)/(IC[i,2]-
      IC[i,1]) + (1/2)*(Ui-Li)/(IC[i,2*j+2]-IC[i,2*j+1]))
      rownames(Intersection)<-rownames(IC)
    }
  }
  return (Intersection)
}
round(Inter(IC),2)
round(apply(Inter(IC),2,mean),3)
```

Code pour reproduire les résultats

```
load("CPS2000RC2.Rdata")

# Statistique descriptive:

prop.table(table(CPS2000RC$ASEX))
prop.table(table(CPS2000RC$ARAC))
prop.table(table(CPS2000RC$AMARITL))
prop.table(table(CPS2000RC$AHGA))
```

```
tab=round(apply(CPS2000[,4:12],2, FUN=function(x){moy=mean(x)
                                        sd=sd(x)
                                        min=min(x)
                                        max=max(x)
                                        med=median(x)
                                        mat=c(moy, sd, min, max, med)
                                        names(mat)<-c("moy", "sd", "min",
                                        "max", "med")
                                        return(mat)}),2)



load("JeuSynU2.Rdata")
a=prop.table(table(as.data.frame(JeuSynU[[1]])$ASEX))
b=prop.table(table(as.data.frame(JeuSynU[[2]])$ASEX))
c=prop.table(table(as.data.frame(JeuSynU[[3]])$ASEX))
d=prop.table(table(as.data.frame(JeuSynU[[4]])$ASEX))
e=prop.table(table(as.data.frame(JeuSynU[[5]])$ASEX))
(a+b+c+d+e)/5
a=prop.table(table(as.data.frame(JeuSynU[[1]])$ARAC))
b=prop.table(table(as.data.frame(JeuSynU[[2]])$ARAC))
c=prop.table(table(as.data.frame(JeuSynU[[3]])$ARAC))
d=prop.table(table(as.data.frame(JeuSynU[[4]])$ARAC))
e=prop.table(table(as.data.frame(JeuSynU[[5]])$ARAC))
(a+b+c+d+e)/5
a=prop.table(table(as.data.frame(JeuSynU[[1]])$AMARITL))
b=prop.table(table(as.data.frame(JeuSynU[[2]])$AMARITL))
c=prop.table(table(as.data.frame(JeuSynU[[3]])$AMARITL))
d=prop.table(table(as.data.frame(JeuSynU[[4]])$AMARITL))
e=prop.table(table(as.data.frame(JeuSynU[[5]])$AMARITL))
(a+b+c+d+e)/5



# Tester les trois régressions.

# Première régression:

# Sur le jeu initial:
```

```
CPS2000<-CPS2000RC
CPS2000_log<-CPS2000[which(CPS2000$HTOTVAL > 0),]
CPS2000_logPS<-CPS2000[which(CPS2000$HTOTVAL > 0),]
CPS2000_log$HOUSESUP1<-CPS2000_log$HNUM>1
CPS2000_log$age2<-(CPS2000_log$AAGE^2)

coef=round(summary(lm(log(HTOTVAL)~factor(ARACE)+AHGA+as.factor(
  HOUSESUP1)+PROPTAX+AAGE+age2+factor(ASEX)*factor(AMARITL),
  data=CPS2000_log))$coefficient[,1],5)
borne=round(confint(lm(log(HTOTVAL)~factor(ARACE)+AHGA+as.factor(
  HOUSESUP1)+PROPTAX+AAGE+age2+factor(ASEX)*factor(AMARITL),
  data=CPS2000_log)),5)

library(xtable)

IC=round(cbind(borne[,1],borne[,2]),5)


#JeuSyn_test11<-replicate(5,GenSyn2(1,CPS2000_logPS,c(1:3)))
#JeuSyn_test12<-replicate(5,GenSyn2(1,CPS2000_logPS,c(1:3)))
#save(JeuSyn_test11,file="JeuSyn11RC.RData")
#save(JeuSyn_test12,file="JeuSyn12RC.RData")
load("JeuSyn11RC.RData")
load("JeuSyn12RC.RData")
Reg.Coe1<-NA
Reg.Std1<-NA
for(i in 1:5)
{
  Temp=as.data.frame(JeuSyn_test11[[i]])
  Temp$HOUSESUP1<-Temp$HNUM>1
  Temp$age2<-(Temp$AAGE^2)
  if(i!=1)
  {
    Reg.Coe1<-cbind(Reg.Coe1,summary(lm(log(HTOTVAL)~factor(ARACE)
    +AHGA+as.factor(HOUSESUP1)+PROPTAX+AAGE+age2+
    factor(ASEX)*factor(AMARITL),data=Temp))$coefficients[,1])
    Reg.Std1<-cbind(Reg.Std1,summary(lm(log(HTOTVAL)~factor(ARACE)
    +AHGA+as.factor(HOUSESUP1)+PROPTAX+AAGE+age2+
    factor(ASEX)*factor(AMARITL),data=Temp))$coefficients[,2]^2)
```

```
  }
  else
  {
    Reg.Coe1<-summary(lm(log(HTOTVAL)~factor(ARACE)+AHGA+as.factor(
    HOUSESUP1)+PROPTAX+AAGE+age2+factor(ASEX)*factor(AMARITL),
    data=Temp))$coefficients[,1]
    Reg.Std1<-summary(lm(log(HTOTVAL)~factor(ARACE)+AHGA+as.factor(
    HOUSESUP1)+PROPTAX+AAGE+age2+factor(ASEX)*factor(AMARITL),
    data=Temp))$coefficients[,2]^2
  }
}

#Coefficient moyen :
qmoy<-apply(Reg.Coe1,MARGIN=1,FUN=mean)
round(qmoy,6)
b=(apply(Reg.Coe1,MARGIN=1,FUN=function(x){sum(x^2)})-5*qmoy^2)/(4)
u=apply(Reg.Std1,MARGIN=1,FUN=mean)
Tm=u+(b/5)
#DL:
Vm=(4)*(1+5*(u/b))^2

#IC :
coef=cbind(coef,round(qmoy,5))
IC=round(cbind(IC,qmoy- qt(0.975,Vm)*sqrt(Tm),qmoy+
qt(0.975,Vm)*sqrt(Tm)),5)

Reg.Coe1<-NA
Reg.Std1<-NA
for(i in 1:5)
{
  Temp=as.data.frame(JeuSyn_test12[[i]])
  Temp$HOUSESUP1<-Temp$HNUM>1
  Temp$age2<-(Temp$AAGE^2)
  if(i!=1)
  {
    Reg.Coe1<-cbind(Reg.Coe1,summary(lm(log(HTOTVAL)~factor(ARACE)+
    AHGA+as.factor(HOUSESUP1)+PROPTAX+AAGE+age2+
    factor(ASEX)*factor(AMARITL),data=Temp))$coefficients[,1])
    Reg.Std1<-cbind(Reg.Std1,summary(lm(log(HTOTVAL)~factor(ARACE)+
```

```
    AHGA+as.factor(HOUSESUP1)+PROPTAX+AAGE+age2+
    factor(ASEX)*factor(AMARITL),data=Temp))$coefficients[,2]^2)
  }
  else
  {
    Reg.Coe1<-summary(lm(log(HTOTVAL)~factor(ARACE)+AHGA+as.factor(
    HOUSESUP1)+PROPTAX+AAGE+age2+factor(ASEX)*factor(AMARITL),
    data=Temp))$coefficients[,1]
    Reg.Std1<-summary(lm(log(HTOTVAL)~factor(ARACE)+AHGA+as.factor(
    HOUSESUP1)+PROPTAX+AAGE+age2+factor(ASEX)*factor(AMARITL),
    data=Temp))$coefficients[,2]^2
  }
}

#Coefficient moyen :
qmoy<-apply(Reg.Coe1,MARGIN=1,FUN=mean)
round(qmoy,6)
b=(apply(Reg.Coe1,MARGIN=1,FUN=function(x){sum(x^2)})-5*qmoy^2)/(4)
u=apply(Reg.Std1,MARGIN=1,FUN=mean)
Tm=u+(b/5)
#DL:
Vm=(4)*(1+5*(u/b))^2

#IC :

coef=cbind(coef,round(qmoy,5))
IC=round(cbind(IC,qmoy- qt(0.975,Vm)*sqrt(Tm),qmoy+ qt(0.975,Vm)*
sqrt(Tm)),5)

xtable(round(coef,2))
xtable(round(IC,5))



#2e régression



CPS2000_HSSVAL<-CPS2000RC[which(CPS2000RC$SSVAL > 0),]
```

```
CPS2000_HSSVAL<-CPS2000_HSSVAL[which(CPS2000_HSSVAL$AAGE > 54),]
CPS2000_HSSVAL<-CPS2000_HSSVAL[,c(3,2,1,4,5,6,7,8,9,10,11,12)]
#JeuSyn2.1<-replicate(5,GenSyn2(1,CPS2000_HSSVAL,c(1:3)))
#JeuSyn2.2<-replicate(5,GenSyn2(1,CPS2000_HSSVAL,c(1:3)))
#save(JeuSyn2.1,file="JeuSyn21RC.RData")
#save(JeuSyn2.2,file="JeuSyn22RC.RData")
load("JeuSyn21RC.RData")
load("JeuSyn22RC.RData")
CPS2000_HSSVAL$wido<-CPS2000_HSSVAL$AMARITL %in% 4
CPS2000_HSSVAL$Divo<-CPS2000_HSSVAL$AMARITL %in% 5
CPS2000_HSSVAL$Single<-CPS2000_HSSVAL$AMARITL %in% c(6,7)

CPS2000_HSSVAL$school<-ifelse(CPS2000_HSSVAL$AHGA %in% c(32:39),1,
                       ifelse(CPS2000_HSSVAL$AHGA %in% 40,2,
                       ifelse(CPS2000_HSSVAL$AHGA %in% c(41:42),3,
                       ifelse(CPS2000_HSSVAL$AHGA %in% c(43:46),4,0))))

coef=summary(lm(sqrt(SSVAL)~as.factor(ASEX)+as.factor(ARACE)+as.factor(
wido)+as.factor(Divo)+as.factor(Single)+as.factor(school)+AAGE,data=
CPS2000_HSSVAL))$coefficients[,1]
IC=round(confint(lm(sqrt(SSVAL)~as.factor(ASEX)+as.factor(ARACE)+
as.factor(wido)+as.factor(Divo)+as.factor(Single)+as.factor(school)+AAGE,
data=CPS2000_HSSVAL)),3)

Reg.Coe2<-NA
Reg.Std2<-NA
for(i in 1:5)
{
  Temp=as.data.frame(JeuSyn2.1[[i]])
  Temp$wido<-Temp$AMARITL %in% 4
  Temp$Divo<-Temp$AMARITL %in% 5
  Temp$Single<-Temp$AMARITL %in% c(6,7)

  Temp$school<-ifelse(Temp$AHGA %in% c(32:39),1,
               ifelse(Temp$AHGA %in% 40,2,
               ifelse(Temp$AHGA %in% c(41:42),3,
               ifelse(Temp$AHGA %in% c(43:46),4,0))))

  if(i!=1)
```

```
  {
    Reg.Coe2<-cbind(Reg.Coe2,summary(lm(sqrt(SSVAL)~as.factor(ASEX)+
    as.factor(ARACE)+as.factor(wido)+as.factor(Divo)+
    as.factor(Single)+as.factor(school)+AAGE,data=Temp))$coefficients[,1])
    Reg.Std2<-cbind(Reg.Std2,summary(lm(sqrt(SSVAL)~as.factor(ASEX)+
    as.factor(ARACE)+as.factor(wido)+as.factor(Divo)+as.factor(Single)+
    as.factor(school)+AAGE,data=Temp))$coefficients[,2]^2)
  }
  else
  {Reg.Coe1<-NA
    Reg.Coe2<-summary(lm(sqrt(SSVAL)~as.factor(ASEX)+as.factor(ARACE)
    +as.factor(wido)+as.factor(Divo)+as.factor(Single)+as.factor(school)
    +AAGE,data=Temp))$coefficients[,1]
    Reg.Std2<-summary(lm(sqrt(SSVAL)~as.factor(ASEX)+as.factor(ARACE)
    +as.factor(wido)+as.factor(Divo)+as.factor(Single)+as.factor(school)+
    AAGE,data=Temp))$coefficients[,2]^2
  }
}

#Coefficient moyen :
qmoy<-apply(Reg.Coe2,MARGIN=1,FUN=mean)
round(qmoy,3)
b=(apply(Reg.Coe2,MARGIN=1,FUN=function(x){sum(x^2)})-5*qmoy^2)/(4)
u=apply(Reg.Std2,MARGIN=1,FUN=mean)
Tm=u+(b/5)
#DL:
Vm=(4)*(1+5*(u/b))^2

#IC:
coef=cbind(coef,qmoy)
IC=round(cbind(IC,qmoy- qt(0.975,Vm)*sqrt(Tm),qmoy+ qt(0.975,Vm)*sqrt(Tm)),5)


Reg.Coe2<-NA
Reg.Std2<-NA
for(i in 1:5)
{
  Temp=as.data.frame(JeuSyn2.2[[i]])
  Temp$wido<-Temp$AMARITL %in% 4
```

```
  Temp$Divo<-Temp$AMARITL %in% 5
  Temp$Single<-Temp$AMARITL %in% c(6,7)

  Temp$school<-ifelse(Temp$AHGA %in% c(32:39),1,
                      ifelse(Temp$AHGA %in% 40,2,
                             ifelse(Temp$AHGA %in% c(41:42),3,
                                    ifelse(Temp$AHGA %in% c(43:46),4,0))))

  if(i!=1)
  {
    Reg.Coe2<-cbind(Reg.Coe2,summary(lm(sqrt(SSVAL)~as.factor(ASEX)+
    as.factor(ARACE)+as.factor(wido)+as.factor(Divo)+as.factor(Single)+
    as.factor(school)+ AAGE,data=Temp))$coefficients[,1])
    Reg.Std2<-cbind(Reg.Std2,summary(lm(sqrt(SSVAL)~as.factor(ASEX)+
    as.factor(ARACE)+as.factor(wido)+as.factor(Divo)+as.factor(Single)+
    as.factor(school)+ AAGE,data=Temp))$coefficients[,2]^2)
  }
  else
  {
    Reg.Coe2<-summary(lm(sqrt(SSVAL)~as.factor(ASEX)+as.factor(ARACE)+
    as.factor(wido)+as.factor(Divo)+as.factor(Single)+as.factor(school)
    +AAGE,data=Temp))$coefficients[,1]
    Reg.Std2<-summary(lm(sqrt(SSVAL)~as.factor(ASEX)+as.factor(ARACE)+
    as.factor(wido)+as.factor(Divo)+as.factor(Single)+as.factor(school)+
    AAGE,data=Temp))$coefficients[,2]^2
  }
}

#Coefficient moyen :
qmoy<-apply(Reg.Coe2,MARGIN=1,FUN=mean)
round(qmoy,3)
b=(apply(Reg.Coe2,MARGIN=1,FUN=function(x){sum(x^2)})-5*qmoy^2)/(4)
u=apply(Reg.Std2,MARGIN=1,FUN=mean)
Tm=u+(b/5)
#DL:
Vm=(4)*(1+5*(u/b))^2

#IC:
coef=cbind(coef,qmoy)
```

```
IC=round(cbind(IC,qmoy- qt(0.975,Vm)*sqrt(Tm),qmoy+ qt(0.975,Vm)*sqrt(Tm)),5)

print(xtable(coef))
print(xtable(IC))


apply(Inter(IC),2,mean)


#3e régression

CPS2000_3<-CPS2000RC[which(CPS2000RC$HCSPVAL > 0),]
CPS2000_3PS<-CPS2000[which(CPS2000$HCSPVAL > 0),]
CPS2000_3PS<-CPS2000_3PS[,c(3,2,1,4:12)]
CPS2000_3$nonwhite=ifelse(CPS2000_3$ARACE %in% c(2:4),1,0)
coef=summary(lm(sqrt(HCSPVAL)~as.factor(ASEX)+as.factor(nonwhite)
+AHGA+HNUML18,data=CPS2000_3))$coefficients[,1]
IC=round(confint(lm(sqrt(HCSPVAL)~as.factor(ASEX)+as.factor(nonwhite)
+AHGA+HNUML18,data=CPS2000_3)),3)
#JeuSyn3.1<-replicate(5,GenSyn2(1,CPS2000_3PS,c(1:3)))
#JeuSyn3.2<-replicate(5,GenSyn2(1,CPS2000_3PS,c(1:3)))
#save(JeuSyn3.1,file="JeuSyn31RC.RData")
#save(JeuSyn3.2,file="JeuSyn32RC.RData")
load("JeuSyn31RC.RData")
load("JeuSyn32RC.RData")



Reg.Coe3<-NA
Reg.Std3<-NA
for(i in 1:5)
{
  Temp=as.data.frame(JeuSyn3.1[[i]])
  Temp$nonwhite=ifelse(Temp$ARACE %in% c(2:4),1,0)

  if(i!=1)
  {
    Reg.Coe3<-cbind(Reg.Coe3,summary(lm(sqrt(HCSPVAL)~as.factor(ASEX)+
    as.factor(nonwhite)+AHGA+HNUML18,data=Temp))$coefficients[,1])
```

```
    Reg.Std3<-cbind(Reg.Std3,summary(lm(sqrt(HCSPVAL)~as.factor(ASEX)+
    as.factor(nonwhite)+AHGA+HNUML18,data=Temp))$coefficients[,2]^2)
  }
  else
  {
    Reg.Coe3<-summary(lm(sqrt(HCSPVAL)~as.factor(ASEX)+as.factor(nonwhite)
    +AHGA+HNUML18,data=Temp))$coefficients[,1]
    Reg.Std3<-summary(lm(sqrt(HCSPVAL)~as.factor(ASEX)+as.factor(nonwhite)
    +AHGA+HNUML18,data=Temp))$coefficients[,2]^2
  }
}

#Coefficient moyen :
qmoy<-apply(Reg.Coe3,MARGIN=1,FUN=mean)
round(qmoy,3)
b=(apply(Reg.Coe3,MARGIN=1,FUN=function(x){sum(x^2)})-5*qmoy^2)/(4)
u=apply(Reg.Std3,MARGIN=1,FUN=mean)
Tm=u+(b/5)
#DL:
Vm=(4)*(1+5*(u/b))^2

#IC:
coef=cbind(coef,qmoy)
IC=round(cbind(IC,qmoy- qt(0.975,Vm)*sqrt(Tm),qmoy+ qt(0.975,Vm)*sqrt(Tm)),5)




Reg.Coe3<-NA
Reg.Std3<-NA
for(i in 1:5)
{
  Temp=as.data.frame(JeuSyn3.2[[i]])
  Temp$nonwhite=ifelse(Temp$ARACE %in% c(2:4),1,0)

  if(i!=1)
  {
    Reg.Coe3<-cbind(Reg.Coe3,summary(lm(sqrt(HCSPVAL)~as.factor(ASEX)+
    as.factor(nonwhite)+AHGA+HNUML18,data=Temp))$coefficients[,1])
```

```
    Reg.Std3<-cbind(Reg.Std3,summary(lm(sqrt(HCSPVAL)~as.factor(ASEX)+
    as.factor(nonwhite)+AHGA+HNUML18,data=Temp))$coefficients[,2]^2)
  }
  else
  {
    Reg.Coe3<-summary(lm(sqrt(HCSPVAL)~as.factor(ASEX)+as.factor(nonwhite)
    +AHGA+HNUML18,data=Temp))$coefficients[,1]
    Reg.Std3<-summary(lm(sqrt(HCSPVAL)~as.factor(ASEX)+as.factor(nonwhite)
    +AHGA+HNUML18,data=Temp))$coefficients[,2]^2
  }
}

#Coefficient moyen :
qmoy<-apply(Reg.Coe3,MARGIN=1,FUN=mean)
round(qmoy,3)
b=(apply(Reg.Coe3,MARGIN=1,FUN=function(x){sum(x^2)})-5*qmoy^2)/(4)
u=apply(Reg.Std3,MARGIN=1,FUN=mean)
Tm=u+(b/5)
#DL:
Vm=(4)*(1+5*(u/b))^2

#IC:
coef=cbind(coef,qmoy)
IC=round(cbind(IC,qmoy- qt(0.975,Vm)*sqrt(Tm),qmoy+ qt(0.975,Vm)*sqrt(Tm)),5)


Inter=function(IC)
{
  Intersection=matrix(NA,dim(IC)[1],((dim(IC)[2]/2)-1))
  for(i in 1:dim(IC)[1])
  {
    for(j in 1:((dim(IC)[2]/2)-1))
    {
      Li=max(IC[i,1],IC[i,2*j+1])
      Ui=min(IC[i,2],IC[i,2*j+2])
      Intersection[i,j]<-ifelse(Li>Ui, 0, (1/2)*(Ui-Li)/(IC[i,2]-IC[i,1])
      + (1/2)*(Ui-Li)/(IC[i,2*j+2]-IC[i,2*j+1]))
      rownames(Intersection)<-rownames(IC)
    }
```

```
  }
  return (Intersection)
}
apply(Inter(IC),2,mean)

print(xtable(coef))
print(xtable(IC))


#Vérification des résultats avec les fonctions reg.


load("JeuSyn11RC.RData")
load("JeuSyn12RC.RData")
reg1(JeuSyn_test11, CPS2000RC)
reg1(JeuSyn_test12, CPS2000RC)

load("JeuSyn21RC.RData")
load("JeuSyn22RC.RData")
reg2(JeuSyn2.1, CPS2000RC)
reg2(JeuSyn2.2, CPS2000RC)

load("JeuSyn31RC.RData")
load("JeuSyn32RC.RData")
reg3(JeuSyn3.1, CPS2000RC)
reg3(JeuSyn3.2, CPS2000RC)
```

Fonctions reg

```
library(xtable)


reg1=function(JeuSyn, CPS2000RC)
{
  CPS2000<-CPS2000RC
  CPS2000_log<-CPS2000[which(CPS2000$HTOTVAL > 0),]
  CPS2000_logPS<-CPS2000[which(CPS2000$HTOTVAL > 0),]
  CPS2000_log$HOUSESUP1<-CPS2000_log$HNUM>1
```

```
CPS2000_log$age2<-(CPS2000_log$AAGE^2)



coef=round(summary(lm(log(HTOTVAL)~factor(ARACE)+AHGA+as.factor(HOUSESUP1)
+PROPTAX+AAGE+age2+factor(ASEX)*factor(AMARITL),data=CPS2000_log))$
coefficient[,1],5)
borne=round(confint(lm(log(HTOTVAL)~factor(ARACE)+AHGA+as.factor(HOUSESUP1)
+PROPTAX+AAGE+age2+factor(ASEX)*factor(AMARITL),data=CPS2000_log)),5)
IC=round(cbind(borne[,1],borne[,2]),5)
Sort<-NULL
Reg.Coe1<-NA
Reg.Std1<-NA
m=length(JeuSyn)
for(i in 1:m)
{
  Temp=as.data.frame(JeuSyn[[i]])
  Temp=Temp[which(Temp$HTOTVAL > 0),]
  Temp$HOUSESUP1<-Temp$HNUM>1
  Temp$age2<-(Temp$AAGE^2)
  if(i!=1)
  {
    Reg.Coe1<-cbind(Reg.Coe1,summary(lm(log(HTOTVAL)~factor(ARACE)+AHGA
    +as.factor(HOUSESUP1)+PROPTAX+AAGE+age2+factor(ASEX)*factor(AMARITL)
    ,data=Temp))$coefficients[,1])
    Reg.Std1<-cbind(Reg.Std1,summary(lm(log(HTOTVAL)~factor(ARACE)+AHGA+
    as.factor(HOUSESUP1)+PROPTAX+AAGE+age2+factor(ASEX)*factor(AMARITL),
    data=Temp))$coefficients[,2]^2)
  }
  else
  {
    Reg.Coe1<-summary(lm(log(HTOTVAL)~factor(ARACE)+AHGA+as.factor(
    HOUSESUP1)+PROPTAX+AAGE+age2+factor(ASEX)*factor(AMARITL),data=
    Temp))$coefficients[,1]
    Reg.Std1<-summary(lm(log(HTOTVAL)~factor(ARACE)+AHGA+as.factor(
    HOUSESUP1)+PROPTAX+AAGE+age2+factor(ASEX)*factor(AMARITL),data=
    Temp))$coefficients[,2]^2
  }
}
```

```
  #Coefficient moyen :
  qmoy<-apply(Reg.Coe1,MARGIN=1,FUN=mean)
  b=(apply(Reg.Coe1,MARGIN=1,FUN=function(x){sum(x^2)})-m*qmoy^2)/(m-1)
  u=apply(Reg.Std1,MARGIN=1,FUN=mean)
  Tm=u+(b/m)
  #DL:
  Vm=(m-1)*(1+m*(u/b))^2
  Sort$coef=cbind(coef,round(qmoy,5))
  Sort$IC=round(cbind(IC,qmoy- qt(0.975,Vm)*sqrt(Tm),qmoy+
  qt(0.975,Vm)*sqrt(Tm)),5)
  Sort$IO=Inter(Sort$IC)
  # Erreur relative
  Sort$ER=abs((Sort$coef[,1]-Sort$coef[,2])/Sort$coef[,1])
  Sort$MER=mean(Sort$ER)
  return(Sort)
}


reg2=function(JeuSyn, CPS2000RC)
{
  CPS2000_HSSVAL<-CPS2000RC[which(CPS2000RC$SSVAL > 0),]
  CPS2000_HSSVAL<-CPS2000_HSSVAL[which(CPS2000_HSSVAL$AAGE > 54),]
  CPS2000_HSSVAL$wido<-CPS2000_HSSVAL$AMARITL %in% 4
  CPS2000_HSSVAL$Divo<-CPS2000_HSSVAL$AMARITL %in% 5
  CPS2000_HSSVAL$Single<-CPS2000_HSSVAL$AMARITL %in% c(6,7)

  CPS2000_HSSVAL$school<-ifelse(CPS2000_HSSVAL$AHGA %in% c(32:39),1,
                          ifelse(CPS2000_HSSVAL$AHGA %in% 40,2,
                          ifelse(CPS2000_HSSVAL$AHGA %in% c(41:42),3,
                          ifelse(CPS2000_HSSVAL$AHGA %in% c(43:46),4,0))))

  coef=summary(lm(sqrt(SSVAL)~as.factor(ASEX)+as.factor(ARACE)+as.factor(
  wido)+as.factor(Divo)+as.factor(Single)+as.factor(school)+AAGE,data=
  CPS2000_HSSVAL))$coefficients[,1]
  IC=round(confint(lm(sqrt(SSVAL)~as.factor(ASEX)+as.factor(ARACE)+
  as.factor(wido)+as.factor(Divo)+as.factor(Single)+as.factor(school)
  +AAGE,data=CPS2000_HSSVAL)),3)
```

```
 Sort<-NULL
 Reg.Coe2<-NA
 Reg.Std2<-NA
 for(i in 1:length(JeuSyn))
 {
   Temp=as.data.frame(JeuSyn[[i]])
   Temp=Temp[which(Temp$SSVAL>0),]
   Temp=Temp[which(Temp$AAGE>54),]
   Temp$wido<-Temp$AMARITL %in% 4
   Temp$Divo<-Temp$AMARITL %in% 5
   Temp$Single<-Temp$AMARITL %in% c(6,7)

   Temp$school<-ifelse(Temp$AHGA %in% c(32:39),1,
    ifelse(Temp$AHGA %in% 40,2,
    ifelse(Temp$AHGA %in% c(41:42),3,
ifelse(Temp$AHGA %in% c(43:46),4,0))))

   if(i!=1)
   {
     Reg.Coe2<-cbind(Reg.Coe2,summary(lm(sqrt(SSVAL)~as.factor(ASEX)
     +as.factor(ARACE)+as.factor(wido)+as.factor(Divo)+as.factor(Single)
     +as.factor(school)+AAGE,data=Temp))$coefficients[,1])
     Reg.Std2<-cbind(Reg.Std2,summary(lm(sqrt(SSVAL)~as.factor(ASEX)
     +as.factor(ARACE)+as.factor(wido)+as.factor(Divo)+as.factor(Single)
     +as.factor(school)+AAGE,data=Temp))$coefficients[,2]^2)
   }
   else
   {
     Reg.Coe2<-summary(lm(sqrt(SSVAL)~as.factor(ASEX)+as.factor(ARACE)
     +as.factor(wido)+as.factor(Divo)+as.factor(Single)+as.factor(school)
     +AAGE,data=Temp))$coefficients[,1]
     Reg.Std2<-summary(lm(sqrt(SSVAL)~as.factor(ASEX)+as.factor(ARACE)
     +as.factor(wido)+as.factor(Divo)+as.factor(Single)+as.factor(school)
     +AAGE,data=Temp))$coefficients[,2]^2
   }
 }

 #Coefficient moyen :
 qmoy<-apply(Reg.Coe2,MARGIN=1,FUN=mean)
```

```
  m=length(JeuSyn)
  b=(apply(Reg.Coe2,MARGIN=1,FUN=function(x){sum(x^2)})-m*qmoy^2)/(m-1)
  u=apply(Reg.Std2,MARGIN=1,FUN=mean)
  Tm=u+(b/m)
  #DL:
  Vm=(m-1)*(1+m*(u/b))^2
  Sort$coef=cbind(coef,round(qmoy,5))
  Sort$IC=round(cbind(IC,qmoy- qt(0.975,Vm)*sqrt(Tm),qmoy+
  qt(0.975,Vm)*sqrt(Tm)),5)
  Sort$IO=Inter(Sort$IC)
  Sort$ER=abs((Sort$coef[,1]-Sort$coef[,2])/Sort$coef[,1])
  Sort$MER=mean(Sort$ER)
  return(Sort)
}


reg3=function(JeuSyn3, CPS2000RC)
{
  CPS2000_3<-CPS2000RC[which(CPS2000RC$HCSPVAL > 0),]
  CPS2000_3$nonwhite=ifelse(CPS2000_3$ARACE %in% c(2:4),1,0)
  coef=summary(lm(sqrt(HCSPVAL)~as.factor(ASEX)+as.factor(nonwhite)+AHGA+
  HNUML18,data=CPS2000_3))$coefficients[,1]
  IC=round(confint(lm(sqrt(HCSPVAL)~as.factor(ASEX)+as.factor(nonwhite)+
  AHGA+HNUML18,data=CPS2000_3)),3)

  Reg.Coe3<-NA
  Reg.Std3<-NA
  for(i in 1:length(JeuSyn3))
  {
    Temp=as.data.frame(JeuSyn3[[i]])
    Temp$nonwhite=ifelse(Temp$ARACE %in% c(2:4),1,0)

    if(i!=1)
    {
      Reg.Coe3<-cbind(Reg.Coe3,summary(lm(sqrt(HCSPVAL)~as.factor(ASEX)+
      as.factor(nonwhite)+AHGA+HNUML18,data=Temp))$coefficients[,1])
      Reg.Std3<-cbind(Reg.Std3,summary(lm(sqrt(HCSPVAL)~as.factor(ASEX)+
      as.factor(nonwhite)+AHGA+HNUML18,data=Temp))$coefficients[,2]^2)
    }
```

```
    else
    {
      Reg.Coe3<-summary(lm(sqrt(HCSPVAL)~as.factor(ASEX)+as.factor(nonwhite
      )+AHGA+HNUML18,data=Temp))$coefficients[,1]
      Reg.Std3<-summary(lm(sqrt(HCSPVAL)~as.factor(ASEX)+as.factor(nonwhite
      )+AHGA+HNUML18,data=Temp))$coefficients[,2]^2
    }
  }
  Sort=NULL
  qmoy<-apply(Reg.Coe3,MARGIN=1,FUN=mean)
  m=length(JeuSyn3)
  b=(apply(Reg.Coe3,MARGIN=1,FUN=function(x){sum(x^2)})-m*qmoy^2)/(m-1)
  u=apply(Reg.Std3,MARGIN=1,FUN=mean)
  Tm=u+(b/m)
  #DL:
  Vm=(m-1)*(1+m*(u/b))^2
  Sort$coef=cbind(coef,round(qmoy,5))
  Sort$IC=round(cbind(IC,qmoy- qt(0.975,Vm)*sqrt(Tm),qmoy+
  qt(0.975,Vm)*sqrt(Tm)),5)
  Sort$IO=Inter(Sort$IC)
  Sort$ER=abs((Sort$coef[,1]-Sort$coef[,2])/Sort$coef[,1])
  Sort$MER=mean(Sort$ER)
  return(Sort)
}
```

Mesure du risque, code R et C++

Code R

```
RiskMeasure<-function(MatOrig,ListMat)
{
  n=nrow(MatOrig)
  m=length(ListMat)
  nt=matrix(0,n,m)
  Indj=matrix(0,n,m)
  Index<-1:n
  MatOrig<-cbind(MatOrig,Index)
  MatOrig_ord<-MatOrig[do.call(order, lapply(1:NCOL
  (MatOrig), function(i) MatOrig[, i])), ]
```

```
  who=NULL
  for(l in 1:m)
  {
    JeuSyn1<-as.matrix(ListMat[[l]])
    Index<-1:n
    JeuSyn1<-cbind(JeuSyn1,Index)
    JeuSyn1<-JeuSyn1[do.call(order, lapply(1:NCOL(Jeu
    Syn1), function(i) JeuSyn1[, i])), ]
    for(j in 1:n)
    {
      if(j==1){
        who=which(apply(JeuSyn1,1,FUN=function(x) all
        (x[1:4]==MatOrig_ord[j,1:4])))
nt[j,l]=length(who)
Indj[j,l]<-ifelse(MatOrig_ord[j,dim(MatOrig_ord)[2
]] %in% JeuSyn1[who,dim(JeuSyn1)[2]],1,0)
        if(nt[j,l]==0) {
          who<-which(apply(JeuSyn1,1,FUN=function(x) x
          [4]==MatOrig_ord[j,4]))
          nt[j,l]<-length(who)
          Indj[j,l]<-ifelse(MatOrig_ord[j,13] %in% Jeu
          Syn1[who,13],1,0)
        }
      }else
      {
        if(all(MatOrig_ord[j,1:4]==MatOrig_ord[j-1,1:4]))
        {
          nt[j,l]<-nt[j-1,l]
  Indj[j,l]<-ifelse(MatOrig_ord[j,dim(MatOrig_ord)[2]
  ] %in% JeuSyn1[who,dim(JeuSyn1)[2]],1,0)
        }else
        {
          who=which(apply(JeuSyn1,1,FUN=function(x) all(x
          [1:4]==MatOrig_ord[j,1:4])))
  nt[j,l]<-length(who)
          Indj[j,l]<-ifelse(MatOrig_ord[j,dim(MatOrig_ord
          )[2]] %in% JeuSyn1[who,dim(JeuSyn1)[2]],1,0)
          if(length(who)==0)
          {
```

```
            who<-which(apply(JeuSyn1,1,FUN=function(x) x
            [4]==MatOrig_ord[j,4]))
            nt[j,l]<-length(who)
            Indj[j,l]<-ifelse(MatOrig_ord[j,13] %in% Jeu
            Syn1[who,13],1,0)
          }
        }
      }
    }
}
UniqueRow=NA
for(i in 1:n)
{
  if(i==1)
  {
    UniqueRow[i]=ifelse(all(MatOrig_ord[i,1:4]==MatOri
    g_ord[i+1,1:4]),0,1)
  }
  else if(i==n)
  {
    UniqueRow[i]=ifelse(all(MatOrig_ord[i,1:4]==MatOrig
    _ord[i-1,1:4]),0,1)
  }
  else
  {
    UniqueRow[i]=ifelse(all(MatOrig_ord[i,1:4]==MatOrig
    _ord[i-1,1:4]) | all(MatOrig_ord[i,1:4]==MatOrig_or
    d[i+1,1:4]),0,1)
  }
}
c=NA
d=NA
k=NA
TM=NA
f=NA
c=nt
d=Indj
k=ifelse(c*d==1,1,0)
TM=mean(apply(k,2,sum)/n)
```

```
  f=ifelse(c*(1-d)==1,1,0)
  g=apply(c,2,FUN=function(x) sum(x==1))
  FM=mean(apply(f,2,sum)/g)
  k=as.matrix(k[which(UniqueRow==1),])
  TM2=mean(apply(k,2,sum)/sum(UniqueRow))
  f=as.matrix(f[which(UniqueRow==1),])
  c=as.matrix(c[which(UniqueRow==1),])
  g=apply(c,2,FUN=function(x) sum(x==1))
  FM2=mean(apply(f,2,sum)/g)
  return(c(TM,FM,TM2,FM2))
}


RiskMeasure(CPS2000,JeuSynU)
```

Code C++

```
/*
 * Test.cpp
 *
 *  Created on: May 20, 2014
 *      Author: M. Caron
 */
#include <R.h>
#include <vector>
#include <Rinternals.h>
#include <Rdefines.h>
using namespace std;
extern "C" void Risk(int *array1, int *array2, int *array3,
int *array4, int *array5, int *array6, int *n, int *p ,
double *match , double *match2);
extern "C" void Risk2(int *array1, int *array2, int *m ,
int *n, int *p ,  double *match );
extern "C" int compare(const vector<int>& left, const
vector<int>& right);
extern "C" int compare2(const vector<int>& left, const
vector<int>& right, int which);
void probabi( const int n1,const vector<vector<int> >&
JeuSyn,const vector<int > T, const int col, vector<
```

```
vector<double> >& proba);
extern "C" vector<int> MaxValueRow(const vector<vector
<double> >& mat, int col);



/* Fonction pour évaluer le risque avec m=5
 * Array1: Jeu Original, les autres "array" représentent
 un jeu synthétique chaque, n représente le nombre
 d'observation et p le nombre de variables
 * match retournera deux valeurs, le pourcentage de véritable
  identification et le pourcentage de fausse identification*/

// Un argument de plus que dans les autres chapitres (chapitre
 4 vs chap 2 !!) soit match2 pour savoir quelles sont les
 observations ré-identifiées
extern "C" void Risk(int *array1, int *array2, int *array3,
int *array4, int *array5, int *array6, int *n, int *p ,
double *match , double *match2)
{
int init_value = 0;//valeur initiale dans les matrices
vector< vector<int> > JeuOrig;
vector< vector<int> > JeuSyn1;
vector< vector<int> > JeuSyn2;
vector< vector<int> > JeuSyn3;
vector< vector<int> > JeuSyn4;
vector< vector<int> > JeuSyn5;
JeuOrig.resize( *n , vector<int>( *p , init_value ) );
// On initialise la matrice avec n lignes et p colonnes
 et la valeur initiale
JeuSyn1.resize( *n , vector<int>( *p, init_value ) );
JeuSyn2.resize( *n , vector<int>( *p , init_value ) );
JeuSyn3.resize( *n , vector<int>( *p, init_value ) );
JeuSyn4.resize( *n , vector<int>( *p , init_value ) );
JeuSyn5.resize( *n , vector<int>( *p , init_value ) );
int n1=*n;//On évite de trainer le pointeur
int p1=*p;
for(int i=0; i<n1;i++)
{
```

```
for(int j=0; j<p1; j++)
{
JeuOrig[i][j]=array1[(i * p1)+j];
// On remplit le tableau original
JeuSyn1[i][j]=array2[(i * p1)+j];
// On remplit le premier jeu synthétique
JeuSyn2[i][j]=array3[(i * p1)+j];
JeuSyn3[i][j]=array4[(i * p1)+j];
JeuSyn4[i][j]=array5[(i * p1)+j];
JeuSyn5[i][j]=array6[(i * p1)+j];
}
}

int d[n1];// Définir comme un vecteur
int c1[n1];// Définir comme un vecteur
int num_of_col = 6;
//Nombre de colonne (5+1 pour la moyenne)
int num_of_row = n1;// Nombre de ligne
double init_value2 = 0;
vector< vector<double> > proba;
proba.resize( num_of_row , vector<double>(  num_of_col,
 init_value2 ) );// On initialise notre vecteur de probabilité

vector<int>& T =JeuOrig[0];
int pres;

for(int i=0; i<n1; i++)
{
vector< vector<double> > proba;
proba.resize( num_of_row , vector<double>
( num_of_col , init_value2 ) );
T=JeuOrig[i];
probabi(n1, JeuSyn1, T, 1, proba);// On calcule
 la probabilité pour une observation
probabi(n1, JeuSyn2, T, 2, proba);
probabi(n1, JeuSyn3, T, 3, proba);
probabi(n1, JeuSyn4, T, 4, proba);
probabi(n1, JeuSyn5, T, 5, proba);
for(int k=0; k<n1; k++)
```

```
proba[k][5]=(proba[k][0]+proba[k][1]+proba[k][2]
+proba[k][3]+proba[k][4])/5;// On calcule la moyenne
vector<int> ligne=MaxValueRow(proba, 5);
// On va chercher ceux qui ont la plus grande probabilité
c1[i]=ligne.size();//On regarde il y en a combien
pres=0;
for(int j=0; j<c1[i]; j++)// On vérifie si l'observation
 de départ est parmi celles qui ont la plus grande probabilité
{
if((i+1)==ligne[j])
pres=1;
}
if(pres){
d[i]=1;
}
else
{
d[i]=0;
}
}

int k=0;
int g=0;
int f=0;
for(int i=0; i<n1; i++)
{
if(c1[i]==1 && d[i]==1)
{
match2[k]=i;
k++;// Vraie identification
}
if(c1[i]==1 && d[i]==0)
f++;// Fausse identification
if(c1[i]==1)
g++;// Nombre d'identification possible
}
match[0]=(double)k/(double)n1;// TM
match[1]=(double)f/(double)g;//FM
}
```

```
// En amélioration: On fournit un long vecteur pour
les jeux synthétiques et on le sépare. (VRAIMENT pas
optimale comme façon de faire présentement, un argument
 de moins aussi)
extern "C" void Risk2(int *array1, int *array2, int *m ,
 int *n, int *p ,  double *match )
{

int init_value = 0;
vector< vector<int> > JeuOrig;
JeuOrig.resize( *n , vector<int>( *p , init_value ) );
int n1=*n;
int p1=*p;
int m1=*m;
int d[n1];// Définir comme un vecteur
int c1[n1];// Définir comme un vecteur
for(int i=0; i<n1;i++)
{
for(int j=0; j<p1; j++)
{
JeuOrig[i][j]=array1[(i * p1)+j];
}
}

// Ne sera pas optimale
if(m1==1)
{
vector< vector<int> > JeuSyn1;
JeuSyn1.resize( *n , vector<int>( *p, init_value ) );
for(int i=0; i<n1;i++)
{
for(int j=0; j<p1; j++)
{
JeuSyn1[i][j]=array2[(i * p1)+j];
}
}

int num_of_col = m1+1;
```

```
int num_of_row = n1;
double init_value2 = 0;
vector< vector<double> > proba;
proba.resize( num_of_row , vector<double>(  num_of_col,
 init_value2 ) );

vector<int>& T=JeuOrig[0];
int pres=0;

for(int i=0; i<n1; i++)
{
vector< vector<double> > proba;
proba.resize( num_of_row , vector<double>( num_of_col ,
init_value2 ) );
T=JeuOrig[i];
probabi(n1, JeuSyn1, T, 1, proba);
vector<int> ligne=MaxValueRow(proba, m1);
c1[i]=ligne.size();
pres=0;
for(int j=0; j<c1[i]; j++)
{
if((i+1)==ligne[j])
pres=1;
}
if(pres){
d[i]=1;
}
else
{
d[i]=0;
}
}
}
if(m1==2)
{
vector< vector<int> > JeuSyn1;
JeuSyn1.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn2;
JeuSyn2.resize( *n , vector<int>( *p, init_value ) );
```

```
for(int i=0; i<n1;i++)
{
for(int j=0; j<p1; j++)
{
JeuSyn1[i][j]=array2[(i * p1)+j];
JeuSyn2[i][j]=array2[((n1 * p1))+(i * p1)+j];
}
}

int num_of_col = m1+1;
int num_of_row = n1;
double init_value2 = 0;
vector< vector<double> > proba;
proba.resize( num_of_row , vector<double>(  num_of_col,
 init_value2 ) );

vector<int>& T=JeuOrig[0];
int pres=0;

for(int i=0; i<n1; i++)
{
vector< vector<double> > proba;
proba.resize( num_of_row , vector<double>(
num_of_col , init_value2 ) );
T=JeuOrig[i];
probabi(n1, JeuSyn1, T, 1, proba);
probabi(n1, JeuSyn2, T, 2, proba);
for(int k=0; k<n1; k++)
proba[k][m1]=(proba[k][0]+proba[k][1])/2;
vector<int> ligne=MaxValueRow(proba, m1);
c1[i]=ligne.size();
pres=0;
for(int j=0; j<c1[i]; j++)
{
if((i+1)==ligne[j])
pres=1;
}
if(pres){
```

```
d[i]=1;
}
else
{
d[i]=0;
}
}
}
if(m1==3)
{
vector< vector<int> > JeuSyn1;
JeuSyn1.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn2;
JeuSyn2.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn3;
JeuSyn3.resize( *n , vector<int>( *p, init_value ) );
for(int i=0; i<n1;i++)
{
for(int j=0; j<p1; j++)
{
JeuSyn1[i][j]=array2[(i * p1)+j];
JeuSyn2[i][j]=array2[(1*(n1 * p1))+(i * p1)+j];
JeuSyn3[i][j]=array2[(2*(n1 * p1))+(i * p1)+j];

}
}

int num_of_col = m1+1;
int num_of_row = n1;
double init_value2 = 0;
vector< vector<double> > proba;
proba.resize( num_of_row , vector<double>(
  num_of_col, init_value2 ) );

vector<int>& T=JeuOrig[0];
int pres=0;

for(int i=0; i<n1; i++)
{
```

```
vector< vector<double> > proba;
proba.resize( num_of_row , vector<double>(
 num_of_col , init_value2 ) );
T=JeuOrig[i];
probabi(n1, JeuSyn1, T, 1, proba);
probabi(n1, JeuSyn2, T, 2, proba);
probabi(n1, JeuSyn3, T, 3, proba);
for(int k=0; k<n1; k++)
proba[k][m1]=(proba[k][0]+proba[k][1]+
proba[k][2])/3;
vector<int> ligne=MaxValueRow(proba, m1);
c1[i]=ligne.size();
pres=0;
for(int j=0; j<c1[i]; j++)
{
if((i+1)==ligne[j])
pres=1;
}
if(pres){
d[i]=1;
}
else
{
d[i]=0;
}
}
}
if(m1==4)
{
vector< vector<int> > JeuSyn1;
JeuSyn1.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn2;
JeuSyn2.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn3;
JeuSyn3.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn4;
JeuSyn4.resize( *n , vector<int>( *p, init_value ) );

for(int i=0; i<n1;i++)
```

```
{
for(int j=0; j<p1; j++)
{
JeuSyn1[i][j]=array2[(i * p1)+j];
JeuSyn2[i][j]=array2[(1*(n1 * p1))+(i * p1)+j];
JeuSyn3[i][j]=array2[(2*(n1 * p1))+(i * p1)+j];
JeuSyn4[i][j]=array2[(3*(n1 * p1))+(i * p1)+j];
}
}

int num_of_col = m1+1;
int num_of_row = n1;
double init_value2 = 0;
vector< vector<double> > proba;
proba.resize( num_of_row , vector<double>(  num_of_col, init_value2 ) );


for(int i=0; i<n1; i++)
{
vector< vector<double> > proba;
proba.resize( num_of_row , vector<double>( num_of_col , init_value2 ) );
vector<int>& T=JeuOrig[i];
probabi(n1, JeuSyn1, T, 1, proba);
probabi(n1, JeuSyn2, T, 2, proba);
probabi(n1, JeuSyn3, T, 3, proba);
probabi(n1, JeuSyn4, T, 4, proba);
for(int k=0; k<n1; k++)
proba[k][m1]=(proba[k][0]+proba[k][1]+proba[k][2]+proba[k][3])/4;
vector<int> ligne=MaxValueRow(proba, m1);
c1[i]=ligne.size();
int pres=0;
for(int j=0; j<c1[i]; j++)
{
if((i+1)==ligne[j])
pres=1;
}
if(pres){
d[i]=1;
}
```

```
else
{
d[i]=0;
}
}
}
if(m1==5)
{
vector< vector<int> > JeuSyn1;
JeuSyn1.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn2;
JeuSyn2.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn3;
JeuSyn3.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn4;
JeuSyn4.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn5;
JeuSyn5.resize( *n , vector<int>( *p, init_value ) );
for(int i=0; i<n1;i++)
{
for(int j=0; j<p1; j++)
{
JeuSyn1[i][j]=array2[(i * p1)+j];
JeuSyn2[i][j]=array2[(1 * (n1 * p1))+(i * p1)+j];
JeuSyn3[i][j]=array2[(2 * (n1 * p1))+(i * p1)+j];
JeuSyn4[i][j]=array2[(3 * (n1 * p1))+(i * p1)+j];
JeuSyn5[i][j]=array2[(4 * (n1 * p1))+(i * p1)+j];
}
}



for(int i=0; i<n1; i++)
{
int num_of_col = 6;
int num_of_row = n1;
double init_value2 = 0;
vector< vector<double> > proba;
proba.resize( num_of_row , vector<double>( num_of_col , init_value2 ) );
```

```
vector<int>& T=JeuOrig[i];
probabi(n1, JeuSyn1, T, 1, proba);
probabi(n1, JeuSyn2, T, 2, proba);
probabi(n1, JeuSyn3, T, 3, proba);
probabi(n1, JeuSyn4, T, 4, proba);
probabi(n1, JeuSyn5, T, 5, proba);
for(int k=0; k<n1; k++)
proba[k][5]=(proba[k][0]+proba[k][1]+proba[k][2]+proba[k][3]+proba[k][4])/5;
vector<int> ligne=MaxValueRow(proba, 5);
c1[i]=ligne.size();
int pres=0;
for(int j=0; j<c1[i]; j++)
{
if((i+1)==ligne[j])
pres=1;
}
if(pres){
d[i]=1;
}
else
{
d[i]=0;
}
}
}
if(m1==6)
{
vector< vector<int> > JeuSyn1;
JeuSyn1.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn2;
JeuSyn2.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn3;
JeuSyn3.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn4;
JeuSyn4.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn5;
JeuSyn5.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn6;
JeuSyn6.resize( *n , vector<int>( *p, init_value ) );
```

```
for(int i=0; i<n1;i++)
{
for(int j=0; j<p1; j++)
{
JeuSyn1[i][j]=array2[(i * p1)+j];
JeuSyn2[i][j]=array2[(1 * (n1 * p1))+(i * p1)+j];
JeuSyn3[i][j]=array2[(2 * (n1 * p1))+(i * p1)+j];
JeuSyn4[i][j]=array2[(3 * (n1 * p1))+(i * p1)+j];
JeuSyn5[i][j]=array2[(4 * (n1 * p1))+(i * p1)+j];
JeuSyn6[i][j]=array2[(5 * (n1 * p1))+(i * p1)+j];

}
}



for(int i=0; i<n1; i++)
{
int num_of_col = m1+1;
int num_of_row = n1;
double init_value2 = 0;
vector< vector<double> > proba;
proba.resize( num_of_row , vector<double>( num_of_col , init_value2 ) );
vector<int>& T=JeuOrig[i];
probabi(n1, JeuSyn1, T, 1, proba);
probabi(n1, JeuSyn2, T, 2, proba);
probabi(n1, JeuSyn3, T, 3, proba);
probabi(n1, JeuSyn4, T, 4, proba);
probabi(n1, JeuSyn5, T, 5, proba);
probabi(n1, JeuSyn6, T, 6, proba);
for(int k=0; k<n1; k++)
proba[k][m1]=(proba[k][0]+proba[k][1]+proba[k][2]+proba[k][3]+proba[k][4]
+proba[k][5])/m1;
vector<int> ligne=MaxValueRow(proba, m1);
c1[i]=ligne.size();
int pres=0;
for(int j=0; j<c1[i]; j++)
{
if((i+1)==ligne[j])
```

```
pres=1;
}
if(pres){
d[i]=1;
}
else
{
d[i]=0;
}
}
}
if(m1==8)
{
vector< vector<int> > JeuSyn1;
JeuSyn1.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn2;
JeuSyn2.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn3;
JeuSyn3.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn4;
JeuSyn4.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn5;
JeuSyn5.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn6;
JeuSyn6.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn7;
JeuSyn7.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn8;
JeuSyn8.resize( *n , vector<int>( *p, init_value ) );
for(int i=0; i<n1;i++)
{
for(int j=0; j<p1; j++)
{
JeuSyn1[i][j]=array2[(i * p1)+j];
JeuSyn2[i][j]=array2[(1 * (n1 * p1))+(i * p1)+j];
JeuSyn3[i][j]=array2[(2 * (n1 * p1))+(i * p1)+j];
JeuSyn4[i][j]=array2[(3 * (n1 * p1))+(i * p1)+j];
JeuSyn5[i][j]=array2[(4 * (n1 * p1))+(i * p1)+j];
JeuSyn6[i][j]=array2[(5 * (n1 * p1))+(i * p1)+j];
```

```
JeuSyn7[i][j]=array2[(6 * (n1 * p1))+(i * p1)+j];
JeuSyn8[i][j]=array2[(7 * (n1 * p1))+(i * p1)+j];
}
}



for(int i=0; i<n1; i++)
{
int num_of_col = m1+1;
int num_of_row = n1;
double init_value2 = 0;
vector< vector<double> > proba;
proba.resize( num_of_row , vector<double>( num_of_col , init_value2 ) );
vector<int>& T=JeuOrig[i];
probabi(n1, JeuSyn1, T, 1, proba);
probabi(n1, JeuSyn2, T, 2, proba);
probabi(n1, JeuSyn3, T, 3, proba);
probabi(n1, JeuSyn4, T, 4, proba);
probabi(n1, JeuSyn5, T, 5, proba);
probabi(n1, JeuSyn6, T, 6, proba);
probabi(n1, JeuSyn7, T, 7, proba);
probabi(n1, JeuSyn8, T, 8, proba);
for(int k=0; k<n1; k++)
proba[k][m1]=(proba[k][0]+proba[k][1]+proba[k][2]+proba[k][3]+proba[k][4]
+proba[k][5]+proba[k][6]+proba[k][7])/m1;
vector<int> ligne=MaxValueRow(proba, m1);
c1[i]=ligne.size();
int pres=0;
for(int j=0; j<c1[i]; j++)
{
if((i+1)==ligne[j])
pres=1;
}
if(pres){
d[i]=1;
}
else
{
```

```
d[i]=0;
}
}
}
if(m1==10)
{
vector< vector<int> > JeuSyn1;
JeuSyn1.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn2;
JeuSyn2.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn3;
JeuSyn3.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn4;
JeuSyn4.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn5;
JeuSyn5.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn6;
JeuSyn6.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn7;
JeuSyn7.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn8;
JeuSyn8.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn9;
JeuSyn9.resize( *n , vector<int>( *p, init_value ) );
vector< vector<int> > JeuSyn10;
JeuSyn10.resize( *n , vector<int>( *p, init_value ) );
for(int i=0; i<n1;i++)
{
for(int j=0; j<p1; j++)
{
JeuSyn1[i][j]=array2[(i * p1)+j];
JeuSyn2[i][j]=array2[(1 * (n1 * p1))+(i * p1)+j];
JeuSyn3[i][j]=array2[(2 * (n1 * p1))+(i * p1)+j];
JeuSyn4[i][j]=array2[(3 * (n1 * p1))+(i * p1)+j];
JeuSyn5[i][j]=array2[(4 * (n1 * p1))+(i * p1)+j];
JeuSyn6[i][j]=array2[(5 * (n1 * p1))+(i * p1)+j];
JeuSyn7[i][j]=array2[(6 * (n1 * p1))+(i * p1)+j];
JeuSyn8[i][j]=array2[(7 * (n1 * p1))+(i * p1)+j];
JeuSyn9[i][j]=array2[(8 * (n1 * p1))+(i * p1)+j];
```

```
JeuSyn10[i][j]=array2[(9 * (n1 * p1))+(i * p1)+j];
}
}



for(int i=0; i<n1; i++)
{
int num_of_col = m1+1;
int num_of_row = n1;
double init_value2 = 0;
vector< vector<double> > proba;
proba.resize( num_of_row , vector<double>( num_of_col , init_value2 ) );
vector<int>& T=JeuOrig[i];
probabi(n1, JeuSyn1, T, 1, proba);
probabi(n1, JeuSyn2, T, 2, proba);
probabi(n1, JeuSyn3, T, 3, proba);
probabi(n1, JeuSyn4, T, 4, proba);
probabi(n1, JeuSyn5, T, 5, proba);
probabi(n1, JeuSyn6, T, 6, proba);
probabi(n1, JeuSyn7, T, 7, proba);
probabi(n1, JeuSyn8, T, 8, proba);
probabi(n1, JeuSyn9, T, 9, proba);
probabi(n1, JeuSyn10, T, 10, proba);
for(int k=0; k<n1; k++)
proba[k][m1]=(proba[k][0]+proba[k][1]+proba[k][2]+proba[k][3]+proba[k][4]+
proba[k][5]+proba[k][6]+proba[k][7]+proba[k][8]+proba[k][9])/m1;
vector<int> ligne=MaxValueRow(proba, m1);
c1[i]=ligne.size();
int pres=0;
for(int j=0; j<c1[i]; j++)
{
if((i+1)==ligne[j])
pres=1;
}
if(pres){
d[i]=1;
}
else
```

```
{
d[i]=0;
}
}
}
int k=0;
int g=0;
int f=0;
for(int i=0; i<n1; i++)
{
if(c1[i]==1 && d[i]==1)
k++;
if(c1[i]==1 && d[i]==0)
f++;
if(c1[i]==1)
g++;
}
match[0]=(double)k/(double)n1;
match[1]=(double)f/(double)g;
}

/*Source: http://stackoverflow.com/questions/5225820/compare-two-vectors-c*/

// Fonction qui permet de conclure si les vecteurs fournies sont identiques
 (diff=0) ou différents
extern "C" int compare(const vector<int>& left, const vector<int>&
 right) {
vector<int>::const_iterator leftIt = left.begin();
vector<int>::const_iterator rightIt = right.begin();
int diff = 0;
while (leftIt != left.end() && rightIt != right.end()) {
if (*leftIt != *rightIt) {
diff++;
}
leftIt++;
rightIt++;
}

return diff;
```

```
}

// Compare en particulier la 4e composante des vecteurs (l'âge) (SUJET
 à modification si on veut!)
extern "C" int compare2(const vector<int>& left, const vector<int>&
 right, int which) {
vector<int>::const_iterator leftIt = left.begin() + which - 1;
vector<int>::const_iterator rightIt = right.begin()+ which - 1;
int diff = 0;
if (*leftIt != *rightIt) {
diff++;
}
// Account for different length vector instances

return diff;
}

//Fonction qui retourne en argument les probabilités pour chaque
 observation d'être l'observation de départ.

void probabi(const int n1,const vector<vector<int> >& JeuSyn,const
 vector<int> T, const int col, vector<vector<double> >& proba)
{
vector<int> vec;
int com=0;
for(int j=0; j<n1; j++)
{
vector<int> T2=JeuSyn[j];
com=compare(T, T2);
if(com==0)// Si com=0, il s'agit d'une identification possible
vec.push_back(j);
}
int who=vec.size();
if(who==0) // S'il est vide, il faut alors vérifier en fonction de l'âge
{
for(int j=0; j<n1; j++)
{
vector<int> T2=JeuSyn[j];
com=compare2(T, T2, 4);
```

```
if(com==0)
vec.push_back(j);
}
who=vec.size();
}
for(int k=0; k<who;k++)
proba[vec[k]][col-1]=(double)(1/(double)who);
}


/*Source: http://stackoverflow.com/questions/16425913/find-how-many-
maximum-column-values-a-row-has*/

// Trouve les lignes associées à la valeur maximale d'une colonne
extern "C" vector<int> MaxValueRow(const vector<vector <double> >&
 mat, int col)
{
int m = mat.size();
vector<int> ind;
ind.resize(m,0);
ind[0]=1;
double maxPoints = mat[0][col];
int k=1;
for (int r = 1; r < m; r++) {
if (mat[r][col] > maxPoints) {
maxPoints = mat[r][col];
ind.erase(ind.begin(), ind.begin()+k);//Efface ce qui
 a été inscrit jusqu'à présent
k=0;
ind[k]=r+1;
k++;
}
else if (mat[r][col] == maxPoints) {
ind[k]=r+1;
k++;
}
}
ind.resize(k);
return ind; // on retourne un vecteur qui contient les lignes
```

```
 avec la proba maximale
}
```

Nombre de jeux synthétiques

```
# Création de différent jeux de données avec différents M
# Besoin de CPSGenSyn2RC_ajoutTest.R

load("CPS2000RC2.Rdata")
#Ordre Etat marital, race sexe
CPS2000=CPS2000RC[,c(3,2,1,4:12)]

#M=2

Jeu2=replicate(2,GenSyn2(1,CPS2000,1:3))
save(Jeu2, file="Jeu2.Rdata")
load("Jeu2.Rdata")
reg1(Jeu2,CPS2000RC)
reg2(Jeu2, CPS2000RC)
#Utilité (reg1): 61.28
#Utilité (reg2): 83.04
#Risk: 3.52
U2=83.04
R=3.52

#M=3

Jeu3=replicate(3,GenSyn2(1,CPS2000,1:3))
save(Jeu3, file="Jeu3.Rdata")
load("Jeu3.Rdata")
reg1(Jeu3,CPS2000RC)
reg2(Jeu3,CPS2000RC)
#Utilité (reg1): 61.65
#Utilité (reg2): 79,77
#Risk:1.48
U2=c(U2, 79.77)
```

```
R=c(R,1.48)

#M=4

Jeu4=replicate(4,GenSyn2(1,CPS2000,1:3))
save(Jeu4, file="Jeu4.Rdata")
load("Jeu4.Rdata")
reg1(Jeu4,CPS2000RC)
reg2(Jeu4,CPS2000RC)
#Utilité (reg1): 61.70
#Utilité (reg2): 78,72
#Risk: 1.58
U2=c(U2, 78.72)
R=c(R,1.58)
#M=5

Jeu5=replicate(5,GenSyn2(1,CPS2000,1:3))
save(Jeu5, file="Jeu5.Rdata")
load("Jeu5.Rdata")
reg1(Jeu5,IC, coef)
reg2(Jeu5, IC, coef)
#Utilité (reg1): 61.8%
#Utilité (reg2): 78.0%
#Risk: 6.76
U=61.8
U2=c(U2,78)
R=c(R,6.76)
#M=10

Jeu10=replicate(10,GenSyn2(1,CPS2000,1:3))
save(Jeu10, file="Jeu10.Rdata")
load("Jeu10.Rdata")
reg1(Jeu10,IC, coef)
reg2(Jeu10, IC, coef)
#Utilité (reg1): 61.2%
#Utilité (reg2): 78.8%
#Risk:9.59
U=c(U,61.2)
U2=c(U2,78.8)
```

```
R=c(R,9.59)

#M=20

Jeu20=replicate(20,GenSyn2(1,CPS2000,1:3))
save(Jeu20, file="Jeu20.Rdata")
load("Jeu20.Rdata")
reg1(Jeu20,IC, coef)
reg2(Jeu20, IC, coef)
#Utilité (reg1): 62.0%
#Utilité (reg2): 79.67%
#Risk:10.99% TM
# 83.00% FM
U=c(U,62.0)
U2=c(U2,79.67)
R=c(R,10.99)
#M=50

Jeu50=replicate(50,GenSyn2(1,CPS2000,1:3))
save(Jeu50, file="Jeu50.Rdata")
load("Jeu50.Rdata")
reg1(Jeu50,IC, coef)
reg2(Jeu50, IC, coef)
#Utilité (reg1): 61.76
#Utilité (reg2): 79.86
#Risk: 12.31
U=c(U,61.76)
U2=c(U2,79.86)
R=c(R,12.31)

plot(U2,R, xlab="Utilit?", ylab="Risque")
names(U2)=c("2","3","4","5","10","20","50")
text(U2, R, names(U2), cex=0.6, pos=4, col="red")
```

```
# On va simuler quelques échantillons avec des M entre 2 et 10.
load("CPS2000RC2.Rdata")
CPS2000=CPS2000RC[,c(3,2,1,4:12)]
dyn.load("Test.so")
k=1
IO=matrix(NA,7,40)
Risk=matrix(NA,7,40)
# Les m tester:
afaire=c(2,3,4,5,6,8,10)
set.seed(4235)
for(i in 1:length(afaire))
{
  for(j in 1:40)# On recommence l'exp?rience 40 fois
  {
    JeuSyn1=replicate(afaire[i], GenSyn2(1,CPS2000, 1:3))
    out=reg2(JeuSyn1, CPS2000)
    IO[k,j]=out[[3]]
    Jeu=as.vector(t(JeuSyn1[[1]][,1:4]))
    for(l in 2:afaire[i])
    {
      Jeu=c(Jeu, as.vector(t(JeuSyn1[[l]][,1:4])))
    }
    out2=.C("Risk2", as.integer(as.vector(t(CPS2000[,1:4]))),
    as.integer((Jeu)), as.integer(afaire[i]), as.integer(10000),
    as.integer(4), as.numeric(c(0,0)))
    Risk[k,j]=out2[[6]][1]
  }
  k=k+1
  JeuSyn1=list()
}
  Risk2=apply(Risk, 1, mean)
  IO2=apply(IO, 1, mean)
save(Risk, file="RiskMdiff19aout.Rdata")
save(IO,file="IOMdiff19aout.Rdata")

load("RiskMdiff19aout.Rdata")
load("IOMdiff19aout.Rdata")
Risk2=apply(Risk, 1, mean)
```

```
IO2=apply(IO, 1, mean)
names(Risk2)=c("2","3","4","5","6","8","10")
library(Hmisc)
png(file="SIMDiffM.png")
plot(c(0.02,0.10),c(0.82,0.94),type="n", xlab="Risque de
réidentification", ylab="Utilité des données", main="Utilité
 en fonction du risque", las=1 )
errbar(Risk2, IO2, IO2+apply(IO,1,sd),IO2-apply(IO,1,sd),add=T )
text(Risk2,IO2,labels=names(Risk2),col="red",pos=2,cex=0.7)
dev.off()
```

Impact de l'ordre sur l'utilité et le risque

```
# Test pour différent ordre pour la synthétisation
# Toujours avec le jeu CPS2000RC2.Rdata

load("CPS2000RC2.Rdata")
dyn.load("Test.so")
# On a besoin de GenSyn2_ajouttest
#


Risk=matrix(NA,20,6)
IO=matrix(NA,20,6)
set.seed(543534)
for(i in 1:20)
{
  # Synthétisons d'abord sex-race-marital
  CPS2000=CPS2000RC
  JeuSyn1=replicate(5,GenSyn2(1,CPS2000,1:3))
  # Évaluons le risque et l'utilité
  out=.C("Risk", as.integer(as.vector(t(CPS2000[,1:4]))),
   as.integer(as.vector(t(JeuSyn1[[1]][,1:4]))), as.integer
   (as.vector(t(JeuSyn1[[2]][,1:4]))),as.integer(as.vector(
   t(JeuSyn1[[3]][,1:4]))),as.integer(as.vector(t
```

```
    (JeuSyn1[[4]
    ][,1:4]))),as.integer(as.vector(t(JeuSyn1[[5]][,1:4]))),
    as.integer(10000), as.integer(4), as.numeric(c(0,0)))
  Risk[i,1]=out[[9]][1]
  out=reg2(JeuSyn1, CPS2000)
  IO[i,1]=out[[3]]
  # marital-race-sex

  CPS2000=CPS2000RC[,c(3,2,1,4:12)]
  JeuSyn1=replicate(5, GenSyn2(1,CPS2000,1:3))
  out=.C("Risk", as.integer(as.vector(t(CPS2000[,1:4]))),
  as.integer(as.vector(t(JeuSyn1[[1]][,1:4]))), as.integer
  (as.vector(t(JeuSyn1[[2]][,1:4]))),as.integer(as.vector
  (t(JeuSyn1[[3]][,1:4]))),as.integer(as.vector(t
  (JeuSyn1[[4]][,1:4]))),as.integer(as.vector(t(JeuSyn1[[5]][,1:4]))),
    as.integer(10000), as.integer(4), as.numeric(c(0,0)))
  Risk[i,2]=out[[9]][1]
  out=reg2(JeuSyn1, CPS2000)
  IO[i,2]=out[[3]]

CPS2000=CPS2000RC[,c(3,1,2,4:12)]
  JeuSyn1=replicate(5, GenSyn2(1,CPS2000,1:3))
  out=.C("Risk", as.integer(as.vector(t(CPS2000[,1:4]))),
  as.integer(as.vector(t(JeuSyn1[[1]][,1:4]))), as.integer
  (as.vector(t(JeuSyn1[[2]][,1:4]))),as.integer(as.vector
  (t(JeuSyn1[[3]][,1:4]))),as.integer(as.vector(t
  (JeuSyn1[[4]][,1:4]))),as.integer(as.vector(t(JeuSyn1[[5]][,1:4]))),
  as.integer(10000), as.integer(4), as.numeric(c(0,0)))
  Risk[i,3]=out[[9]][1]
  out=reg2(JeuSyn1, CPS2000)
  IO[i,3]=out[[3]]

CPS2000=CPS2000RC[,c(1,3,2,4:12)]
  JeuSyn1=replicate(5, GenSyn2(1,CPS2000,1:3))
  out=.C("Risk", as.integer(as.vector(t(CPS2000[,1:4]))),
  as.integer(as.vector(t(JeuSyn1[[1]][,1:4]))), as.integer
  (as.vector(t(JeuSyn1[[2]][,1:4]))),as.integer(as.vector
  (t(JeuSyn1[[3]][,1:4]))),as.integer(as.vector(t
  (JeuSyn1[[4]][,1:4]))),as.integer(as.vector(t(JeuSyn1[[5]][,1:4]))),
```

```
  as.integer(10000), as.integer(4), as.numeric(c(0,0)))
  Risk[i,4]=out[[9]][1]
  out=reg2(JeuSyn1, CPS2000)
  IO[i,4]=out[[3]]

CPS2000=CPS2000RC[,c(2,1,3,4:12)]
  JeuSyn1=replicate(5, GenSyn2(1,CPS2000,1:3))
  out=.C("Risk", as.integer(as.vector(t(CPS2000[,1:4]))),
  as.integer(as.vector(t(JeuSyn1[[1]][,1:4]))), as.integer
  (as.vector(t(JeuSyn1[[2]][,1:4]))),as.integer(as.vector
  (t(JeuSyn1[[3]][,1:4]))),as.integer(as.vector(t(
  JeuSyn1[[4]][,1:4]))),as.integer(as.vector(t(JeuSyn1[[5]][,1:4]))),
   as.integer(10000), as.integer(4), as.numeric(c(0,0)))
  Risk[i,5]=out[[9]][1]
  out=reg2(JeuSyn1, CPS2000)
  IO[i,5]=out[[3]]

CPS2000=CPS2000RC[,c(2,3,1,4:12)]
  JeuSyn1=replicate(5, GenSyn2(1,CPS2000,1:3))
  out=.C("Risk", as.integer(as.vector(t(CPS2000[,1:4]))),
   as.integer(as.vector(t(JeuSyn1[[1]][,1:4]))), as.integer
   (as.vector(t(JeuSyn1[[2]][,1:4]))),as.integer(as.vector
   (t(JeuSyn1[[3]][,1:4]))),as.integer
   (as.vector(t(JeuSyn1[[4]][,1:4]))),
   as.integer(as.vector(t(JeuSyn1[[5]][,1:4]))),
    as.integer(10000), as.integer(4), as.numeric(c(0,0)))
  Risk[i,6]=out[[9]][1]
  out=reg2(JeuSyn1, CPS2000)
  IO[i,6]=out[[3]]
}

IO2=apply(IO, 2, FUN=mean)
Mean2=apply(Risk, 2,FUN=mean)



cor(CPS2000RC[,1:3], CPS2000RC)
#           ASEX      ARACE    AMARITL
#ASEX    1.00000000 0.05110617 0.29731949
```

```
#ARACE   0.05110617 1.00000000 0.09350763
#AMARITL 0.29731949 0.09350763 1.00000000

summary(lm(CPS2000RC[,1]~(CPS2000RC[,3])))
summary(lm(CPS2000RC[,1]~(CPS2000RC[,2])))
summary(lm(CPS2000RC[,2]~(CPS2000RC[,1])))
summary(lm(CPS2000RC[,2]~(CPS2000RC[,3])))
summary(lm(CPS2000RC[,3]~(CPS2000RC[,1])))
summary(lm(CPS2000RC[,3]~(CPS2000RC[,2])))



summary(glm((CPS2000RC[,1]-1)~(CPS2000RC[,3]), family=binomial))




load("CPS2000RC2.Rdata")

a=ifelse(CPS2000RC[,1]==2, rbinom(sum(CPS2000RC[,1]==2),
7,0.87),rbinom(sum(CPS2000RC[,1]==1),3,0.75))

a=ifelse(a==0,1,a)

cor(a,CPS2000RC[,1])

# Présent dans le fichier CPSGenSyn2RC_ajoutTest.R


dyn.load("Test.so")

Risk=matrix(NA,20,6)
IO=matrix(NA,20,6)
for(i in 1:20)
{
  JeuSyn1=replicate(5,GenSyn2(1,cbind(CPS2000RC[,1:2],a,
  CPS2000RC[,4:12]),1:3))
  # Évaluons le risque et l'utilité
  out=.C("Risk", as.integer(as.vector(t(CPS2000[,1:4]))),
```

```
   as.integer(as.vector(t(JeuSyn1[[1]][,1:4]))),
    as.integer(as.vector(t(JeuSyn1[[2]][,1:4]))),
    as.integer(as.vector(t(JeuSyn1[[3]][,1:4]))),
   as.integer(as.vector(t(JeuSyn1[[4]][,1:4]))),
   as.integer(as.vector(t(JeuSyn1[[5]][,1:4]))),
   as.integer(10000), as.integer(4), as.numeric(c(0,0)))
  Risk[i,1]=out[[9]][1]
  out=reg2(JeuSyn1, CPS2000)
  IO[i,1]=out[[3]]

  JeuSyn1=replicate(5,GenSyn2(1,cbind(a,CPS2000RC[,2],
  CPS2000RC[,c(1,4:12)]),1:3))
  # Évaluons le risque et l'utilité
  out=.C("Risk", as.integer(as.vector(t(CPS2000[,1:4]))),
   as.integer(as.vector(t(JeuSyn1[[1]][,1:4]))),
   as.integer(as.vector(t(JeuSyn1[[2]][,1:4]))),
   as.integer(as.vector(t(JeuSyn1[[3]][,1:4]))),
   as.integer(as.vector(t(JeuSyn1[[4]][,1:4]))),
   as.integer(as.vector(t(JeuSyn1[[5]][,1:4]))),
   as.integer(10000), as.integer(4), as.numeric(c(0,0)))
  Risk[i,1]=out[[9]][1]
  out=reg2(JeuSyn1, CPS2000)
  IO[i,1]=out[[3]]
  # marital-race-sex

}




AMARITL=randomForest(as.factor(CPS2000RC[,3])~.,
data=CPS2000RC[,c(2,4:12)], xtest=CPS2000RC[,c(2,4:12)]
)$test$predicted




table(CPS2000RC[,2])
```

```
set.seed(43255465)
ARACE=(14*CPS2000RC[,1]+2*CPS2000RC[,3])/2
ARACE2=ifelse(ARACE<11, 1, ifelse(ARACE<15,2,ifelse
(ARACE<19,3,4)))+
 (rbinom(10000,2,0.5)-1)
cor(CPS2000RC[,1],ARACE2)
cor(CPS2000RC[,3],ARACE2)
cor(CPS2000RC[,c(1,3)])

CPS2000RC[,2]=ARACE2

Risk=matrix(NA,50,6)

IO=matrix(NA,50,6)
set.seed(5435345)
for(i in 1:50)
{
  # Synthétisons d'abord sex-race-marital
  CPS2000=CPS2000RC
  JeuSyn1=replicate(5,GenSyn2(1,CPS2000,1:3))
  # Évaluons le risque et l'utilité
  out=.C("Risk", as.integer(as.vector(t(CPS2000[,1:4]))),
  as.integer(as.vector(t(JeuSyn1[[1]][,1:4]))), as.integer
  (as.vector(t(JeuSyn1[[2]][,1:4]))),as.integer(as.vector
  (t(JeuSyn1[[3]][,1:4]))),as.integer(as.vector(t
  (JeuSyn1[[4]][,1:4]))),as.integer(as.vector(t(JeuSyn1[[5]][,1:4]))),
  as.integer(10000), as.integer(4), as.numeric(c(0,0)))
  Risk[i,1]=out[[9]][1]
  out=reg2(JeuSyn1, CPS2000)
  IO[i,1]=out[[3]]
  # marital-race-sex

  CPS2000=CPS2000RC[,c(3,2,1,4:12)]
  JeuSyn1=replicate(5, GenSyn2(1,CPS2000,1:3))
  out=.C("Risk", as.integer(as.vector(t(CPS2000[,1:4]))),
  as.integer(as.vector(t(JeuSyn1[[1]][,1:4]))), as.integer
  (as.vector(t(JeuSyn1[[2]][,1:4]))),as.integer(as.vector
  (t(JeuSyn1[[3]][,1:4]))),as.integer(as.vector(t
```

```
(JeuSyn1[[4]][,1:4]))),as.integer(as.vector(t(JeuSyn1[[5]][,1:4]))),
as.integer(10000), as.integer(4), as.numeric(c(0,0)))
Risk[i,2]=out[[9]][1]
out=reg2(JeuSyn1, CPS2000)
IO[i,2]=out[[3]]

CPS2000=CPS2000RC[,c(3,1,2,4:12)]
JeuSyn1=replicate(5, GenSyn2(1,CPS2000,1:3))
out=.C("Risk", as.integer(as.vector(t(CPS2000[,1:4]))),
 as.integer(as.vector(t(JeuSyn1[[1]][,1:4]))), as.integer
 (as.vector(t(JeuSyn1[[2]][,1:4]))),as.integer(as.vector
 (t(JeuSyn1[[3]][,1:4]))),as.integer(as.vector(t
 (JeuSyn1[[4]][,1:4]))),as.integer(as.vector(t(JeuSyn1[[5]][,1:4]))),
 as.integer(10000), as.integer(4), as.numeric(c(0,0)))
Risk[i,3]=out[[9]][1]
out=reg2(JeuSyn1, CPS2000)
IO[i,3]=out[[3]]

CPS2000=CPS2000RC[,c(1,3,2,4:12)]
JeuSyn1=replicate(5, GenSyn2(1,CPS2000,1:3))
out=.C("Risk", as.integer(as.vector(t(CPS2000[,1:4]))),
as.integer(as.vector(t(JeuSyn1[[1]][,1:4]))), as.integer
(as.vector(t(JeuSyn1[[2]][,1:4]))),as.integer(as.vector
(t(JeuSyn1[[3]][,1:4]))),as.integer(as.vector(t
(JeuSyn1[[4]][,1:4]))),as.integer(as.vector(t(JeuSyn1[[5]][,1:4]))),
 as.integer(10000), as.integer(4), as.numeric(c(0,0)))
Risk[i,4]=out[[9]][1]
out=reg2(JeuSyn1, CPS2000)
IO[i,4]=out[[3]]

CPS2000=CPS2000RC[,c(2,1,3,4:12)]
JeuSyn1=replicate(5, GenSyn2(1,CPS2000,1:3))
out=.C("Risk", as.integer(as.vector(t(CPS2000[,1:4]))),
as.integer(as.vector(t(JeuSyn1[[1]][,1:4]))), as.integer
(as.vector(t(JeuSyn1[[2]][,1:4]))),as.integer(as.vector
(t(JeuSyn1[[3]][,1:4]))),as.integer(as.vector(t
(JeuSyn1[[4]][,1:4]))),as.integer(as.vector(t(JeuSyn1[[5]][,1:4]))),
as.integer(10000), as.integer(4), as.numeric(c(0,0)))
Risk[i,5]=out[[9]][1]
```

```
  out=reg2(JeuSyn1, CPS2000)
  IO[i,5]=out[[3]]

  CPS2000=CPS2000RC[,c(2,3,1,4:12)]
  JeuSyn1=replicate(5, GenSyn2(1,CPS2000,1:3))
  out=.C("Risk", as.integer(as.vector(t(CPS2000[,1:4]))),
   as.integer(as.vector(t(JeuSyn1[[1]][,1:4]))), as.integer
   (as.vector(t(JeuSyn1[[2]][,1:4]))),as.integer(as.vector
   (t(JeuSyn1[[3]][,1:4]))),as.integer(as.vector(t
   (JeuSyn1[[4]][,1:4]))),as.integer(as.vector(t(JeuSyn1[[5]][,1:4]))),
   as.integer(10000), as.integer(4), as.numeric(c(0,0)))
  Risk[i,6]=out[[9]][1]
  out=reg2(JeuSyn1, CPS2000)
  IO[i,6]=out[[3]]
}

IO2=apply(IO, 2, FUN=mean)
Mean2=apply(Risk, 2,FUN=mean)

save(IO, file="IO6aout.Rdata")
save(Risk, file="Risk6aout.Rdata")
names(Mean2)=c("SRM","MRS","MSR","SMR","RSM","RMS")
names(IO2)=c("SRM","MRS","MSR","SMR","RSM","RMS")
png("Erreur_bar_IO11aout.png")
plot(x=c(1,6), y= c(0.73,0.83),xaxt='n',main="Barres d'erreur de
l'utilité en fonction des différents ordres", type='n', xlab="Ordre
 des variables", ylab="Chevauchement des intervalles", las=1 )
errbar(1:6, IO2, IO2+apply(IO,2, sd), IO2-apply(IO,2,sd),xlab="
Ordre des variables", ylab="Mesure d'utilité", add=TRUE)
axis(1,at=c(1:6),labels=names(IO2))
dev.off()

png("Erreur_bar_Risk11aout.png")
plot(x=c(1,6), y= c(0.082,0.096),xaxt='n',main="Barres d'erreur
 du risque en fonction des différents ordres", type='n', xlab="
 Ordre des variables", ylab="Risque de réidentification", las=1 )
errbar(1:6, Mean2, Mean2+apply(Risk,2, sd), Mean2-apply(Risk,2,sd), add=T)
axis(1,at=c(1:6),labels=names(Mean2))
dev.off()
```

Annexe C

Code Chapitre 3

Exemple 1

```
# Estimating the mean with sigma2 known
n=100
sigma2=100
B=10000
mu_moy=NULL
var_mu_IM=NULL
var_theo=NULL
set.seed(431423)
for( k in 1:B)
{
  Y=rnorm(n,0,sqrt(sigma2))
  Pri=sample(n,35,F)
  #ToImp=rbinom(n,1,0.387)
  ToImp=as.numeric(1:n %in% Pri)
  Y[ToImp==1]=NA

  #Prop of missing:
  PiM=sum(ToImp)/n
  m=100
  mu_IM=NULL
  S_k2=NULL
  for(i in 1:m)
  {

    Ymoy=mean(Y,na.rm=T)
```

```
    eps=rnorm(sum(ToImp),0,sqrt(sigma2))
    Sk=rnorm(1,0,sqrt(sigma2/sum(ToImp==0)))
    mu_IM[i]=Ymoy+PiM*(mean(eps)+Sk)
    S_k2[i]=(1/(n-1))* (sum((Y-Ymoy)^2, na.rm=T)+
    sum((eps-mean(eps))^2)+ n*(1-PiM)*PiM*(mean(eps)+Sk)^2)

  }
  var_mu_IM[k]=(mean(S_k2)/n)+(1+1/m)*var(mu_IM)

  #théorique:

  var_theo[k]=sigma2/sum(ToImp==0) * (1+ PiM/m)

  mu_moy[k]=mean(mu_IM)
}

mean(var_mu_IM)
mean(var_theo)
var(mu_moy)
```

Exemple 2

```
Beta1=3
Beta2=12
sigma2=100
B=5000
n=100
m=100
Beta_Jeu_j=matrix(NA,m,2)
Beta_rep_i=matrix(NA,B,2)
var_Beta_rep_i=matrix(NA,B,4)
var_Beta_rep_i2=matrix(NA,B,4)
set.seed(3542354)
for(i in 1:B)
{
  x1=rbinom(n,10,0.463)
  x2=ifelse(x1>5,rbinom(sum(x1>5),5,0.73),
  rbinom(sum(x1<5),5,0.41))
  y=Beta1*x1+ Beta2*x2 + rnorm(n,0,sqrt(sigma2))
```

```
  X=cbind(x1,x2)
  Pri=sample(n,35,F)
  #ToImp=rbinom(n,1,0.387)
  ToImp=as.numeric(1:n %in% Pri)
  y[ToImp==1]=NA
  Beta_Obs_Est=(solve(t(X[ToImp==0,])%*%X[ToImp==0,]))
  %*% t(X[ToImp==0,])%*%y[ToImp==0]
  V_obs=sigma2* (solve(t(X[ToImp==0,])%*%X[ToImp==0,]))
  var_beta_j=matrix(NA,m,4)
  for(j in 1:m)
  {
    bk=rmvnorm(1,rep(0,2),V_obs)
    ek=rnorm(sum(ToImp==1),0,sqrt(sigma2))
    y[ToImp==1]= X[ToImp==1,]%*%(Beta_Obs_Est+t(bk)) + ek
    Beta_Jeu_j[j,]=t(Beta_Obs_Est+ solve(t(X)%*% X)%*%
    t(X[ToImp==1,])%*%(X[ToImp==1,]%*%t(bk)+ek))
    var_beta_j[j,]=as.vector(vcov(lm(y~X-1)))
  }
  Beta_rep_i[i,]=apply(Beta_Jeu_j,2,mean)
  var_Beta_rep_i[i,]= as.vector(V_obs+ (1/m)*(V_obs-
  sigma2*(solve(t(X)%*%X))))
  var_Beta_rep_i2[i,]= apply(var_beta_j,2,mean) +
  (1+1/m)*as.vector(var(Beta_Jeu_j))
}

Beta_mean_rep=apply(Beta_rep_i,2,mean)
var(Beta_rep_i)
apply(var_Beta_rep_i,2,mean)
apply(var_Beta_rep_i,2,sd)
apply(var_Beta_rep_i2,2,mean)
```

Exemple 3

```
# Calcul de moyenne après imputation par
forêt aléatoire, variable continue

B=3000
m=100
n=300
```

```
y_moyen=rep(NA,m)
var_y=rep(NA,m)
Beta_rep_i=rep(NA,B)
var_Beta_rep_i=rep(NA,B)
moy=c(0,0)
sigma=matrix(c(12,5,5,7), 2,2)
set.seed(3542354)
for(i in 1:B)
{
  X=rmvnorm(n,moy, sigma)
  y=X[,1]+ X[,2] + rnorm(n,0,10)
  Pri=sample(n,90,F)
  #ToImp=rbinom(n,1,0.387)
  ToImp=as.numeric(1:n %in% Pri)
  y[ToImp==1]=NA
  y_moyen2=list()
  var_y2=list()
  for(j in 1:m)
  {
    #y[ToImp==1]=mice.impute.rf(y,(ToImp==0),X, ntree=10)
    modele=randomForest(y[ToImp==0]~.,
    data=as.data.frame(X[ToImp==0,]),
    xtest=as.data.frame(X[ToImp==1,]))
    y[ToImp==1]= modele$test$predicted
    y_moyen[j]=mean(y)
    var_y[j]=var(y)/n
    y_moyen2[[j]]=mean(y)
    var_y2[[j]]=var(y)/n
  }
  Beta_rep_i[i]=mean(y_moyen)
  var_Beta_rep_i[i]= (mean(var_y)) + (1+1/m)* var(y_moyen)

}

Beta_mean_rep=mean(Beta_rep_i)
var(Beta_rep_i)
mean(var_Beta_rep_i)

# Résultat: avec B=3000, m=10 et n=100
```

```
# Retrait selon binom param 0.387
> var(Beta_rep_i)
[1] 1.93998
> mean(var_Beta_rep_i)
[1] 0.9529256

# Avec n=300
# Retrait selon binom param 0.387
> var(Beta_rep_i)
[1] 0.6716313
> mean(var_Beta_rep_i)
[1] 0.3199292

# Avec n=300 et m=100
# Retrait selon binom param 0.387
> var(Beta_rep_i)
[1] 0.6787556
> mean(var_Beta_rep_i)
[1] 0.3215069

# n=300 m= 100 var x1=12 var x2=7 cov=5
# Retrait selon binom param 0.387
> var(Beta_rep_i)
[1] 0.719412
> mean(var_Beta_rep_i)
[1] 0.3187418


# Retrait de 90 m=100, n=300, repet=3000
> var(Beta_rep_i)
[1] 0.5961739
> mean(var_Beta_rep_i)
[1] 0.3437665
```

Exemple 4

```
# Même que précédent mais avec var Y discrète.

B=10000
```

```
m=100
n=300
y_moyen=rep(NA,m)
var_y=rep(NA,m)
Beta_rep_i=rep(NA,B)
var_Beta_rep_i=rep(NA,B)
moy=c(0,0)
sigma=matrix(c(8,5,5,8), 2,2)
set.seed(3542354)
for(i in 1:B)
{
  X=rmvnorm(n,moy, sigma)
  y=round((X[,1]+ X[,2] + rnorm(n,0,10))/2)
  Pri=sample(n,90,F)
  #ToImp=rbinom(n,1,0.387)
  ToImp=as.numeric(1:n %in% Pri)
  y[ToImp==1]=NA
  for(j in 1:m)
  {
    #y[ToImp==1]=mice.impute.rf(y,(ToImp==0),X, ntree=10)
    modele=randomForest(as.factor(y[ToImp==0])~.,
    data=as.data.frame(X[ToImp==0,]),
    xtest=as.data.frame(X[ToImp==1,]))
    Ynew= modele$test$predicted
    #y[ToImp==1]=mice.impute.norm.predict(y,(ToImp==0),X)
    #y[ToImp==1]=mice.impute.norm.nob(y,(ToImp==0),X)
    #y[ToImp==1]=mice.impute.norm(y,(ToImp==0),X)
    y[ToImp==1]=as.numeric(levels(Ynew)[as.numeric(Ynew)])
    y_moyen[j]=mean(y)
    var_y[j]=var(y)
  }
  Beta_rep_i[i]=mean(y_moyen)
  var_Beta_rep_i[i]= (mean(var_y)/n) + (1+1/m)* var(y_moyen)
}

Beta_mean_rep=mean(Beta_rep_i)
var(Beta_rep_i)
mean(var_Beta_rep_i)
```

```
# Même que précédent mais avec var Y discrète et tirage

B=3000
m=100
n=300
y_moyen=rep(NA,m)
var_y=rep(NA,m)
Beta_rep_i=rep(NA,B)
var_Beta_rep_i=rep(NA,B)
moy=c(0,0)
sigma=matrix(c(12,7,7,12), 2,2)
set.seed(3544)
for(i in 1:B)
{
  X=rmvnorm(n,moy, sigma)
  y=round((X[,1]+ X[,2] + rnorm(n,0,10))/2)
  Pri=sample(n,90,F)
  #ToImp=rbinom(n,1,0.387)
  ToImp=as.numeric(1:n %in% Pri)
  y[ToImp==1]=NA
  for(j in 1:m)
  {
    #y[ToImp==1]=mice.impute.rf(y,(ToImp==0),X, ntree=10)
    modele=randomForest(as.factor(y[ToImp==0])~.,
     data=as.data.frame(X[ToImp==0,]),xtest=as.data.frame
    (X[ToImp==1,]))
    proba= modele$test$votes
    y[ToImp==1]=apply(proba,1,FUN=function(x)
    as.numeric(row.names(which(rmultinom(1,1,prob=x)
    !=0,arr.ind=T))))
    #y[ToImp==1]=mice.impute.norm.predict(y,(ToImp==0),X)
    #y[ToImp==1]=mice.impute.norm.nob(y,(ToImp==0),X)
    #y[ToImp==1]=mice.impute.norm(y,(ToImp==0),X)
    #y[ToImp==1]=as.numeric(levels(Ynew)[as.numeric(Ynew)])
    y_moyen[j]=mean(y)
    var_y[j]=var(y)
```

```
  }
  Beta_rep_i[i]=mean(y_moyen)
  var_Beta_rep_i[i]= (mean(var_y)/n) + (1+1/m)* var(y_moyen)
}

Beta_mean_rep=mean(Beta_rep_i)
var(Beta_rep_i)
mean(var_Beta_rep_i)
```

Exemple 5

```
# Calcul de moyenne après la synth par régression
B=10000
m=100
n=100
y_moyen=rep(NA,m)
var_y=rep(NA,m)
Beta_rep_i=rep(NA,B)
var_Beta_rep_i=rep(NA,B)
var_Beta_rep_i2=rep(NA,B)
var_Beta_rep_i3=rep(NA,B)

moy=c(0,0)
sigma=matrix(c(10,5,5,10), 2,2)
set.seed(3542354)
for(i in 1:B)
{
  X=rmvnorm(n,moy, sigma)
  y=X[,1]+ X[,2] + rnorm(n,0,10)
  Pri=sample(n,35,F)
  #ToImp=rbinom(n,1,0.387)
  ToImp=as.numeric(1:n %in% Pri)
  Beta_Obs_Est=(solve(t(X)%*%X))%*% t(X)%*%y
  V_obs= vcov(summary(lm(y~X-1)))
  for(j in 1:m)
  {
    bk=rmvnorm(1,rep(0,2),V_obs)
    ek=rnorm(1,0,sqrt((V_obs/solve(t(X)%*%X))[1,1]/sum(ToImp==0)))
    y[ToImp==1]= X[ToImp==1,]%*%(Beta_Obs_Est+t(bk)) + ek
```

```
    y_moyen[j]=mean(y)
    var_y[j]=var(y)
  }
  Beta_rep_i[i]=mean(y_moyen)
  var_Beta_rep_i[i]= (mean(var_y)/n) + (1+1/m)* var(y_moyen)
  var_Beta_rep_i2[i]= (mean(var_y)/n) + (1/m)* var(y_moyen)
  var_Beta_rep_i3[i]= (1+1/m)* var(y_moyen)- (mean(var_y)/n)

}
#ne pas se fier au nom
mean(Beta_rep_i)
var(Beta_rep_i)
mean(var_Beta_rep_i)
mean(var_Beta_rep_i2)
mean(var_Beta_rep_i3)
```

Exemple 6

```
# Confi. avec foret aléatoire sans tirage. y discret
n=300
Beta1=3
Beta2=-1
moyenne=c(0,0)
sigma=matrix(c(1,0.6, 0.6,1),2,2)
set.seed(532523)

repet=3000#nombre de réplication
qmoy=matrix(NA,repet,2)
B=matrix(NA,repet,2)
U=matrix(NA,repet,2)
Tm=matrix(NA,repet,2)
k=10
set.seed(53253)
for(index in 1:repet)
{
  X=rmvnorm(n,moyenne, sigma)
  Y=round(Beta1*X[,1]+ Beta2*X[,2] +rnorm(n,0,3))
  VecBeta=c(Beta1, Beta2)
```

```
  Jeu=cbind(Y,X)
  Beta_jeu_i=matrix(NA,k,2)
  Var_Beta_jeu_i=matrix(NA,k,2)
  for(i in 1:k)
  {
    modele=randomForest(as.factor(Y)~.,data=as.data.frame(Jeu[,2:3]))
    Ynew=as.numeric(modele$predicted)
    IC=round(confint(lm(Ynew~.-1, data=as.data.frame(Jeu[,2:3]))),3)
    Beta_jeu_i[i,]=summary(lm(Ynew~.-1,
    data=as.data.frame(Jeu[,2:3])))$coefficient[,1]
    Var_Beta_jeu_i[i,]=summary(lm(Ynew~.-1,
    data=as.data.frame(Jeu[,2:3])))$coefficient[,2]^2
  }

  qmoy[index,]=apply(Beta_jeu_i,2,mean)
  B[index,]=diag(var(Beta_jeu_i))
  U[index,]=apply(Var_Beta_jeu_i,2,mean)

}
Tm=U+(B/k)
m=k
vm=(m-1)*(1+m*(U/B))^2
IC2=cbind(qmoy-qt(0.975,vm)*sqrt(Tm), qmoy+qt(0.975,vm)*sqrt(Tm))
apply(qmoy,2,var)
apply(Tm,2,mean)

#> apply(qmoy,2,var)
#[1] 0.7491549 0.7282531
#> apply(Tm,2,mean)
#[1] 0.7396704 0.7425105


# Avec GenSyn2 (donc avec tirage), y discret
n=300
Beta1=3
Beta2=-1
moyenne=c(0,0)
sigma=matrix(c(1,0.6, 0.6,1),2,2)
set.seed(532523)
```

```
repet=3000#nombre de réplication
qmoy=matrix(NA,repet,2)
B=matrix(NA,repet,2)
U=matrix(NA,repet,2)
Tm=matrix(NA,repet,2)
k=10
set.seed(53253)
for(index in 1:repet)
{
  X=rmvnorm(n,moyenne, sigma)
  Y=round(Beta1*X[,1]+ Beta2*X[,2] +rnorm(n,0,5))
  VecBeta=c(Beta1, Beta2)

  Jeu=cbind(Y,X)
  Couverture=matrix(NA,k,2)
  Beta_jeu_i=matrix(NA,k,2)
  Var_Beta_jeu_i=matrix(NA,k,2)
  JeuSyn=replicate(k,GenSyn2(1,Jeu,1))
  for(i in 1:k)
  {
    Ynew=JeuSyn[[i]][,1]
    IC=round(confint(lm(Ynew~.-1, data=as.data.frame(Jeu[,2:3]))),3)
    Beta_jeu_i[i,]=summary(lm(Ynew~.-1,
    data=as.data.frame(Jeu[,2:3])))$coefficient[,1]
    Var_Beta_jeu_i[i,]=summary(lm(Ynew~.-1,
    data=as.data.frame(Jeu[,2:3])))$coefficient[,2]^2
  }

  qmoy[index,]=apply(Beta_jeu_i,2,mean)
  B[index,]=diag(var(Beta_jeu_i))
  U[index,]=apply(Var_Beta_jeu_i,2,mean)


}
Tm=U+(B/k)
m=k
vm=(m-1)*(1+m*(U/B))^2
IC2=cbind(qmoy-qt(0.975,vm)*sqrt(Tm), qmoy+qt(0.975,vm)*sqrt(Tm))
```

```
apply(qmoy,2,var)
apply(Tm,2,mean)

#m=10 n=300 rep=3000
#> apply(qmoy,2,var)
#[1] 0.04704437 0.04621456
#> apply(Tm,2,mean)
#[1] 0.04936550 0.04922549


#> apply(qmoy,2,var)
#[1] 0.1340127 0.1272386
#> apply(Tm,2,mean)
#[1] 0.1416611 0.1413754

#2493 repl. (vraiment long) k=50 (fait en parallèle finalement..)
#> apply(qmoy,2,var,na.rm=T)
#[1] 0.1277458 0.1216112
#> apply(Tm,2,mean, na.rm=T)
#[1] 0.1349704 0.1350344
```

Exemple 7

```
# Vérifier si synthétise partiellement une variable vs une variable complet.

n=300
Beta1=3
Beta2=-1
moyenne=c(0,0)
sigma=matrix(c(3,1.6, 1.6,3),2,2)/3
repet=3000#nombre de réplication
qmoy=NULL
B=NULL
U=NULL
Tm=NULL
k=50
set.seed(53253)
for(index in 1:repet)
{
```

```
  X=rmvnorm(n,moyenne, sigma)
  Y=round(Beta1*X[,1]+ Beta2*X[,2] +rnorm(n,0,3))
  ToSyn=sample(1:n,100,replace=F)
  VecBeta=c(Beta1, Beta2)
  Jeu=cbind(Y,X)
  Beta_jeu_i=NULL
  Var_Beta_jeu_i=NULL
  for(i in 1:k)
  {
    modele=randomForest(as.factor(Y)~.,data=as.data.frame(Jeu[,2:3]))
    Ynew=modele$predicted[ToSyn] # MODIFIER SUR CODEPARALLELE
    Ynew[101:300]=Y[-ToSyn]
    Ynew=as.numeric(levels(Ynew)[as.numeric(Ynew)])
    Beta_jeu_i[i]=mean(Ynew)
    Var_Beta_jeu_i[i]=var(Ynew)/n
  }

  qmoy[index]=mean(Beta_jeu_i)
  B[index]=var(Beta_jeu_i)
  U[index]=mean(Var_Beta_jeu_i)

}
Tm=U+(B/k)
# matrice var cov: 1 0.6. 0.6 1
var(qmoy)
#[1] 0.06781878
mean(Tm)
#[1] 0.04951934
```

Fonction GenSyn partielle

```
GenSyn2PAR= function(n=1,MatIni,ColSyn,ToSyn,nArbre=500,
sortie=list())
{
  library("randomForest")
  MatFin=matrix(NA, nrow(MatIni),ncol(MatIni))
  VarExp=NA
  toChange=1:ncol(MatIni) %in% ColSyn
  MatFin[,toChange==F]=as.matrix(MatIni[,toChange==F])
```

```
  MatFin[-ToSyn,toChange==T]=as.matrix(MatIni[-ToSyn,toC
  hange==T])
  for(i in 1:length(ColSyn))
  {
    VarSyn=NA
    VarSyn<-as.factor(MatIni[,ColSyn[i]])
    VarExp<-as.matrix(MatIni[,c(which(toChange==F), which
    (toChange==T)[1:i-1])])
    RF.VarSyn=randomForest(VarExp,VarSyn,ntree=nArbre,xtest
    =as.matrix(MatFin[,c(which(toChange==F), which(toChange
    ==T)[1:i-1])]))
    for(j in 1:length(ToSyn))
    {
      proba=RF.VarSyn$test$votes[ToSyn[j],]
      MatFin[ToSyn[j],ColSyn[i]]=as.numeric(row.names(which(
      rmultinom(1,1,prob=proba)!=0,arr.ind=T)))
    }
  }
  colnames(MatFin)=colnames(MatIni)
  sortie[[n]]=MatFin
  ifelse(n==1,return(sortie),sortie[[n-1]]<-return(GenSyn2
  (n-1,MatIni,ColSyn,nArbre,sortie)))
}
```

Exemple 8

```
n=300
Beta1=3
Beta2=-1
Beta3=2
Beta4=-3
moyenne=c(0,0,0,0)
sigma=matrix(c(10,6, 3, 7, 6 ,10, 4, 6, 3, 4, 10, 7, 7,6,7,10),4,4)/10
repet=5000#nombre de réplication
qmoy=matrix(NA,repet,4)
B=matrix(NA,repet,4)
U=matrix(NA,repet,4)
Tm=NULL
k=10
```

```
set.seed(533)
for(index in 1:repet)
{
  X=rmvnorm(n,moyenne, sigma)
  X=round(X)
  Y=round(Beta1*X[,1]+ Beta2*X[,2]+Beta3*X[,3]+Beta4*X[,4] +rnorm(n,0,3))
  VecBeta=c(Beta1, Beta2,Beta3,Beta4)
  Y=round(Y)
  Jeu=cbind(Y,X)
  Beta_jeu_i=matrix(NA,k,4)
  Var_Beta_jeu_i=matrix(NA,k,4)
  JeuSyn=replicate(k,GenSyn2(1,Jeu,1:2))
  for(i in 1:k)
  {
    Ynew=JeuSyn[[i]][,1]
    IC=round(confint(lm(Ynew~.-1, data=as.data.frame(JeuSyn[[i]][,2:5]))),3)
    Beta_jeu_i[i,]=summary(lm(Ynew~.-1,
    data=as.data.frame(JeuSyn[[i]][,2:5])))$coefficients[,1]
    Var_Beta_jeu_i[i,]=summary(lm(Ynew~.-1,
    data=as.data.frame(JeuSyn[[i]][,2:5])))$coefficients[,2]^2
  }

  qmoy[index,]=apply(Beta_jeu_i,2,mean)
  B[index,]=diag(var(Beta_jeu_i))
  U[index,]=apply(Var_Beta_jeu_i,2,mean)

}
Tp=U+(B/k)

m=k
vm=(m-1)*(1+m*(U/B))^2
IC2=cbind(qmoy-qt(0.975,vm)*sqrt(Tp), qmoy+qt(0.975,vm)*sqrt(Tp))
apply(qmoy,2,var)
apply(Tp,2,mean)

> apply(qmoy,2,var)
[1] 0.05206587 0.03782886 0.04272594 0.06402625
> apply(Tp,2,mean)
[1] 0.05646942 0.03935430 0.04400276 0.07238836
```

Exemple 9

```
load("CPS2000RC2.Rdata")

CPS2000_HSSVAL<-CPS2000RC[which(CPS2000RC$SSVAL > 0),]
CPS2000_HSSVAL<-CPS2000_HSSVAL[which(CPS2000_HSSVAL$AAGE > 54),]
CPS2000_HSSVAL$wido<-CPS2000_HSSVAL$AMARITL %in% 4
CPS2000_HSSVAL$Divo<-CPS2000_HSSVAL$AMARITL %in% 5
CPS2000_HSSVAL$Single<-CPS2000_HSSVAL$AMARITL %in% c(6,7)

CPS2000_HSSVAL$school<-ifelse(CPS2000_HSSVAL$AHGA %in% c(32:39),1,
                      ifelse(CPS2000_HSSVAL$AHGA %in% 40,2,
                      ifelse(CPS2000_HSSVAL$AHGA %in% c(41:42),3,
                      ifelse(CPS2000_HSSVAL$AHGA %in% c(43:46),4,0))))

theta_hat=summary(lm(sqrt(SSVAL)~as.factor(ASEX)+as.factor(ARACE)
+as.factor(wido)+as.factor(Divo)+as.factor(Single)+as.factor(school)
+AAGE,data=CPS2000_HSSVAL))$coefficients[,1]




#Couverture:
n=200
theta_syn=matrix(NA,n,13)
Couverture=matrix(NA,n,13)
set.seed(543223)
JeuSyn=replicate(n,GenSyn2(1,CPS2000RC,1:3)) # SRM
for(i in 1:n)
{
  JeuSynt=as.data.frame(JeuSyn[[i]])
  JeuSynt<-JeuSynt[which(JeuSynt$SSVAL > 0),]
  JeuSynt<-JeuSynt[which(JeuSynt$AAGE > 54),]
  JeuSynt$wido<-JeuSynt$AMARITL %in% 4
  JeuSynt$Divo<-JeuSynt$AMARITL %in% 5
  JeuSynt$Single<-JeuSynt$AMARITL %in% c(6,7)

  JeuSynt$school<-ifelse(JeuSynt$AHGA %in% c(32:39),1,
                 ifelse(JeuSynt$AHGA %in% 40,2,
```

```
                  ifelse(JeuSynt$AHGA %in% c(41:42),3,
                  ifelse(JeuSynt$AHGA %in% c(43:46),4,0))))
  theta_syn[i,]=summary(lm(sqrt(SSVAL)~as.factor(ASEX)+as.factor(ARACE)
  +as.factor(wido)+as.factor(Divo)+as.factor(Single)+as.factor(school)+
  AAGE,data=JeuSynt))$coefficients[,1]
  IC=round(confint(lm(sqrt(SSVAL)~as.factor(ASEX)+as.factor(ARACE)
  +as.factor(wido)+as.factor(Divo)+as.factor(Single)+as.factor(school)+AAGE,
  data=JeuSynt)),3)
  Couverture[i,]=as.numeric(theta_hat > IC[,1] & theta_hat < IC[,2])
}
colnames(Couverture)=names(theta_hat)
apply(Couverture,2,sum)/2



# Exemple avec jeu simulé
n=300
Beta1=3
Beta2=-1
moyenne=c(0,0)
sigma=matrix(c(1,0.6, 0.6,1),2,2)
set.seed(4342)
X=rmvnorm(n,moyenne, sigma)
Y=round(Beta1*X[,1]+ Beta2*X[,2] +rnorm(n,0,3))
VecBeta=c(Beta1, Beta2)
k=300
Jeu=cbind(Y,X)
set.seed(4325)
JeuSyn=replicate(k,GenSyn2(1,Jeu,1))
Couverture=matrix(NA,k,2)
for(i in 1:k)
{
  JeuSynt=JeuSyn[[i]]
  Y=JeuSynt[,1]
  IC=round(confint(lm(Y~.-1, data=as.data.frame(JeuSynt[,2:3]))),3)
  Couverture[i,]=as.numeric(VecBeta > IC[,1] & VecBeta < IC[,2])
}
colnames(Couverture)=names(VecBeta)
apply(Couverture,2,mean)*100
```

```
#[1] 67 98



load("CPS2000RC2.Rdata")

CPS2000_HSSVAL<-CPS2000RC[which(CPS2000RC$SSVAL > 0),]
CPS2000_HSSVAL<-CPS2000_HSSVAL[which(CPS2000_HSSVAL$AAGE > 54),]
CPS2000_HSSVAL$wido<-CPS2000_HSSVAL$AMARITL %in% 4
CPS2000_HSSVAL$Divo<-CPS2000_HSSVAL$AMARITL %in% 5
CPS2000_HSSVAL$Single<-CPS2000_HSSVAL$AMARITL %in% c(6,7)

CPS2000_HSSVAL$school<-ifelse(CPS2000_HSSVAL$AHGA %in% c(32:39),1,
                       ifelse(CPS2000_HSSVAL$AHGA %in% 40,2,
                       ifelse(CPS2000_HSSVAL$AHGA %in% c(41:42),3,
                       ifelse(CPS2000_HSSVAL$AHGA %in% c(43:46),4,0))))

theta_hat=summary(lm(sqrt(SSVAL)~as.factor(ASEX)+as.factor(ARACE)
+as.factor(wido)+as.factor(Divo)+as.factor(Single)+as.factor(school)
+AAGE,data=CPS2000_HSSVAL))$coefficients[,1]




#Couverture:
n=200
theta_syn=matrix(NA,n,13)
var_theta_syn=matrix(NA,n,13)
Couverture=matrix(NA,n,13)
set.seed(543223)
JeuSyn=replicate(n,GenSyn2(1,CPS2000RC,1:3)) # SRM
for(i in 1:n)
{
  JeuSynt=as.data.frame(JeuSyn[[i]])
  JeuSynt<-JeuSynt[which(JeuSynt$SSVAL > 0),]
  JeuSynt<-JeuSynt[which(JeuSynt$AAGE > 54),]
  JeuSynt$wido<-JeuSynt$AMARITL %in% 4
  JeuSynt$Divo<-JeuSynt$AMARITL %in% 5
```

```
  JeuSynt$Single<-JeuSynt$AMARITL %in% c(6,7)

  JeuSynt$school<-ifelse(JeuSynt$AHGA %in% c(32:39),1,
                 ifelse(JeuSynt$AHGA %in% 40,2,
                 ifelse(JeuSynt$AHGA %in% c(41:42),3,
                 ifelse(JeuSynt$AHGA %in% c(43:46),4,0))))
  theta_syn[i,]=summary(lm(sqrt(SSVAL)~as.factor(ASEX)+as.factor(ARACE)
  +as.factor(wido)+as.factor(Divo)+as.factor(Single)+as.factor(school)
  +AAGE,data=JeuSynt))$coefficients[,1]

  var_theta_syn[i,]=summary(lm(sqrt(SSVAL)~as.factor(ASEX)+as.factor(ARACE)
  +as.factor(wido)+as.factor(Divo)+as.factor(Single)+as.factor(school)+
  AAGE,data=JeuSynt))$coefficients[,2]^2
  IC=round(confint(lm(sqrt(SSVAL)~as.factor(ASEX)+as.factor(ARACE)+
  as.factor(wido)+as.factor(Divo)+as.factor(Single)+as.factor(school)
  +AAGE,data=JeuSynt)),3)
  Couverture[i,]=as.numeric(theta_hat > IC[,1] & theta_hat < IC[,2])
}
qmoy=matrix(NA,n/5,13)
B=matrix(NA,n/5,13)
U=matrix(NA,n/5,13)
for(j in 1:(n/5))
{
  qmoy[j,]=apply(theta_syn[c(j:(j+4)),],2,mean)
  B[j,]=diag(var(theta_syn[j:(j+4),]))
  U[j,]=apply(var_theta_syn[j:(j+4),],2,mean)
}
Tm=U+(B/n)
m=5
vm=(m-1)*(1+m*(U/B))^2
IC2=cbind(qmoy-qt(0.975,vm)*sqrt(Tm), qmoy+qt(0.975,vm)*sqrt(Tm))
Couverture2=matrix(NA,(n/5),13)
for(j in 1:13)
{
  for(k in 1:(n/5))
    Couverture2[k,j]=as.numeric(theta_hat[j] > IC2[k,j] &
    theta_hat[j] < IC2[k,j+13])
}
colnames(Couverture)=names(theta_hat)
```

```
apply(Couverture,2,sum)/2
colnames(Couverture2)=names(theta_hat)
apply(Couverture2,2,sum)



# Exemple avec jeu simulé
n=300
Beta1=3
Beta2=-1
Beta3=2
moyenne=c(0,0,0)
sigma=matrix(c(1,0.6, 0.7, 0.6,1, 0.5, 0.7, 0.5, 1),3,3)
set.seed(532523)
X=rmvnorm(n,moyenne, sigma)
Y=round(Beta1*X[,1]+ Beta2*X[,2] +Beta3*X[,3] +rnorm(n,0,3))
VecBeta=c(Beta1, Beta2, Beta3)
k=500
IC_ori=round(confint(lm(Y~.-1,data=as.data.frame(X))),3)
Couverture_ori=as.numeric(VecBeta > IC_ori[,1] & VecBeta < IC_ori[,2])
Jeu=cbind(Y,X)
set.seed(53253)
JeuSyn=replicate(k,GenSyn2(1,Jeu,1))
Couverture=matrix(NA,k,3)
Beta_jeu_i=matrix(NA,k,3)
Var_Beta_jeu_i=matrix(NA,k,3)
for(i in 1:k)
{
  JeuSynt=JeuSyn[[i]]
  Y=JeuSynt[,1]
  IC=round(confint(lm(Y~.-1, data=as.data.frame(JeuSynt[,2:4]))),3)
  Beta_jeu_i[i,]=summary(lm(Y~.-1,data=as.data.frame(
  JeuSynt[,2:4])))$coefficient[,1]
  Var_Beta_jeu_i[i,]=summary(lm(Y~.-1,data=as.data.frame(
  JeuSynt[,2:4])))$coefficient[,2]^2
  Couverture[i,]=as.numeric(VecBeta > IC[,1] & VecBeta < IC[,2])
}
qmoy=matrix(NA,k/5,3)
```

```
B=matrix(NA,k/5,3)
U=matrix(NA,k/5,3)
for(j in 1:(k/5))
{
  qmoy[j,]=apply(Beta_jeu_i[c(j:(j+4)),],2,mean)
  B[j,]=diag(var(Beta_jeu_i[j:(j+4),]))
  U[j,]=apply(Var_Beta_jeu_i[j:(j+4),],2,mean)
}
Couverture2=matrix(NA,k/5,3)
Tm=U+(B/n)
m=5
vm=(m-1)*(1+m*(U/B))^2
IC2=cbind(qmoy-qt(0.975,vm)*sqrt(Tm), qmoy+qt(0.975,vm)*sqrt(Tm))
for(j in 1:3)
{
  for(l in 1:(k/5))
    Couverture2[l,j]=as.numeric(VecBeta[j] > IC2[l,j] &
     VecBeta[j] < IC2[l,j+3])
}
colnames(Couverture)=names(VecBeta)
apply(Couverture,2,mean)*100
colnames(Couverture2)=names(VecBeta)
apply(Couverture2,2,mean)*100

> apply(Couverture,2,mean)*100
[1] 79.0 93.0 84.6
> colnames(Couverture2)=names(VecBeta)
> apply(Couverture2,2,mean)*100
[1] 100 100 100


# Exemple avec jeu simulé
n=300
Beta1=3
Beta2=-1
moyenne=c(0,0)
sigma=matrix(c(1,0.6, 0.6,1),2,2)
set.seed(532523)
X=rmvnorm(n,moyenne, sigma)
```

```
Y=round(Beta1*X[,1]+ Beta2*X[,2] +rnorm(n,0,3))
VecBeta=c(Beta1, Beta2)
k=300
Jeu=cbind(Y,X)
Couverture=matrix(NA,k,2)
Beta_jeu_i=matrix(NA,k,2)
Var_Beta_jeu_i=matrix(NA,k,2)
set.seed(53253)
for(i in 1:k)
{
  modele=randomForest(as.factor(Y)~.,data=as.data.frame(Jeu[,2:3]))
  Ynew=as.numeric(modele$predicted)
  IC=round(confint(lm(Ynew~.-1, data=as.data.frame(Jeu[,2:3]))),3)
  Beta_jeu_i[i,]=summary(lm(Ynew~.-1,data=as.data.
  frame(Jeu[,2:3])))$coefficient[,1]
  Var_Beta_jeu_i[i,]=summary(lm(Ynew~.-1,data=as.data.
  frame(Jeu[,2:3])))$coefficient[,2]^2
  Couverture[i,]=as.numeric(VecBeta > IC[,1] & VecBeta < IC[,2])
}
qmoy=matrix(NA,k/5,2)
B=matrix(NA,k/5,2)
U=matrix(NA,k/5,2)
for(j in 1:(k/5))
{
  qmoy[j,]=apply(Beta_jeu_i[c(j:(j+4)),],2,mean)
  B[j,]=diag(var(Beta_jeu_i[j:(j+4),]))
  U[j,]=apply(Var_Beta_jeu_i[j:(j+4),],2,mean)
}
Couverture2=matrix(NA,k/5,2)
Tm=U+(B/n)
m=5
vm=(m-1)*(1+m*(U/B))^2
IC2=cbind(qmoy-qt(0.975,vm)*sqrt(Tm), qmoy+qt(0.975,vm)*sqrt(Tm))
for(j in 1:2)
{
  for(l in 1:(k/5))
    Couverture2[l,j]=as.numeric(VecBeta[j] > IC2[l,j] &
    VecBeta[j] < IC2[l,j+2])
}
```

```
colnames(Couverture)=names(VecBeta)
apply(Couverture,2,mean)*100
colnames(Couverture2)=names(VecBeta)
apply(Couverture2,2,mean)*100
```

Annexe D

Code Chapitre 4

Code vérification de α

Fonction GenSyn pour Dirichlet-Multinomiale

```
GenSyn_DM= function(n=1,MatIni,ColSyn,nArbre=500,
alpha,sortie=list())
{
  library("randomForest")
  library("gtools")
  MatFin=matrix(NA, nrow(MatIni),ncol(MatIni))
  VarExp=NA
  toChange=1:ncol(MatIni) %in% ColSyn
  MatFin[,toChange==F]=as.matrix(MatIni[,toChange==F])
  for(i in 1:length(ColSyn))
  {
    VarSyn=NA
    VarSyn<-as.factor(MatIni[,ColSyn[i]])
    VarExp<-as.matrix(MatIni[,c(which(toChange==F),
    which(toChange==T)[1:i-1])])
    RF.VarSyn=randomForest(VarExp,VarSyn,ntree=nArbre,
    xtest=as.matrix(MatFin[,c(which(toChange==F), which
    (toChange==T)[1:i-1])]))
    for(j in 1:nrow(MatIni))
    {
      X=RF.VarSyn$test$votes[j,]*nArbre
      alpha=rep(alpha,length(RF.VarSyn$votes[j,]))
      #ajout de Dirichlet-Multi.
      thetatilde=rdirichlet(1,as.vector(X+alpha))
```

```
      MatFin[j,ColSyn[i]]=as.numeric(names(RF.VarSyn$
      votes[j,])[which(rmultinom(1,1,prob=thetatilde)!=0)])
      alpha=alpha[1]
    }
  }
  colnames(MatFin)=colnames(MatIni)
  sortie[[n]]=MatFin
  ifelse(n==1,return(sortie),sortie[[n-1]]<-return(GenSyn2
  (n-1,MatIni,ColSyn,nArbre,sortie)))
}



# Vérification des formules pour obtenir la variance
 avec différents alpla

# Fichier GenSyn_DM.R pour la fonction GenSyn_DM
library(mvtnorm)
alp=10
repet=3000
k=20
n=300
moy=c(0,0)
sigma=matrix(c(10,7,7,10),2,2)
qmoy=NULL
B=NULL
U=NULL
set.seed(5324)
for(i in 1:repet)
{
  X=rmvnorm(n,moy,sigma)
  Y=round(X[,1]+X[,2]+rnorm(n,0,3))
  Jeu=cbind(Y,X)
  JeuSyn=replicate(k,GenSyn_DM(1,Jeu,1,500, alpha=alp))
  q=NULL
  var_j=NULL
  for(j in 1:k)
  {
    Ynew=JeuSyn[[j]][,1]
    q[j]=mean(Ynew)
```

```
    var_j[j]=var(Ynew)/n
  }
  qmoy[i]=mean(q)
  B[i]=var(q)
  U[i]=mean(var_j)
}
Tp=U+(B/k)

var(qmoy)
mean(Tp)
```

Pour faire les graphiques

```
\begin{verbatim}
# Clear workspace
rm(list=ls())

# load snowfall package
require(snowfall)

# load chron package
require(mvtnorm)

#the_hosts <- rep(c("dms1","dms2","dms3","dms4","que
telet"),5)
the_hosts <- rep(c("dms9", "dms8", "dms7", "dms6","d
ms5", "dms4", "dms3", "dms2", "dms1"),8)
the_cpus <- 72


# Load in datasets needed in calculation
df1 = df2 = df3 = NULL



## Initialize parallel operation
sfInit(socketHosts=the_hosts, parallel=TRUE, cpus=the_cpus )
#sfInit(parallel=TRUE, cpus=the_cpus )
```

```
#########################################################
## specify functions that will be used:

set.seed( 12345 )
Eval_Utilite_Risk=function(k=5)
{
  setwd("/users/mcaron/Codemaitrise/Chapitre4")
  load("CPS2000RC2.Rdata")
  Jeu=CPS2000RC
  dyn.load("Test.so")
  VarY=12
  VarExp=c(1:4,8)
  RiskDA=NULL
  VarSyn=1:3
  # Doit se faire avec le jeu de données de
  Caiola et Reiter
  choix=sample(1:10000,4000,replace=T)
  JeuOrig=Jeu[choix,]
  JeuOrig2=JeuOrig[which(JeuOrig[,VarY]>0),]
  CoefOri=lm(sqrt(JeuOrig2[,VarY])~.-1,data=as.
  data.frame(JeuOrig2[,VarExp]))$coefficients
  alpha=10
  JeuSyn=replicate(5,GenSyn_DM(1,JeuOrig,VarSyn,
  500,alpha))# On crée les jeux synthétiques
  #JeuSyn=replicate(5,GenSyn2(1,JeuOrig,VarSyn,500))
  ER=matrix(NA,5,length(VarExp))
  Coef=matrix(NA,5,length(VarExp))
  U=matrix(NA,5,length(VarExp))
  for(j in 1:5)
  {
    Jeu2=JeuSyn[[j]]
    Jeu3=Jeu2[which(Jeu2[,VarY]>0),]
    Coef[j,]=lm(sqrt(Jeu3[,VarY])~.-1,data=as.
    data.frame(Jeu3[,VarExp]))$coefficients
    U[j,]=diag(vcov(lm(sqrt(Jeu3[,VarY])~.-1,
    data=as.data.frame(Jeu3[,VarExp]))))
    ER[j,]=abs(abs(Coef[j,]-CoefOri)/CoefOri)
  }
```

```
  UtilERC=apply(ER,2,mean)
  qmoy=apply(Coef,2,mean)
  B=apply(Coef,2,var)
  Um=apply(U,2,mean)
  out=.C("Risk", as.integer(as.vector(t(JeuOrig
  [,1:4]))), as.integer(as.vector(t(JeuSyn[[1]]
  [,1:4]))), as.integer(as.vector(t(JeuSyn[[2]]
  [,1:4]))),as.integer(as.vector(t(JeuSyn[[3]]
  [,1:4]))),as.integer(as.vector(t(JeuSyn[[4]]
  [,1:4]))),as.integer(as.vector(t(JeuSyn[[5]]
  [,1:4]))), as.integer(4000), as.integer(4),
   as.numeric(c(0,0)),as.numeric(rep(0,4000)))
  RiskDA=out[[9]][1]
  TP=Um+(B/k)
  list(c(as.vector(UtilERC),as.vector(TP),as.vector(RiskDA)
  ,as.vector(qmoy)))

}

GenSyn_DM= function(n=1,MatIni,ColSyn,nArbre=500,
alpha,sortie=list())
{
  library("randomForest")
  library("gtools")
  MatFin=matrix(NA, nrow(MatIni),ncol(MatIni))
  VarExp=NA
  toChange=1:ncol(MatIni) %in% ColSyn
  MatFin[,toChange==F]=as.matrix(MatIni[,toChange==F])
  for(i in 1:length(ColSyn))
  {
    VarSyn=NA
    VarSyn<-as.factor(MatIni[,ColSyn[i]])
    VarExp<-as.matrix(MatIni[,c(which(toChange==F),
    which(toChange==T)[1:i-1])])
    RF.VarSyn=randomForest(VarExp,VarSyn,ntree=nArbre,
    xtest=as.matrix(MatFin[,c(which(toChange==F), which
    (toChange==T)[1:i-1])]))
    for(j in 1:nrow(MatIni))
    {
```

```
      X=RF.VarSyn$test$votes[j,]*nArbre
      alpha=rep(alpha,length(RF.VarSyn$votes[j,]))
      #ajout de Dirichlet-Multi.
      thetatilde=rdirichlet(1,as.vector(X+alpha))
      MatFin[j,ColSyn[i]]=as.numeric(names(RF.VarSyn$
      votes[j,])[which(rmultinom(1,1,prob=thetatilde)!=0)])
      alpha=alpha[1]
    }
  }
  colnames(MatFin)=colnames(MatIni)
  sortie[[n]]=MatFin
  ifelse(n==1,return(sortie),sortie[[n-1]]<-return(GenSyn2
  (n-1,MatIni,ColSyn,nArbre,sortie)))
}


iter=3000
##########################################################
## 'Export' all functions, packages, and dataframes needed
##  for to all "slaves" so that parallel calculations can
##  occur simultaneously.

# functions
sfExport(list=list("Eval_Utilite_Risk"))
sfExport(list=list("GenSyn_DM"))


#packages
sfLibrary(mvtnorm)
sfLibrary(randomForest)

# dataframes
sfExport('df1')
sfExport('df2')
sfExport('df3')

# Set up random number generator
sfClusterSetupRNG(seed=53253)
```

```
## call function using sfApply; will return values as a list object
out = sfLapply(1:iter, fun=Eval_Utilite_Risk)
## stop parallel computing job
sfStop()

save(out, file="outChap4Alpha_10_28nov.Rdata")




## Mettre tous les résultats dans une liste
# Load chaque jeu puis l'assigné au alpha.
Alpha_0=out
Alpha_1=out
Alpha_2=out
Alpha_5=out
Alpha_10=out
Alpha_50=out
Alpha_100=out



save(Alpha_0,Alpha_1,Alpha_2,Alpha_5,Alpha_10,Alpha_50,
Alpha_100,file="Resultat.Rdata")
load("Resultat.Rdata")
res=unlist(Alpha_0)
res2=unlist(Alpha_1)
res3=unlist(Alpha_2)
res4=unlist(Alpha_5)
res5=unlist(Alpha_10)
res6=unlist(Alpha_50)
res7=unlist(Alpha_100)
a=seq(1,48000,by=16)
UtilERC=matrix(NA,length(a),5)
Tp=matrix(NA,length(a),5)
Risk=matrix(NA,length(a),1)
qmoy=matrix(NA,length(a),5)
UtilERC2=matrix(NA,length(a),5)
```

```
Tp2=matrix(NA,length(a),5)
Risk2=matrix(NA,length(a),1)
qmoy2=matrix(NA,length(a),5)
UtilERC3=matrix(NA,length(a),5)
Tp3=matrix(NA,length(a),5)
Risk3=matrix(NA,length(a),1)
qmoy3=matrix(NA,length(a),5)
UtilERC4=matrix(NA,length(a),5)
Tp4=matrix(NA,length(a),5)
Risk4=matrix(NA,length(a),1)
qmoy4=matrix(NA,length(a),5)
UtilERC5=matrix(NA,length(a),5)
Tp5=matrix(NA,length(a),5)
Risk5=matrix(NA,length(a),1)
qmoy5=matrix(NA,length(a),5)
UtilERC6=matrix(NA,length(a),5)
Tp6=matrix(NA,length(a),5)
Risk6=matrix(NA,length(a),1)
qmoy6=matrix(NA,length(a),5)
UtilERC7=matrix(NA,length(a),5)
Tp7=matrix(NA,length(a),5)
Risk7=matrix(NA,length(a),1)
qmoy7=matrix(NA,length(a),5)
for(i in 1:length(a))
{
  UtilERC[i,]=res[a[i]:(a[i]+4)]
  Tp[i,]=res[(a[i]+5):(a[i]+9)]
  Risk[i]=res[(a[i]+10)]
  qmoy[i,]=res[(a[i]+11):(a[i]+15)]

  UtilERC2[i,]=res2[a[i]:(a[i]+4)]
  Tp2[i,]=res2[(a[i]+5):(a[i]+9)]
  Risk2[i]=res2[(a[i]+10)]
  qmoy2[i,]=res2[(a[i]+11):(a[i]+15)]

  UtilERC3[i,]=res3[a[i]:(a[i]+4)]
  Tp3[i,]=res3[(a[i]+5):(a[i]+9)]
  Risk3[i]=res3[(a[i]+10)]
  qmoy3[i,]=res3[(a[i]+11):(a[i]+15)]
```

```
  UtilERC4[i,]=res4[a[i]:(a[i]+4)]
  Tp4[i,]=res4[(a[i]+5):(a[i]+9)]
  Risk4[i]=res4[(a[i]+10)]
  qmoy4[i,]=res4[(a[i]+11):(a[i]+15)]

  UtilERC5[i,]=res5[a[i]:(a[i]+4)]
  Tp5[i,]=res5[(a[i]+5):(a[i]+9)]
  Risk5[i]=res5[(a[i]+10)]
  qmoy5[i,]=res5[(a[i]+11):(a[i]+15)]

  UtilERC6[i,]=res6[a[i]:(a[i]+4)]
  Tp6[i,]=res6[(a[i]+5):(a[i]+9)]
  Risk6[i]=res6[(a[i]+10)]
  qmoy6[i,]=res6[(a[i]+11):(a[i]+15)]

  UtilERC7[i,]=res7[a[i]:(a[i]+4)]
  Tp7[i,]=res7[(a[i]+5):(a[i]+9)]
  Risk7[i]=res7[(a[i]+10)]
  qmoy7[i,]=res7[(a[i]+11):(a[i]+15)]
}

UtilERC_T=apply(UtilERC,2,function(x) sort(x)[75:2925])
UtilERC_T2=apply(UtilERC2,2,function(x) sort(x)[75:2925])
UtilERC_T3=apply(UtilERC3,2,function(x) sort(x)[75:2925])
UtilERC_T4=apply(UtilERC4,2,function(x) sort(x)[75:2925])
UtilERC_T5=apply(UtilERC5,2,function(x) sort(x)[75:2925])
UtilERC_T6=apply(UtilERC6,2,function(x) sort(x)[75:2925])
UtilERC_T7=apply(UtilERC6,2,function(x) sort(x)[75:2925])
mean(Risk)
median(Risk6)
mean(Risk2)
mean(Risk3)
mean(Risk4)
mean(Risk5)
mean(Risk6)
mean(Risk7)
ERV=t(apply(Tp,1,FUN=function(x) abs(x-apply(qmoy,2,var))
/apply(qmoy,2,var)))
```

```
ERV2=t(apply(Tp2,1,FUN=function(x) abs(x-apply(qmoy,2,var))
/apply(qmoy,2,var)))
ERV3=t(apply(Tp3,1,FUN=function(x) abs(x-apply(qmoy,2,var))
/apply(qmoy,2,var)))
ERV4=t(apply(Tp4,1,FUN=function(x) abs(x-apply(qmoy,2,var))
/apply(qmoy,2,var)))
ERV5=t(apply(Tp5,1,FUN=function(x) abs(x-apply(qmoy,2,var))
/apply(qmoy,2,var)))
ERV6=t(apply(Tp6,1,FUN=function(x) abs(x-apply(qmoy,2,var))
/apply(qmoy,2,var)))
ERV7=t(apply(Tp7,1,FUN=function(x) abs(x-apply(qmoy,2,var))
/apply(qmoy,2,var)))
png(filename="ERCChap4.png")
boxplot(cbind(UtilERC_T7,UtilERC_T6,UtilERC_T5,UtilERC_T4,
UtilERC_T3,UtilERC_T2,UtilERC_T)*100,at =c(1,2,3,4,5, 7,8,
9,10,11, 13,14,15,16,17, 19,20,21,22,23, 25,26,27,28,29,
31,32,33,34,35, 37,38,39,40,41), main= "Erreur relative
des coefficients en fonction du risque" , xaxt="n",xlab=
"Risque de réidentification",ylab="Erreur relative des
coefficients (en %)",cex=0.5, col=c(rep("skyblue",5),
rep("red",5),rep("yellow",5),rep("green",5),rep("orange"
,5), rep("purple",5),rep("limegreen",5)),outline=F,las=1)
axis(1,at=c(3,9,15,21,27,33,39),labels=c("6.39%","8.99%",
"11.3%","11.5%","11.6%","11.6%","11.7%"))
legend("topright",title="Valeur d'alpha",c("0","1","2",
"5","10","50","100"), fill=c("limegreen","purple","orange"
,"green","yellow","red","skyblue"), horiz=F, ncol=3)
text(c(1,2,3,4,5, 7,8,9,10,11, 13,14,15,16,17, 19,20,21,22
,23, 25,26,27,28,29, 31,32,33,34,35, 37,38,39,40,41),
c(apply(UtilERC_T7,2,median),apply(UtilERC_T6,2,median),
apply(UtilERC_T5,2,median),apply(UtilERC_T4,2,median),
apply(UtilERC_T3,2,median),apply(UtilERC_T2,2,median),
apply(UtilERC_T,2,median))*100,labels=rep(c("B1","B2",
"B3","B4","B5"),6),cex=0.6,pos=3)
dev.off()
png(filename="ERVChap4.png")
plot(c(0,47),c(0,110),type="n",xaxt="n", xlab="Risque
 de réidentification", ylab="Erreur relative de la
  variance (en %)", main="Erreur relative de la variance
```

```
   en fonction du risque",lwd=2, las=1)
boxplot(cbind(ERV7,ERV6,ERV5,ERV4,ERV3,ERV2,ERV)*100,
at =c(1,2,3,4,5, 8,9,10,11,12, 15,16,17,18,19, 22,23,
24,25,26, 29,30,31,32,33, 36,37,38,39,40, 43,44,45,46,
47),xaxt="n",yaxt="n", col=c(rep("skyblue",5),rep("red",
5),rep("yellow",5),rep("green",5),rep("orange",5), rep(
"purple",5),rep("limegreen",5)),add=T,outline=F)
axis(1,at=c(3,10,17,24,31,38,45),labels=c("6.39%","8.99%",
"11.3%","11.5%","11.6%","11.6%","11.7%"))
legend("topright",title="Valeur d'alpha",c("0","1","2",
"5","10","50","100"), fill=c("limegreen","purple","orange"
,"green","yellow","red","skyblue"), horiz=TRUE)
text(c(1,2,3,4,5, 8,9,10,11,12, 15,16,17,18,19, 22,23,24,
25,26, 29,30,31,32,33, 36,37,38,39,40, 43,44,45,46,47),
c(apply(ERV7,2,median),apply(ERV6,2,median),apply(ERV5,2,
median),apply(ERV4,2,median),apply(ERV3,2,median),apply
(ERV2,2,median),apply(ERV,2,median))*100,labels=rep(c("B1",
"B2","B3","B4","B5"),6),cex=0.6,pos=3)
dev.off()
```

Annexe E

Code Chapitre 5

Exemple de code pour réaliser les équations structurelles

```
# Usefull library
library(colortools)
library(plspm)

Jeu=read.csv("Jeu_LIS1.csv")


#rows of the inner model matrix

humain=c(0,0,0,0,0)
innov=c(1,0,0,0,0)
struc=c(1,1,0,0,0)
relat=c(1,0,1,0,0)
perform=c(0,1,1,1,0)

PME_C_path=rbind(humain,innov,struc,relat,perform)
colnames(PME_C_path)=rownames(PME_C_path)
PME_C_path

innerplot(PME_C_path)

PME_C_blocks=list(2,3:4,5:6,7:9,10)

PME_C_modes=c("A","A","A","A","A")
```

```
PME_C_pls=plspm(Jeu,PME_C_path,PME_C_blocks,modes=PME_C_modes)

plot(PME_C_pls)

summary(PME_C_pls)



load("Jeu_LIS_NE.Rdata")


#write.table(round(Jeu_LIS_NE,2), "Jeu_LIS_NE.txt", quote=F, sep=" ",
row.names=F)

nmiss=rep(0,ncol(Jeu_LIS_NE))
Jeu_LIS_NE=as.data.frame(Jeu_LIS_NE)
for(i in 1:ncol(Jeu_LIS_NE))
{
  count=0
  for(j in 1:nrow(Jeu_LIS_NE))
  {
    if(is.na(Jeu_LIS_NE[j,i]))
    {
      count=count+1
    }
  }
  nmiss[i]=count
}

humain=c(0,0,0,0,0)
innov=c(1,0,0,0,0)
struc=c(1,1,0,0,0)
relat=c(1,0,1,0,0)
perform=c(0,1,1,1,0)

PME_C_path=rbind(humain,innov,struc,relat,perform)
colnames(PME_C_path)=rownames(PME_C_path)
PME_C_path
```

```
PME_C_blocks=list(c(1:8),9:14,15:20,21:30,31:33)

PME_C_modes=c("A","A","A","A","A")

#scal=c(rep("num",2),"ord",rep("num",6),"ord",rep
#("num",2),rep("ord",2),"num",rep("ord",3),"num","ord"
#          ,rep("num",2),"ord","num","num","ord",
#rep("num",3),"ord",rep("num",3))

#scal2=list(scal[1:8],scal[9:14],scal[15:20],scal[21:30],
scal[31:33])

library(plspm)
PME_C_pls=plspm(Jeu_LIS_NE[,-8],PME_C_path,PME_C_blocks,
modes=PME_C_modes)

plot(PME_C_pls)

summary(PME_C_pls)
library(xtable)
print(xtable(PME_C_pls$outer_model[,c(1,4)]))

JeuSyn=replicate(5,GenSyn2(1,Jeu_LIS_NE[,-8],c(13,14)))

# age
save(JeuSyn,file="JeuSyn_LIS_NE_age.Rdata")
# pas age
save(JeuSyn,file="JeuSyn_LIS_NE.Rdata")
PME_C_pls=plspm(JeuSyn[[1]],PME_C_path,PME_C_blocks,
modes=PME_C_modes)
PME_C_pls$outer_model[,c(1,4)]
PME_C_pls$path_coefs

PME_C_pls=plspm(JeuSyn[[2]],PME_C_path,PME_C_blocks,
modes=PME_C_modes)
PME_C_pls$outer_model[,c(1,4)]
PME_C_pls$path_coefs

PME_C_pls=plspm(JeuSyn[[3]],PME_C_path,PME_C_blocks,
```

```
modes=PME_C_modes)
PME_C_pls$outer_model[,c(1,4)]
PME_C_pls$path_coefs



PME_C_pls=plspm(JeuSyn[[4]],PME_C_path,PME_C_blocks,
modes=PME_C_modes)
PME_C_pls$outer_model[,c(1,4)]
PME_C_pls$path_coefs

PME_C_pls=plspm(JeuSyn[[5]],PME_C_path,PME_C_blocks,
modes=PME_C_modes)
PME_C_pls$outer_model[,c(1,4)]
PME_C_pls$path_coefs
```