Anonymisation de données sensibles

Mohamed Ali Batita — **Benjamin** Gras — **Anthony** Perez

Le contexte

In a crowd it's hard to spot him, but anonymity can cost.

La composition des groupes. Le projet devra être réalisé par groupe de 4. **Aucune dérogation ne sera accordée**, sauf évidemment si le nombre de personnes devant réaliser le projet n'est pas un multiple de 4.

Le rendu. Chaque groupe possèdera un identifiant unique, associé à un compte `git` déployé sur les serveurs de l'université. Ces derniers pourront faire office de rendu du projet, et être mis à jour régulièrement lors des différentes sessions de travail. Un dépôt Célène sera également ouvert pour le rendu, mais nous vous invitons à utiliser `git` en utilisant le dernier TP vu en Outils du développeur.

Le projet sera noté suivant trois critères distincts et de (quasi-)égale importance :

- un rapport entre 5 et 10 pages, contenant *a minima* les choix d'implémentation réalisés, une modélisation des classes de votre application et la répartition du travail au sein du groupe.
  Il n'est pas demandé d'explication du code JAVA : cette dernière sera à fournir dans la JavaDoc.
- le code source de votre projet, commenté au format JavaDoc (**il n'est pas demandé de l'inclure dans le rendu**).
- une soutenance, où la contribution de chaque membre du groupe sera analysée. De plus amples informations sur cette dernière seront données au cours du semestre.

L'évaluation

Le sujet. L'anonymisation des données est devenue depuis quelques années un enjeu majeur. L'augmentation des données collectées et le besoin d'analyser ces dernières tout en respectant la vie privée imposent d'utiliser des techniques d'anonymisation efficaces. Une illustration de cet intérêt est la récente mise en place du RGPD (**règlement général sur la protection des données**). L'objectif de ce projet sera d'analyser cette problématique et de produire une implémentation en langage `JAVA` d'un algorithme d'anonymisation de données.

Les données. Les données à traiter sont des bases de données, représentées sous la forme de tableurs `.ods`. Elles sont donc composées d'un ensemble d'**attributs**, qui peuvent être catégorisés en différents groupes : **Identifiants** (`I`), **Quasi-identifiants** (`Q`) et **Données Sensibles** (`DS`). Dans ce sujet, on suppose que les trois groupes sont disjoints. Les exemples suivants seront utilisés dans la suite du sujet (C.E. étant Consommation électrique) :

Nom	CP	Age	C.E.
Sue	18000	22	50
Pat	69000	27	70
Bob	18500	21	90
Bill	18510	20	60
Dan	69100	26	70
Sam	69300	28	75

Age	Sexe	CP	Maladie
25	H	53711	Grippe
25	F	53712	Hépatite
26	H	53711	Bronchite
27	H	53710	Fracture
27	F	53712	Angine
28	H	53711	Entorse

Les attributs de type `I` permettent d'identifier directement un individu, ceux de type `Q` demandent un autre jeu de données mais représentent un risque, et ceux de type `DS` sont ceux rendant l'anonymisation obligatoire. L'objectif du projet est donc d'implémenter un algorithme permettant de retirer les identifiants, et de *flouter* suffisamment les quasi-identifiants sans nuire à la qualité et l'utilité des données fournies.

Toutes les données fournies auront les identifiants en premières colonnes, suivis des quasi-identifiants, d'éventuels attributs non-sensibles et enfin d'une **unique** colonne de données sensibles.

Les données

La pseudonymisation

Personal responsibility.

La technique la plus simple permettant de retirer les **Identifiants** d'un jeu de données consiste à remplacer ces attributs par un unique attribut `Pseudo` (voir Figure 1 avec l'identifiant `SSN`).

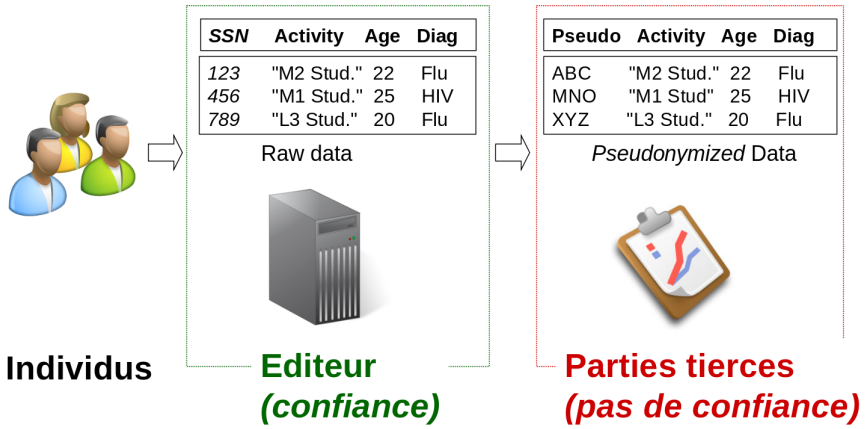


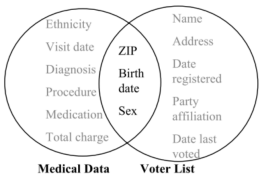
FIGURE 1 – Illustration de la notion de pseudonymisation [1].

1. Modéliser et implémenter les classes permettant de gérer les données (fournies au format `.ods`) et construisant un document **pseudonymisé**.

Pseudonymser

Inconvénients. Cette approche n'est pas suffisante pour anonymiser des données. Dans l'exemple ci-après, à partir d'un jeu de données pseudonymisé et d'un autre jeu de données, complet, il est possible de recouper les informations pour obtenir les données sensibles du jeu de données initial.

Prosecutor risk



A partir de données médicales anonmysées et d'une liste d'électeurs publique, Latanya SWEENEY [2] a réussi à obtenir les informations médicales du gouverneur **Weld**. L'ensemble d'attributs ayant permis ce recoupement (*{ZIP, Birth date, Sex}* sur l'exemple) est donc un **quasi-identifiant**.

K-Anonymisation. L'exemple précédent permet de préciser une technique d'anonymisation : les identifiants **doivent** être pseudonymisés [1], et les **quasi-identifiants doivent** être *floutés*. En particulier, il faut s'assurer que dans le jeu de données anonymisé, chaque tuple distinct du quasi-identifiant est associé à **au moins** K données sensibles différentes dans la base de données : on parle alors de **K-anonymisation**.

1. Par souci de lisibilité, ils seront simplement retirés dans tous les exemples du sujet. **Ne pas oublier de toujours pseudonymiser les tables.**

3-anonymisation de la consommation électrique

La première table est correcte, la seconde ne l'est pas : certains tuples (*e.g.* `(Cher, [20-22])`) ne sont pas associés à suffisamment de données sensibles.

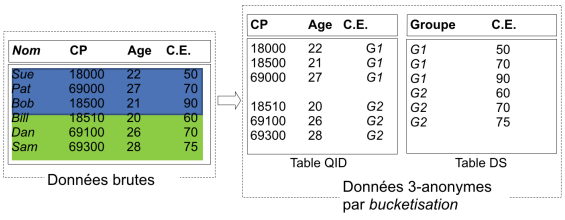
CP	Age	C.E.
Cher	[20-24]	50
Rhône	[25-29]	70
Cher	[20-24]	90
Cher	[20-24]	60
Rhône	[25-29]	70
Rhône	[25-29]	75

CP	Age	C.E.
Cher	[20-24]	50
Rhône	[25-29]	70
Cher	[20-24]	90
Cher	[20-22]	60
Rhône	[25-29]	70
Rhône	[25-29]	75

La technique de *bucketisation*

I paid five bucks for my Bucket T.

Une première technique simple pour K-anonymiser est de grouper le jeu de données à anonymiser en ensembles de taille k, et de représenter le jeu de données anonymisé sous deux tables : `QID` contenant les attributs du quasi-identifiant et leur groupe, et `DS` contenant toutes les paires `(groupes, donnée sensible)` construites.



Dans cet exemple, les données sont 3-anonymes : à chaque quasi-identifiant (représenté par les attributs *{CP, Age}* et regroupés en deux ensembles G_1 et G_2) correspondent trois données sensibles. Par exemple, le quasi-identifiant *{18000, 22}* étant dans le groupe G_1, ses données sensibles associées sont *{50, 70, 90}*.

2. Modéliser et implémenter les classes permettant d'appliquer la technique de bucketisation sur un jeu de données au format `.ods`. La sortie devra consister en deux fichiers `.ods`, respectivement pour les tables `QID` et `DS`. **Le code devra vérifier le bon formatage du jeu de données.**

Bucketiser

Un inconvénient de cette méthode est que si un *bucket* de taille k contient toujours la même donnée sensible, il peut être possible de savoir de manière sûre les données sensibles des quasi-identifiants associés. Afin d'éviter cela, on introduit la notion de *l-diversité* : chaque quasi-identifiant doit apparaître au moins k fois dans la base **et être associé à au moins l valeurs différentes.**

Table brute ; 4-anonyme sans diversité ; 4-anonyme 3-diverse

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	13053	28	Russian	Heart Disease
2	13068	29	American	Heart Disease
3	13068	21	Japanese	Viral Infection
4	13053	23	American	Viral Infection
5	14853	50	Indian	Cancer
6	14853	55	Russian	Heart Disease
7	14850	47	American	Viral Infection
8	14850	49	American	Viral Infection
9	13053	31	American	Cancer
10	13053	37	Indian	Cancer
11	13068	36	Japanese	Cancer
12	13068	35	American	Cancer

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	130**	< 30	*	Heart Disease
2	130**	< 30	*	Heart Disease
3	130**	< 30	*	Viral Infection
4	130**	< 30	*	Viral Infection
5	1485*	≥ 40	*	Cancer
6	1485*	≥ 40	*	Heart Disease
7	1485*	≥ 40	*	Viral Infection
8	1485*	≥ 40	*	Viral Infection
9	130**	3*	*	Cancer
10	130**	3*	*	Cancer
11	130**	3*	*	Cancer
12	130**	3*	*	Cancer

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	1305*	≤ 40	*	Heart Disease
4	1305*	≤ 40	*	Viral Infection
9	1305*	≤ 40	*	Cancer
10	1305*	≤ 40	*	Cancer
5	1485*	> 40	*	Cancer
6	1485*	> 40	*	Heart Disease
7	1485*	> 40	*	Viral Infection
8	1485*	> 40	*	Viral Infection
2	1306*	≤ 40	*	Heart Disease
3	1306*	≤ 40	*	Viral Infection
11	1306*	≤ 40	*	Cancer
12	1306*	≤ 40	*	Cancer

3. Définir une méthode permettant de vérifier si une base de données k-anonymisée est l-diverse. Il n'est pas demandé d'**implémenter** une *bucketisation* respectant cette propriété, simplement de vérifier si elle est respectée.

l-diversité

Un algorithme plus efficace

I'm an alpha I divide and conquer.

L'un des autres inconvénients majeur de cette méthode est l'utilité des données obtenues : comment les analyser ? Une manière de remédier à ce problème serait de ne plus regrouper les entrées du jeu de données de manière aléatoire, mais en utilisant une hiérarchie pré-définie, appelée **arbre de généralisation** (voir Figure 2).

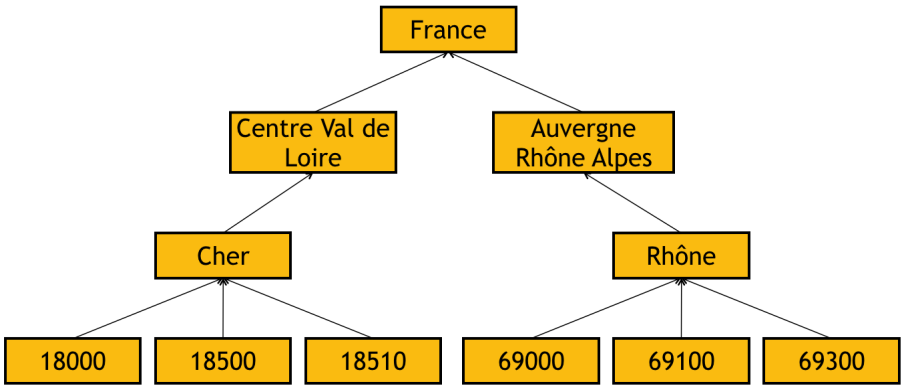


FIGURE 2 – Un exemple d'arbre de généralisation pour l'attribut *CP* [1].

L'exemple des données 3-anonymes présenté en début de sujet est obtenu via cette technique. Il n'est cependant pas nécessaire de gérer explicitement des arbres pour arriver à ce niveau d'anonymisation. L'**Algorithme 1** est une implémentation permettant de réaliser cette généralisation.

Les codes postaux de l'exemple précédent seraient donc généralisés en utilisant les intervalles de valeurs $[18000 - 18510]$ et $[69000 - 69300]$. Par souci de compréhension, l'explication qui suit est donnée pour des quasi-identifiants de taille (*dimension*) 2, mais **doit être implémentée en dimension** d.

> Pour simplifier, les attributs considérés comme **quasi-identifiant** seront **tous des valeurs numériques**.

Soient A_1 et A_2 les deux attributs composant le **quasi-identifiant**. Chaque valeur prise par ces attributs peut être vue comme un point dans le plan d'abscisse A_1 et d'ordonnée A_2. Le principe de l'Algorithme 1 de K-anonymisation est le suivant : tant qu'il est possible de **séparer** le plan en différentes régions contenant toutes au moins K points, l'algorithme continue. Dès que cette séparation n'est plus possible, l'ensemble des régions ainsi calculé est retourné. La généralisation peut être déduite en utilisant une correspondance donnant, pour chaque attribut du **quasi-identifiant**, l'intervalle de valeurs le représentant. Deux versions existent pour cet algorithme :

- **Unidimensionnelle** : l'attribut pour lequel une séparation est cherchée est le même à chaque itération.
- **Multi-dimensionnelle** : un choix d'attribut pour lequel chercher une séparation est nécessaire, et (demandant l'implémentation d'une heuristique (Algorithme 1))

Algorithme 1: Anonyme($\mathcal{D}$)

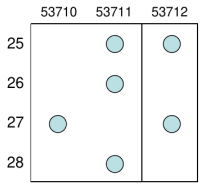
Entrée: Un jeu de données $\mathcal{D}$
Sortie: L'ensemble des régions multi-dimensionnelles calculées

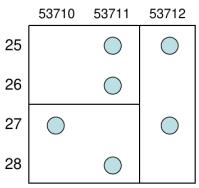
si *aucune séparation n'est possible* **alors**
  retourner les régions calculées;
$att \leftarrow$ `choisir_attribut`($\mathcal{D}$);
$fs \leftarrow$ `frequences`($\mathcal{D}$, att);
$med \leftarrow$ `mediane`(fs);
$lhs \leftarrow \{t \in \mathcal{D} : t.att \leqslant med\}$;
$rhs \leftarrow \{t \in \mathcal{D} : t.att > med\}$;
Retourner `Anonyme`(lhs) $\cup$ `Anonyme`(rhs);

La table présentée ci-dessous est le résultat de l'application de l'Algorithme 1 multi-dimensionnel sur la table présentée en début de sujet. Dans les Figures, les coupes sont réalisées sur le **quasi-identifiant** `CP` uniquement, puis en rajoutant `Age`.

Données 2-anonyme

Age	Sexe	CP	Maladie
25-26	H	53711	Grippe
25-27	F	53712	Hépatite
25-26	H	53711	Bronchite
27-28	H	53710-53711	Fracture
25-27	F	53712	Angine
27-28	H	53710-53711	Entorse





4. Modéliser et implémenter les classes permettant d'appliquer l'Algorithme 1 **unidimensionnel** sur un jeu de données au format `.ods`. **Le code devra vérifier le bon formatage du jeu de données.**
5. Modéliser et implémenter les classes permettant d'appliquer l'Algorithme 1 **multi-dimensionnel** sur un jeu de données au format `.ods`. **Le code devra vérifier le bon formatage du jeu de données.**

Implémentations de l'Algorithme 1

Cet algorithme contient plusieurs parties, qu'il sera important de décomposer pour obtenir un code lisible et facilement modulable.

Interface Homme-Machine

Man Machine, superhuman being.

L'objectif de cette dernière partie est de créer une interface graphique **simple** permettant de demander à l'utilisateur de choisir un fichier (au format `.ods` **exclusivement**) à anonymiser, en appliquant soit la méthode de *bucketisation* soit l'Algorithme 1. Une fois le fichier **valide** et la méthode choisis, votre interface graphique demandera à l'utilisateur le ou les noms à donner pour le fichier de sortie, qui sera enregistré au format `.ods`.

Quelques pointeurs.

La conception d'interfaces graphiques en `JAVA` peut se faire au moyen de plusieurs librairies. Si aucune n'est imposée, les suivantes sont conseillées :
- `Java` : `Swing` et `awt`, voir par exemple ici ou là.

6. Rajouter l'utilisation d'une interface au projet. Cette partie peut être implémentée même sans un projet totalement fonctionnel. Par exemple, si seule la méthode *bucketisation* fonctionne, l'interface graphique pourra ne proposer que cette dernière.

7. En plus de l'enregistrement, les tables anonymisées pourront être directement affichées dans l'interface graphique.

Extension.

Références

[1] Benjamin Nguyen. Techniques d'anonymisation classiques. Séminaire du LIFO, Orléans, Novembre 2019.

[2] Latanya Sweeney. k-anonymity : A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05) :557–570, 2002.