Appariements de données individuelles : concepts, méthodes, conseils

Document de travail

N° M2023-03 – Juin 2023

Lucas MALHERBE

Insee
Mesurer pour comprendre

INSTITUT NATIONAL DE LA STATISTIQUE ET DES ÉTUDES ÉCONOMIQUES
Série des documents de travail « Méthodologie Statistique »
de la Direction de la Méthodologie et de la Coordination Statistique et Internationale

M 2023/03

Appariements de données individuelles : concepts, méthodes, conseils

LUCAS MALHERBE

Insee

Juin 2023

Direction de la méthodologie et de la coordination statistique et internationale
Département des Méthodes Statistiques -Timbre L001 -
88 Avenue Verdier - CS 70058 - 92541 Montrouge Cedex - France -
Tél. : 33 (1) 87 69 55 00 - E-mail : DG75-L001@insee.fr - Site Web Insee : http://www.insee.fr

Appariements de données individuelles : concepts, méthodes, conseils

Lucas Malherbe*

Résumé

Un appariement consiste à rapprocher deux bases de données d'origine distincte partageant des unités statistiques communes mais contenant des informations différentes. Cette opération permet d'accroître l'information disponible dans les deux bases. Un cas classique pour le statisticien est l'appariement de fichiers de personnes. Les enjeux sont multiples : enrichissement de données d'enquête par des sources administratives, repérage de doublons, constitution de panels...

La tâche est aisée lorsque les deux sources de données disposent d'un identifiant unique commun pour tous les individus, mais ce cas n'est pas fréquent. En l'absence d'un tel identifiant, il est nécessaire de mobiliser les champs caractérisant l'identité d'une personne (état civil, adresse, etc.) pour repérer les individus communs dans les deux bases. Ces traits d'identité pouvant être entachés d'erreurs, certains individus doivent être appariés malgré des informations légèrement différentes sur les champs servant à l'identification.

Ce document constitue une introduction pratique aux appariements de données individuelles sur traits d'identité. Après avoir introduit le vocabulaire et les notions clés, il expose et illustre les enjeux et les méthodes associés aux différentes étapes d'un appariement. Un ensemble d'outils d'appariement *open source* sont présentés, et leurs avantages et inconvénients sont discutés. Des conseils pratiques sont également proposés afin de choisir l'approche la plus adaptée au cas rencontré, paramétrer au mieux l'appariement et éviter de tomber dans certains écueils.

Un dépôt de code évolutif contenant des exemples d'appariements ainsi que des ressources d'autoformation est disponible à l'adresse suivante : `https://github.com/InseeFrLab/appariement`

Mots clés : appariement, dédoublonnage, données individuelles, Fellegi-Sunter

Classification JEL : C50, C81

* Insee

Abstract

Record linkage involves matching two databases from distinct sources pertaining to common statistical units but containing different information. This operation increases the information available in the two databases. Statisticians most often face the case of linking people. The challenges are manifold: enriching survey data with administrative data sources, identifying duplicate records, creating panels, and more.

The task is straighforward when both data sources possess a common unique identifier for all individuals, but this is not often the case. In the absence of such an identifier, it is necessary to mobilise personal data (first and last names, date of birth, address, etc.) to identify common individuals in both databases. The data may be prone to errors, therefore some individuals need to be matched despite slight variations in the identification fields.

This document serves as a practical introduction to record linkage using personal data. After introducing the vocabulary and key concepts, it discusses and illustrates the challenges and methods associated with the different stages of record linkage. A set of open-source record linkage tools is presented, along with a discussion of their advantages and disadvantages. Practical advice is also provided to help choose the most suitable approach for a given use case, optimise the linkage parameters, and avoid common pitfalls.

An evolving code repository containing examples of record linkage and self-training resources is available at the following address: `https://github.com/InseeFrLab/appariement`

Keywords : record linkage, data matching, individual data, Fellegi-Sunter

JEL Classification : C50, C81

Table des matières

Introduction

Avec de plus en plus de sources de données individuelles existantes, l'information disponible pour le statisticien public devient conséquente. Cependant, toutes ces sources ne sont pas pensées et construites de la même façon, elles ne poursuivent pas le même objectif et ne couvrent pas la même population. C'est dans ce contexte que l'appariement de données individuelles prend tout son sens. Il consiste à rapprocher deux bases de données d'origines distinctes et contenant des informations différentes mais portant sur des unités statistiques communes.

La tâche est aisée si les deux bases disposent d'un identifiant unique commun pour tous les enregistrements, comme un numéro de sécurité sociale ou un numéro d'identification statistique. Ce document porte sur l'autre cas, lorsqu'un tel identifiant n'est pas disponible ou que celui-ci n'est pas de bonne qualité. L'appariement se fait alors sur une combinaison d'autres champs (état civil, adresse, etc.). Toute la difficulté réside dans le fait que ces champs eux-mêmes peuvent être entachés d'erreurs. Il n'est donc pas possible d'apparier uniquement les individus dont les informations identifiantes sont strictement identiques. Il s'agit ainsi de trouver une méthode autorisant de légères variations, sans toutefois être trop permissive pour éviter d'apparier des individus à tort.

Les enjeux sont multiples, allant de la construction de répertoires au repérage de doublons en passant par l'enrichissement de données d'enquêtes.

Il faut dans un premier distinguer les appariements à visée administrative de ceux à visée statistique. Les premiers se placent à un niveau plus fin et le résultat de l'appariement a un impact direct sur l'individu apparié dans le cadre d'un processus administratif, comme le calcul d'une retraite ou la gestion d'une demande d'aide de l'État. La fréquence des erreurs doit dans ces cas-là rester extrêmement faible, ce qui influence fortement la façon dont est conduit l'appariement. Les appariements à visée statistique s'intéressent moins aux individus eux-mêmes qu'à des agrégats. L'objectif est souvent d'enrichir les données à disposition. Les erreurs sont permises tant qu'elles n'ont pas d'impact trop grand sur les distributions sous-jacentes et qu'elles n'altèrent pas la validité des résultats de l'étude statistique qui en découle. Ce document sera principalement consacré aux appariements à visée statistique.

Dans le contexte de la statistique publique, les appariements sont utilisés depuis près d'un siècle mais ils prennent aujourd'hui une place de plus en plus importante. Les besoins concernent en particulier la constitution de panels, l'utilisation de données administratives en complément ou à la place de données d'enquêtes ainsi que l'appariement de données administratives entre elles en vue de constituer des bases de données présentant un fort taux de couverture de la population d'intérêt. Ce dernier cas peut être illustré par la problématique du programme de Répertoires Statistiques d'Individus et de Logements (Résil) qui vise à construire un système

de répertoires statistiques anonymisés d'individus, de ménages et de locaux d'habitation, durable et évolutif, mis à jour à partir de sources administratives diverses. Les appariements occuperont ainsi une place centrale au sein de ce programme, à la fois pour la construction des répertoires mais aussi dans le cadre du service d'enrichissement de données qui sera proposé à terme.

L'appariement est donc central dans l'exploitation de sources administratives. Les avantages sont multiples, avec en premier lieu une réduction de la charge de réponse des enquêtés ainsi que du coût de réalisation des enquêtes (l'information provenant de certaines d'entre elles pouvant être déduite du rapprochement de sources administratives) mais aussi dans le cas de sources exhaustives une meilleure couverture de la population qu'avec des enquêtes.

Les bénéfices peuvent également se manifester sur la qualité des données. Par exemple, dans le cas de variables complexes comme le revenu disponible ou même le salaire, les données obtenues en appariant avec des sources administratives sont souvent de meilleure qualité que celles obtenues par voie d'enquête, car les concepts sont relativement difficiles à appréhender pour les enquêtés.

L'appariement de données individuelles constitue ainsi une problématique récurrente et cruciale pour le statisticien. Pourtant, il n'existe aucun mode opératoire de référence en la matière.

Une partie non négligeable dépend des données à apparier, mais la mutualisation reste possible. Celle-ci passe notamment par le partage de connaissances. Ce document constitue donc avant tout une introduction pratique aux appariements pour le statisticien, pour lui permettre de trouver plus facilement et plus rapidement la solution adaptée à sa situation et à ses données.

Les deux premières parties constituent une bonne introduction au domaine des appariements de données individuelles pour un lecteur découvrant le sujet. Le lecteur averti pourra s'attarder sur les deux autres parties, qui ont pour but de l'aiguiller dans le choix des méthodes et des outils ainsi que dans leur mise en oeuvre.

La première partie de ce document vise à poser les bases du sujet en définissant des termes clés. La deuxième partie est consacrée à une présentation des enjeux et méthodes associées aux différentes étapes d'un appariement de données individuelles. La troisième partie consiste en une étude comparative d'une sélection d'outils *open source*. La quatrième et dernière partie propose un ensemble d'enseignements pratiques pour réussir un appariement.

Ce document est accompagné d'un dépôt de code évolutif contenant des exemples d'appariements ainsi que des ressources d'autoformation. Ce dépôt est accessible via le lien suivant : `https://github.com/InseeFrLab/appariement`

1 Concepts et définitions

Le domaine des appariements fait intervenir des termes spécifiques qu'il convient de définir.

Les individus

Ce document traite avant tout des appariements de personnes physiques, ce qui constitue le cas le plus fréquent pour le statisticien. Une ligne d'un fichier à apparier sera donc désignée par le terme "**individu**". La majorité du document s'applique cependant à des appariements concernant d'autres unités statistiques, tant qu'il est possible de procéder à des comparaisons sur différents champs.

Les paires

La matière première d'un appariement de données individuelles est une **paire** de deux individus, pour lesquels on tente de déterminer s'ils désignent ou non la même personne. L'ensemble des paires à traiter est le **produit cartésien** des deux fichiers à apparier, c'est-à-dire l'ensemble des couples possibles faisant intervenir un individu du premier fichier et un individu du second fichier. Pour deux fichiers de taille n, le produit cartésien est de taille n^2.

Une paire peut être soit une **paire d'individus identiques**, soit une **paire d'individus différents**, et ce sont ces dénominations qui seront utilisées dans tout le document pour désigner le **statut réel** d'une paire (à défaut de disposer d'un mot dédié comme *match* en anglais).

Le but d'un travail d'appariement est de construire un **modèle** qui tente de déterminer le statut de chaque paire. Une paire est dite **liée** si le modèle considère que les individus la constituant sont les mêmes, dans le cas contraire c'est une **paire non liée**. Les termes « lié » et « non lié » font donc référence uniquement à la décision du modèle, sans rapport avec le statut réel de chaque paire.

Par ailleurs, une **paire annotée** est une paire dont le statut réel est connu ou supposé connu, par exemple suite à un **examen manuel**. L'examen manuel désigne le fait qu'un humain décide du statut d'une paire suite à une évaluation visuelle des similarités et différences entre les deux individus.

Variables

Un appariement mobilise des variables présentes dans les deux fichiers, qui sont comparées pour lier ou non les paires. Ces variables sont appelées **variables identifiantes**, **champs identifiants** ou encore **variables d'appariement**.

Pour des données individuelles, il s'agit le plus souvent de **traits d'identité**, comme le nom, le prénom, le sexe ainsi que la date et la commune de naissance.

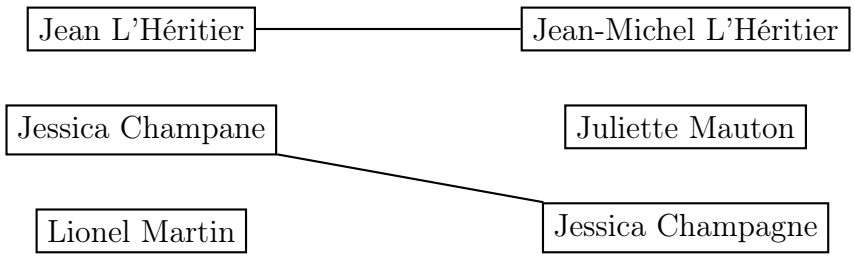
Indexation

Le terme « **indexation** » est employé dans différents domaines, notamment dans le vocabulaire des bases de données. Il possède cependant un sens propre dans le domaine des appariements.

L'indexation correspond à l'étape de réduction du nombre de paires, c'est-à-dire au processus de filtrage des paires qui seront comparées de façon plus approfondie. Cette étape, décrite dans la section 2.3, est le plus souvent indispensable pour limiter les ressources informatiques nécessaires à l'appariement.

Appariement *one-to-one*, *many-to-one* ou *many-to-many*

Le type de relations qui existe entre les deux fichiers à apparier est une caractéristique à prendre en compte. Il existe trois types de relations : *one-to-one*, *many-to-one* ou *many-to-many*.

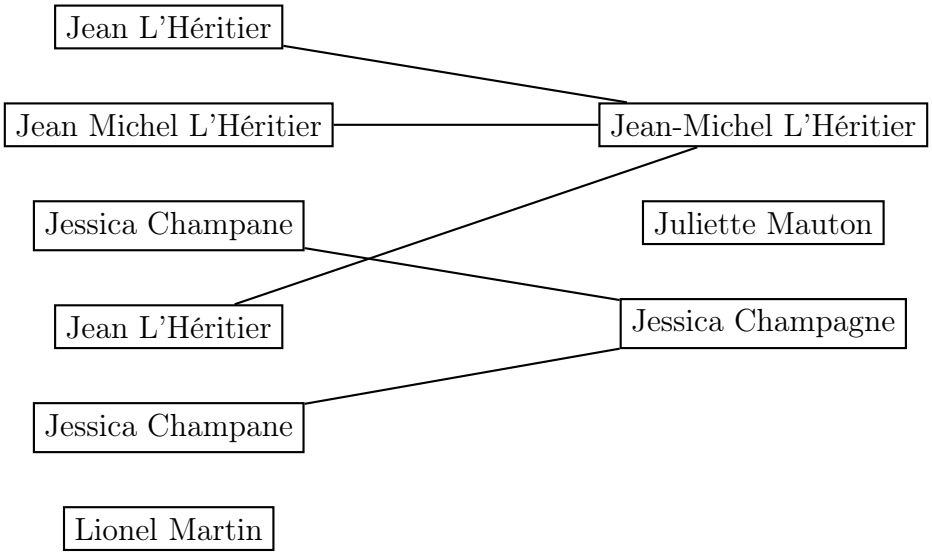


GRAPHIQUE **1 – Exemple d'appariement *one-to-one***

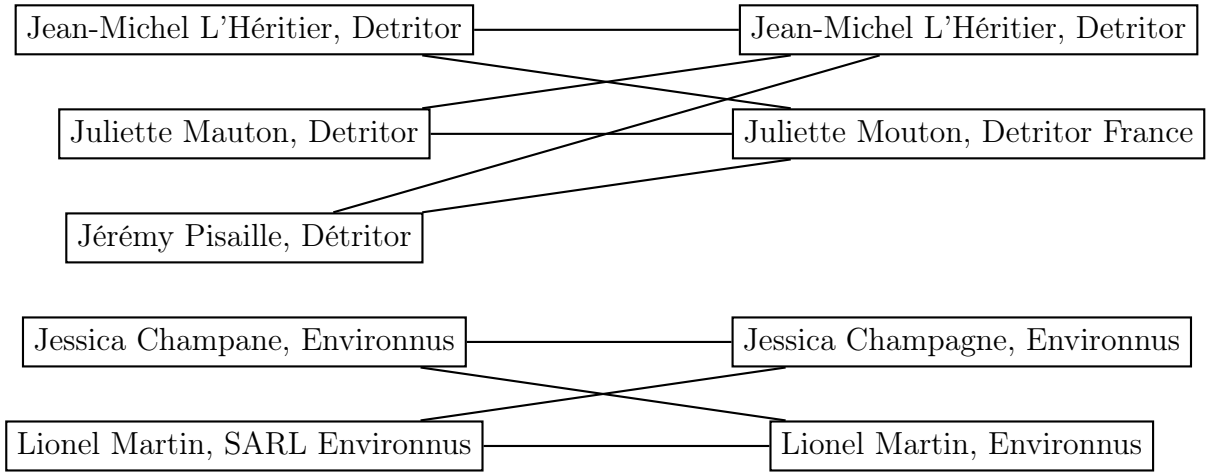
Le cas le plus fréquent en statistique publique est celui de l'appariement ***one-to-one***. Il se présente notamment lorsqu'une source administrative vient enrichir des données d'enquête. Chacune des deux sources est supposée exempte de doublons et on s'interdit donc d'apparier deux individus d'un fichier à un seul individu de l'autre fichier.

Dans un appariement de type ***many-to-one***, plusieurs individus d'un fichier peuvent être appariés à un seul individu de l'autre fichier, mais ce n'est pas le cas en sens inverse. Cela se produit lorsque l'un des deux fichiers contient des doublons mais pas l'autre. C'est un cas typique lorsque les données utilisées s'apparentent à des transactions. Par exemple, l'appariement de données de santé couvrant des actes de soins avec une source fiscale est un appariement *many-to-one*. Un même individu peut en effet avoir bénéficié de plusieurs actes de soins mais il est supposé n'apparaître qu'une seule fois dans les données fiscales.

Le cas ***many-to-many*** est plus rare dans les appariements de données individuelles. À titre d'exemple, à partir de fichiers de salariés, apparier tous les individus travaillant dans la même entreprise est une tâche *many-to-many*.



GRAPHIQUE **2 – Exemple d'appariement *many-to-one***



GRAPHIQUE **3 – Exemple d'appariement *many-to-many* avec données nominatives et noms d'entreprise**

Appariement exact et appariement flou

L'appariement le plus simple est un **appariement exact**, dans lequel on exige que toutes les variables identifiantes soient identiques pour lier deux individus. Si les fichiers à apparier contiennent des erreurs sur les champs identifiants, comme c'est souvent le cas, l'appariement exact manque des paires d'individus identiques et il faut plutôt procéder à un **appariement flou** pour tenter de les repérer.

L'appariement flou consiste à autoriser des erreurs sur les variables identifiantes et à les mesurer pour déterminer s'il faut tout de même lier la paire ou non. La mesure des erreurs se fait notamment via des calculs de distance, qui sont détaillés en section 2.4.

Ainsi, si l'un des fichiers à apparier contient l'individu « Jean L'Héritier » tandis que « Jean-Michel L'Héritier » est présent dans l'autre, un appariement exact conduirait à rejeter cette paire, mais un appariement flou reconnaîtrait la proximité entre ces deux noms et validerait peut-être la paire grâce aux autres informations disponibles, comme une correspondance sur la date de naissance par exemple.

2 Étapes d'un appariement

Cette partie vise à présenter les principales étapes d'un processus d'appariement. Elle aborde les méthodes les plus communes pour chaque étape. Le lecteur intéressé pourra trouver plus de détails dans la littérature sur les appariements, en particulier dans Christen (2012).

2.1 Vue d'ensemble

La plupart des appariements suivent une structure commune. Les différentes étapes présentées ici de manière succincte sont détaillées aux paragraphes suivants.

Préparation et mise en qualité des données
Cette première phase consiste en un examen de la qualité des données et une succession d'étapes de nettoyage et de normalisation pour préparer les deux bases à la comparaison. Il s'agit de se débarrasser des fioritures, qui n'apportent pas particulièrement d'information mais nuisent à l'appariement, comme les accents ou la capitalisation des lettres dans un nom. Le résultat d'un appariement est intimement lié à la qualité des données utilisées. Porter un soin particulier à cette étape est un investissement indispensable pour réaliser un bon appariement.

Indexation : réduction du nombre de paires
L'idéal pour effectuer un appariement entre deux fichiers serait de comparer l'ensemble des paires du produit cartésien. Malheureusement, il est souvent impossible de procéder de la sorte en raison du temps de calcul associé aux comparaisons de champs textuels. L'objectif de l'indexation est alors de réduire la dimension en ne considérant que les paires qui ont une chance raisonnable de correspondre à la même entité. La méthode la plus répandue est celle du blocage.

Comparaison des champs
Cette étape consiste à calculer des mesures de similarité pour chaque champ intervenant dans l'appariement et pour chaque paire potentielle conservée à l'issue de l'indexation. De nombreuses mesures existent pour chaque type de champ (texte, nombre, date, etc.) et leur choix est étroitement lié à celui de la méthode de classification choisie.

Comparaison des paires : classification
Les mesures de similarité calculées à l'étape précédente sont mobilisées pour classer les paires en deux ou trois catégories : les paires liées, les paires non liées, et parfois des paires laissées en suspens pour un éventuel examen manuel. De nombreuses méthodes existent et se rangent en général en deux groupes, les méthodes

déterministes et les méthodes probabilistes. Ces dernières se distinguent par l'utilisation d'une probabilité : celle que les deux éléments d'une paire correspondent à un même individu. Aucune méthode ne s'impose réellement face aux autres, le choix doit être guidé par le type de données à apparier ainsi que les objectifs poursuivis.

Résolution des conflits

Cette étape n'a lieu que s'il existe des restrictions sur les combinaisons de paires qui peuvent être liées, comme dans le cas d'un appariement *one-to-one* ou *many-to-one* (cf. section 1 pour la définition de ces termes). Des conflits peuvent apparaître, comme le fait de lier deux individus différents du premier fichier à un même individu du second fichier. Il est alors nécessaire de trancher, en s'appuyant sur les distances ou les probabilités issues des étapes précédentes.

Évaluation de la qualité

Une fois l'appariement terminé, l'évaluation de sa qualité est primordiale, à la fois pour procéder à d'éventuels ajustements dans le processus (un appariement est un processus fortement itératif) et pour déterminer si son résultat satisfait les critères nécessaires à une utilisation dans une étude statistique par exemple. L'objectif est de rassembler autant d'informations que possible sur la qualité de l'appariement. La difficulté réside dans le fait que le vrai statut de chaque paire n'est en général pas connu. Il est nécessaire dans ce cas d'annoter manuellement un échantillon de paires pour être en mesure d'en déduire différents indicateurs.

2.2 Préparation et mise en qualité des données

Cette première phase consiste en un examen de la qualité des données et en une succession d'étapes de **nettoyage** et de **normalisation** pour préparer les deux bases à la comparaison.

Le point de départ consiste en une **expertise qualité** des variables qui serviront ensuite à l'appariement. Ce travail a pour but d'identifier tout ce qui peut gêner l'appariement, comme des valeurs manquantes, des modalités fréquemment erronées ou encore une sous-population présentant des données de moindre qualité.

Une fois les données explorées, plusieurs types de transformations sont en général effectuées.

Il s'agit d'abord de **se débarrasser du superflu**, qui n'apporte pas particulièrement d'information et nuit à l'appariement, comme les accents ou la capitalisation des lettres dans un nom. En effet, sur ce type de caractères, l'usage n'est pas toujours uniforme entre deux fichiers différents ou parfois au sein même d'un fichier.

Ensuite, il est important de s'assurer que les champs servant à l'appariement sont stockés dans le même format dans les deux bases. Il existe une multitude de formats différents pour les dates par exemple, et la comparaison n'a de sens que si le format est identique de chaque côté. Il faut également veiller à la question de la segmentation de l'information. Une date peut correspondre à trois champs (jour, mois et année) ou un seul regroupant toute l'information. De même pour une adresse avec le numéro, le nom de la voie, le code postal et la ville. En général, il est plus intéressant de segmenter au maximum mais ce n'est pas toujours possible. Dans tous les cas, il faut procéder aux transformations nécessaires (découpage de l'information, ou au contraire agrégation de plusieurs variables) pour que les deux bases contiennent bien les mêmes champs identifiants.

Enfin, procéder à des contrôles et corrections sur les données permet d'en améliorer la qualité générale. Certaines valeurs ou combinaisons de modalités sont impossibles et correspondent à des anomalies, comme un âge supérieur à 120 ans. L'analyse de ces anomalies permet parfois d'en comprendre le processus générateur et de le corriger. Dans d'autres cas, la bonne solution peut être de les supprimer du fichier à apparier pour éviter qu'elles polluent le résultat de l'appariement.

Arbitrage bruit / information

L'étape de normalisation est bénéfique et bien souvent nécessaire, à condition qu'elle ne soit pas effectuée de manière trop brutale. En effet, toute opération de normalisation réduit la variance intrinsèque du jeu de données traité et supprime de l'information. Le but ultime est de retirer toute l'information parasite (bruit) pour mieux faire ressortir l'information pertinente. Une faute de frappe dans un nom contient de l'information, mais elle détériore souvent l'appariement ; il vaut mieux s'en débarrasser lorsqu'il est possible de la détecter. *A contrario*, conserver uniquement le premier prénom lorsque plusieurs sont fournis permet certes de normaliser son jeu de données, mais les prénoms suivants contiennent de l'information qui peut être utile pour apparier certains individus, notamment en cas d'inversion ou d'omission de prénoms ; il est donc souvent pertinent de conserver cette information afin de l'utiliser à bon escient.

Cette étape est à penser en combinaison avec celle de comparaison des champs (section 2.4) : il est parfois plus pertinent de ne pas corriger ou supprimer les anomalies rencontrées, mais de les prendre en compte dans les fonctions de comparaison.

La préparation des données est une étape fastidieuse, mais l'expérience prouve que le résultat d'un appariement est intimement lié à la qualité des données utilisées. Des gains considérables peuvent être obtenus suite au nettoyage des deux bases à apparier et à la prise en compte de leurs spécificités.

Exemples de traitements de nettoyage et normalisation

- **Capitalisation des lettres** : Dupont → DUPONT
- **Suppression des caractères superflus** : JEAN-MICHEL L'HÉRITIER → JEANMICHEL LHERITIER
- **Suppression des mots vides (*stop words*)** : le, la, de, du, ce...
- **Prise en compte des variations d'écriture communes** : AV. → AVENUE
- **Contrôle et correction des anomalies** : âge > 120, ou négatif
- **Segmentation de l'information** : date de naissance = 13/01/1970 → jour = 13, mois = 01 et année = 1970
- **Passage d'un libellé à un code** : libellé de commune = NÎMES → code officiel géographique = 30189

2.3 Indexation : réduction du nombre de paires

La solution naturelle pour apparier deux fichiers consiste à comparer chaque paire du produit cartésien. Cependant, sa taille évolue comme le carré de la taille des fichiers. Même pour deux bases de taille moyenne, il devient informatiquement impossible de procéder à toutes les comparaisons en un temps raisonnable. Pour deux fichiers de 10 000 lignes, le produit cartésien représente déjà 100 millions de paires potentielles !

De plus, la proportion de paires d'individus identiques parmi le produit cartésien devient alors très faible. Cette écrasante majorité d'exemples négatifs biaise l'estimation dans l'étape de classification.

Il est de toute manière peu pertinent de faire toutes les comparaisons puisque la plupart des paires peuvent être écartées très facilement, tant les variables identifiantes sont différentes pour les deux individus.

L'idée de la phase d'**indexation** est alors de réduire la dimension du problème en n'effectuant des comparaisons précises que sur les paires qui ont une chance raisonnable de correspondre à un même individu. De nombreuses stratégies existent pour effectuer cette étape de filtrage. Deux des méthodes les plus communes sont présentées dans cette section : le **blocage** et l'approche du **voisinage trié**.

2.3.1 Le blocage

Le **blocage** est de loin la méthode la plus classique. L'un des champs identifiants est choisi comme **clé de blocage**, et autant de blocs que de valeurs distinctes de la clé de blocage sont créés. Chaque bloc contient les individus présentant une valeur particulière de la clé de blocage et seules les paires d'individus appartenant à un même bloc sont conservées comme paires potentielles.

Par exemple, si la clé de blocage est l'année de naissance, un individu du fichier A né en 1970 sera comparé à tous les individus du fichier B nés en 1970, et à aucun autre. Cela implique en particulier qu'en cas d'erreur sur l'année de naissance dans l'une des deux bases, l'individu en question ne pourra pas être apparié correctement. La clé de blocage doit donc être de très bonne qualité. Dans le cas contraire, l'étape d'indexation crée beaucoup de faux négatifs, c'est-à-dire de paires qui ont été éliminées alors qu'elles auraient dû être liées.

En plus d'être de très bonne qualité, la clé de blocage idéale doit également posséder un grand nombre de modalités différentes. Dans le cas contraire, elle n'est pas suffisamment intéressante d'un point de vue informatique puisqu'elle ne partage pas bien la population. Procéder à un blocage sur le sexe peut être un bon début, mais cela n'est pas suffisant si les fichiers sont trop volumineux.

Enfin, il faut aussi être vigilant à la distribution des modalités. Bloquer sur le code commune par exemple conduit à des blocs de taille très faible pour les villages mais aussi à des blocs colossaux pour les plus grandes villes.

2.3.2 Variations autour du blocage

Pour dépasser les limites du blocage classique, de nombreuses variations existent.

Pour autoriser les erreurs sur une clé de blocage, il est possible d'en utiliser plusieurs. Par exemple, en bloquant d'abord sur l'année de naissance pour obtenir un premier ensemble de paires, ensuite sur le code postal pour en obtenir un second, et en conservant l'union de ces deux ensembles comme paires potentielles.

Si un champ identifiant présente trop peu de modalités différentes ou ne permet pas assez de réduire la dimension par lui-même, une clé de blocage peut-être construite en combinant plusieurs champs. Par exemple, une paire n'est conservée que si à la fois le sexe et le département de résidence sont les mêmes.

Il peut également être judicieux de bloquer en utilisant une distance (comme la distance de Levenshtein, détaillée dans la section 2.4) en complément ou à la place d'une comparaison exacte. Imposer une année de naissance identique ainsi qu'un seuil sur la distance de Levenshtein entre les noms de famille permet de ne conserver que des paires d'individus très similaires, au prix d'un temps de calcul plus important, les calculs de distance étant bien plus coûteux que les comparaisons exactes.

Par ailleurs, la transcription phonétique de certains champs identifiants peut être utilisée comme clé de blocage. Il s'agit de transformer un nom ou une adresse en un code phonétique puis de ne conserver que les paires d'individus présentant un code, et donc une prononciation, identiques ou proches. Parmi les algorithmes phonétiques classiques se trouvent le Soundex (Russel (1918), la version originale concerne la langue anglaise, mais des versions existent pour le français) et le Double Metaphone (Philips (2000)). Le Soundex a été utilisé sur les recensements américains il y a près d'un siècle par le Census Bureau, la problématique de l'appariement n'est donc pas nouvelle. La transcription phonétique pour les appariements est bien adaptée à certains processus générateurs d'erreurs, par exemple dans le cas où un individu transmet oralement ses informations d'identité à un opérateur qui les enregistre dans la base. En effet, ce mode de fonctionnement crée naturellement des erreurs d'orthographe sur les noms et prénoms qui n'en modifient pas la prononciation, comme le fait d'écrire « Thibault » au lieu de « Thibaud ». L'approche phonétique est moins pertinente dans d'autres cas, comme pour des données issues d'un questionnaire auto-administré sur Internet, où une partie des erreurs sera due à des fautes de frappe pouvant assez fréquemment changer la prononciation.

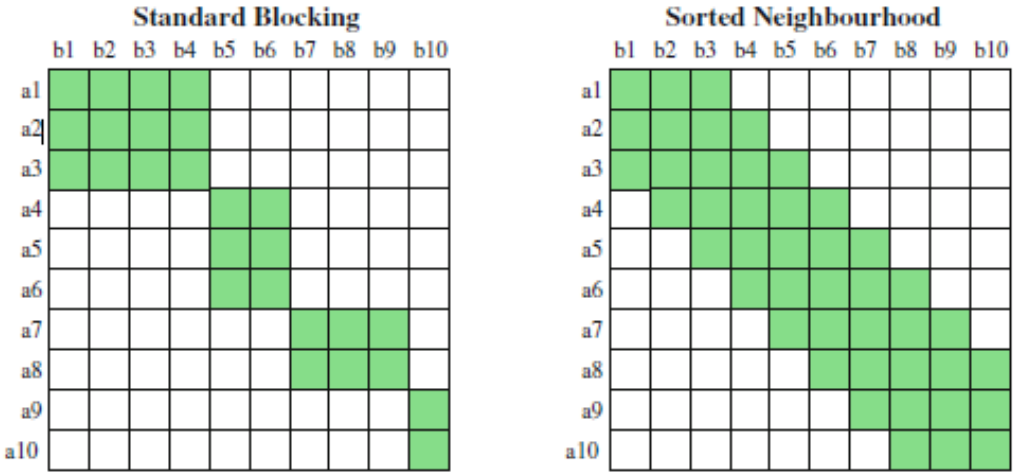
2.3.3 L'approche du voisinage trié

Au-delà du blocage, l'approche du voisinage trié (*sorted neighbourhood* en anglais) offre une solution différente au problème d'indexation. Elle consiste à trier les deux bases selon une même clé de tri et à conserver les paires pour lesquelles cette clé est proche. La clé de tri est construite à partir d'un champ identifiant ou plusieurs, transformés ou non.

La différence avec le blocage réside dans la façon de créer les blocs et est illustrée par le graphique 4. Le blocage classique crée des blocs disjoints de taille variable, et autant de blocs que de modalités de la clé de blocage. L'approche du voisinage trié crée des blocs de taille fixe pour chaque individu et les blocs se recouvrent.

Une fois les deux bases triées, une fenêtre glissante de taille fixe et centrée sur la valeur de la clé de tri permet de sélectionner les paires.

Comme pour le blocage, le choix de la clé de tri est décisif. L'une des limites de cette approche est la très grande importance qu'elle donne aux premiers caractères de la clé de tri. Ainsi pour ne pas se reposer trop sur un seul champ identifiant, il est possible de générer plusieurs ensembles de paires avec différentes clés de tri et de conserver l'union de toutes les paires obtenues.



GRAPHIQUE 4 – **Illustration du blocage et de l'approche du voisinage trié**

Source : Christen (2012)
À gauche, avec le blocage standard, chaque rectangle correspond à un ensemble d'individus partageant une valeur commune de la clé de blocage. Chaque bloc peut être de taille différente.
À droite, avec l'approche du voisinage trié, chaque individu est comparé avec un nombre fixe d'individus de l'autre fichier, et les individus comparés sont ceux présentant des valeurs proches (mais pas nécessairement exactes) de la clé de tri.

2.3.4 En résumé

Indispensable lorsque la taille des fichiers dépasse les milliers de lignes, la phase d'indexation est une affaire de compromis. Garder un grand nombre de paires permet d'être conservateur mais ne résout pas le problème des ressources informatiques. *A contrario*, une indexation trop drastique conduit à la création de faux négatifs : on manque d'emblée des paires d'individus identiques. Il y a donc un équilibre à trouver entre qualité de l'appariement et réduction de la dimension du problème.

Exemples de stratégies d'indexation

— Blocage sur **le département OU l'année de naissance** (l'union des deux ensembles de paires est conservée)

— Variante pour réduire plus fortement la dimension : blocage sur **le code postal OU la date de naissance**

— Blocage flou, avec calculs de distance : **[même année de naissance OU code postal] ET [distance de Levenshtein sur prénom OU nom inférieure à 3]**

2.4 Comparaison des champs

L'étape d'indexation ayant permis d'éliminer un certain nombre de paires, une attention particulière doit être portée aux paires restantes. Une solution naturelle est d'effectuer des comparaisons exactes sur tous les champs identifiants. Cependant cette approche est presque toujours insuffisante en raison des problèmes de qualité des données. Il existe probablement des erreurs dans les fichiers à apparier ; dans ce cas, les comparaisons exactes sont trop strictes et ne permettent pas de capter toute l'information.

Il est donc plus intéressant d'effectuer des comparaisons floues et de calculer des distances pour tous les champs identifiants. Le choix de la distance dépend avant tout du type de champ (texte, nombre, date, etc.).

2.4.1 Champs textuels

Le type le plus commun en appariements est la chaîne de caractères, qui correspond à des champs textuels. Les principales distances utilisées pour comparer des chaînes de caractère sont la distance de Levenshtein et la distance de Jaro ou de Jaro-Winkler.

Distance de Levenshtein

La distance de Levenshtein entre deux chaînes de caractères est définie comme le nombre minimal de caractères à supprimer, insérer ou remplacer pour passer de l'une à l'autre. Son temps de calcul est approximativement proportionnel au produit des deux chaînes de caractères à comparer. Á titre d'exemple, la distance de Levenshtein entre 'ELOI' et 'ELLIOT' est de 3, comme illustré ci-dessous.

$$\text{ELOI} \xrightarrow{\text{substitution}} \text{ELLI} \xrightarrow{\text{insertion}} \text{ELLIO} \xrightarrow{\text{insertion}} \text{ELLIOT}$$

En appariements, il est généralement plus pratique de travailler avec des **mesures de similarité**, inversement proportionnelles à la distance et normalisées. Les valeurs vont alors de 0 à 1, 1 correspondant à deux chaînes identiques et 0 à des chaînes extrêmement différentes. La distance de Levenshtein entre deux chaînes de caractères s_1 et s_2 se transforme en une mesure de similarité via la formule suivante :

$$sim_L(s1, s2) = 1 - \frac{dist_L(s_1, s_2)}{max(|s_1|, |s_2|)}$$

avec :
— sim_L la similarité de Levenshtein,
— $dist_L$ la distance de Levenshtein,
— et $|s_i|$ la longueur de la chaîne s_i.

Similarité de Jaro

La similarité de Jaro est plus récente que la distance de Levenshtein et a été développée particulièrement pour la comparaison de noms et prénoms en vue de faire des appariements. Voici sa définition :

$$sim_j(s_1, s_2) = \begin{cases} 0 \text{ si } m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) \text{ sinon} \end{cases}$$

avec :
— m le nombre de caractères correspondants,
— et t le nombre de transpositions.

La définition des *caractères correspondants* est un peu particulière. Deux caractères de s_1 et s_2 sont considérés comme correspondants si les deux conditions suivantes sont remplies :

1. ils sont identiques,
2. leur éloignement (la différence entre leurs positions respectives dans les chaînes s_1 et s_2) ne dépasse pas $\left\lfloor \frac{\max(|s_1|,|s_2|)}{2} \right\rfloor - 1$

ELOI
ELLIOT

Dans l'exemple ci-dessus, les deux chaînes de caractères comportent 4 caractères correspondants : $m = 4$

Pour obtenir le nombre de *transpositions* t, il reste à comparer les caractères correspondants de chaque chaîne de caractères un à un et dans l'ordre. Á chaque fois que les caractères sont différents, une demi-transposition est comptée.

E L O I
↕ ↕ ↕ ↕
E L I O

Dans cet exemple, les caractères en 3^e et 4^e position ne correspondent pas, ainsi $t = 2/2 = 1$. Finalement,

$$sim_J(\text{'ELOI', 'ELLIOT'}) = \frac{1}{3}\left(\frac{4}{4} + \frac{4}{6} + \frac{4-1}{4}\right) = 0,81$$

Similarité de Jaro-Winkler

Winkler a proposé une variante à la similarité de Jaro en ajoutant un terme mesurant la correspondance des premiers caractères. Les erreurs sont en effet relativement moins fréquentes en début de mot qu'au milieu ou à la fin. La similarité de Jaro-Winkler favorise ainsi la présence d'un préfixe commun aux deux chaînes de caractères. Elle se définit comme suit :

$$sim_{JW} = sim_J + \frac{p}{10}(1 - sim_J)$$

avec :
- sim_J la similarité de Jaro,
- et p la longueur du plus grand préfixe commun entre les deux chaînes de caractères, avec un maximum fixé à 4.

Dans l'exemple précédent de comparaison des prénoms 'ELOI' et 'ELLIOT', les deux premiers caractères sont identiques mais le troisième est différent, donc $p = 2$ et :

$$sim_{JW}(\text{'ELOI', 'ELLIOT'}) = 0,806 + \frac{2}{10}(1 - 0,806) = 0,84$$

La similarité de Jaro-Winkler est toujours supérieure à la similarité de Jaro puisqu'on y ajoute un terme positif. C'est un point à garder en tête lors de la comparaison des deux mesures. D'un point de vue informatique, la similarité de Jaro-Winkler se calcule un peu plus rapidement que la distance de Levenshtein.

Similarité reposant sur les n-grammes

Une autre approche de comparaisons de champs textuels fait appel aux n-grammes. Dans une chaîne de caractères, les n-grammes correspondent à toutes les sous-séquences de n caractères. Par exemple, voici les trigrammes (dénomination commune des 3-grammes) des prénoms 'ELOI' et 'ELLIOT' :

— ELOI : ELO et LOI
— ELLIOT : ELL, LLI, LIO et IOT

La similarité associée repose sur le nombre de n-grammes communs entre les deux chaînes de caractères. Les deux prénoms précédents ne possèdent aucun trigramme commun et sont donc considérés comme extrêmement différents. Il est intéressant de noter que le résultat est sensiblement différent de celui obtenu avec les mesures précédentes.

Cet exemple souligne l'une des limites de l'approche des n-grammes. Elle pénalise en effet fortement les erreurs en milieu de mots ou dans des mots courts. Cela va à l'encontre du principe de la similarité de Jaro-Winkler, qui pénalise plutôt les divergences en début de mot car elles sont plus discriminantes. Une méthode visant à rééquilibrer ce comportement consiste à insérer des caractères fictifs en début et / ou en fin de mot. Si le symbole "#" représente le caractère fictif, les trigrammes des prénoms 'ELOI' et 'ELLIOT' deviennent :

— ELOI : ##E, #EL, ELO, LOI, OI# et I##
— ELLIOT : ##E, #EL, ELL, LLI, LIO, IOT, OT# et T##

Il y a maintenant deux trigrammes communs : ##E et #EL.

Cette mesure, comme la distance de Levenshtein, dépend de la taille des chaînes de caractères comparées. Une normalisation est nécessaire pour obtenir une mesure comparable entre différentes paires, ce qui est souvent plus pratique. Une méthode de normalisation qui convient bien à cette approche est l'**indice de Jaccard** sur les deux ensembles de n-grammes des deux chaînes de caractères comparées. Elle conduit à une mesure de similarité classique, dont les valeurs vont de 0 si aucun n-gramme commun n'existe, à 1 pour des chaînes strictement identiques :

$$sim_{NG}(s1, s2) = 1 - \frac{t_{commun}}{t_1 + t_2 - t_{commun}}$$

avec :

— t_1 le nombre de n-grammes dans la chaîne s1,
— t_2 le nombre de n-grammes dans la chaîne s2,
— et t_{commun} le nombre de n-grammes communs entre s1 et s2.

Distance phonétique

Dans certains cas, une distance phonétique peut être utilisée. La distance *Editex* par exemple consiste à encoder les deux chaînes de caractère via un algorithme phonétique comme le *Soundex* ou le *Double Metaphone*, puis à calculer une distance sur ces deux codes.

2.4.2 Dates

Pour des dates, une différence temporelle fonctionne mais il est généralement préférable de procéder à une comparaison textuelle, le plus souvent en segmentant au préalable l'information sur le jour, le mois et l'année. Il est alors possible d'accorder des poids différents à une correspondance sur chacun de ces trois champs et d'ajouter des règles pour prendre en compte les cas typiques comme l'inversion du jour et du mois.

Il est également utile de tenir compte de dates spécifiques. En particulier, le 1er janvier et le 31 décembre font souvent office de valeurs par défaut lorsque l'information est manquante. Si l'analyse des données fait apparaître une surreprésentation de ces dates, c'est qu'il existe probablement une règle de gestion en amont.

2.4.3 Champs numériques

Pour des champs numériques tels que des âges, des numéros de voie voire des revenus de référence, deux solutions existent. La première consiste à calculer la différence. La seconde option est de considérer le champ comme une chaîne de caractères et d'utiliser une distance textuelle.

Les deux approches peuvent mener à des résultats sensiblement différents. Par exemple, pour la comparaison des deux revenus de référence 22641 et 42641, la différence numérique est élevée par rapport à l'ordre de grandeur des deux valeurs, mais leur distance textuelle est faible puisqu'un seul caractère diffère.

2.4.4 En résumé

Quelle que soit la distance utilisée, il peut être intéressant de procéder à une normalisation *a posteriori* afin d'obtenir pour chaque champ identifiant une mesure de similarité comprise entre 0 et 1. Cette approche permet de rendre plus comparables les valeurs obtenues avec différentes méthodes et de simplifier l'interprétation. De cette manière, quel que soit le champ considéré, une valeur proche de 1 sera toujours synonyme d'une grande similarité et inversement pour une valeur proche de 0.

Contrairement aux étapes précédentes où les décisions prises pouvaient avoir une influence énorme sur la qualité de l'appariement, le choix d'une distance plutôt

			Naissance			
Champ	Nom	Prénom	Jour	Mois	Année	Commune
Individu a1	MARTIN	ELOI	01	01	1980	36248
Individu b1	FORTIN	ELLIOT	01	01	1970	48089
Similarité	0,78	0,84	1	1	0	0
Individu a2	GOUSSELOT	ANNE-CLAIRE	13	12	1964	75115
Individu b2	GOUSSEL	ANNE	12	12	1964	75115
Similarité	0,93	0,79	0	1	1	1
Mesure	Jaro-Winkler		Exacte			

TABLEAU **1 – Exemples de calculs de similarités sur variables d'état civil**

Champ	Numéro de voie	Libellé	Commune résidence
Individu a1	2	avenue des Lilas	13206
Individu b1	2	avenue de la Porte Dorée	13206
Similarité	1	0,41	1
Individu a2	37	rue du Commandant Bouchet	75114
Individu b2	57	rue du Commandant Louis Bouchet	75114
Similarité	0	0,81	1
Mesure	Exacte	Levenshtein	Exacte

TABLEAU **2 – Exemples de calculs de similarités sur variables d'adresse**

qu'une autre change relativement peu les résultats obtenus (tant que les distances utilisées sont adaptés aux types des champs). Il est en revanche capital de s'assurer que l'information soit stockée dans un format adapté au modèle de classification utilisé par la suite. En effet, certains algorithmes seront plus performants avec des mesures de similarité continues et normalisées, tandis que d'autres (les méthodes probabilistes) ne fonctionneront qu'avec des distances catégorisées, avec un nombre fixe de modalités. Dans ce dernier cas, il faut fixer des seuils afin de définir les tranches. Voici un exemple de seuils, en notant s_{prenom} la similarité sur le champ prénom :

$$\begin{cases} 2 \text{ si } s_{prenom} > 0,92 \\ 1 \text{ si } 0,88 < s_{prenom} \leq 0,92 \\ 0 \text{ sinon} \end{cases}$$

Exemples de stratégies de comparaison

— Pour les champs courts (code postal ou année de naissance par exemple) : **comparaison exacte**.

— Pour les chaînes de caractères plus longues : distance de **Jaro-Winkler** ou de **Levenshtein**.

— Pour des dates :
- distance de **Levenshtein** sur une date au format **AAAAMMJJ**,
- ou **comparaison exacte des jours, des mois et des années**. Un score intermédiaire peut également être attribué en cas d'inversion jour-mois (jour A = mois B et jour B = mois A).

2.5 Comparaison des paires : classification

L'étape précédente consiste à calculer pour chaque champ identifiant une distance, ou plus vraisemblablement une mesure de similarité. Dans l'étape de classification, ces mesures sont agrégées afin de classer les paires en deux ou trois catégories : les paires liées, les paires non liées, et parfois des paires laissées en suspens pour un éventuel examen manuel. La liberté qui existe, à la fois dans l'agrégation des mesures et dans la façon d'utiliser cette information pour classer, laisse la place à des méthodes très diverses.

Ces méthodes de classification sont généralement rangées en deux groupes : les méthodes déterministes et les méthodes probabilistes. Ces dernières reposent sur l'utilisation d'une probabilité : celle que les deux éléments d'une paire correspondent à un même individu. La théorie sous-jacente est celle de l'estimation statistique, avec notamment des liens très étroits avec les tests d'hypothèse.

Les méthodes déterministes sont bien plus diverses et se caractérisent avant tout par leur non-appartenance à l'autre groupe. Les principales méthodes de classification sont présentées dans la suite de cette section.

Quelle que soit la méthode utilisée, il est d'usage de procéder en premier lieu à un appariement exact afin de lier d'emblée les paires les plus évidentes, pour lesquelles des méthodes de comparaison plus coûteuses ne sont pas nécessaires.

2.5.1 Méthode des tours de clés successifs

Cette méthode consiste à apparier les deux fichiers au cours de **plusieurs étapes successives en commençant par des règles strictes et en relâchant progressivement les contraintes**. Les individus appariés à une étape ne sont plus considérés pour les étapes suivantes. Cette méthode est spécifique aux appariements *one-to-one*.

La première étape est en général un appariement exact. Les étapes suivantes autorisent des erreurs et deviennent de moins en moins strictes. Autoriser des erreurs peut se faire soit en excluant simplement un champ de la comparaison, soit en imposant une contrainte plus souple que l'exactitude, comme une distance de Levenshtein maximale fixée (3, par exemple). L'idée étant de relâcher progressivement les contraintes, ce sont plutôt les champs les moins discriminants ou ceux contenant le plus d'erreurs qui sont relâchés en premier, par exemple le jour de naissance plutôt que le nom de famille.

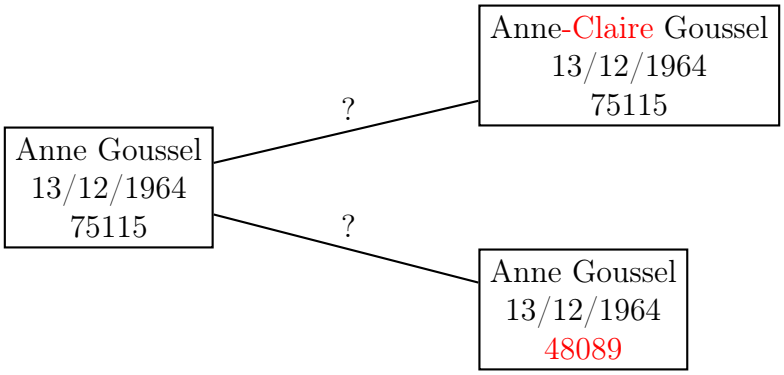
La méthode des tours de clés successifs a l'avantage d'être **facile à mettre en œuvre**. Elle est de plus **facilement explicable** et compréhensible, même pour un non-initié. Pour cette raison, elle peut être adaptée pour des appariements *ad hoc*.

Par ailleurs, **le temps de calcul associé est en général acceptable**, à condition de ne pas abuser des calculs de distance. En effet, d'une part, les champs sont souvent comparés de façon exacte, ce qui est relativement peu coûteux informatiquement parlant ; d'autre part, le nombre de paires à traiter diminue à chaque étape.

Du côté des inconvénients, **le choix de l'ordre des étapes est critique** et peut engendrer des erreurs à cause du caractère séquentiel de l'approche. Certains individus peuvent être appariés à tort à l'étape 2 alors qu'ils auraient été appariés correctement à l'étape 4, par exemple. Cette difficulté est illustrée par le graphique 5.

Le **choix des règles et contraintes appliquées à chaque étape** est tout aussi important. Il n'existe pas de vérité absolue en la matière, c'est le plus souvent une approche par essais et erreurs qui permet d'obtenir un bon appariement en procédant plusieurs fois à de petits ajustements. C'est donc **un processus qui peut prendre du temps** car il doit être adapté aux données à apparier.

En outre, **cette méthode ne fournit pas de score numérique pour les paires**, comme une probabilité ou une mesure de similarité globale au niveau de la paire. Or, un tel score se montre souvent utile, en particulier pour résoudre les éventuels conflits d'appariements ou évaluer la qualité.



GRAPHIQUE **5 – Illustration de l'importance de l'ordre des étapes dans la méthode des tours de clés successifs**

La paire liée ne sera pas la même selon que l'on choisit de relâcher d'abord la commune de naissance ou le prénom.

Voici un exemple fictif de procédure d'appariement par la méthode des tours de clés successifs en 5 étapes ; les champs identifiants utilisés étant le nom, le prénom ainsi que la date et la commune de naissance :

1. appariement exact ;
2. relâche totale sur la commune de naissance (les autres champs sont comparés de façon exacte) ;
3. relâche totale sur la date de naissance (les autres champs sont comparés de façon exacte) ;
4. comparaison exacte de la date et de la commune de naissance, et distance de Levenshtein maximale de 3 sur le nom et le prénom ;
5. appariement exact avec une interversion des champs noms et prénoms (prénom A est comparé à nom B et prénom B est comparé à nom A).

2.5.2 Méthode du plus proche écho

La méthode du plus proche écho **consiste à calculer des mesures de similarité pour chaque champ identifiant puis à les agréger pour obtenir une mesure globale de similarité pour chaque paire retenue après l'étape d'indexation**. La similarité globale au niveau de la paire s'obtient par une somme pondérée des similarités de chaque champ.

Cette méthode s'accompagne presque systématiquement de la sélection d'un seuil. Dans ce cas, une paire n'est liée que si sa similarité globale dépasse le seuil. Ce seuil peut être choisi au juger, mais il peut être intéressant de s'appuyer sur un échantillon de paires dont le statut est connu afin d'optimiser sa valeur.

Cette méthode, comme la précédente, est **facile à mettre en œuvre et à comprendre**. Elle se montre en général **assez efficace** si les poids associés à chaque variable sont bien ajustés.

Néanmoins, **ces poids sont à déterminer manuellement**, les ajuster peut ainsi se révéler ardu et fastidieux.

Par ailleurs, **les calculs de similarité** de type Levenshtein ou Jaro-Winkler **sont coûteux informatiquement**. Il peut donc être nécessaire avec cette approche de réduire drastiquement le nombre de paires en amont, ou bien de limiter ces comparaisons à certains champs seulement (le nom et le prénom par exemple).

Voici un exemple fictif de pondérations sur des variables d'état civil :
— nom, prénom et commune de naissance : 1 ;
— jour et mois de naissance : $0,25$;
— année de naissance : $0,5$.

Avec ces poids, la similarité théorique maximale pour une paire est de 4. Le seuil d'acceptation des paires peut être fixé à 3.

Dans le cas des paires d'individus du tableau 1, la paire a1-b1 obtient un score de $2,12$ et la paire a2-b2 un score de $3,47$. La valeur de 3 choisie pour le seuil conduirait donc à lier la paire a2-b2 et à ne pas lier la paire a1-b1.

2.5.3 *Machine learning* supervisé

Un appariement est une tâche de classification binaire sur un ensemble de paires. Le but est de prédire si une paire correspond à une paire d'individus identiques ou à une paire d'individus différents. Les méthodes de *machine learning* supervisé ayant prouvé leur efficacité sur ce type de tâches, il est naturel que de nouvelles approches d'appariement reposant sur ce principe soient apparues au cours des dernières années. Les méthodes d'appariement reposant sur du *machine learning* supervisé consistent à **construire un modèle qui, à partir d'un échantillon de paires annotées, apprend à prédire le statut de nouvelles paires**. Pour se placer dans un cadre classique de *machine learning*, il faut considérer les correspondances suivantes :
— les éléments à classer sont les paires ;
— la variable à prédire est le statut de la paire ;
— les variables explicatives sont les mesures de similarité sur chaque champ obtenues à l'étape précédente ;
— la base d'apprentissage est un ensemble de paires dont le statut réel est connu.

Les méthodes de classification binaire sont très nombreuses et elles peuvent toutes en théorie être employées pour faire des appariements. Cependant, les plus

communes sont la **régression logistique**, les ***Support Vector Machines*** et les modèles reposant sur des arbres de décision comme les ***Gradient Boosting Machines***. La présentation de ces méthodes sort du cadre de ce document. Le lecteur intéressé pourra trouver des éléments d'explication dans Hastie, Tibshirani, et Friedman (2001).

L'avantage principal de cette approche est lié au fondement même du *machine learning* : **le modèle apprend lui-même la façon optimale de combiner les différentes mesures de similarités**, il n'y a donc pas de poids à choisir comme dans la méthode du plus proche écho par exemple. L'expertise humaine se porte plutôt sur le choix du modèle ainsi que de ses hyperparamètres (par exemple, le degré de pénalisation L1 ou L2 dans une régression logistique).

Le frein principal à l'utilisation de cette méthode est peut-être la **nécessité de fournir des paires annotées au modèle**. Le plus souvent, aucun échantillon annoté n'est disponible, il faut donc prendre le temps de le constituer. La façon d'obtenir un tel échantillon est discutée dans l'encadré 3 page 39.

Par ailleurs, **certains modèles de *machine learning* peuvent souffrir du fort déséquilibre qui existe entre les deux classes** : même avec une étape d'indexation, il y a en général beaucoup plus d'exemples négatifs (les paires d'individus différents) que d'exemples positifs (les paires d'individus identiques). Ce déséquilibre perturbe l'apprentissage du modèle et se traduit par des résultats peu satisfaisants.

L'effet du déséquilibre des classes peut toutefois être contré en partie en portant une attention particulière à la constitution de l'échantillon d'apprentissage. Plutôt que de tirer un échantillon aléatoire uniforme parmi l'ensemble des paires, il s'agit de choisir celles qui seront les plus utiles au modèle. L'*active learning* (Settles (2009)) constitue une approche intéressante pour remplir cet objectif.

L'***active learning*** consiste à **faire intervenir le modèle dans le choix des paires à annoter**. Plus précisément, le modèle est d'abord entraîné sur un petit échantillon sélectionné arbitrairement, puis il est utilisé pour prédire le statut de l'ensemble des autres paires. Ces prédictions sont ensuite mises à profit pour choisir les prochaines paires à annoter. Ce procédé peut éventuellement être répété plusieurs fois, en calibrant à chaque fois un nouveau modèle à partir de l'ensemble des paires annotées jusque-là. L'idée est toujours de **sélectionner les paires qui apporteront le plus d'information au modèle**. En général, il s'agit des cas les plus ambigus, pour lesquels la prédiction est la plus incertaine, par exemple les paires dont la probabilité est proche de $0,5$ si le modèle fournit une probabilité.

C'est l'approche qui permet **d'obtenir l'échantillon d'apprentissage le plus riche**, pour une taille fixée. Le temps d'annotation nécessaire pour obte-

nir un échantillon satisfaisant s'en voit donc réduit, ce qui rend le processus moins coûteux.

Une telle approche est cependant **un peu plus lourde à mettre en œuvre** en raison de son caractère itératif, puisqu'il faut calibrer le modèle plusieurs fois.

2.5.4 Méthodes probabilistes

Les méthodes d'appariement dites "probabilistes" dérivent toutes du **cadre décrit par Fellegi et Sunter (1969)**. Elles se caractérisent par un processus d'**inférence bayésienne** qui conduit au **calcul d'une probabilité** pour chaque paire considérée. Cette partie expose le principe général et les particularités de l'approche probabiliste. Une présentation plus technique de la théorie de Fellegi-Sunter est proposée en annexe A.

Dans le modèle probabiliste classique, **les paires sont classées en trois catégories** (les paires liées, les paires non liées et une zone grise de paires laissées en suspens) **grâce à deux seuils définis en fonction des taux d'erreurs autorisés**.

La classification des paires repose sur l'utilisation de deux probabilités conditionnelles, appelées m et u, calculées pour chaque paire et chaque champ identifiant. Ainsi, pour une paire et un champ identifiant donnés :

- $\boldsymbol{m}$ correspond à la probabilité d'observer une valeur identique au sein de la paire sur ce champ, sachant qu'il s'agit d'une paire d'individus identiques. $\mathbf{1 - m}$ **représente en quelque sorte le taux d'erreurs lors de la saisie des données**. Plus les données sont propres et de qualité, plus m est grand.
- $\boldsymbol{u}$ correspond à la probabilité d'observer une valeur identique au sein de la paire sur ce champ, sachant qu'il s'agit d'une paire d'individus différents. $\boldsymbol{u}$ **représente la probabilité d'observer la même valeur par chance pour deux individus pris au hasard** (par exemple $\sim 1/12$ pour la variable mois de naissance).

Des **poids** sont ensuite calculés pour chaque variable identifiante et pour les deux cas de figure : valeur identique pour cette variable identifiante au sein d'une paire, ou valeur différente. Ces poids sont définis comme le ratio des probabilités m et u ou de leurs opposés : m / u pour une valeur identique au sein d'une paire, (1-m) / (1-u) pour une valeur différente.

Les poids représentent le pouvoir prédictif de chaque champ pour déterminer si deux individus d'une paire sont identiques ou non. Par exemple, un nom de famille ne porte pas la même information qu'un genre.

Plus un poids est important, plus il incite à lier la paire ; et inversement. Ainsi, observer un nom de famille identique au sein d'une paire sera associé à un poids très

important car c'est un indice intéressant pour rapprocher deux individus, tandis qu'avoir le même genre correspondra à un poids modéré.

Pour un même champ, les poids ont souvent une importance différente selon que la valeur observée est identique ou différente au sein de la paire. En effet, si le fait d'avoir le même genre donne peu d'informations, observer un genre différent au sein d'une paire sera associé à un poids assez faible car, sauf erreur dans les données, les deux individus sont forcément différents.

Le modèle de Fellegi-Sunter permet d'attribuer des poids à chaque champ mais, comme la plupart des autres méthodes d'appariement, il ne tient pas compte de la rareté des modalités rencontrées. Par exemple en France, « Martin » est un nom de famille bien plus courant que « Milton ». Il y a donc plus de chances d'observer une correspondance sur le nom « Martin » par hasard. Pour tenir compte de cet effet, il est possible d'**ajouter une correction liée à la fréquence des modalités**. De cette façon, une correspondance sur une modalité rare sera associée à un poids plus important (et donc une probabilité prédite plus grande) qu'une correspondance sur une modalité fréquente.

La décision de lier une paire ou pas est prise sur la base de la mesure obtenue en agrégeant les poids pour chaque variable identifiante en fonction des caractéristiques observées au sein de la paire (valeur identique ou différente pour chaque champ).

Toute la difficulté réside dans le fait d'estimer les probabilités m et u. L'algorithme Espérance-Maximisation (EM, Dempster, Laird, et Rubin (1977)) constitue la méthode de référence (Jaro (1989)). Il a l'avantage de ne nécessiter aucune donnée ou connaissance extérieure, contrairement à la plupart des autres méthodes d'estimation existantes.

À condition de laisser le modèle classer toutes les paires (c'est-à-dire de ne rien laisser dans la zone grise), la méthode probabiliste est peut-être celle qui permet d'aboutir à un résultat le plus rapidement (en temps de réflexion et de développement, pas forcément en temps de calcul informatique) puisqu'elle s'applique directement à n'importe quel jeux de données. En effet, d'une part, **il n'y a pas de poids ou de paramètres à déterminer soi-même**, ils sont estimés à partir des données. D'autre part, **un échantillon de paires annotées n'est pas non plus nécessaire** puisque l'estimation s'effectue de manière non supervisée.

L'approche probabiliste est aussi celle qui offre en théorie le meilleur contrôle sur les erreurs puisque dans la théorie de Fellegi-Sunter, les paires sont classées de façon à ne pas dépasser des taux de faux positifs et faux négatifs fixés à l'avance. Cependant, en pratique, l'estimation de ces taux d'erreurs est biaisée lorsqu'il y a une étape de réduction du nombres de paires ou lorsque la proportion de paires

d'individus identiques parmi l'ensemble des paires est déséquilibrée, ce qui est très souvent le cas sur des données réelles.

Du côté des inconvénients, les méthodes probabilistes ont en général un **temps de calcul assez long** par rapport à des approches déterministes optimisées. Certains outils sont donc assez limités sur la taille des fichiers qu'ils permettent d'apparier.

Par ailleurs, la théorie probabiliste est relativement complexe. Les méthodes déterministes sont plus directes, plus facilement explicables et plus faciles à développer *ad hoc*. Cependant, un utilisateur peut tout à fait apparier deux fichiers de manière probabiliste sans connaître la théorie sous-jacente s'il dispose d'un outil clés en main.

Plusieurs instituts nationaux de statistiques ont fait le choix d'investir dans les méthodes d'appariement probabilistes. Ainsi, Statistique Canada a développé l'outil G-Link (Chevrette (2011), logiciel propriétaire), tandis que Istat est à l'origine de RELAIS (Tuoto, Cibella, Fortini, et Scannapieco (2010), logiciel libre sous licence EUPL-1.1).

Encadré 1 : Examen manuel

Certains modèles ne classent pas toutes les paires en deux catégories mais laissent une zone grise pour les cas jugés particulièrement ambigus. Ce fonctionnement est particulièrement observé dans la théorie traditionnelle de l'appariement probabiliste. L'idée est de classer automatiquement la plupart des paires, mais de laisser la décision à un humain pour les cas les plus complexes.

C'est un processus coûteux, le volume de paires traité de façon manuelle doit rester raisonnable. L'utilisation d'une interface montrant clairement les différences facilite grandement la tâche.

L'examen manuel de paires peut également intervenir à d'autres étapes du processus d'appariement. Par exemple, en amont de la classification, les modèles reposant sur du *machine learning* nécessitent un ensemble de paires annoté, c'est-à-dire dont le vrai statut est connu. C'est également le cas pour le calcul de certaines mesures de performance.

2.5.5 Utiliser un moteur de recherche

En plus des méthodes traditionnelles d'appariement présentées dans ce document, une autre approche se place comme un candidat intéressant : **l'utilisation**

d'un moteur de recherche.

Ce type d'outils n'est pas initialement conçu pour faire des appariements de données individuelles. En particulier, la grande majorité des cas d'usage de ces moteurs de recherche portent sur des champs textuels longs, comme le contenu d'une page web ou la description d'un produit ; ce qui contraste avec les noms, prénoms, et autres champs usuels en appariements. L'outil étant détourné de son usage initial, cela induit un paramétrage spécifique et les valeurs par défaut ne sont pas toujours les plus adaptées.

Cette approche n'est pas discutée dans la littérature, néanmoins **plusieurs appariements au sein du service statistique public français ont montré par exemple le potentiel du moteur ElasticSearch**.

Cette méthode se distingue nettement des autres car elle **brouille les frontières entre les différentes étapes**, en particulier entre celle de réduction du nombre de paires et celle de comparaison des champs.

Fonctionnement

Pour apparier deux fichiers avec un moteur de recherche, il faut **d'abord créer un index inversé de l'un des deux fichiers**, de préférence le plus volumineux, **puis rechercher les individus de l'autre fichier dans cet index**.

D'après la documentation d'ElasticSearch, « un index inversé liste chaque mot unique qui apparaît dans n'importe quel document et identifie tous les documents dans lesquels chaque mot apparaît ». La création d'un index inversé est relativement longue, mais elle rend la recherche dans le fichier indexé considérablement plus efficace. Ainsi, ElasticSearch est capable de déterminer de façon extrêmement rapide par exemple que le prénom « Jason » apparaît 3 fois dans le fichier indexé et de renvoyer l'identifiant des 3 individus en question. Attention, le terme indexation prend un autre sens dans le contexte des moteurs de recherche. Il désigne le fait de créer l'index inversé et non l'étape de filtrage des paires.

Pour rechercher les individus du second fichier dans cet index inversé, il faut ensuite effectuer des requêtes qui imposent des contraintes sur certains champs (ce qui agit comme une étape de filtrage) et qui calculent des similarités de différentes manières (correspondance exacte, calcul de distance, etc.). Le moteur de recherche renvoie alors une liste d'individus du premier fichier qui répondent à ces critères et les classe par score décroissant, comme le ferait un moteur de recherche de sites web.

Cas d'usage

Tous les appariements se prêtent à l'utilisation d'un moteur de recherche. Cependant, ces outils prennent tout leur sens lors d'une **recherche d'individus**

dans un fichier de référence volumineux.

L'attribution d'un code statistique non signifiant (CSNS) [1] est un cas d'usage typique qui tire toute la puissance d'ElasticSearch. L'opération consiste en un service permettant d'associer à chaque individu d'un fichier un CSNS, qui pourra servir d'identifiant unique pour les traitements statistiques. Ce service a vocation à faciliter les appariements, tout en préservant la vie privée des individus concernés puisque l'identifiant est non signifiant, c'est-à-dire qu'il ne dévoile aucune information personnelle. Le CSNS est calculé à partir du numéro d'inscription au répertoire (NIR) et d'une procédure de chiffrement. Pour obtenir ce NIR [2], s'il est absent du fichier soumis, les individus sont recherchés dans le répertoire national d'identification des personnes physiques (RNIPP) sur leurs traits d'identités. Ainsi, dans le cadre du service de délivrance du CSNS, de nombreux appariements vont avoir lieu entre différents fichiers et le fichier de référence qu'est le RNIPP. Puisque le RNIPP évolue très peu, une seule indexation permettra d'effectuer plusieurs appariements, ce qui permet de **mutualiser le temps de traitement**.

En plus des appariements de données individuelles, les moteurs de recherche sont particulièrement efficaces sur d'autres types d'appariements, notamment avec des libellés textuels plus longs comme des descriptions de produits.

Principales méthodes de classification des paires

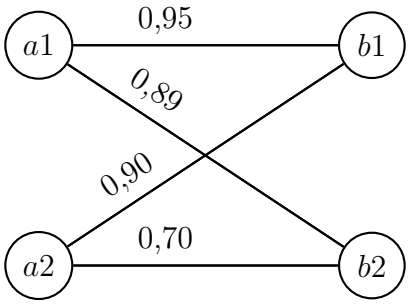
- Approches **déterministes** :
  - méthode des tours de clés successifs,
  - méthode du plus proche écho,
  - classification par *machine learning*.
- Approche **probabiliste** : toutes les méthodes dérivant du cadre de Fellegi et Sunter (1969).
- **Moteur de recherche** textuelle (ex : ElasticSearch)

Aucune de ces méthodes n'est uniformément meilleure que les autres, le choix doit être guidé par le type de données à apparier ainsi que les objectifs poursuivis. Des éléments de décision seront présentés dans la section 4.

1. Le code statistique non signifiant trouve son origine dans la loi pour une République numérique de 2016. Il est à usage exclusif du service statistique public pour des seules finalités statistiques.
2. Le NIR n'est *in fine* jamais délivré dans cette opération d'attribution, seul le CSNS l'est.

2.6 Résolution des conflits

Les algorithmes de classification traitent généralement les paires de façon indépendante, mais bien souvent il existe des restrictions sur les combinaisons de paires qui peuvent être liées. Par exemple, dans le cas d'un appariement *one-to-one*, un seul individu du fichier A doit être apparié à un seul individu du fichier B.



GRAPHIQUE **6 – Exemple de conflit d'appariement**

Le poids d'une arête désigne le score obtenu pour la paire associée.

Dans l'exemple de la figure 6, le calcul des scores a créé un conflit et il n'est pas évident de décider quels individus doivent être appariés.

Appariement *one-to-one*

Les appariements *one-to-one* représentent la majorité des cas rencontrés en statistique publique. Dans ce cas, deux solutions se distinguent pour résoudre les éventuels conflits.

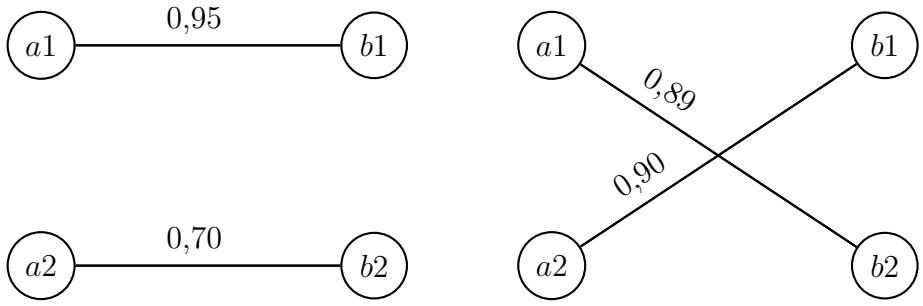
La première solution repose sur un **algorithme glouton**. Ce type d'approche consiste, dans un processus itératif, à choisir à chaque instant la solution locale optimale, sans garantie d'obtenir un optimum global. C'est un compromis entre la recherche de l'optimum dans un ensemble énorme de solutions et le temps de calcul : l'objectif est d'aboutir à une bonne solution en un temps raisonnable.

L'algorithme peut être appliqué si les méthodes choisies dans les étapes précédentes permettent d'obtenir un score pour chaque paire, c'est-à-dire une probabilité ou une mesure de similarité globale. Dans ce cas, les paires sont d'abord triées par ordre décroissant de score puis elles sont traitées une à une. Pour commencer, les deux individus de la première paire (i.e. la paire la plus probable) sont appariés. Ils sont ensuite retirés et ne pourront plus être appariés avec un autre individu. Le processus se poursuit de façon itérative en parcourant l'ensemble des paires.

Il est éventuellement possible de fixer un score seuil en dessous duquel les individus ne sont de toute façon pas appariés. Par exemple, toutes les paires dont

le score est inférieur à 0,7 sont d'emblée écartées et l'algorithme de résolution des conflits n'est appliqué que sur les paires restantes.

Dans l'exemple de la figure 6, cette stratégie conduirait à former les paires a1-b1 et a2-b2.



(a) Méthode gloutonne **(b) Solution optimale**

GRAPHIQUE **7 – Comparaison de deux approches de résolution des conflits**

La seconde approche consiste à trouver une solution optimale, c'est-à-dire une solution qui maximise la somme des scores de toutes les paires retenues. Cette tâche est connue en informatique sous le nom de problème d'affectation. Une méthode de résolution de ce problème est l'algorithme hongrois (Kuhn (1955)).

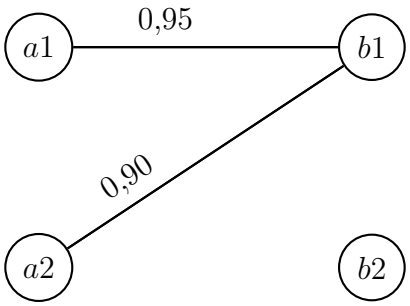
Privilégier cette approche est tentant *a priori*, mais la recherche d'une solution optimale s'accompagne d'un temps de calcul important. Lorsque les fichiers traités deviennent trop volumineux, il est alors nécessaire de se rabattre sur l'approche gloutonne.

Dans l'exemple précédent, ce sont les paires a1-b2 et a2-b1 qui permettent de maximiser la somme des scores. La solution obtenue est donc différente de celle retenue par la méthode gloutonne.

Appariement *many-to-one*

Dans le cas d'un appariement *many-to-one*, il est possible de se ramener à une situation *one-to-one* en passant au préalable par un dédoublonnage du fichier pouvant contenir des individus identiques.

Autrement, il existe une solution gloutonne très similaire à celle présentée précédemment. Les paires sont toujours traitées de manière successive dans l'ordre décroissant du score, mais au lieu d'éliminer au fur et à mesure les deux individus de chaque paire retenue, on conserve l'individu provenant du fichier exempt de doublons pour qu'il puisse éventuellement être lié à nouveau. Cette solution est illustrée par le graphique 8.



GRAPHIQUE 8 – **Résolution des conflits dans un cas many- (fichier A) to-one (fichier B)**

Dans cet exemple, le fichier A peut contenir des doublons, mais pas le fichier B.

Méthodes de résolution des conflits

— Appariement *one-to-one* :
- **Méthode itérative gloutonne** (solution sous-optimale mais au temps de calcul raisonnable)
- Résolution du problème d'affectation (recherche de solution optimale) : **algorithme hongrois** par exemple

— Appariement *many-to-one* :
- **Dédoublonnage** du fichier contenant des doublons pour se ramener à une situation *one-to-one*
- **Méthode itérative gloutonne** adaptée

Encadré 2 : Fusion des fichiers appariés

Les deux fichiers appariés contiennent les mêmes champs identifiants avec des informations potentiellement différentes des deux côtés. Si ces variables présentent un intérêt après l'appariement, quelles valeurs conserver ?

Lorsque l'un des deux fichiers est considéré comme de meilleure qualité générale, il est classique de le choisir comme fichier de référence et de conserver les informations issues de ce fichier.
Dans le cas contraire, plusieurs solutions existent. Les méthodes automatiques requièrent en général des choix forts pour assurer qu'une valeur puisse être choisie, comme conserver toujours la valeur la plus grande pour un champ numérique. Bleiholder et Naumann (2009) ont étudié en détail le processus de fusion et ont recensé les méthodes les plus communes, parmi lesquelles (et de façon non exclusive) :

- utiliser une information temporelle pour garder l'information la plus récente,
- conserver les deux valeurs,
- vérifier la cohérence des combinaisons de variables,
- ou même examiner manuellement tous les conflits.

2.7 Évaluation de la qualité

À ce stade, l'appariement est terminé et les deux fichiers peuvent techniquement être fusionnés. Cependant, il serait dangereux de s'arrêter là sans contrôler les résultats. Il reste donc une dernière étape capitale, qui consiste à rassembler autant d'informations que possible sur la qualité de l'appariement. L'objectif est double : **quantifier la performance du processus** et **trouver des pistes d'amélioration**.

Évaluer la qualité d'un appariement permet en effet de disposer de mesures chiffrées, à partir desquelles l'appariement pourra être jugé satisfaisant ou non. De nombreux appariements ont pour finalité une étude statistique. Dans ce cas, il est important de s'assurer que certaines contraintes sont vérifiées pour garantir la qualité des données utilisées, ou d'évaluer l'impact potentel de l'appariement sur les résultats de l'étude. Par ailleurs, les mesures de qualité peuvent être mises à profit pour comparer différentes méthodes (approche déterministe ou probabiliste, différentes clés de blocage, etc.) et sélectionner la plus performante.

D'autre part, cette étape est également l'occasion d'identifier d'éventuelles pistes d'amélioration pour l'appariement. Par exemple, si les résultats montrent qu'une sous-population en particulier est mal appariée, il peut être intéressant d'y porter une attention particulière, notamment en retravaillant le nettoyage et la

normalisation des données sur cette sous-population. L'examen manuel de paires donne aussi l'opportunité de repérer les erreurs fréquentes et d'adapter l'appariement pour les éviter.

2.7.1 Taux d'individus appariés

Une première mesure de la qualité est le taux d'individus appariés. Il peut être différent pour les deux bases si celles-ci ne couvrent pas exactement la même population. Avantage non négligeable, il peut être calculé pour tous les appariements sans information supplémentaire, contrairement aux mesures abordées dans la suite de cette section. Pour cette raison, il est tentant d'évaluer un appariement sur le taux d'individus appariés. Cependant, cet indicateur ne donne qu'une information partielle sur la qualité de l'appariement. Apparier tous les individus de façon aléatoire conduit à un taux d'appariement parfait de 100%, pourtant aucun individu ou presque ne sera apparié correctement. Ainsi, le taux d'appariement doit être complété par d'autres mesures de qualité.

Encadré 3 : Obtenir un échantillon de paires annotées

Si le taux d'individus appariés se calcule très facilement pour n'importe quel appariement, ce n'est pas le cas de la plupart des autres mesures de qualité. Celles-ci nécessitent en effet des informations supplémentaires sur les fichiers appariés, le plus souvent un échantillon de paires annotées.

Dans le meilleur des cas, on dispose d'un ***gold standard***. Il s'agit d'un échantillon de paires **représentatif** des fichiers et dont le vrai statut est connu. La manière d'obtenir un tel échantillon est propre à chaque cas.

Dans la majorité des cas cependant, il n'existe pas de *gold standard* et il faut donc passer par une étape d'**annotation manuelle**. Plus l'interface est ergonomique et pensée pour aider visuellement l'annotateur, plus le processus sera efficace.
L'examen manuel de paires à l'issue d'un appariement poursuit deux objectifs :

— identifier des axes d'amélioration,
— constituer un échantillon représentatif pour l'évaluation.

Pour faire évoluer le processus d'appariement, les paires les plus intéressantes à examiner sont des cas très particuliers, ou les paires les plus incertaines pour le modèle. Toutefois, en agissant de la sorte, on ne constitue pas un échantillon **représentatif** et les indicateurs de qualité calculés ne peuvent pas être généralisés à l'ensemble des paires. Il faut donc être conscient de l'objectif poursuivi avant de sélectionner les paires à annoter. Une stratégie possible pour constituer un échantillon représentatif est de faire des strates par valeur du score et de tirer aléatoirement les paires selon ces strates.

2.7.2 Matrice de confusion

Un appariement peut se résumer à une tâche de classification binaire sur des paires d'individus. À condition de disposer d'un échantillon représentatif de paires annotées, les indicateurs de qualité classiques des problèmes de classification binaire sont donc tout à fait pertinents pour évaluer un appariement. Parmi ceux-ci, on trouve en premier lieu la **matrice de confusion**, dont découlent plusieurs autres indicateurs importants.

La matrice de confusion sépare les paires en quatre groupes en fonction de leur statut réel et du statut prédit par le modèle :

— les **vrais positifs** (VP) sont les paires d'individus identiques qui ont été liés par le modèle ;
— les **vrais négatifs** (VN) sont les paires d'individus différents qui n'ont pas

		Statut réel	
		Individus identiques	Individus différents
Statut prédit	Lié	Vrais positifs (VP)	Faux positifs (FP)
	Non lié	Faux négatifs (FN)	Vrais négatifs (VN)

TABLEAU **3 – Matrice de confusion**

été liés par le modèle ;
— les **faux positifs** (FP) sont les paires d'individus différents qui ont été liés par le modèle ;
— les **faux négatifs** (FN) sont les paires d'individus identiques qui n'ont pas été liés par le modèle.

La matrice de confusion est un outil visuel intéressant mais elle ne constitue pas une évaluation quantitative de la performance. Il est cependant possible de définir de telles mesures à partir des valeurs inscrites dans la matrice de confusion. La **justesse** (connue sous le nom d'*accuracy* en anglais) en est un exemple. Elle représente la proportion de prédictions correctes du modèle. Visuellement, il s'agit de la proportion de paires se trouvant dans les cases vertes de la matrice. Sa définition est la suivante :

$$\begin{aligned} justesse &= \frac{VP + VN}{VP + VN + FP + FN} \\ &= \frac{nombre\ de\ paires\ classées\ correctement}{nombre\ total\ de\ paires} \end{aligned}$$

La justesse est un indicateur simple et très utile dans certains cas. Cependant, dans un problème d'appariement, la distribution des deux classes est extrêmement déséquilibrée : pour des fichiers de taille n, le nombre de paires d'individus identiques est proche de n tandis que le nombre de paires d'individus différents est approximativement de n^2. Le nombre de vrais négatifs écrase les trois autres quantités et l'indicateur se rapproche mécaniquement de 1. Des mesures comme la précision et le rappel sont dans ce cas plus adaptées.

2.7.3 Précision, rappel et F-mesure

La **précision** se définit de la façon suivante :

$$
\begin{aligned}
\textit{précision} &= \frac{VP}{VP + FP} \\
&= \frac{\textit{nombre de paires d'individus identiques liées par le modèle}}{\textit{nombre de paires liées par le modèle}} \\
&= \textit{Taux de réussite sur les paires liées par le modèle}
\end{aligned}
$$

Une précision élevée signifie que le modèle se trompe rarement lorsqu'il lie une paire. Cependant cela ne dit rien sur sa capacité à en identifier un grand nombre. Dans le cas extrême, un modèle liant une seule paire et ayant raison aura une précision parfaite de 1. Un tel modèle n'est pourtant pas satisfaisant. C'est pourquoi le rappel intervient souvent en complément de la précision.

Le **rappel**, aussi appelé sensibilité, correspond en effet à la proportion de cas positifs identifiés comme tels par le modèle. Il se définit comme suit :

$$
\begin{aligned}
\textit{rappel} &= \frac{VP}{VP + FN} \\
&= \frac{\textit{nombre de paires d'individus identiques liées par le modèle}}{\textit{nombre de paires d'individus identiques}} \\
&= \textit{Taux d'identification des paires d'individus identiques}
\end{aligned}
$$

Bien que prises séparément les mesures de précision et de rappel soient insuffisantes pour caractériser pleinement la performance d'un modèle, leur analyse conjointe permet d'en avoir une représentation assez précise. Plusieurs mesures combinant précision et rappel existent, la plus répandue étant sans doute le **F1-score**. Il se définit comme la moyenne harmonique de la précision et du rappel :

$$
\textit{F1-score} = \frac{2}{\textit{précision}^{-1} + \textit{rappel}^{-1}} = 2 \cdot \frac{\textit{précision} \cdot \textit{rappel}}{\textit{précision} + \textit{rappel}}
$$

Cette mesure s'interprète plutôt par comparaison avec un autre F1-score que par sa valeur. Le F1-score associe la même importance à la précision et au rappel. Il est possible d'ajouter une pondération pour prendre en compte l'importance relative accordée aux faux positifs et aux faux négatifs.

2.7.4 Analyse de distributions

Une méthode complémentaire d'évaluation de la qualité, qui se montre d'autant plus utile si une annotation manuelle de paires est impossible, consiste à **analyser les distributions des paires liées et des paires non liées** par le modèle. Dans le cas où un échantillon annoté est disponible, il est en plus possible d'étudier **la distribution des paires identifiées comme incorrectes**.

L'étude des distributions peut porter en premier lieu sur les champs identifiants eux-mêmes, comme le sexe, l'âge ou le département de naissance. Il s'agit de rechercher des anomalies, des modifications de distributions dues à l'appariement. Par exemple, si les deux fichiers initiaux contiennent approximativement la même proportion d'hommes et de femmes, et que les hommes sont très majoritairement représentés dans les paires liées. Ce comportement est peut-être dû à la mauvaise prise en compte de certaines caractéristiques propres aux femmes, comme la présence d'un nom marital et d'un nom de naissance. Ainsi, en plus d'informer sur la qualité générale de l'appariement, l'analyse des distributions permet de pointer du doigt certaines sources d'erreur.

L'étude des distributions des autres variables disponibles, celles qui ne servent pas à l'appariement, est tout aussi intéressante. Voici un exemple. Deux fichiers couvrant la même population sont appariés. L'un des deux contient une variable liée aux revenus. Parmi les paires liées, se trouvent majoritairement des individus à hauts revenus, alors que la distribution était bien plus étalée dans la population initiale. Il y a donc un problème de représentativité dans la population appariée.

Les informations sur le manque de représentativité sont capitales si des études statistiques reposant sur les résultats de l'appariement sont menées par la suite, car les résultats seront altérés par ce problème. Le meilleur indicateur de qualité dans ce cas est une **évaluation de l'impact des erreurs d'appariement sur les études qui en découlent**, mais il n'existe pas de méthode unique de détection de ces erreurs comme pour les indicateurs classiques. De plus, la personne effectuant l'appariement n'est pas toujours la même que celle qui mène l'étude statistique, ce qui demande d'impliquer cette dernière dans l'évaluation de la qualité de l'appariement.

Encadré 4 : Exemple de prise en compte des erreurs d'appariement

On souhaite étudier l'incidence d'une maladie dans une population. Pour cela, on dispose de deux fichiers. Le premier liste l'ensemble des individus ayant contracté cette maladie sur une période donnée et contient seulement des informations d'état-civil. Le second contient tous les individus de la population d'intérêt et comprend des informations telles que leur région de résidence et leur situation d'emploi.

L'analyse des résultats de l'appariement peut révéler une plus grande difficulté à apparier les personnes jeunes, par exemple en raison d'informations de qualité médiocre pour cette sous-population dans l'un des deux fichiers. Dans ce cas, si l'appariement est considéré (à tort) comme parfait, l'incidence de cette maladie chez les jeunes sera sous-estimée. Il est donc intéressant d'estimer un taux de faux positifs par catégorie d'âge pour discuter de l'effet de ces paires manquées sur le taux d'incidence estimé et éventuellement proposer des chiffres ajustés.

Indicateurs de qualité d'un appariement

— Sans échantillon représentatif annoté :
- **Taux d'individus appariés**
- **Distributions des paires liées et non liées**

— Avec un échantillon représentatif annoté :
- **Matrice de confusion**
- **Précision** : $\dfrac{VP}{VP+FP}$
- **Rappel** : $\dfrac{TP}{TP+FN}$
- **F1-score** : $2 \cdot \dfrac{précision \cdot rappel}{précision + rappel}$
- **Distribution des paires incorrectes**

3 Principaux outils d'appariement *open source*

Cette section propose une présentation de différents outils d'appariement *open source*. La plupart sont disponibles sous la forme de packages R ou Python. Ils ont l'avantage de s'installer facilement et permettent à tout-un-chacun d'avoir accès immédiatement à un ensemble de fonctions utiles pour faire des appariements, évitant ainsi d'avoir à développer soi-même les méthodes de blocage ou les algorithmes de classification des paires.

Des notebooks d'exemples et de prise en main sont disponibles pour certains de ces outils dans le dépôt GitHub associé à ce document : `https://github.com/InseeFrLab/appariement`

3.1 reclin2 (R)

reclin2 (van der Laan (2022)) est un package R d'appariement développé par CBS, l'institut national de statistiques néerlandais. Il facilite leur mise en œuvre par le biais d'un ensemble de fonctions auxiliaires pour chaque étape du processus, dont une méthode de classification probabiliste via l'approche de Fellegi et Sunter décrite en 2.5.4.

Avantages

reclin2 est développé avec un objectif de performance informatique afin de limiter au maximum les ressources (processeurs et mémoire vive) nécessaires pour réaliser un appariement. Pour atteindre ce but, le package repose notamment sur des algorithmes développés en C++.

En ce qui concerne la prise en main du package, des fonctions auxiliaires sont disponibles pour faciliter toutes les étapes d'un appariement et elles se combinent bien entre elles, ce qui en fait un outil assez complet et plutôt clés en main si l'objectif est de réaliser un appariement via la méthode probabiliste. Pour les autres méthodes de classification, il est nécessaire d'écrire le code soi-même ou de faire appel à d'autres packages (pour classer les paires en utilisant des algorithmes de *machine learning* spécifiques par exemple).

Inconvénients

Le package propose plusieurs exemples de mise en œuvre sur lesquels s'appuyer pour démarrer mais pour aller plus loin, la documentation manque parfois de précision. Il peut donc être nécessaire de fouiller un peu pour trouver des informations sur les arguments supplémentaires de certaines fonctions ou la méthodologie sous-jacente.

Par ailleurs, comme la plupart des packages R et Python, la mémoire vive va limiter le volume des données qu'il est possible d'apparier via **reclin2**. À titre d'exemple, il a fallu un peu plus de 30Go de mémoire vive pour apparier deux fichiers de 100 000 lignes sur 5 variables identifiantes, avec un blocage limitant le nombre de paires à 247 millions, des comparaisons floues (Jaro-Winkler) sur deux champs et des comparaisons exactes sur les trois autres.

Cas d'usage

reclin2 est l'outil le plus pratique pour réaliser en R des appariements (notamment par l'approche probabiliste) sur des fichiers n'excédant pas un million de lignes.

3.2 fastLink (R)

fastLink (Enamorado, Fifield, et Imai (2020)) est un package R permettant d'apparier des données via l'approche probabiliste. La méthode sous-jacente est détaillée dans Enamorado, Fifield, et Imai (2019).

Avantages

fastLink propose une mise en œuvre optimisée de la méthode d'appariement probabiliste, notamment plus rapide que celles du package Python **recordLinkage** et du package R **reclin2**. À titre d'exemple, il a fallu environ 30 secondes pour apparier deux fichiers de 10 000 lignes sans blocage avec **fastLink**, contre 2 minutes avec **reclin2**. [3]

Par ailleurs, le modèle probabiliste mis en œuvre dans ce package est un peu plus flexible que celui décrit initialement par Fellegi et Sunter. Les variables de comparaison n'ont en effet pas besoin d'être binaires mais peuvent avoir plusieurs modalités (correspondant à différents niveaux de concordance). Il est également possible de prendre en compte la fréquence des modalités pour comparer certains champs (de façon à ce qu'une concordance sur un nom rare donne plus de poids par exemple).

Inconvénients

L'inconvénient majeur de ce package concerne l'étape d'indexation. Les méthodes de blocage proposées sont en effet trop restrictives, ce qui rend le package

3. Le test a été effectué sur une machine virtuelle comportant 8 CPU et 16 Go de RAM. L'appariement portait sur 6 variables : prénom, nom de famille, libellé d'adresse, date de naissance, code postal, libellé de commune. Les 3 premières variables ont été comparées grâce à une similarité de Jaro-Winkler et les 3 dernières de façon exacte.

difficilement utilisable dans de nombreux cas.

Plus précisément, l'algorithme de classification des paires est estimé et appliqué sur chaque bloc séparément. Cette stratégie n'est pas standard et convient uniquement si les blocs sont peu nombreux et de tailles équivalentes. Ce cas n'est cependant pas le plus fréquent et présente plusieurs limites :

— Si la clé de blocage possède des modalités peu représentées, comme avec un code commune de résidence ou une année de naissance par exemple, certains blocs seront trop restreints pour que les paramètres estimés par l'algorithme aient un sens.
— Si le nombre de blocs devient grand, il devient coûteux en temps de traitement d'estimer des paramètres sur chacun des blocs.
— Il n'est pas forcément souhaitable d'obtenir des paramètres différents pour chaque bloc. En effet, cela pourrait conduire à une décision d'appariement différente pour deux paires d'individus présentant les mêmes motifs de comparaison (mêmes nom et prénoms, mais date et commune de naissance différentes par exemple), en fonction du bloc dans lequel elles se trouvent.

Par ailleurs, il est impossible de définir des règles de blocage complexes. En particulier, l'usage de conditions logiques (ET, OU) dans les règles de blocage est usuel, mais pas prévu au sein du package.

Cas d'usage

fastLink, dans la version ayant été testée (v0.6), conviendra pour des appariements de données peu volumineuses, sans blocage ou avec des clés de blocage très simples. Cependant, **reclin2** ou les packages d'appariement probabiliste en Python couvrent un ensemble plus large de cas d'usage.

3.3 recordLinkage (Python)

recordLinkage (De Bruin (2022)) est un package Python se présentant comme une boîte à outils complète pour les appariements. Il propose un ensemble de fonctions pour mener à bien chacune des étapes d'un appariement : méthodes d'indexation, fonctions de comparaison des champs et algorithmes de classification.

Avantages

Parmi les outils présentés ici, le package **recordLinkage** est celui qui propose le plus de méthodes différentes. Il donne en effet la possibilité de procéder à des appariements de façon déterministe (y compris via du *machine learning*) ou probabiliste. En outre, il permet de comparer les champs par l'intermédiaire de toutes les distances communes avec une syntaxe unique. Du côté de l'indexation, différentes

fonctions sont mises en œuvre pour créer ses clés de blocage sur mesure (via des conditions logiques ET/OU), et l'approche du voisinage trié est également prise en charge.

Au-delà de son côté très complet, le package présente une série d'autres avantages :

— une syntaxe agréable et un partitionnement clair des différentes étapes d'un appariement,
— de la flexibilité, puisqu'il est possible de définir soi-même de nouvelles façons d'indexer ou de classer les paires (au prix d'un investissement plus important en temps de développement puisqu'il est nécessaire d'écrire un peu plus de code),
— et une documentation claire.

Inconvénients

Le seul véritable inconvénient de ce package est sa résistance à l'augmentation du volume. Passé un certain volume, la mémoire vive nécessaire et le temps de calcul deviennent rédhibitoires. Le volume critique dépend des ressources informatiques disponibles, mais il est probable de rencontrer des soucis au-delà de 100 000 lignes par fichier.

Cas d'usage

recordLinkage est donc idéal pour des appariements de fichiers de taille moyenne. Il est facile à prendre en main et donc bien adapté à un utilisateur novice en appariements.

3.4 dedupe (Python)

dedupe (Gregg et Eder (2022)) est un package Python destiné en premier lieu à effectuer du dédoublonnage. Le cas de l'appariement est tout de même pris en charge, ces deux problématiques étant très proches. **dedupe** se décline également sous la forme d'un outil d'appariement avec une interface graphique, nommé Dedupe.io. Celui-ci repose sur les mêmes méthodes que le package, mais son utilisation est payante.

dedupe adopte une approche ambitieuse reposant sur du *machine learning*. Il se démarque clairement de tous les autres outils, en particulier sur deux points : des règles de blocage apprises automatiquement et une annotation de paires par *active learning*.

Le principe de l'*active learning* est décrit au paragraphe 2.5.3. Concrètement, le package **dedupe** nécessite d'annoter un ensemble de paires, mais ce n'est pas

l'utilisateur qui choisit les paires à annoter. L'*active learning* consiste en effet à laisser ce choix au modèle, qui va proposer à l'annotation les paires sur lesquelles il est le plus incertain. Ce procédé permet de maximiser l'information obtenue pour un nombre de paires annoté donné.

La seconde particularité de ce package concerne l'étape d'indexation, qui est automatisée. **dedupe** teste un grand nombre de règles de blocage différentes. Les règles retenues sont définies de façon à limiter au maximum le nombre de paires conservées, tout en excluant le moins possible de paires ayant été annotées positivement. Les règles de blocage testées peuvent être simples (égalité d'un champ) ou plus raffinées (concordance des premiers caractères d'un champ, distance de Levenshtein inférieure à un seuil, etc.).

Avantages

La méthode, sur le papier, est intéressante. Elle demande de faire peu voire aucun choix de paramètres. Il suffit surtout d'avoir ne serait-ce qu'un peu de temps à accorder à l'annotation.

Inconvénients

Cette approche innovante s'accompagne d'un côté « boîte noire » : le déroulé de l'appariement n'est pas aussi clair qu'avec des outils plus classiques.

Par ailleurs, à l'usage, le package souffre d'une grande variabilité dans les temps de calcul, y compris à taille de fichiers fixée. Le calcul des règles de blocage optimales peut en effet être assez rapide comme très long, sans que l'utilisateur n'aie vraiment d'emprise là-dessus. Cela ne posera pas de problème majeur pour apparier des fichiers de 10 000 lignes, mais cela devient gênant au-delà de 100 000 lignes.

Enfin, la méthode utilisée oblige à annoter des paires, et pour profiter des avantages de l'*active learning*, à les annoter au moment de l'appariement et non en amont. Une intervention humaine est donc nécessaire au cours du processus. En revanche, celle-ci peut s'avérer assez rapide puisqu'il est possible d'obtenir de bons résultats en annotant un petit nombre de paires (50 paires devraient suffire pour des fichiers de 10 000 lignes).

Cas d'usage

dedupe est utile pour obtenir de premiers résultats rapidement sur des fichiers de taille moyenne. Il est relativement facile à utiliser et l'*active learning* rend le processus d'annotation efficace. Les résultats sont toutefois peu reproductibles, et il paraît plus adapté à des appariements ponctuels qu'à un processus régulier.

3.5 splink (Python)

splink (Linacre (2022)) est un package Python dédié aux appariements probabilistes et qui fonctionne sur de grands volumes. La méthodologie est celle de l'algorithme du package R **fastLink** présenté ci-dessus, avec plus d'options de paramétrisation. Initialement, **splink** reposait sur Spark, un moteur de traitement de données distribué qui permet d'effectuer efficacement des opérations en parallèle sur de grands volumes de données. Par la suite, il est devenu possible d'utiliser d'autres systèmes de manipulation de données comme AWS Athena pour les données massives ou DuckDB pour les données qui tiennent en mémoire (jusqu'à 1 à 2 millions de lignes pour un ordinateur personnel standard).

Avantages

L'avantage majeur de **splink** est sa capacité à gérer de gros volumes de données. Parmi les packages R et Python, c'est le seul à pouvoir apparier deux fichiers de quelques millions de lignes.

En outre, le package est en développement actif. Des améliorations et de nouvelles fonctionnalités sont régulièrement ajoutées. La documentation est assez riche et aussi régulièrement mise à jour.

Enfin, **splink** propose un degré élevé de paramétrisation. Le package laisse en effet une flexibilité intéressante pour la définition des règles de blocage. De plus, le modèle probabiliste de **splink**, comme celui de **fastLink** dont il est inspiré, laisse la possibilité de définir plusieurs niveaux de comparaison pour chaque variable au lieu d'une règle binaire et propose une option pour ajuster les estimations grâce à la fréquence des modalités.

Inconvénients

L'utilisation de **splink** sur des fichiers volumineux nécessite de configurer un moteur de traitement de données massives comme Spark au préalable, ce qui le rend moins accessible que les autres packages.

Cas d'usage

splink est un excellent choix pour un appariement probabiliste en Python, aussi bien sur des données très volumineuses que sur des fichiers de taille plus raisonnable. Pour des fichiers dépassant quelques centaines de milliers de lignes, c'est même la seule option viable, avec l'utilisation d'ElasticSearch.

3.6 ElasticSearch

ElasticSearch est un moteur de recherche textuelle. C'est un outil conçu pour renvoyer rapidement des résultats lors de recherches dans un ensemble de textes volumineux. Ses cas d'usage classiques sont la recherche de produits sur un site web de e-commerce ou encore le stockage et l'analyse des logs sur un site web. ElasticSearch peut aussi être mis à profit pour apparier des données, notamment en présence de fichiers volumineux. Son utilisation dans ce cadre est détaillée dans la section 2.5.5.

D'un point de vue pratique, l'un des deux fichiers (en général le plus volumineux) est indexé dans ElasticSearch. Puis chaque individu du second fichier est recherché dans l'index, grâce à une requête qui définit les champs qui doivent correspondre et le type de correspondance (exacte ou approchée via différentes méthodes). Un exemple de requête est proposé à la fin de cette section. Les requêtes pour chaque individu sont indépendantes et peuvent être effectuées en parallèle.

Les manipulations d'ElasticSearch peuvent être réalisées via une API. Il existe également des packages R et Python pour travailler directement dans ces langages. Il reste toutefois nécessaire de disposer d'un cluster ElasticSearch au préalable.

Avantages

ElasticSearch est conçu pour traiter de gros volumes de données efficacement, ce qui rend possible l'appariement de fichiers de plusieurs millions de lignes.

Sur le plan de la méthodologie d'appariement, les requêtes ElasticSearch sont très flexibles. Une requête permet de spécifier des critères obligatoires pour qu'un individu fasse partie des résultats de la recherche (ce qui correspond à l'indexation dans un processus d'appariement classique), ainsi que des critères facultatifs, qui jouent seulement sur le score associé à l'individu (ce qui s'apparente à la comparaison des champs identifiants). Il existe de multiples manières de définir les critères, ce qui permet réellement d'adapter la requête à ses données et à ses besoins.

Par ailleurs, la recherche est optimisée. Ainsi, une fois l'un des deux fichiers indexé dans ElasticSearch (ce qui peut prendre du temps), les calculs sont rapides.

Inconvénients

L'installation d'ElasticSearch est un peu moins immédiate que celle des packages R et Python présentés ci-dessus. De plus, il est nécessaire de configurer le *cluster*[4] avant de pouvoir l'utiliser.

4. Un cluster ElasticSearch est un ensemble de noeuds, dont chacun correspond à une instance d'ElasticSearch. Le fonctionnement en *cluster* permet de paralléliser les opérations et de répartir la charge sur plusieurs machines.

La prise en main peut paraître complexe, d'une part pour la configuration au départ, d'autre part pour écrire les requêtes. L'approche est en effet assez différente des outils classiques d'appariement et les possibilités sont nombreuses. Il est donc nécessaire de passer du temps sur la documentation pour tirer pleinement partie d'ElasticSearch.

En ce qui concerne la performance de l'outil lui-même, le temps d'indexation au début du processus d'appariement peut annuler l'intérêt de la rapidité d'exécution des requêtes. Cela peut être le cas pour de petits fichiers ou lorsque l'utilisateur demande trop d'opérations lors de l'indexation (comme le calcul de tous les n-grammes pour différents valeurs de n, la combinaison de plusieurs champs ou des opérations de modifications complexes reposant sur des expressions régulières).

Enfin, ElasticSearch est peu adapté aux calculs des distances usuelles en appariement (Levenshtein, Jaro-Winkler). Il est possible d'effectuer des recherches avec une distance de Levenshtein maximale fixée, mais cela peut vite faire exploser le temps de calcul. L'outil gère beaucoup mieux en revanche les comparaisons exactes ou à base de n-grammes, ce qui peut forcer à revoir sa stratégie dans les requêtes.

Cas d'usage

ElasticSearch est particulièrement adapté à des appariements de fichiers volumineux. Il est même l'un des rares outils permettant d'apparier des fichiers de plusieurs millions de lignes. Parmi les cas d'usage possibles, celui de l'appariement de plusieurs fichiers différents à un même fichier de référence prend tout son sens avec ElasticSearch, car l'indexation est mutualisée.

Exemple de requête

```
{
    "query":{
        "bool":{
            "must":[
                {
                    "match":{
                        "date_naissance":"19641213"
                    }
                },
                {
                    "multi_match":{
                        "query":"Gousselot",
                        "fields":[
                            "nom_naissance",
                            "nom_marital"
                        ]
                    }
                },
                {
                    "match":{
                        "prenom":{
                            "query":"Anne-Claire",
                            "fuzziness":2
                        }
                    }
                }
            ]
        }
    }
}
```

Cette requête renvoie tous les individus de l'index dont la date de naissance est exactement "19641213", dont le nom de naissance ou le nom marital est exactement Gousselot et dont le prénom est à une distance de Levenshtein maximale de 2 de "Anne-Claire".

Il n'est pas nécessaire d'écrire une requête par individu à identifier, il est tout à fait possible d'utiliser la même requête pour tous en remplaçant les exemples de nom, prénom et date de naissance utilisés ici par des variables.

3.7 Logiciels d'instituts nationaux de statistiques

Compte tenu de l'importance que revêt ce sujet pour les instituts nationaux de statistiques, il n'est pas étonnant que certains d'entre eux aient développé leurs propres outils d'appariement. Ainsi, l'institut italien Istat a développé le logiciel **RELAIS** (Tuoto, Cibella, Fortini, et Scannapieco (2010)), tandis que Statistique Canada est à l'origine du logiciel **G-Link** (Chevrette (2011)).

Ces deux outils présentent de nombreuses similitudes dans leur philosophie. Il s'agit avant tout de logiciels d'appariement probabiliste, qui fonctionnent comme des boîtes à outils permettant de traiter toutes les étapes d'un projet d'appariement, de la préparation des données à l'évaluation de la qualité. Les deux logiciels sont accompagnés d'une interface graphique et incluent de nombreuses options de paramétrisation de l'appariement de façon à pouvoir tenir compte de la diversité de cas d'usage rencontrés au sein de ces instituts.

RELAIS est un projet *open source*, il est donc facilement utilisable. En revanche, il est à réserver à des appariements de fichiers peu volumineux, au maximum à l'échelle d'un département. **G-Link** n'est pas *open source*, Statistique Canada le distribue de façon commerciale.

3.8 Outils non spécifiques aux appariements

L'utilisation d'outils dédiés aux appariements est pratique mais pas indispensable. Ainsi, certains packages plus généraux, mettant en œuvre notamment des fonctions de calcul de distance, peuvent permettre de développer des solutions *ad hoc*. Par exemple, un appariement par la méthode du plus proche écho (détaillée au paragraphe 2.5.2) se prête bien à l'utilisation d'outils génériques.

L'avantage de ces outils est de sortir du cadre imposé par les packages spécialisés d'appariement. Libéré de cette contrainte, il devient possible, à condition d'y passer du temps, de développer des solutions uniques pouvant répondre à des besoins particuliers qui ne seraient pas couverts par les outils d'appariement existants. Par ailleurs, de par leur caractère générique, ces outils possèdent en général une plus grande communauté d'utilisateurs, ce qui permet de trouver plus facilement des réponses à ses interrogations.

Comparaison des champs

La grande majorité des appariements nécessitent de calculer des distances entre chaînes de caractères. Les distances usuelles sont proposées dans différents packages, à la fois en R et Python. Inutile donc de développer soi-même ces fonctions.

En Python, le package **rapidfuzz** propose une mise en œuvre optimisée de la distance de Levenshtein, tandis que **jellyfish** offre un choix assez large de distances,

dont celle de Jaro-Winkler.

En R, le package **stringdist** permet lui-aussi de calculer différents types de distances.

Nettoyage et normalisation

Les étapes de nettoyage et de normalisation des fichiers à apparier sont généralement aussi des traitements standard. Ainsi, de nombreux packages facilitent la réalisation d'opérations usuelles : suppression des *stopwords* et des caractères spéciaux, passage des caractères en minuscule, etc.

Les packages Python **nltk** et **spacy** sont des outils très complets pour le traitement du langage.

En R, le package **tm** permet d'effectuer ces opérations.

Évaluation de la qualité

Enfin, il peut être utile de s'appuyer sur des packages dédiés au calcul de mesures de performance des modèles de classification binaire, telles que la précision et le rappel. Les packages de *machine learning* **scikitlearn** en Python et **caret** en R remplissent cet objectif et permettent de tracer des graphiques, par exemple en vue de choisir un seuil, à condition de disposer d'un échantillon de paires annotées.

4 Conseils pratiques

4.1 Choix méthodologiques décisifs

Réaliser un appariement peut être déstabilisant tant les choix possibles d'outils et de méthodes sont divers, d'autant plus que la plupart de ces choix sont interdépendants. Cette partie présente un ensemble de questions essentielles, chacune associée à des éléments de réponse. Les choix les plus déterminants sont présentés en premier.

4.1.1 Quel algorithme de classification des paires choisir ?

Une tâche d'appariement consiste avant tout à classer des paires. La sélection de l'algorithme de classification est donc fondamentale, et elle influence largement les décisions suivantes.

La **méthode probabiliste** repose sur une théorie un peu plus complexe que les autres, mais elle est la plus « clés en main ». En effet, elle ne nécessite pas d'annoter un échantillon de paires et peut ainsi fournir un résultat immédiat. La limite du volume peut cependant être problématique car la plupart des outils mettant en oeuvre cette méthode éprouvent des difficultés avec les volumes importants.

Si l'objectif est de développer soi-même un outil pour un cas d'usage identifié à l'avance, les méthodes des **tours de clés successifs** et du **plus proche écho** sont sans doute les plus adaptées. Elles sont les plus simples à développer en partant de zéro et fournissent de très bons résultats si les différents paramètres sont bien ajustés.

Si des paires annotées sont disponibles, il est intéressant de tester du ***machine learning***. Des librairies très complètes existent dans la plupart des langages de programmation pour appliquer les principaux algorithmes d'apprentissage supervisé. En raison du fort déséquilibre entre les exemples positifs (les paires d'individus identiques) et les exemple négatifs (les paires d'individus différents), il sera toutefois nécessaire de choisir un algorithme gérant correctement le déséquilibre des classes ou d'adapter l'apprentissage en conséquence (par exemple en ajoutant des poids sur les exemples positifs ou en ajustant certains hyperparamètres).

Par ailleurs, **le caractère ponctuel ou régulier de l'appariement est un élément clé de la décision**. Si l'appariement a vocation a être répété, les méthodes qui demandent le moins d'ajustements manuels, comme l'approche probabiliste ou les méthodes de *machine learning*, sont à privilégier.

De plus, s'il s'agit d'apparier régulièrement différents fichiers à un référentiel qui évolue peu, les moteurs de recherche textuelle comme **ElasticSearch** s'avèrent particulièrement adaptés.

Encadré 5 : Appariement probabiliste ou déterministe : évidences empiriques

Afin de comparer de façon pratique les méthodes d'appariement, des tests ont été menés dans un article présenté lors des Journées de Méthodologie Statistique de l'Insee (Haag, Koumarianos, et Malherbe (2022)).

Il s'agissait d'apparier une source fiscale, le fichier d'imposition des personnes, à l'enquête annuelle de recensement 2019. En raison des limitations de certains outils liées au volume des données, les tests ont porté sur deux départements : l'Ille-et-Vilaine (35) et la Lozère (48).
Trois outils ont été comparés :

- Le package Python **recordlinkage**, et plus particulièrement sa méthode probabiliste ;
- L'outil d'appariement probabiliste **RELAIS**, développé par l'institut italien de statistiques Istat ;
- **Rapsodie**, un outil interne de l'Insee qui met en œuvre des appariements déterministes par la méthode du plus proche écho (via une somme pondérée des mesures de similarité calculées sur les différents champs identifiants).

Les trois méthodes conduisent à des taux d'appariement similaires, situés entre 90 et 95 % pour les deux départements. Par ailleurs, l'examen visuel d'un échantillon de paires fait ressortir un taux de faux positifs légèrement plus faible pour l'appariement via Rapsodie. Il faut toutefois noter que cet outil est conçu pour effectuer des appariements avec les sources fiscales, et que de nombreux ajustements ont eu lieu pour prendre en compte les spécificités de ces données.

En conclusion, cette expérimentation suggère que l'approche probabiliste fournit des résultats très corrects, et de façon rapide puisque les principaux paramètres sont estimés par le modèle. Cependant, il est sans doute possible d'obtenir des résultats légèrement meilleurs avec une méthode déterministe aux paramètres ajustés avec soin et avec des prétraitements adaptés aux deux fichiers à apparier.

4.1.2 Comment réduire le nombre de paires à traiter ?

Quelle que soit la méthode de classification adoptée, il n'est jamais possible de comparer précisément l'ensemble des paires (hormis avec des jeux de données très réduits). Il faut donc en rejeter d'emblée la majorité afin de se concentrer sur celles qui correspondent le plus probablement à des individus identiques. Cette sélection a un impact critique sur la qualité de l'appariement. Le danger est de créer des faux négatifs en rejetant des paires qui auraient dû être liées.

Cette sélection de paires potentielles s'effectue le plus souvent grâce à des règles de blocage, dont le principe est détaillé dans la section 2.3. La connaissance des données à apparier est essentielle pour définir ces règles au mieux. Une bonne variable de blocage doit être :

— **suffisamment discriminante**, de façon à ce que la taille des blocs reste raisonnable ;
— **d'excellente qualité**, pour éviter de créer des faux négatifs.

Il est souvent judicieux de **combiner plusieurs variables de blocage** pour éviter de donner trop d'importance à une seule d'entre elles, surtout si la qualité des variables est douteuse.

Une fois ces variables de blocage potentielles identifiées, il est pratique de **procéder par itération**. L'idée est de démarrer par un blocage relativement souple (qui conserve un grand nombre de paires), puis de le rendre plus strict si l'appariement est trop long ou trop consommateur de ressources informatiques.

4.1.3 Quels prétraitements effectuer sur les données ?

La qualité du résultat d'un appariement dépend avant tout de celle des données. En effet, dans un cas extrêmement favorable, avec des données de parfaite qualité, il suffirait d'un appariement exact. Tous les prétraitements pouvant être appliqués pour **mettre en qualité les données *ex post*** sont donc bons à prendre.

Certains traitements sont standards et applicables à la plupart des appariements. Des exemples sont proposés dans l'encadré de la section 2.2.

Cependant, les opérations les plus importantes à effectuer sont **celles qui vont corriger les erreurs courantes rencontrées dans les fichiers à apparier**, ce qui est forcément très dépendant des données manipulées. Il peut s'agir d'un champ avec des valeurs manquantes, de l'utilisation d'une modalité par défaut ou encore d'une inversion régulière de deux champs. Une bonne connaissance des données est donc là-aussi primordial pour prendre les bonnes décisions.

4.1.4 Quelles mesures de comparaison utiliser ?

Afin de classer les paires, il est nécessaire dans la plupart des approches de disposer pour chaque champ identifiant d'une mesure de la proximité des valeurs.

Les champs peuvent être comparés de manière **exacte** ou de manière **floue** (par l'intermédiaire de calculs de distance), et c'est ce choix pour chaque champ disponible qui est déterminant.

En effet, sur le plan méthodologique, **certains champs ne se prêtent pas à une comparaison floue**. En particulier, pour les codes ou les champs composés uniquement de chiffres comme un code postal ou une date de naissance, une omission ou une interversion de caractères peut conduire à une modalité valide mais très différente de la valeur initiale. Dans ce cas, une comparaison floue peut rapprocher de façon abusive des modalités qui ne devraient pas l'être. Par exemple, les dates 19270801 et 19870201 sont proches selon les mesures de similarité classiques, mais elles racontent deux histoires différentes. Pour ce type de champs, il vaut mieux s'en tenir par défaut à des comparaisons exactes, sauf si l'analyse des données à apparier a montré que la distribution des erreurs courantes s'y prêtait.

Sur le plan informatique, **les calculs de distance sont beaucoup plus consommateurs de ressources que les comparaisons exactes**. Si les fichiers à apparier sont volumineux, il est nécessaire de sélectionner soigneusement une poignée de champs qui bénéficieront de comparaisons floues (en privilégiant en général les champs de nom et prénom qui souffrent assez souvent d'erreurs orthographiques).

Pour les comparaisons floues, **le choix de la mesure de similarité** est une question naturelle, mais **l'expérience prouve qu'elle n'est pas déterminante**. Les différences sont minces entre les distances les plus communes (Levenshtein, Jaro-Winkler, etc.). Ainsi, il n'est pas forcément pertinent de passer du temps sur ce point.

4.1.5 Quelles paires lier ?

La plupart des algorithmes de classification fournissent un score pour chaque paire. Il peut prendre différentes formes (une probabilité par exemple) mais il représente toujours une mesure agrégée de la correspondance d'une paire. La dernière question consiste ainsi à **décider d'un seuil au-delà duquel les paires seront liées**.

Ce choix a une grande influence sur les résultats de l'appariement. **Plus le seuil est bas, plus l'appariement devient "permissif"**, avec un grand nombre de paires liées. Si le seuil choisi est trop bas, le rappel (comme défini dans la section

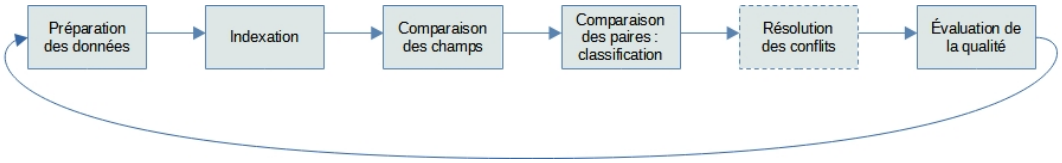
2.7) sera bon puisqu'un grand nombre de paires d'individus identiques seront effectivement liées, mais il y aura également de nombreux faux-positifs. Inversement, **si le seuil est trop haut, l'appariement devient trop strict**, les paires ne sont liées que lorsque le score est excellent. Cela conduit à une meilleure précision, mais aussi à des faux négatifs, c'est-à-dire à des paires d'individus identiques qui ne sont pas repérées comme telles.

Avec un échantillon de paires annotées, il est possible de faire varier le seuil afin d'obtenir le meilleur résultat. Ce choix est propre à chaque appariement. En effet, le critère d'optimalité **dépend de l'objectif recherché et du type d'erreur à minimiser** (les faux positifs ou les faux négatifs).

4.2 Réussir un appariement prend du temps

La réussite d'un appariement tient à deux facteurs principaux :
— la qualité des données ;
— le temps passé à ajuster les différents paramètres et à prendre en compte la particularité des fichiers appariés.

Il est souvent impossible d'influer sur le premier point, le conseil est donc d'**accepter d'y passer du temps**.



GRAPHIQUE 9 – **Processus d'appariement itératif**

Il est tentant de s'en remettre à une méthode entièrement automatique. Toutefois, aucun algorithme ne peut être parfaitement adapté à tous les fichiers. La recherche d'un résultat rapide passe forcément par un compromis sur la qualité de l'appariement.

À l'inverse, si l'objectif est d'aboutir au meilleur résultat possible, l'appariement prend alors la forme d'un **processus itératif**. L'analyse des résultats permet de repérer à chaque itération les erreurs les plus fréquentes et de les corriger. Les pistes d'amélioration ainsi identifiées peuvent être très diverses : une meilleure normalisation pour certaines sous-populations, un paramètre à ajuster dans un sens ou dans l'autre, etc.

Pour réaliser ce travail, il est important de disposer d'indicateurs de qualité fiables. En effet, sans indication sur les paires classées de façon incorrecte, il est

impossible de repérer les erreurs et de rentrer dans un travail d'amélioration incrémentale de l'appariement. Ce constat conduit au deuxième conseil.

4.3 Prendre au sérieux l'évaluation de la qualité

Garder un regard critique sur l'appariement est essentiel, et il ne faut surtout **pas avoir une confiance aveugle en ses résultats**. En particulier, il est dangereux de s'en remettre uniquement au taux d'appariement, qui peut facilement donner une fausse impression de qualité.

Une première piste peu coûteuse pour compléter le taux d'appariement est de vérifier que la **distribution des individus appariés** est cohérente avec celle des fichiers initiaux.

Cependant, le plus souvent, seul **le recours à un échantillon de paires annotées** permet d'avoir une réelle idée de la qualité de l'appariement. Constituer cet échantillon prend du temps, mais le processus peut être rendu plus efficace par une interface mettant visuellement en évidence les différences au sein des paires.

La sélection des paires à annoter mérite également de la réflexion. Un tirage aléatoire uniforme entraînerait une proportion écrasante d'exemples négatifs et très faciles à classer. Si la méthode de classification choisie associe un score (ou une probabilité) à chaque paire, il est plus intéressant de procéder à un **tirage stratifié sur ce score**, afin d'inclure à la fois des exemples positifs et négatifs, ainsi que des paires plus ou moins faciles à classer.

Enfin, calculer des indicateurs de qualité fiables permet aussi de définir un critère d'arrêt dans l'optimisation d'un appariement : soit lorsque les améliorations incrémentales deviennent minimes, soit lorsque des seuils de qualité (sur la précision et le rappel par exemple) fixés à l'avance ont été atteints.

4.4 Prendre en compte les principaux cas particuliers

Comme évoqué dans la section 4.1.4, le choix de la mesure de similarité pour comparer les champs ne fait pas vraiment de différence. En revanche, il est possible d'affiner la comparaison des champs en prenant en compte les principaux cas particuliers des fichiers à apparier. En effet, le calcul de distances ou de similarités ne permet pas de gérer tous les cas. En voici quelques exemples.

Noms composés Les prénoms composés par exemple peuvent causer des erreurs atypiques. Dans les exemples de paires « Éloi / Elliot » et « Anne / Anne-Claire » de la section 2.4, quelle est l'erreur la plus probable ? C'est sans doute la seconde,

avec l'oubli de la deuxième partie du prénom composé. Pourtant, la première paire obtient un meilleur score de similarité avec les mesures classiques. Pour pallier cette incohérence, il est possible et intéressant d'adapter la comparaison des prénoms pour moins pénaliser l'oubli d'une partie d'un prénom composé.

Plusieurs prénoms Le conseil précédent est également valable lorsque plusieurs prénoms sont disponibles dans l'un des deux fichiers. Il est judicieux dans ce cas de les comparer séparément, de façon à mieux apparier les individus qui utilisent leur deuxième ou troisième prénom comme prénom d'usage.

Interversion nom-prénom L'interversion du nom et du prénom est un autre cas d'erreur classique. La comparaison champ-par-champ d' « Éloi Martin » à « Martin Éloi » conduira sans doute à rejeter la paire, alors qu'il s'agit probablement du même individu. Ajouter une comparaison croisée prenom A - nom B et prenom B - nom A rend la méthode d'appariement plus robuste (au prix cependant d'un coût computationnel plus élevé).

Nom marital L'idée est la même pour la gestion des noms maritaux. Il est fréquent de retrouver un nom marital à la place d'un nom de naissance, ou inversement. La prise en compte des deux noms dans les comparaisons permet d'éviter quelques erreurs.

Dates Les cas particuliers concernent aussi les dates. Ainsi, il peut être utile d'appliquer un traitement particulier au 1er janvier car il s'agit souvent d'une valeur par défaut lorsque la date est manquante. En ce qui concerne les années, il peut y avoir des erreurs de siècle lorsqu'elles sont renseignées sur deux chiffres, ce qui se traduit par une valeur en 19** au lieu de 20** par exemple.

Codes communes Enfin, il est important de connaître l'historique de ses données. En particulier, en France, les changement de codes communes sont fréquents, ce qui peut occasionner des erreurs si les fichiers ont été produits à des périodes différentes. Par exemple, pour identifier des individus nés à Malakoff avant 1968, il est intéressant de tester le code commune 75047 en plus du code actuel 92046.

4.5 Les appariements de fichiers volumineux

Le volume des fichiers appariés est un facteur limitant de nombreux appariements. Le volume critique dépend des choix techniques effectués, mais certains

outils atteignent leurs limites dès 100 000 lignes. À partir d'un million de lignes, il faut la plupart du temps prendre des mesures spécifiques.

Les limitations se manifestent principalement de deux manières, qui sont liées :

— **la mémoire vive (RAM)** : lorsque des données volumineuses sont chargées en mémoire ou s'il y a trop d'opérations simultanées, la mémoire vive peut être saturée. Cela se traduit par des ralentissements, voire une terminaison brutale du programme avec une erreur. Ce type de problème concerne particulièrement les outils qui reposent sur du code R ou Python, puisque ces deux langages fonctionnent essentiellement en mémoire vive.

— **le temps de calcul** : plus de lignes implique plus d'opérations (des calculs de distance par exemple) donc un temps d'exécution plus long et pouvant devenir rédhibitoire.

Quelques pistes existent tout de même pour mener à bien un appariement de fichiers volumineux.

Rendre l'indexation plus stricte

Pour limiter la charge informatique, il faut souvent ajuster l'étape d'indexation en conservant moins de paires potentielles. Il faut alors définir des règles plus strictes, au risque de créer des faux-négatifs.

Si cette modification crée trop de faux-négatifs, une option est d'effectuer plusieurs tours d'appariement en écartant pour les tours suivants les individus déjà appariés. Le premier tour est effectué avec l'indexation la plus stricte, puis les contraintes sont relâchées progressivement pour accepter plus de paires. Le fait de ne pas considérer toutes les paires de façon simultanée peut cependant créer des faux-positifs : il peut arriver d'apparier un individu lors du premier tour alors qu'une meilleure correspondance aurait été trouvée dans un tour ultérieur.

Limiter les comparaisons floues

Les calculs de distance sur des chaînes de caractères sont très consommateurs de ressources informatiques et sont nettement plus longs que des comparaisons exactes. Sur des fichiers volumineux, il faut donc les limiter au maximum et les réserver aux champs tels que le nom et le prénom.

Utiliser des outils spécifiques pour les gros volumes

Lorsque les ajustements dans le processus d'appariement ne suffisent plus, il faut reconsidérer certains choix techniques et envisager de changer d'outil. Spark constitue par exemple une option intéressante. Spark est un système de calcul

distribué conçu spécialement pour traiter de gros volumes de données rapidement. Il permet d'effectuer des opérations en parallèle et offre une gestion optimisée de la mémoire. Spark peut être utilisé via différents langages de programmation, dont R et Python.

Utiliser un moteur de recherche textuelle

Les appariements de données individuelles ne sont pas le cas d'usage principal des moteurs de recherche textuelle, mais ceux-ci sont conçus pour traiter efficacement de grandes quantités de textes. L'utilisation d'un outil comme **ElasticSearch** (voir section 2.5.5) peut donc permettre de résoudre les problèmes de performance liés aux appariements de fichiers volumineux.

4.6 Privilégier les outils *open source*

La section 3 donne des clés pour choisir un outil en fonction du cas d'usage rencontré. Le principal conseil cependant est d'**opter, si possible, pour une solution *open source***. Ceci est valable quelle que soit l'envergure du projet.

Pour des appariements à usage unique qui n'ont pas vocation à s'intégrer à une chaîne de production, utiliser un outil *open source* permet en général d'aller plus vite puisque qu'il est plus facile d'accès. S'il existe en plus une communauté d'utilisateurs autour de cet outil, il est assez facile de trouver des réponses à ses questions.

Pour des appariements qui doivent être répétés de manière régulière ou qui doivent s'intégrer dans une chaîne de production, l'*open source* est aussi l'idéal. Il permet d'être moins dépendant de l'entité à l'origine de l'outil et sa maintenance est facilitée.

Références

Bleiholder, J., et F. Naumann (2009) : "Data fusion," *ACM computing surveys (CSUR)*, 41(1), 1–41.

Chevrette, A. (2011) : "G-Link : A probabilistic record linkage system," in *NORC Conference Proceedings, May*.

Christen, P. (2012) : "Data matching : concepts and techniques for record linkage, entity resolution, and duplicate detection," *Data-centric systems and applications*.

De Bruin, J. (2022) : "Python Record Linkage Toolkit : A toolkit for record linkage and duplicate detection in Python. Version 0.15," https ://-doi.org/10.5281/zenodo.3559043.

Dempster, A. P., N. M. Laird, et D. B. Rubin (1977) : "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society : Series B (Methodological)*, 39(1), 1–22.

Enamorado, T., B. Fifield, et K. Imai (2019) : "Using a probabilistic model to assist merging of large-scale administrative records," *American Political Science Review*, 113(2), 353–371.

——— (2020) : "fastLink : Fast Probabilistic Record Linkage with Missing Data. Version 0.6.0," https ://github.com/kosukeimai/fastLink.

Fellegi, I. P., et A. B. Sunter (1969) : "A theory for record linkage," *Journal of the American Statistical Association*, 64(328), 1183–1210.

Fortini, M. (2020) : "An Improved Fellegi-Sunter Framework for Probabilistic Record Linkage Between Large Data Sets," *Journal of Official Statistics*, 36(4), 803–825.

Gregg, F., et D. Eder (2022) : "Dedupe. Version 2.0.17," https ://github.com/dedupeio/dedupe.

Haag, O., H. Koumarianos, et L. Malherbe (2022) : "Probabilistes ou Déterministes, des méthodes d'appariement au banc d'essai du programme Résil," *Journées de méthodologie statistique de l'Insee*.

Hastie, T., R. Tibshirani, et J. Friedman (2001) : *The Elements of Statistical Learning*, Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

Jaro, M. A. (1989) : "Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida," *Journal of the American Statistical Association*, 84(406), 414–420.

Kuhn, H. W. (1955) : "The Hungarian method for the assignment problem," *Naval research logistics quarterly*, 2(1-2), 83–97.

Linacre, R. (2022) : "splink : Fast, accurate and scalable probabilistic data linkage. Version 3.5.2," https ://github.com/moj-analytical-services/splink.

Murray, J. S. (2016) : "Probabilistic record linkage and deduplication after indexing, blocking, and filtering," *arXiv preprint arXiv :1603.07816*.

Philips, L. (2000) : "The double metaphone search algorithm," *C/C++ users journal*, 18(6), 38–43.

Russel, R. C. (1918) : "US patent 1261167," .

Settles, B. (2009) : "Active learning literature survey," *Computer Sciences Technical Report*, 1648.

Tuoto, T., N. Cibella, M. Fortini, et M. Scannapieco (2010) : "From theory to practice : the software RELAIS as a solution for record linkage," *Atti della XLV riunione scientifica della Società Italiana di Statistica, Padova : SIS*.

van der Laan, J. (2022) : "reclin2 : Record Linkage Toolkit. Version 0.2.0," https ://github.com/djvanderlaan/reclin2.

ANNEXES

A Théorie des appariements probabilistes

A.1 Calcul de la probabilité estimée

L'introduction de quelques notations est nécessaire pour commencer.

À chaque paire est associé un couple $(\boldsymbol{\gamma}, \boldsymbol{M})$ correspondant à la réalisation de deux variables aléatoires :

- $\boldsymbol{\gamma} = (\boldsymbol{\gamma_1}, ..., \boldsymbol{\gamma_K})$ représente le vecteur de comparaison des paires sur K champs présents dans les deux fichiers à apparier.
- **M** représente le vrai statut de la paire (1 pour une paire d'individus identiques, 0 sinon).

Par exemple, avec des comparaisons exactes,
$\gamma = (\gamma_{nom}, \gamma_{prenom}, \gamma_{date\_nais}, \gamma_{com\_nais}) = (1, 0, 1, 0)$ signifie que les deux individus ont les mêmes noms et dates de naissance, mais des prénoms et communes de naissance différents.

Ce paragraphe est consacré au calcul de $\boldsymbol{P(M = 1|\gamma)}$.

Deux probabilités conditionnelles jouent un rôle fondamental dans le modèle :

$$\boldsymbol{m(\gamma) = P(\gamma|M = 1)}$$
$$\boldsymbol{u(\gamma) = P(\gamma|M = 0)}$$

- m mesure la qualité d'un champ identifiant.
  Par exemple, $m_{prenom}(1) = 0,9$ signifie que des erreurs surviennent 1 fois sur 10 sur le champ *prénom*.
- u mesure la probabilité d'observer la même valeur sur un champ identifiant par hasard. La probabilité u dépend de la cardinalité de chaque champ. Par exemple, $u_{mois\_nais} \approx 1/12$ tandis que u_{nom} sera très faible.

Dans la théorie classique de Fellegi-Sunter, les composantes du vecteur de comparaison sont binaires : pour une variable i, γ_i vaut 0 ou 1. Cela implique donc $m_i(0) = 1 - m_i(1)$ et $u_i(0) = 1 - u_i(1)$.

Afin de bien comprendre comment est calculée la probabilité $P(M = 1|\gamma)$, il est intéressant de considérer les cas extrêmes dans lesquels il n'y a aucun ou un seul champ identifiant disponible pour effectuer l'appariement. Ces cas sont peu réalistes mais permettent d'assimiler les notations introduites précédemment.

En l'absence de champ identifiant, le vecteur de comparaison γ est vide et la probabilité devient $P(M = 1)$. **Cette quantité est notée $\boldsymbol{\lambda}$**. Elle correspond à la proportion de paires d'individus identiques parmi l'ensemble des paires, de l'ordre de $1/n$. **Il s'agit de l'*a priori***.

Avec un unique champ identifiant, par le théorème de Bayes :

$$\begin{aligned} P(M = 1|\gamma_1) &= \frac{P(\gamma_1|M = 1) \cdot P(M = 1)}{P(\gamma_1|M = 1) \cdot P(M = 1) + P(\gamma_1|M = 0) \cdot P(M = 0)} \\ &= \frac{m_1(\gamma_1) \cdot \lambda}{m_1(\gamma_1) \cdot \lambda + u_1(\gamma_1) \cdot (1 - \lambda)} \end{aligned}$$

Avant de généraliser à un appariement faisant intervenir plusieurs champs identifiants, une hypothèse fondamentale du modèle de Fellegi-Sunter doit être posée :

Hypothèse d'indépendance conditionnelle. *On suppose que les composantes du vecteur de comparaison γ sont mutuellement indépendantes conditionnellement au statut réel de la paire M.*

Cette hypothèse signifie notamment que les erreurs ne doivent pas être corrélées entre les différentes variables. Elle serait violée par exemple si à la fois l'âge et la date de naissance étaient utilisées comme variables d'appariement, puisqu'une erreur sur la date de naissance serait presque systématiquement associée à une erreur sur l'âge.

L'hypothèse d'indépendance conditionnelle simplifie nettement les calculs et implique notamment :

$$\begin{aligned} m(\gamma) &= m_1(\gamma_1)m_2(\gamma_2)\dots m_K(\gamma_K) \\ u(\gamma) &= u_1(\gamma_1)u_2(\gamma_2)\dots u_K(\gamma_K) \end{aligned}$$

Dans le cas d'un appariement avec deux champs identifiants, une première manière d'envisager les choses est de façon séquentielle :

$$P(M = 1|\gamma_1, \gamma_2) = \frac{\tilde{\lambda} \cdot m_2(\gamma_2)}{\tilde{\lambda} \cdot m_2(\gamma_2) + (1 - \tilde{\lambda}) \cdot u_2(\gamma_2)}$$

avec $\tilde{\lambda} = P(M = 1|\gamma_1) = \frac{\lambda \cdot m_1(\gamma_1)}{\lambda \cdot m_1(\gamma_1) + (1-\lambda) \cdot u_1(\gamma_1)}$

La seconde manière consiste à calculer directement :

$$\begin{aligned} P(M = 1|\gamma_1, \gamma_2) &= \frac{P(M = 1) \cdot P(\gamma_1, \gamma_2|M = 1)}{P(\gamma_1, \gamma_2)} \\ &= \frac{\lambda \cdot m(\gamma_1, \gamma_2)}{\lambda \cdot m(\gamma_1, \gamma_2) + (1 - \lambda) \cdot u(\gamma_1, \gamma_2)} \\ &= \frac{\lambda \cdot m_1(\gamma_1) \cdot m_2(\gamma_2)}{\lambda \cdot m_1(\gamma_1) \cdot m_2(\gamma_2) + (1 - \lambda) \cdot u_1(\gamma_1) \cdot u_2(\gamma_2)} \end{aligned}$$

Finalement, dans le cas général avec K variables identifiantes :

$$P(M=1|\gamma) = \frac{\lambda m_1(\gamma_1)m_2(\gamma_2)\dots m_K(\gamma_K)}{\lambda m_1(\gamma_1)m_2(\gamma_2)\dots m_K(\gamma_K) + (1-\lambda)\cdot u_1(\gamma_1)u_2(\gamma_2)\dots u_K(\gamma_K)}$$

A.2 Les poids

Afin d'interpréter l'impact de chaque champ, il est plus facile de raisonner sur les cotes (*odds* en anglais). Cette transformation de la probabilité fait apparaître des poids, qui représentent **l'importance relative des variables dans la discrimination des paires**.

$$\begin{aligned}\frac{P(M=1|\gamma)}{1-P(M=1|\gamma)} &= \frac{\lambda m_1(\gamma_1)m_2(\gamma_2)\dots m_K(\gamma_K)}{(1-\lambda)u_1(\gamma_1)u_2(\gamma_2)\dots u_K(\gamma_K)} \\ &= \frac{\lambda}{1-\lambda} w_1(\gamma_1)w_2(\gamma_2)\dots w_K(\gamma_K)\end{aligned}$$

avec $w_j(\gamma_j) = \frac{m_j(\gamma_2)}{u_j(\gamma_2)}$ le poids associé au champ j.

Variable	$m_{var}(1)$	$u_{var}(1)$	$w_{var}(1)$	$w_{var}(0)$
Prénom	0,8	0,02	$\frac{0,8}{0,02} = 40$	$\frac{1-0,8}{1-0,02} \approx 0,20$
Genre	0,99	0,5	$\frac{0,99}{0,5} = 1,98$	$\frac{1-0,99}{1-0,5} = 0,02$

TABLEAU **4 – Exemples de poids**

Pour une paire donnée, **le poids associé à chaque variable dépend de la valeur du vecteur de comparaison**. Par exemple, si les deux prénoms au sein d'une paire sont identiques, le poids associé sera $w_{prenom}(1)$ tandis que s'ils sont différents, le poids sera $w_{prenom}(0)$. Une valeur de poids supérieure à 1 fait augmenter la côte, et donc la probabilité qu'il s'agisse d'une paire d'individus identiques ; et inversement.

Le pouvoir discriminant de chaque variable dépend du cas de figure rencontré (valeurs identiques ou différentes au sein de la paire). Le tableau 4 illustre ce phénomène avec des valeurs plausibles des probabilités m et u pour les champs de prénom et de genre ainsi que les poids associés. Ainsi, dans cet exemple, un prénom identique au sein d'une paire donne un poids très important de 40, tandis que le fait d'observer un prénom différent est moins informatif. Inversement pour le genre, c'est la différence de genre qui donne l'information la plus significative avec un poids de $0,02$, qui fait nettement diminuer la cote.

A.3 Règle de décision

En sortie de l'appariement, **les paires sont classées dans trois ensembles disjoints** : les paires liées $\mathcal{M}$, les paires non liées $\mathcal{U}$ et une zone grise de paires laissées en suspens $\mathcal{P}$. Cette règle de décision est matérialisée par une fonction d qui à un vecteur de comparaison γ associe l'une de ces trois catégories.

Le modèle probabiliste a l'avantage de permettre d'estimer les taux de faux négatifs et faux positifs. Ainsi, le **taux de faux négatifs** peut être estimé sur l'ensemble des paires par :

$$\begin{aligned} E[\mathbb{1}[d(\gamma) \in \mathcal{U}])|M = 1)] &= \sum_{j=1}^{n} P(\gamma^j|M = 1)\mathbb{1}[d(\gamma) \in \mathcal{U}]) \\ &= \sum_{d(\gamma)\in\mathcal{U}} m(\gamma^j) \end{aligned}$$

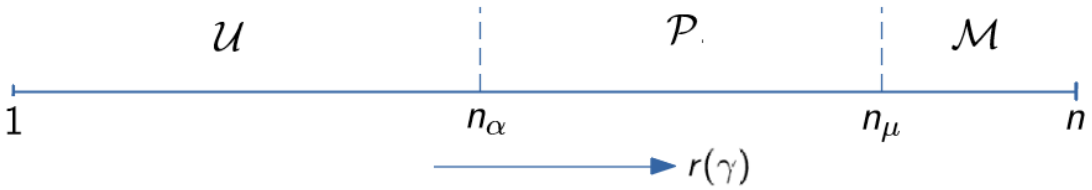
De même, le **taux de faux positifs** est estimé par :

$$E[\mathbb{1}[d(\gamma) \in \mathcal{M}])|M = 0)] = \sum_{d(\gamma)\in\mathcal{M}} u(\gamma^j)$$

Ces taux d'erreurs vont permettre de définir la règle de décision du modèle, en combinaison avec une nouvelle quantité, définie ici :

$$\boldsymbol{r(\gamma) = \frac{m(\gamma)}{u(\gamma)}}$$

Les paires sont réindexées dans l'ordre décroissant du rapport $r(\gamma)$ puis classées dans les trois catégories conformément à la figure 10.



GRAPHIQUE **10 – Règle de décision du modèle probabiliste**

Les seuils n_α et n_μ sont fixés de façon à **laisser le moins de paires possibles dans la zone grise tout en respectant les contraintes établies à l'avance sur les taux d'erreurs**.

Étant donné un **niveau toléré de faux négatifs** α, n_α est le plus petit indice vérifiant $\sum_{j>n_\alpha} m(\gamma^j) < \alpha$.

Pour un **niveau toléré de faux positifs** μ, n_μ est le plus grand indice vérifiant $\sum_{j<n_\mu} u(\gamma^j) < \mu$.

Toutes les paires d'indice supérieur ou égal à n_μ sont liées, toutes celles d'indice inférieur ou égal à n_α ne sont pas liées et les paires situées entre n_α et n_μ sont laissées en suspens pour un examen manuel.

Cette règle de décision est **optimale**, dans le sens où elle minimise la taille de la zone grise à niveaux d'erreur α et μ donnés. Cette optimalité est intimement liée à la théorie des tests d'hypothèse.

En considérant le statut d'une paire M comme un paramètre, le classement d'une paire comme un *match* est équivalent à la réalisation d'un **test du rapport de vraisemblance de l'hypothèse** $H_0 : M = 1$ **contre** $H_1 : M = 0$ **de niveau** α.

La statistique de test s'écrit : $\dfrac{P(\gamma|M=1)}{P(\gamma|M=0)} = \dfrac{\boldsymbol{m(\gamma)}}{\boldsymbol{u(\gamma)}}$

D'après le lemme de Neymann-Pearson, **ce test est le plus puissant de niveau** α.

Le raisonnement est identique pour le **test de l'hypothèse** $H_0 : M = 0$ **contre** $H_1 : M = 1$ **de niveau** μ.

Il est intéressant de noter que **cette règle de décision est cohérente avec la probabilité estimée** pour chaque paire. En effet, le classement des paires s'effectue selon l'ordre du rapport $\dfrac{m(\gamma)}{u(\gamma)} = w_1(\gamma_1)w_2(\gamma_2)\dots w_K(\gamma_K)$

Il s'agit, à un facteur multiplicatif près, de la cote associée la probabilité $P(M = 1|\gamma)$. Le passage d'une cote à une probabilité étant une transformation croissante, **la règle de décision proposée équivaut à classer les paires selon l'ordre des probabilités**.

A.4 Estimation des paramètres

Plusieurs méthodes existent pour estimer les paramètres du modèle, mais c'est l'estimation par **l'algorithme Espérance-Maximisation (EM)**, proposée par Jaro (1989), qui s'est imposée comme **référence dans la littérature des appariements probabilistes**.

L'algorithme EM (Dempster, Laird, et Rubin (1977)) est une adaptation de l'estimation par maximum de vraisemblance en présence de **variables latentes non observables**. Dans le cas de l'appariement probabiliste, **les variables observées sont les composantes du vecteur de comparaison** $\boldsymbol{\gamma}$ et **la variable latente est le vrai statut de la paire** $\boldsymbol{M}$.

L'estimation s'effectue de façon **non supervisée** : elle ne nécessite pas d'information supplémentaire sur les jeux de données.

En notant $\theta = (m, u, \lambda)$ le vecteur des paramètres à estimer, la log-vraisemblance du modèle s'écrit :

$$log\mathcal{L}(\theta, M_1, \ldots, M_n, \gamma^1, \ldots, \gamma^n) = \sum_{j=1}^{n} M_j log(\lambda \cdot m(\gamma^j)) + (1 - M_j) log((1 - \lambda) \cdot u(\gamma^j))$$

Le processus est **itératif**, alternant les phases de **calcul d'espérance de la vraisemblance** et d'ajustement des paramètres par **maximisation de cette quantité**.

L'**hypothèse d'indépendance conditionnelle** simplifie les calculs et permet d'obtenir des **formules en forme close**.

Les autres méthodes d'estimation des paramètres reposent pour la plupart sur des **calculs de fréquences**. Cependant, **elles nécessitent en général des informations** comme la fréquence des noms et prénoms dans une langue, ce qui les rend moins faciles à mettre en œuvre.

A.5 Au-delà de la théorie classique

A.5.1 Variables de comparaison à plusieurs modalités

Dans le modèle tel qu'exposé par Fellegi et Sunter, les composantes du vecteur de comparaison sont binaires (valant 1 si les champs sont similaires, 0 sinon). On peut choisir d'attribuer la modalité 1 uniquement pour des valeurs identiques, ou bien lorsqu'une mesure de similarité dépasse un seuil, par exemple lorsque la similarité de Jaro-Winkler dépasse $0,92$. Toutefois, des variables binaires ne captent probablement pas toute l'information pertinente.

Pour pallier cette perte d'information, il est possible d'adapter le modèle pour que les composantes du vecteur γ soient des variables à plusieurs modalités. Par exemple, une première amélioration peut être d'ajouter une modalité afin de distinguer les paires qui présentent des valeurs strictement identiques, celles qui présentent des valeurs similaires mais pas identiques et celles qui ont des valeurs très différentes. L'estimation du modèle fonctionne de la même façon que dans le cadre classique, cependant **l'inconvénient de cette approche est l'augmentation du nombre de paramètres à estimer**. Le temps de calcul et les ressources informatiques nécessaires à l'estimation du modèle sont plus importantes, ce qui peut poser un problème lorsque les fichiers à apparier sont très volumineux.

A.5.2 Prise en compte de la fréquence dans les comparaisons

Le modèle de Fellegi-Sunter prend en compte les différences de pouvoir prédictif de chaque variable via les probabilités m et u. En particulier, le paramètre u mesure la probabilité d'observer une valeur identique par hasard sur un champ identifiant donné. Toutefois, le modèle suppose que ce paramètre ne varie pas au sein de la population. Pourtant, comparer deux individus qui s'appellent « Jean » et deux individus prénommés « Dimitri » ne donne pas la même information. **Observer des valeurs identiques au sein d'une paire sur une modalité rare devrait faire augmenter plus fortement la probabilité de lier cette paire**. L'article Enamorado, Fifield, et Imai (2019) propose une méthode pour tenir compte de la fréquence des modalités des variables identifiantes, sous la forme d'un ajustement *ex-post* de la probabilité associée à chaque paire.

A.5.3 Gestion du volume et effet de l'indexation sur les paramètres

La phase d'indexation réduit le champ des paires étudiées et **modifie la distribution** du vecteur de comparaison γ et du statut des paires M. Quelle que soit sa forme (blocage ou filtrage plus complexe), **elle modifie la valeur des paramètres estimés**. Sans adaptation de l'algorithme d'estimation des paramètres, **les taux d'erreur ne sont plus fiables**.

L'indexation reste néanmoins nécessaire lorsque les fichiers deviennent grands :

— d'abord pour des raisons opérationnelles de temps de calcul,
— mais aussi car lorsque la proportion de paires d'individus identiques dans le produit cartésien devient trop faible, l'estimation des paramètres est biaisée.

L'indexation induit particulièrement des variations sur les valeurs des $u_i(\gamma_i)$ et sur l'*a priori* λ parce qu'elle retire essentiellement des paires dont le vrai statut est négatif. En général, si l'indexation retire un nombre significatif de paires, λ augmente fortement. Les $u_i(\gamma_i)$ augmentent légèrement car les paires retenues sont globalement plus similaires qu'une paire aléatoire du produit cartésien. Jaro (1989) propose d'**estimer les paramètres $u_i(\gamma_i)$ sur le produit cartésien et les paramètres $m_i(\gamma_i)$ sur les données indexées**.

Murray (2016) étudie de façon approfondie **l'effet de l'indexation sur l'estimation des paramètres**. Les paramètres estimés sont effectivement différents, en revanche **le classement des paires** (l'ordre des probabilités) **peut être conservé sous certaines conditions**. C'est le cas par exemple lorsque l'indexation ne fait intervenir que des éléments du vecteur de comparaison γ. L'article propose une adaptation de la phase d'estimation des paramètres, principalement en repérant

les modalités du vecteur γ qui ne peuvent pas apparaître dans les paires conservées en raison de choix d'indexation et leur appliquant un traitement particulier.

Fortini (2020) traite des **défis liés au volume des fichiers à apparier** et propose deux adaptations, indépendantes l'une de l'autre, au modèle classique. D'abord, lors de la phase de comparaison des paires, il s'agit de conserver toutes celles pour lesquelles au plus un champ identifiant diffère, puis de procéder à une estimation par échantillonnage pour les autres modalités du vecteur γ. Ensuite, il introduit un algorithme d'estimation EM robuste, qui reste non-biaisé même lorsque la proportion de paires d'individus identiques dans le produit cartésien devient très faible. Il consiste à donner plus d'importance aux paires qui correspondent sur la plupart des champs dans l'étape de maximisation.

Série des Documents de Travail « Méthodologie Statistique »

9601 : Une méthode synthétique, robuste et efficace pour réaliser des estimations locales de population.
G. DECAUDIN, J.-C. LABAT

9602 : Estimation de la précision d'un solde dans les enquêtes de conjoncture auprès des entreprises.
N. CARON, P. RAVALET, O. SAUTORY

9603 : La procédure FREQ de SAS - Tests d'indépendance et mesures d'association dans un tableau de contingence.
J. CONFAIS, Y. GRELET, M. LE GUEN

9604 : Les principales techniques de correction de la non-réponse et les modèles associés.
N. CARON

9605 : L'estimation du taux d'évolution des dépenses d'équipement dans l'enquête de conjoncture : analyse et voies d'amélioration.
P. RAVALET

9606 : L'économétrie et l'étude des comportements. Présentation et mise en œuvre de modèles de régression qualitatifs. Les modèles univariés à résidus logistiques ou normaux (LOGIT, PROBIT).
S. LOLLIVIER, M. MARPSAT, D. VERGER

9607 : Enquêtes régionales sur les déplacements des ménages : l'expérience de Rhône-Alpes.
N. CARON, D. LE BLANC

9701 : Une bonne petite enquête vaut-elle mieux qu'un mauvais recensement ?
J.-C. DEVILLE

9702 : Modèles univariés et modèles de durée sur données individuelles.
S. LOLLIVIER

9703 : Comparaison de deux estimateurs par le ratio stratifiés et application aux enquêtes auprès des entreprises.
N. CARON, J.-C. DEVILLE

9704 : La faisabilité d'une enquête auprès des ménages.
1. au mois d'août.
2. à un rythme hebdomadaire

C. LAGARENNE, C. THIESSET

9705 : Méthodologie de l'enquête sur les déplacements dans l'agglomération toulousaine.
P. GIRARD.

9801 : Les logiciels de désaisonnalisation TRAMO & SEATS : philosophie, principes et mise en œuvre sous SAS.
K. ATTAL-TOUBERT, D. LADIRAY

9802 : Estimation de variance pour des statistiques complexes : technique des résidus et de linéarisation.
J.-C. DEVILLE

9803 : Pour essayer d'en finir avec l'individu Kish.
J.-C. DEVILLE

9804 : Une nouvelle (encore une !) méthode de tirage à probabilités inégales.
J.-C. DEVILLE

9805 : Variance et estimation de variance en cas d'erreurs de mesure non corrélées ou de l'intrusion d'un individu Kish.
J.-C. DEVILLE

9806 : Estimation de précision de données issues d'enquêtes : document méthodologique sur le logiciel POULPE.
N. CARON, J.-C. DEVILLE, O. SAUTORY

9807 : Estimation de données régionales à l'aide de techniques d'analyse multidimentionnelle.
K. ATTAL-TOUBERT, O. SAUTORY

9808 : Matrices de mobilité et calcul de la précision associée.
N. CARON, C. CHAMBAZ

9809 : Échantillonnage et stratification : une étude empirique des gains de précision.
J. LE GUENNEC

9810 : Le Kish : les problèmes de réalisation du tirage et de son extrapolation.
C. BERTHIER, N. CARON, B. NEROS

9901 : Perte de précision liée au tirage d'un ou plusieurs individus Kish.
N. CARON

9902 : Estimation de variance en présence de données imputées : un exemple à partir de l'enquête Panel Européen.
N. CARON

0001 : L'économétrie et l'étude des comportements. Présentation et mise en œuvre de modèles de régression qualitatifs. Les modèles univariés à résidus logistiques ou normaux (LOGIT, PROBIT) (version actualisée).
S. LOLLIVIER, M. MARPSAT, D. VERGER

0002 : Modèles structurels et variables explicatives endogènes.
J.-M. ROBIN

0003 : L'enquête 1997-1998 sur le devenir des personnes sorties du RMI - Une présentation de son déroulement.
D. ENEAU, D. GUILLEMOT

0004 : Plus d'amis, plus proches ? Essai de comparaison de deux enquêtes peu comparables.
O. GODECHOT

0005 : Estimation dans les enquêtes répétées : application à l'Enquête Emploi en Continu.
N. CARON, P. RAVALET

0006 : Non-parametric approach to the cost-of-living index.
F. MAGNIEN, J. POUGNARD

0101 : Diverses macros SAS : Analyse exploratoire des données, Analyse des séries temporelles.
D. LADIRAY

0102 : Économétrie linéaire des panels : une introduction.
T. MAGNAC

0201 : Application des méthodes de calages à l'enquête EAE-Commerce.
N. CARON

C 0201 : Comportement face au risque et à l'avenir et accumulation patrimoniale - Bilan d'une expérimentation.
L. ARRONDEL, A. MASSON, D. VERGER

C 0202 : Enquête Méthodologique Information et Vie Quotidienne - Tome 1 : bilan du test 1, novembre 2002.
J.-A. VALLET, G. BONNET, J.-C. EMIN, J. LEVASSEUR, T. ROCHER, P. VRIGNAUD, X. D'HAULTFOEUILLE, F. MURAT, D. VERGER, P. ZAMORA

0203 : General principles for data editing in business surveys and how to optimise it.
P. RIVIERE

0301 : Les modèles logit polytomiques non ordonnés : théories et applications.
C. AFSA ESSAFI

0401 : Enquête sur le patrimoine des ménages - Synthèse des entretiens monographiques.
V. COHEN, C. DEMMER

0402 : La macro SAS CUBE d'échantillonnage équilibré
S. ROUSSEAU, F. TARDIEU

0501 : Correction de la non-réponse et calage de l'enquêtes Santé 2002
N. CARON, S. ROUSSEAU

0502 : Correction de la non-réponse par répondération et par imputation
N. CARON

0503 : Introduction à la pratique des indices statistiques - notes de cours
J-P BERTHIER

0601 : La difficile mesure des pratiques dans le domaine du sport et de la culture - bilan d'une opération méthodologique
C. LANDRE, D. VERGER

0801 : Rapport du groupe de réflexion sur la qualité des enquêtes auprès des ménages
D. VERGER

M2013/01 : La régression quantile en pratique
P. GIVORD,
X. D'HAULTFOEUILLE

M2014/01 : La microsimulation dynamique : principes généraux et exemples en langage R
D. BLANCHET

M2015/01 : la collecte multimode et le paradigme de l'erreur d'enquête totale
T. RAZAFINDROVONA

M2015/02 : Les méthodes de Pseudo-Panel
M. GUILLERM

M2015/03 : Les méthodes d'estimation de la précision pour les enquêtes ménages de l'Insee tirées dans Octopusse
E. GROS
K. MOUSSALAM

M2016/01 : Le modèle Logit Théorie et application.
C. AFSA

M2016/02 : Les méthodes d'estimation de la précision de l'Enquête Emploi en Continu
E. GROS
K.MOUSSALAM

M2016/03 : Exploitation de l'enquête expérimentale Vols, violence et sécurité.
T. RAZAFINDROVONA

M2016/04 : Savoir compter, savoir coder. Bonnes pratiques du statisticien en programmation.
E. L'HOUR
R. LE SAOUT
B. ROUPPERT

M2016/05 : Les modèles multiniveaux
P. GIVORD
M. GUILLERM

M2016/06 : Econométrie spatiale : une introduction pratique
P. GIVORD
R. LE SAOUT

M2016/07 : La gestion de la confidentialité pour les données individuelles
M. BERGEAT

M2016/08 : Exploitation de l'enquête expérimentale Logement internet-papier
T. RAZAFINDROVONA

M2017/01: Exploitation de l'enquête expérimentale Qualité de vie au travail
T. RAZAFINDROVONA

M2018/01: Estimation avec le score de propension sous
S. QUANTIN

M2018/02: Modèles semi-paramétriques de survie en temps continu sous
S. QUANTIN

M2019/01 : Les méthodes de décomposition appliquées à l'analyse des inégalités
B. BOUTCHENIK
E. COUDIN
S. MAILLARD

M2020/01 : L'économétrie en grande dimension
J. L'HOUR

M2021/01 : R Tools for JDemetra+ - Seasonal adjustment made easier
A. SMYK
A. TCHANG

M2021/02 : Le traitement du biais de sélection endogène dans les enquêtes auprès des ménages par modèle de Heckman
L. CASTELL
P. SILLARD

M2021/03 : Conception de questionnaires auto-administrés
H. KOUMARIANOS
A. SCHREIBER

M2022/01 : Introduction à la géomatique pour le statisticien : quelques concepts et outils innovants de gestion, traitement et diffusion de l'information spatiale
F. SEMECURBE
E. COUDIN

M2022/02 : Le zonage en unites urbaines 2020
V. COSTEMALLE
S. OUJIA
C. GUILLO
A. CHAUVET

M2023/01 : Les réseaux de neurones appliqués à la statistique publique : méthodes et cas d'usages
D. BABET
Q. DELTOUR
T. FARIA
S. HIMPENS

M2023/02 : Redressements de la première vague de l'enquête epicov : un exemple de correction des effets de sélection dans les enquêtes multimodes
L.CASTELL
C. FAVRE-MARTINOZ
N. PALIOD
P. SILLARD

M2023/03 : Appariements de données individuelles : concepts, méthodes, conseils
L.MALHERBE