OTrecod: An R Package for Data Fusion using Optimal Transportation Theory

Grégory Guernec, Valérie Garès, Jérémy Omer, Nicolas Savy, Philippe Saint-Pierre

OTrecod: An R Package for Data Fusion using Optimal Transportation Theory

by Gregory Guernec[*], Valerie Gares[†], Jeremy Omer[‡], Philippe Saint-Pierre[§] and Nicolas Savy[¶]

October 24, 2022

Abstract

The advances of information technologies often confront users with a large amount of data which is essential to integrate easily. In this context, creating a single database from multiple separate data sources can appear as an attractive but complex issue when same information of interest is stored in at least two distinct encodings. In this situation, merging the data sources consists in finding a common recoding scale to fill the incomplete information in a synthetic database. The OTrecod package provides R-users two functions dedicated to solve this recoding problem using optimal transportation theory. Specific arguments of these functions enrich the algorithms by relaxing distributional constraints or adding a regularization term to make the data fusion more flexible. The OTrecod package also provides a set of support functions dedicated to the harmonization of separate data sources, the handling of incomplete information and the selection of matching variables. This paper gives all the keys to quickly understand and master the original algorithms implemented in the OTrecod package, assisting step by step the user in its data fusion project.

1 Introduction

The large amount of data produced by information technology requires flexible tools to facilitate its handling. Among them, the field of data fusion (Hall and Llinas, 1997; Klein, 2004; Castanedo, 2013) also known as statistical matching (Adamek, 1994; D'Orazio et al., 2006; Vantaggi, 2008) aims to integrate the overall information from multiple data sources for a better understanding of the phenomena that interact in the population. Assuming that two heterogeneous databases A and B share a set of common variables X while an information of interest is encoded in two distinct scales respectively: Y in A and Z in B. If Y and Z are never jointly observed, a basic data fusion objective consists in the recoding of Y in the same scale of Z (or conversely), to allow the fusion between the databases as illustrated in Table 1.

Providing a solution to this recoding problem is often very attractive because it aims at giving access to more accurate and consistent information with no additional costs in a unique and bigger database. Despite this, if we exclude all R data integration packages applied in the context of genetic area like the **MultitDataset** package (Hernandez-Ferrer et al., 2017), the **OMICsPCA** package (Das and Tripathy, 2020), or the **MixOmics** package (Rohart et al., 2017) which are often only effective for quantitative data integration. To our knowledge, the StatMatch package (D'Orazio and D'Orazio, 2019) is actually the only one that provide a concrete solution to the problem using hot deck

[*]Center for Epidemiology and Research in POPulation health (CERPOP), Université de Toulouse, INSERM, UPS, France. gregory.guernec@inserm.fr

[†]CNRS, IRMAR, UMR 6625, INSA, Université de Rennes, France.

[‡]CNRS, IRMAR, UMR 6625, INSA, Université de Rennes, France.

[§]CNRS UMR 5219, IMT, Université Paul Sabatier, Toulouse, France.

[¶]CNRS UMR 5219, IMT, Université Paul Sabatier, Toulouse, France.

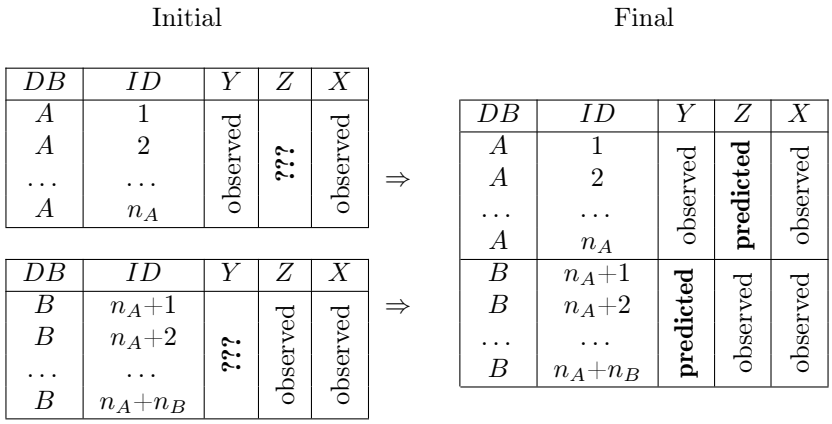

Table 1: Formulation of a recoding problem for 2 distinct databases A and B

imputation procedures. The main reason for this relative deficiency is that this problem is, in fact, often assimilated and solved like a missing data imputation problem. According to this idea, a very large amount of works and reference books now exist about the handling of missing data (Zhu et al., 2019; Little and Rubin, 2019). Moreover, we can enumerate several R packages that we can sort by types of imputation methods (Mayer et al., 2019): mice (Van Buuren and Groothuis-Oudshoorn, 2011) and missforest (Stekhoven and Bühlmann, 2012) which use conditional models, softImpute (Hastie et al., 2015) and missMDA (Josse and Husson, 2016) which apply low-rank based models. For all these packages, imputation performances can sometimes fluctuate a lot according to the structure and the proportion of non-response encountered. Regressions and non parametric imputation approaches (like hot-deck methods from donor based family) seem to use partially, or not at all, the available information of the two databases to provide the individual predictions. Contrary to our approach, all these methods only use the set of shared variables X for the prediction of Z in A (or conversely Y in B) without really taking into account Y and its interrelations with X in their process of predictions.

The purpose of this paper is to present a new package to the R community (R Core Team, 2016) called OTrecod which provides a simple and intuitive access to two original algorithms (Gares et al., 2019; Gares and Omer, 2020) dedicated to solve these recoding problems in the data fusion context by considering them as applications of Optimal Transportation theory (OT). In fact, this theory was already applied in many areas: for example, the transport package (Schuhmacher et al., 2020) solves optimal transport problems in the field of image processing. A specific package called `POT: Python Optimal Transport` also exists in the `Python` software (`https:/pythonot.github.io`) to solve optimization problems using optimal transportation theory in the fields of signal theory, image processing and domain adaptation. Nevertheless, all these available tools were still not really adapted to our recoding problem and the performances established by OT-based approaches to predict missing information compared to more standard processes (Gares et al., 2019; Gares and Omer, 2020; Muzellec et al., 2020) have finished to confirm our decision.

In OTrecod the first provided algorithm, called OUTCOME and integrated in the `OT_outcome` function consists in finding a map that pushes the distribution of Y forward to the distribution of Z (Gares et al., 2019) while the second one, called JOINT and integrated in the `OT_joint` function, pushes the distribution of (Y, X) forward to the distribution of (Z, X). Consequently, by building, these two algorithms take advantage of all the potential relationships between Y, Z, and X for the prediction of the incomplete information of Y and/or Z in A and B. Enrichments related to these algorithms and described in (Gares et al., 2019; Cuturi, 2013) are also available via the optional arguments of these two functions. In its current version, these algorithms are accompanied by original preparation (`merge_dbs`, `select_pred`) and validation (`verif_OT`) functions which are key steps of any standard statistical matching project and can be used independently of the OUTCOME

and JOINT algorithms.

2 Solving recoding problems using optimal transportation theory

2.1 The optimal transportation problem

The optimal transportation (OT) problem was originally stated by Monge (1781) and consists in finding the cheapest way to transport a pile of sand to fill a hole. Formally the problem writes as follows: Consider two (Radon) spaces $\mathbb{X}$ and $\mathbb{Y}$, μ^X a probability measure on $\mathbb{X}$, and μ^Y a probability measure on $\mathbb{Y}$ and c a Borel-measurable function from $\mathbb{X}\times\mathbb{Y}$ to $[0,\infty]$. The Kantorovich's formulation of the optimal transportation problem (Kantorovich, 1942) consists in finding a measure $\gamma \in \Gamma(\mu^X, \mu^Y)$ that realizes the infimum:

$$\inf\left\{ \int_{\mathbb{X}\times\mathbb{Y}} c(x,y)\,\mathrm{d}\gamma(x,y) \,\middle|\, \gamma \in \Gamma(\mu^X,\mu^Y) \right\}, \tag{1}$$

where $\Gamma(\mu^X,\mu^Y)$ is the set of measures on $\mathbb{X}\times\mathbb{Y}$ with marginals μ^X on $\mathbb{X}$ and μ^Y on $\mathbb{Y}$.

This theory is applied here to solve a recoding problem of missing distributions in a data fusion area. To do so, we make use of Kantovorich's formulation adapted to the discrete case, known as Hitchcock's problem (Hitchcock, 1941). Therefore, by construction, the proposed algorithms are usable for specific target variables only: categorical variables, ordinal or nominal, and discrete variables with finite number of values.

2.2 Optimal transportation of outcomes applied to data recoding

Let A and B be two databases corresponding to two independent sets of subjects. We assume without loss of generality that the two databases have equal sizes, so that they can be written as $A = \{i_1,\dots,i_n\}$ and $B = \{j_1,\dots,j_n\}$.

Let $\big((X_i,Y_i,Z_i)\big)_{i\in A}$ and $\big((X_j,Y_j,Z_j)\big)_{j\in B}$ be two sequences of i.i.d. discrete random variables with values in $\mathcal{X}\times\mathcal{Y}\times\mathcal{Z}$, where $\mathcal{X}$ is a finite subset of $\mathbb{R}^P$, and $\mathcal{Y}$ and $\mathcal{Z}$ are finite subsets of $\mathbb{R}$. Variables $(X_i,Y_i,Z_i), i\in A$, are i.i.d copies of (X^A,Y^A,Z^A) and $(X_j,Y_j,Z_j), j\in B$, are i.i.d copies of (X^B,Y^B,Z^B). Moreover assume that $\big\{(X_i,Y_i,Z_i), i\in A\big\}$ are independent of $\big\{(X_j,Y_j,Z_j), j\in B\big\}$. The first version using the optimal transportation algorithm approach, described in (Gares et al., 2019), assumes that:

Assumption 1. *Y^A and Z^A respectively follow the same distribution as Y^B and Z^B.*

Assumption 2. *For all $x\in\mathcal{X}$ the probability distributions of Y^A and Z^A given that $X^A = x$ are respectively equal to those of Y^B and Z^B given that $X^B = x$.*

In this setting, the aim is to solve the recoding problem given by equation (1) that pushes μ^{Y^A} forward to μ^{Z^A}. The variable γ of (1) is a discrete measure with marginals μ^{Y^A} and μ^{Z^A}, represented by a $|\mathcal{Y}|\times|\mathcal{Z}|$ matrix. The cost function denoted as c is a $|\mathcal{Y}|\times|\mathcal{Z}|$ matrix, $(c_{y,z})_{y\in\mathcal{Y},z\in\mathcal{Z}}$. The goal is in the identification of:

$$\gamma^* \in \operatorname{argmin}_{\gamma\in\mathbb{R}_+^{|\mathcal{Y}|\times|\mathcal{Z}|}} \left\{ \langle\gamma|c\rangle : \gamma\mathbf{1}_{|\mathcal{Z}|} = \mu^{Y^A}, \gamma^T\mathbf{1}_{|\mathcal{Y}|} = \mu^{Z^A} \right\}, \tag{2}$$

where $\langle\cdot|\cdot\rangle$ is the dot product, $\mathbf{1}$ is a vector of ones with appropriate dimension and M^T is the transpose of matrix M. The cost function considered by Gares et al. (2019), $c_{y,z}$, measures the average distance between the profiles of shared variables of A satisfying $Y = y$ and subjects of B satisfying $Z = z$, that is:

$$c_{y,z} = \mathbb{E}\left[d(X^A,X^B) \mid Y^A = y, Z^B = z \right], \tag{3}$$

where d is a given distance function to choose on $\mathcal{X}\times\mathcal{X}$.

In fact, the above situation cannot be solved in reality, since the distributions of X^A, X^B, Y^A and Z^A are never jointly observed. As a consequence, the following unbiased empirical estimators are used: $\hat{\mu}_n^{X^A}$ of μ^{X^A} and $\hat{\mu}_n^{X^B}$ of μ^{X^B}. Because Y and Z are only available in A and B respectively, two distinct empirical estimators have to be defined:

$$\begin{aligned} \hat{\mu}_{n,y}^{Y^A} = & \quad \frac{1}{n} \sum_{i \in A} \mathbf{1}_{\{Y_i = y\}}, \ \forall y \in \mathcal{Y} \\ \hat{\mu}_{n,z}^{Z^A} = & \quad \frac{1}{n} \sum_{j \in B} \mathbf{1}_{\{Z_j = z\}}, \ \forall z \in \mathcal{Z}, \end{aligned} \tag{4}$$

where $\mathbf{1}_{\{Y=y\}} = 1$ if $Y = y$ and 0 otherwise. The assumption 1 gives: $\mu^{Z^A} = \mu^{Z^B}$ from which we can conclude that $\hat{\mu}_{n,z}^{Z^B} = \hat{\mu}_{n,z}^{Z^A}$. Finally, denoting:

$$\kappa_{n,y,z} \equiv \sum_{i \in A} \sum_{j \in B} \mathbf{1}_{\left\{Y_i = y, Z_j = z\right\}}$$

the number of pairs $(i,j) \in A \times B$ such that $Y_i = y$ and $Z_j = z$, the cost matrix c is estimated by:

$$\hat{c}_{n,y,z} = \begin{cases} \frac{1}{\kappa_{n,y,z}} \sum_{i \in A} \sum_{j \in B} \mathbf{1}_{\left\{Y_i = y, Z_j = z\right\}} \times d(X_i, X_j), & \forall y \in \mathcal{Y}, z \in \mathcal{Z} : \kappa_{n,y,z} \neq 0, \\ 0, & \forall y \in \mathcal{Y}, z \in \mathcal{Z} : \kappa_{n,y,z} = 0. \end{cases} \tag{5}$$

Plugging the values observed for these estimators in (2) yields to a linear programming model denoted:

$$\hat{\mathcal{P}}_n^0 : \begin{cases} \min \ < \hat{c}_n, \gamma > \\ \text{s.t.} \ \sum_{z \in Z} \gamma_{y,z} = \mu_{n,y}^{Y^A}, \ \forall y \in \mathcal{Y} \\ \quad \sum_{y \in Y} \gamma_{y,z} = \mu_{n,z}^{Z^A}, \ \forall z \in \mathcal{Z} \\ \quad \gamma_{y,z} \geq 0, \ \forall y \in \mathcal{Y}, \forall z \in \mathcal{Z} \end{cases} \tag{6}$$

The solution $\hat{\gamma}_n$ can then be interpreted as an estimator $\hat{\mu}_n^{(Y^A, Z^A)}$ of the joint distribution of Y^A and Z^A, $\mu^{(Y^A, Z^A)}$. If this estimate is necessary, it is nevertheless insufficient here to provide the individual predictions on Z in A. These predictions are done in a second step using a nearest neighbor algorithm from which we deduce an estimation of $\mu^{Z^A | X^A = x, Y^A = y}$ (see Gares and Omer (2020) for details). In the remainder, the overall algorithm described in this section is referred to as OUTCOME. To improve the few drawbacks of this algorithm described in Gares et al. (2019), derived algorithms from OUTCOME have been developed (Gares and Omer, 2020) and described in the following part.

2.3 Optimal transportation of outcomes and covariates

Using the same notations, Gares et al. (2019) propose to search for an optimal transportation map between the two joint distributions of (X^A, Y^A) and (X^A, Z^A) with marginals $\mu^{(X^A, Y^A)}$ and $\mu^{(X^A, Z^A)}$ respectively. Under Kantorovich's formulation in a discrete setting, they search for:

$$\gamma^* \in \text{argmin}_{\gamma \in \mathcal{D}} < c, \gamma >,$$

where c is a given cost matrix and $\mathcal{D}$ is the set of joint distributions with marginals $\mu^{(X^A, Y^A)}$ and $\mu^{(X^A, Z^A)}$. It is natural to see any element $\gamma \in \mathcal{D}$ as the vector of joint probabilities $\mathbb{P}((X^A = x, Y^A = y), (X^A = x', Z^A = z))$ for any $x, x' \in \mathcal{X}^2$, $y \in \mathcal{Y}$ and $z \in \mathcal{Z}$. Since this probability nullifies for all $x \neq x'$, $\gamma \in \mathcal{D}$ is defined as a vector of $\mathbb{R}^{|\mathcal{X}| \times |\mathcal{Y}| \times |\mathcal{Z}|}$, where $\gamma_{x,y,z}$ stands for an estimation of the joint probability $\mathbb{P}(X^A = x, Y^A = y, Z^A = z)$. These notations lead to the more detailed model:

$$\mathcal{P}: \begin{cases} \min \ < c, \gamma > \\ \text{s.t.} \ \sum_{z \in Z} \gamma_{x,y,z} = \mu_{x,y}^{(X^A, Y^A)}, \ \forall x \in \mathcal{X}, \forall y \in \mathcal{Y} \\ \quad \sum_{y \in Y} \gamma_{x,y,z} = \mu_{x,z}^{(X^A, Z^A)}, \ \forall x \in \mathcal{X}, \forall z \in \mathcal{Z} \\ \quad \gamma_{x,y,z} \geq 0, \ \forall x \in \mathcal{X}, \forall y \in \mathcal{Y}, \forall z \in \mathcal{Z} \end{cases} \tag{7}$$

The above algorithm can be solved only if the marginals $\mu^{(X^A, Y^A)}$ and $\mu^{(X^A, Z^A)}$ are known, but, based on assumption 2, unbiased estimators $\hat{\mu}_n^{X^A, Y^A}$ and $\hat{\mu}_n^{X^A, Z^A}$ can be built according to the previous subsection. For the first one it gives:

$$\hat{\mu}_n^{X^A, Y^A} = \quad \frac{1}{n} \sum_{i \in A} \mathbf{1}_{\{Y_i = y, X_i = x\}}, \ \forall x \in \mathcal{X}, \ \forall y \in \mathcal{Y}, \tag{8}$$

The cost matrix introduced in the OUTCOME algorithm is used (3) and estimated by (5). Formally we can write:

$$c_{x,y,z} = c_{y,z}, \ \forall x \in \mathcal{X}, \forall y \in \mathcal{Y}, \forall z \in \mathcal{Z}. \tag{9}$$

which does not depend on the value of x.

Plugging the values observed for these estimators in (7) yield a linear programming model denoted as $\hat{\mathcal{P}}_n$. In contrast to OUTCOME, the algorithm that consists in solving $\hat{\mathcal{P}}_n$ to solve the recoding problem is referred to as JOINT in what follows.

An estimation of the distribution of Z^A given the values of X^A and Y^A is then given by:

$$\tilde{\mu}_{n,z}^{Z^A | X^A = x, Y^A = y} = \begin{cases} \frac{\hat{\gamma}_{n,x,y,z}}{\hat{\mu}_{n,x,y}^{(X^A, Y^A)}}, & \forall x \in \mathcal{X}, y \in \mathcal{Y}, z \in \mathcal{Z} : \hat{\mu}_{n,x,y}^{(X^A, Y^A)} \neq 0, \\ 0, & \forall x \in \mathcal{X}, y \in \mathcal{Y}, z \in \mathcal{Z} : \hat{\mu}_{n,x,y}^{(X^A, Y^A)} = 0. \end{cases} \tag{10}$$

and an individual prediction of Z^A is then deduced using the maximum a posterior rule:

$$\widehat{z}_i^A = \text{argmax}_{z \in \mathcal{Z}} \, \tilde{\mu}_{n,z}^{Z^A | X^A = x_i, Y^A = y_i}$$

Due to potential errors in the estimations of $\mathcal{P}$, the constraints of $\hat{\mathcal{P}}_n$ may derive from the true values of the marginals of $\mu^{(X^A, Y^A, Z^A)}$. To deal with this situation, small violations of the constraints of $\hat{\mathcal{P}}_n$ are allowed by enriching the initial algorithm as described in Gares and Omer (2020).

The equality constraints of $\hat{\mathcal{P}}_n$ are then relaxed as follows:

$$\sum_{z \in Z} \gamma_{x,y,z} = \hat{\mu}_{n,x,y}^{(X^A, Y^A)} + e_{x,y}^{X,Y}, \ \forall x \in \mathcal{X}, \forall y \in \mathcal{Y} \tag{11}$$

$$\sum_{y \in Y} \gamma_{x,y,z} = \tilde{\mu}_{n,x,z}^{(X^A, Z^A)} + e_{x,z}^{X,Z}, \ \forall x \in \mathcal{X}, \forall z \in \mathcal{Z} \tag{12}$$

$$\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} e_{x,y}^{X,Y} = 0, \sum_{x \in \mathcal{X}, z \in \mathcal{Z}} e_{x,z}^{X,Z} = 0 \tag{13}$$

$$-e_{x,y}^{X,Y,+} \leq e_{x,y}^{X,Y} \leq e_{x,y}^{X,Y,+}, \ \forall x \in \mathcal{X}, \forall y \in \mathcal{Y} \tag{14}$$

$$-e_{x,z}^{X,Z,+} \leq e_{x,z}^{X,Z} \leq e_{x,z}^{X,Z,+}, \ \forall x \in \mathcal{X}, \forall z \in \mathcal{Z} \tag{15}$$

$$\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} e_{x,y}^{X,Y,+} \leq \alpha_n, \sum_{x \in \mathcal{X}, z \in \mathcal{Z}} e_{x,z}^{X,Z,+} \leq \alpha_n \tag{16}$$

This relaxation is possible by introducing extra-variables $e^{X,Y,+}$ and $e^{X,Z,+}$ as additional constraints (14)–(15). Gares and Omer (2020) suggests to consider $\alpha_n := \frac{\alpha}{\sqrt{n}}$ from (16), with a parameter α to calibrate numerically but proposes also a default value fixed to 0.4.

A regularization term λ given by $(\frac{\pi_{x,y,z}}{\hat{\mu}^{X^A}_{n,x}})_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}}$ can also be added to improve regularity in the variations of the conditional distribution $\mu^{Y^A,Z^A|X^A=x}$ with respect to x. The corresponding regularized algorithm is:

$$\hat{\mathcal{P}}^R_n : \begin{cases} \min \ < \hat{c}_n, \gamma > + \lambda \displaystyle\sum_{(x_i,x_j)\in E_{\mathcal{X}}} w_{i,j} \sum_{y\in\mathcal{Y},z\in\mathcal{Z}} r^+_{i,j,y,z} \\ \quad \text{s.t. constraints (11)–(16)} \\ \quad \dfrac{\gamma_{x_i,y,z}}{\hat{\mu}^{X^A}_{n,x_i}} - \dfrac{\gamma_{x_j,y,z}}{\hat{\mu}^{X^A}_{n,x_j}} \leq r^+_{i,j,y,z}, \ \forall \{x_i,x_j\} \in E_{\mathcal{X}}, y \in \mathcal{Y}, z \in \mathcal{Z} \\ \quad \dfrac{\gamma_{x_i,y,z}}{\hat{\mu}^{X^A}_{n,x_i}} - \dfrac{\gamma_{x_j,y,z}}{\hat{\mu}^{X^A}_{n,x_j}} \geq -r^+_{i,j,y,z}, \ \forall \{x_i,x_j\} \in E_{\mathcal{X}}, y \in \mathcal{Y}, z \in \mathcal{Z} \\ \quad \gamma_{x,y,z} \geq 0, \ \forall x \in \mathcal{X}, \forall y \in \mathcal{Y}, \forall z \in \mathcal{Z} \end{cases} \tag{17}$$

The constant $\lambda \in \mathbb{R}^+$ is a regularization parameter to be calibrated numerically (0.1 can be considered as default value) and $E_{\mathcal{X}} \subset \mathcal{X}^2$ includes the pairs of elements of X defined as neighbors: $\{x_i, x_j\} \in E_{\mathcal{X}}$ if x_j is among the k nearest neighbors of x_i for some parameter $k \geq 1$. The method that computes a solution to the recoding problem with regularization and relaxation is called R-JOINT.

In a same way, a relaxation of the assumption 1 is also proposed and added to the OUTCOME algorithm: This resulting method is denoted R-OUTCOME and is related to the following program:

$$\hat{\mathcal{P}}^{0-R}_n : \begin{cases} \min \ < \hat{c}_n, \gamma > \\ \quad \displaystyle\sum_{z\in\mathcal{Z}} \gamma_{y,z} = \hat{\mu}^{Y^A}_{n,y} + e^Y_y, \ \forall y \in \mathcal{Y} \\ \quad \displaystyle\sum_{y\in\mathcal{Y}} \gamma_{y,z} = \tilde{\mu}^{Z^A}_{n,z} + e^Z_z, \ \forall z \in \mathcal{Z} \\ \quad \displaystyle\sum_{y\in\mathcal{Y}} e^Y_y = 0, \ \sum_{z\in\mathcal{Z}} e^Z_z = 0 \\ \quad - e^{Y,+}_y \leq e^Y_y \leq e^{Y,+}_y, \ \forall y \in \mathcal{Y} \\ \quad - e^{Z,+}_z \leq e^Z_z \leq e^{Z,+}_z, \ \forall z \in \mathcal{Z} \\ \quad \displaystyle\sum_{y\in\mathcal{Y}} e^{Y,+}_y \leq \alpha_n, \ \sum_{z\in\mathcal{Z}} e^{Z,+}_z \leq \alpha_n \\ \quad \gamma_{y,z} \geq 0, \ \forall y \in \mathcal{Y}, \forall z \in \mathcal{Z} \end{cases} \tag{18}$$

Note that algorithms JOINT and R-JOINT do not require assumption 1. The relaxation in R-OUTCOME alleviates its dependence to the satisfaction of this assumption. However, algorithms JOINT and R-JOINT require that X is a set of discrete variables (factors ordered or not are obviously allowed) while the absence of X in the linear algorithms OUTCOME and R-OUTCOME allow X to be a set of discrete and/or continuous variables. In this case, the nature of the variables X need to be considered when choosing the distance d.

3 Package installation and description

3.1 Installation

The OTrecod package can be installed from the Comprehensive R Archive Network (CRAN) by running the `install.packages("OTrecod")` command. The development version of **OTrecod** (`https://github.com/otrecoding/OTrecod`) is also available and can be directly installed from **GitHub** by loading the devtools package and following the command:

```
R> devtools::install_github("otrecoding/OTrecod")
```

3.2 Main functions

The two types of optimal transportation (OT) algorithms previously introduced (OUTCOME and JOINT) and their respective enrichments (R-OUTCOME and R-JOINT) are available in the **OTrecod** package via two core functions denoted `OT_outcome` and `OT_joint`. Details about their implementations in `R` are described in the following section. In this package, these algorithms of recoding are seen as fundamental steps of data fusion projects that also require often adapted preparation and validation functions. In this way, the table 2 introduces the main functions proposed by **OTrecod** to handle a data fusion project.

`R` Function	Description
Pre-process functions	
`merge_dbs`	Harmonization of the data sources
`select_pred`	Selection of matching variables
Functions of data fusion	
`OT_outcome`	Data fusion with OT theory using the OUTCOME or R-OUTCOME algorithms.
`OT_joint`	Data fusion with OT theory using the JOINT or R-JOINT algorithms.
Post-process function	
`verif_OT`	Quality assessment of the data fusion

Table 2: A brief description of the main functions of **OTrecod**

All the intermediate functions integrated in the `OT_outcome` and `OT_joint` functions (`proxim_dist`, `avg_dist_closest`, `indiv_grp_closest`, `indiv_grp_optimal`), and their related documentations, are all included and usable separately in the package. They have been kept available for users to ensure a great flexibility as other interesting functions like `power_set` that returns the power set of a set. This function did not exist on `R` until now and could be of interest for specialists of algebra. These functions are not described here but detailed in the related pdf manual of the package.

4 Functionalities overview

4.1 Expected structure of the input databases as arguments

The functions of recoding `OT_outcome` and `OT_joint` require a specific structure of data.frame as input arguments. Described in Table 3, it must be the result of two overlayed databases made of at least four variables:

- A first variable, discrete or categorical, corresponding to the database identifier, stored in factor or not, but with only two classes or levels (for example: A and B, 1 and 2 or otherwise).
- The target variable of the first database (or top database) denoted Y for example, whose values related to the second database are missing. This variable can be discrete or categorical stored in factor, ordered factor or not.
- In the same way, the target variable of the second database (or below database) denoted Z for example, whose values related to the first database are missing.
- At least one shared variable (defined as a variable with the same label and the same encoding in the two distinct data sources). The type of shared variables can be continuous, categorical stored in factor or not, complete or not. Nevertheless, few constraints must be noticed related to this question. First, in a critical situation where only one shared variable exists, this latter cannot be incomplete. Second, continuous shared variables are actually not allowed in the current version of the function `OT_joint`, therefore, these variables must be transformed beforehand.

DB	Y	Z	X_1	X_2	X_3
1	(600-800]	<NA>	M	Yes	50
1	(600-800]	<NA>	M	No	32
1	[200-600]	<NA>	W	No	31
2	<NA>	G1	M	No	47
2	<NA>	G3	W	Yes	43
2	<NA>	G2	W	No	23
2	<NA>	G4	M	Yes	22
2	<NA>	G2	W	Yes	47

Table 3: Example of expected structure for two databases 1 and 2 with three shared variables X_1, X_2, X_3

As additional examples, users can also refer to the databases `simu_data` and `tab_test` provided in the package with expected structures. Note that class objects are not expected here as input arguments of these functions to allow users to freely work with or without the use of the pre-process functions provided in the package.

4.2 Choice of solver

The package **OTrecod** uses the `ROI` optimization infrastructure (Theußl et al., 2017) to solve the optimization problems related to the OUTCOME and JOINT algorithms. The solver GLPK (The GNU Linear Programming Kit Makhorin (2011)) is the default solver actually integrated in the `OT_outcome` and `OT_joint` functions for handling linear problems with linear constraints. The ROI infrastructure makes easy for users to switch solvers for comparisons. In many situations, some of them can noticeably reduce the running time of the functions.

For example, the solver Clp (Forrest et al., 2004) for COINT-OR Linear Programming, known to be particularly convenient in linear and quadratic situations, can be easily installed by users via the related plug-in available in `ROI` (searchable with the instruction `ROI_available_solvers()`) and following the few instructions detailed in Theußl et al. (2020) or via the dedicated website.

4.3 An illustrative example

In California (United States), from 1999 to 2018, the Public Schools Accountability Act (PSAA) imposed on its California Department of Education (CDE) to provide annually the results of an Academic Performance Index (API) which established a ranking of the best public schools of the state.

This numeric score, indicator of school's performance levels, could vary from 200 to 1000 and the performance objective to reach for each school was 800. Information related to the 418 schools (identified by `cds`) of Nevada (County 29) and to the 362 schools of San Benito (County 35), was respectively collected in two databases, `api29` and `api35`, available in the package. The distributions of all the variables in the two databases are provided by following the `R` commands:

```
R> library(OTrecod)
R> data(api29); data(api35)
R> summary(api29) #--------------------------------------------------
 cds                apicl_2000       stype   awards     acs.core
 Length:418         [200-600] : 93   E:300   No  : 98   Min.   :16.00
 Class :character   (600-800] :180   M: 68   Yes :303   1st Qu.:26.00
 Mode  :character   (800-1000]:145   H: 50   NA's: 17   Median :30.00
                                                       Mean   :31.97
                                                       3rd Qu.:39.00
                                                       Max.   :50.00
```

```
 api.stu          acs.k3.20     grad.sch   ell            mobility
 Min.   : 108.0   <=20   :190   0   : 86   [0-10]  :153   [0-20]  :362
 1st Qu.: 336.2   >20    :108   1-10:180   (10-30] : 83   (20-100]: 56
 Median : 447.5   unknown:120   >10 :152   (30-50] : 60
 Mean   : 577.8                            (50-100]: 93
 3rd Qu.: 641.5                            NA's    : 29
 Max.   :2352.0
 meals          full
 [0-25]  :200   1: 85
 (25-50] : 56   2:333
 (50-75] :100
 (75-100]: 62
R> summary(api35) #-------------------------------------------------
 cds                apicl_1999 stype   awards     acs.core
 Length:362         G1:91      E:257   No  :111   Min.   :16.00
 Class :character   G2:90      M: 67   Yes :237   1st Qu.:25.00
 Mode  :character   G3:90      H: 38   NA's: 14   Median :30.00
                    G4:91                         Mean   :31.81
                                                  3rd Qu.:39.00
                                                  Max.   :50.00
 api.stu          acs.k3.20     grad.sch   ell            mobility
 Min.   : 102.0   <=20   :227   0   : 50   [0-10]  :164   1:213
 1st Qu.: 363.2   >20    : 30   1-10:241   (10-30] : 99   2:149
 Median : 460.0   unknown:105   >10 : 71   (30-50] : 64
 Mean   : 577.2                            (50-100]: 10
 3rd Qu.: 624.0                            NA's    : 25
 Max.   :2460.0
 meals          full
 [0-25]  : 77   1:244
 (25-50] : 92   2:118
 (50-75] :122
 (75-100]: 71
```

The two databases seem to share a majority of variables (same labels, same encodings) of different types, therefore inferential statistics could be ideally considered by combining all the information of the two databases to study the effects of social factors on the results of the API score in 2000.

Nevertheless, while this target variable called `apicl_2000` is correctly stored in the database `api29` and encoded as a three levels ordered factors clearly defined: `[200-600]`, `(600-800]` and `(800-1000]`[1], the only information related to the API score in the database `api35` is the variable `apicl_1999` for the API score collected in 1999, encoded in four unknown balanced classes (`G1`, `G2`, `G3` and `G4`). As no school is common to the two counties, we easily deduce that these two variables have never been jointly observed.

By choosing these two variables as outcomes (called also target variables), the objective of the following examples consists in creating a synthetic database where the missing information related to the API score in 2000 is fully completed in `api35` by illustrating the use of the main functions of the package.

4.4 Harmonization of two datasources using merge_dbs

The function `merge_dbs` is an optional pre-process function dedicated to data fusion projects that merges two raw databases by detecting possible discrepancies among the respective sets of variables from one database to another. The current discrepancy situations detected by the function follow specific rules described below:

[1] a parenthesis removed

- any variable (other than a target one) whose label (or name) is not common to the two data sources is automatically excluded. By default, the remaining variables are denoted shared variables.
- among the subset of shared variables, any variable stored in a different format (or type) from one data source to another will be automatically discarded from the subset previously generated and its label saved in an output object called `REMOVE1`.
- among the remaining subset of shared variables, a factor variable (ordered or not) stored with different levels or number of levels from one data source to another will be automatically discarded from the subset and its label saved in an output object called `REMOVE2`.

These situations sometimes require reconciliation actions which are not supported by the actual version of the `merge_dbs` function. Therefore, when reconciliation actions are required by the user, they will have to be treated a posteriori and outside the function.

Applied to our example and according to the introduced rules, the first step of our data fusion project consists in studying the coherence between the variables of the databases `api29` and `api35` via the following R code:

```
R> step1 = merge_dbs(DB1 = api29, DB2 = api35,
                     NAME_Y = "apicl_2000", NAME_Z = "apicl_1999",
                     row_ID1 = 1, row_ID2 = 1,
                     ordinal_DB1 = c(2:3, 8:12),
                     ordinal_DB2 = c(2:3, 8:12))

DBS MERGING in progress. Please wait ...
DBS MERGING OK
-----------------------
SUMMARY OF DBS MERGING:
Nb of removed subjects due to NA on targets: 0(0%)
Nb of removed covariates due to differences between the 2 bases: 1
Nb of remained covariates: 9
Imputation on incomplete covariates: NO
```

As entry, the raw databases must be declared separately in the `DB1` and `DB2` arguments and the name of the related target variables of each database must be specified via the `NAME_Y` and `NAME_Z` for `DB1` and `DB2` respectively. In presence of row identifiers, the respective column indexes of each database must be set in the argument `row_ID1` and `row_ID2`. The arguments `ordinal_DB1` and `ordinal_DB2` list the related column indexes of all the variables defined as ordinal in the two databases (including also the indexes of the target variables if necessary). Here, `apicl_2000` is clearly an ordinal variable, and, by default, we suppose that the unknown encoding related to `apicl_1999` is also ordinal: the corresponding indexes (2 and 3) are so added in these two arguments.

After running, the function informs users that no row was dropped from the databases during the merging because each target variable is fully completed in the two databases. Nine potential predictors are kept while only one variable is discarded because of discrepancies between the databases: its identity is consequently stored in output and informs user about the nature of the problem: the `mobility` factor has different levels from one database to the other.

```
R> summary(step1)
R> step1$REMOVE1   # List of removed variables because of type's problem
NULL
R> step1$REMOVE2   # List of removed factors because of levels' problem
"mobility"
R> levels(api29$mobility)  # Verification
"[0-20]"   "(20-100]"
R> levels(api29$mobility); levels(api35$mobility)
"[0-20]"   "(20-100]"      # levels in api29
"1" "2"                    # levels in api35
R> step1$REMAINING_VAR
```

```
"acs.core"  "acs.k3.20"  "api.stu"  "awards"  "ell"  "full"
"grad.sch"  "meals"  "stype"
```

The function returns a list object and notably `DB_READY`, a data.frame whose structure corresponds to the expected structure introduced in the previous subsection: a unique database, here result of the superimposition of `api29` on `api35` where the first column (`DB`) corresponds to the database identifier (1 for `api29` and 2 for `api35`), the second and third columns (`Y`, `Z` respectively) corresponds to the target variables of the two databases. Missing values are automatically assigned by the function to the unknown part of each target variable: in `Y` when the identifier equals to 2, in `Z` when the identifier equals to 1. The subset of shared variables whose information is homogeneous between the databases are now stored from the fourth column to the last one. Their identities are available in the output object called `REMAINING_VAR`.

The `merge_dbs` function can handle incomplete shared variables by using the `impute` argument. This option allows to keep the missing information unchanged (the choice by default, and also the case in this example), to work with complete cases only, to do fast multivariate imputations by chained equations approach (the function integrates the main functionalities of the `mice` function of the mice package), or to impute data using the dimensionality reduction method introduced in the missMDA package (see the pdf manual for details).

4.5 Selection of matching variables using select_pred

In data fusion projects, a selection of shared variables (also called matching variables) appears as an essential step for two main reasons. First, the proportion of shared variables X between the two databases can be important (much higher than three variables) and keeping a pointless part of variability between variables could strongly reduce the quality of the fusion. Second, this selection greatly influences the quality of the predictions regardless of the matching technique which is chosen a posteriori (Adamek, 1994). The specific context of data fusion is subject to the following constraints:

1. Y, Z and X are never jointly observed in database A or B, so the relationships between Y and X, and between Z and X must be investigated separately.
2. matching variables need to be good predictors of both target variables (outcomes) Y and Z (Cohen, 1991), in the sense that they explain relevant parts of variability of the targets.

These particularities suppose that the investigations have to be done separately but in the same manner in both databases. In this way, a predictor which appears at the same time as highly correlated to Y and Z will be automatically selected for the fusion. On the contrary, predictors whose effects on Y and Z seem not obvious will be discarded. Additional recommended rules also emerge from literature:

- concerning the subset of variables that would predict only one of the two targets Y or Z, D'Orazio et al. (2006) suggests a moderate parsimonious selection remaining too many predictors could complicate the fusion procedure.
- Cibella (2010) and Scanu (2010) suggests to select quality predictors with no error and just a small amount of missing data.

The function of the package dedicated to this task is `select_pred`. From the `DB_READY` database generated in the previous subsection, studying the outputs related to each database and produced by the following `R` commands assist users in selecting the best predictors:

```
# For the dataset api29 --------------
R> step2a = select_pred(step1$DB_READY,
                        Y = "Y", Z = "Z", ID = 1, OUT = "Y",
                        quanti = c(4,6), nominal = c(1,5,7),
                        ordinal = c(2,3,8:12), thresh_cat = 0.50,
                        thresh_num = 0.70, RF_SEED = 3011)

# For the dataset api35 --------------
```

```
R> step2b = select_pred(step1$DB_READY,
                        Y = "Y", Z = "Z", ID = 1, OUT = "Z",
                        quanti = c(4,6), nominal = c(1,5,7),
                        ordinal = c(2,3,8:12), thresh_cat = 0.50,
                        thresh_num = 0.70, RF_SEED = 3011)
```

The `quanti`, `nominal`, and `ordinal` arguments requires vectors of column indexes related to the type of each variable of the input database. The `ID` argument specifies the column index of the database identifier. `Y` and `Z` expected the respective names of the target variables while `OUT` provides the target variable to predict (`Y` or `Z`).

To detect the subset of best predictors, `select_pred` studies the standard pairwise associations between each shared variable and the outcomes Y and Z, taken separately (by only varying the argument `OUT`), according to its type: for numeric and/or ordered factor variables, `select_pred` calculates the associations using rank correlation coefficients (Spearman) while the Cramer's V criterion (Bergsma, 2013) is used for categorical variables and not ordered factors. The related ranking tables of top scoring predictors are available in two distinct output objects: `cor_OUTC_num` and `cor_OUTC_cat`. In our example, the corresponding results, for each database, are:

```
### ASSOCATIONS BETWEEN TARGET VARIABLES AND SHARED VARIABLES

## Results for the api29 dataset -----

R> step2a$cor_OUTC_num   # Y versus numeric or ordinal predictors

name1    name2 RANK_COR pv_COR_test   N
    Y    meals  -0.8030      0.0000 418
    Y      ell  -0.7514      0.0000 389
    Y     full   0.3919      0.0000 418
    Y grad.sch   0.3346      0.0000 418
    Y  api.stu  -0.1520      0.0018 418
    Y    stype  -0.1520      0.0018 418
    Y acs.core  -0.0556      0.2566 418

R> step2a$vcrm_OUTC_cat  # Y versus nominal or ordinal predictors

name1     name2 V_Cramer CorrV_Cramer   N
    Y     meals   0.6871       0.6835 418
    Y       ell   0.6015       0.5966 389
    Y      full   0.4508       0.4459 418
    Y  grad.sch   0.3869       0.3816 418
    Y    awards   0.2188       0.2073 401
    Y acs.k3.20   0.1514       0.1349 418
    Y     stype   0.1370       0.1185 418


## Results for the api35 dataset -----

R> step2b$cor_OUTC_num   # Z versus numeric or ordinal predictors

name1    name2 RANK_COR pv_COR_test   N
    Z      ell  -0.7511      0.0000 337
    Z    meals  -0.7291      0.0000 362
    Z grad.sch   0.6229      0.0000 362
    Z     full   0.3997      0.0000 362
    Z  api.stu  -0.0563      0.2851 362
    Z    stype   0.0359      0.4959 362
```

```
    Z acs.core   0.0119      0.8219 362

R> step2b$vcrm_OUTC_cat  # Z versus nominal or ordinal predictors

name1     name2 V_Cramer CorrV_Cramer   N
    Z     meals   0.5181       0.5121 362
    Z  grad.sch   0.5131       0.5063 362
    Z       ell   0.4775       0.4701 337
    Z      full   0.4033       0.3934 362
    Z    awards   0.2040       0.1818 348
    Z     stype   0.1320       0.0957 362
    Z acs.k3.20   0.1223       0.0817 362
```

The two first tables related to `api29` highlights the strong associations between Y and the variables `meals`, `ell`, `full` and `grad.sch` in this order of importance, while the summary tables related to `api35` highlights the strong associations between Z and the variables `meals`, `grad.sch`, `ell` and `full`.

It is often not unusual to observe that one or more predictors are in fact linear combinations of others. In supervised learning areas, these risks of collinearity increase with the number of predictors, and must be detected beforehand to keep only the most parsimonious subset of predictors for fusion. To avoid collinearity situations, the result of a Farrar and Glauber (FG) test is provided (Farrar and Glauber, 1967). This test is based on the determinant of the rank correlation matrix of the shared variables D (Spearman) and the corresponding test statistic is given by:

$$S_{FG} = -\left(n - 1 - \frac{1}{6}(2k+5) \times \ln(\det(D))\right)$$

where n is the number of rows and k is the number of covariates. Th null hypothesis supposes that S_{FG} follows a chi square distribution with $k(k-1)/2$ dof, and its acceptation indicates an absence of collinearity. In presence of a large number of numeric covariates and/or ordered factors, the approximate Farrar-Glauber test, based on the normal approximation of the null distribution (Kotz et al., 2004) can be more adapted and the statistic test becomes:

$$\sqrt{2S_{FG}} - \sqrt{2k-1}$$

Users will note that these two tests can often appear highly sensitive in the sense that they tend to easily conclude in favor of multicollinearity. Thus, it is suggested to consider these results as indicators of collinearity between predictors rather than an essential condition of acceptability. The results related to this test are stored in the `FG_test` object of the `select_pred` output.

In presence of collinearity, `select_pred` tests the pairwise associations between all the potential predictors according to their types (Spearman or Cramér's V). The `thres_num` argument fixed the threshold beyond which two ranked predictors are considered as redundant while `thres_cat` fixed the threshold of acceptability for the Cramér's V criterion in the subgroup of factors. In output, the results stored in the `collinear_PB` object permit to identify the corresponding variables. In our example, we observe:

```
### DETECTION OF REDUNDANT PREDICTORS

## Results for the api29 dataset -----

## Results of the Farrar-Glauber test
R> step2a$FG_test

     DET_X      pv_FG_test pv_FG_test_appr
0.04913067      0.00000000      0.00000000
```

```
## Identity of the redundant predictors

R> step2a$collinear_PB
$VCRAM
    name1 name2 V_Cramer CorrV_Cramer   N
acs.k3.20 stype   0.6988       0.6971 418
      ell meals   0.6696       0.6664 389
     full meals   0.5215       0.5152 418

$SPEARM
  name1 name2 RANK_COR pv_COR_test   N
    ell meals   0.9047           0 389
api.stu stype   0.7069           0 418


#### Results for the api35 dataset -----

R> step2b$FG_test    # Significant result
R> step2b$collinear_PB

$VCRAM
    name1 name2 V_Cramer CorrV_Cramer   N
acs.k3.20 stype   0.6977       0.6957 362

$SPEARM
    name1 name2  RANK_COR  pv_COR_test   N
<0 rows> (or 0-length row.names)
```

The FG test warns the user against the risks of collinearity between predictors, and the function notably detects strong collinearities between the variables `meals`, `ell`, `full` in the `api29` (less strong trends in `api35`): an information that have to be taken into account during the selection. The part of predictors finally kept for the data fusion must be small to improve its quality. When the initial number of shared variables is not too important as in this example, choosing the best candidates between groups of redundant predictors can be made manually by selecting highest ranked predictors in the summary tables previously described. In this way, the variable `meals` could be preferred to `ell`, and `full`, while the variable `stype` could be dropped. Consequently, a possible final list of predictors could be only composed of the variables `meals` and `grad.sch`.

When the number of predictors is important, or when users prefer that an automatic process performs the variable selection, a random forest procedure can also be used via the `select_pred` function. Random forest approaches (Breiman, 2001) are here particularly convenient (Grajski et al., 1986) for multiple reasons: it works fine when the number of variables exceeds the number of rows, whatever the types of covariates, it allows to deal with non linearity, to consider correlated predictors, ordered or not ordered outcomes and to rank good candidate predictors through an inbuilt criterion: the variable importance measure.

In few words, random forest processes aggregates many CART models (Breiman et al., 1984) with `RF_ntree` bootstrap samples from the raw data source and averaging accuracies from each model permits to reduce the related variances and also the errors of prediction. A standard random forest process provides two distinct measures of importance of each variable for the prediction of the outcome, the Gini importance criterion and the permutation importance criterion, which depends on the appearance frequency of the predictor but also on its place taken up in each tree. For more details about random forest theory, user can consult Breiman (2001) and/or the pdf manual of the randomForest package.

Strobl et al. (2009) suggests that the permutation importance criterion, which works with permuted samples (subsampling without replacements) instead of bootstrap ones, is particularly conve-

nient with uncorrelated predictors, but must be replaced by a conditional permutation measurement in presence of strong correlations. `select_pred` provides these assessments by integrating the main functionalities of the `cforest` and `varimp` functions of the package party (Hothorn et al., 2006b; Zeileis et al., 2008; Hothorn et al., 2006a; Strobl et al., 2007, 2008). However, these measurements must be used with caution, by accounting the following constraints:

- the Gini importance criterion can produce bias in favor of continuous variables and variables with many categories. This criterion is thus not available in the function.
- the permutation importance criterion can overestimate the importance of highly correlated predictors and therefore redundant predictors will be discarded beforehand using the first steps of the process integrated in the function.

The `select_pred` function allows to proceed with different scenarios according to the type of predictors (Table 4 can help users to choose). The first one consists in boiling down to a set of categorical variables (ordered or not) by categorizing all the continuous predictors using the dedicated argument (`convert_num` and `convert_clss`) and to work with the conditional importance assessments that directly provide unbiased estimations (by setting the `RF_condi` argument to TRUE). Users can consult (Hothorn et al., 2006b) for more details about the approach and consult the pdf manual of the package for details about the related arguments of the `select_pred` function. This approach does not take into account incomplete information, so that the method will be applied to complete data only (incomplete rows will be temporarily removed from the study). It is nevertheless possible to impute missing data beforehand by using dedicated pre-existing `R` packages like mice (Van Buuren et al., 1999) or by using the `imput_cov` function provided in this package.

The second possible scenario (always usable in presence of mixed type predictors), consists in the execution of a standard random forest procedure after taking care to rule out collinearity issues by first selecting unique candidates between potential redundant predictors (in this case, the discarded predictors are stored in the `drop_var` output object). This is the default approach used by `select_pred` as soon as the `RF_condi` argument is set to FALSE while `RF` is set to TRUE. This scenario can work in presence of incomplete predictors. By constructing, note that results from random forest procedures stay dependent on successive random draws carried out for the constitution of trees, and it is so suggested to check this stability by testing different random seeds (`RF_SEED` argument) before concluding.

The `R` command previously written provides automatically the results related to the second approach as additional results. The results from the two datasets show here the permutation importance estimates of each variable ranked in order of importance and expressed as percentage, after resolving collinearity problems:

```
R> step2a$RF_PRED   # For the api29 dataset

  meals    stype grad.sch   awards acs.core
71.5529  11.6282  11.1181   5.4674   0.2334

R> step2b$RF_PRED   # For the api35 dataset
```

Possible scenarios	Correlation between predictors	State of the RF_condi argument	Incomplete information
Same type predictors	NO	FALSE	Allowed
Same type predictors	YES	TRUE	Not allowed
Different type predictors	NO	FALSE	Allowed
Different type predictors	YES	TRUE	Not Allowed

Table 4: Completing the `RF_condi` argument according to predictors

```
  meals      ell grad.sch     full    stype  api.stu acs.core   awards
35.9974  28.3051  22.2094   5.2143   4.0192   1.8965   1.5643   0.7937
```

The results confirm that the variable `meals` appeared as the best predictor of the target variables in `api29` and `api35` respectively. The variable `ell` is not present in the first list (see `RF_PRED` from `step2a`) because the variables `meals` and `ell` has been detected as strongly collinear (according to the initial chosen threshold) and so `select_pred` keep the best choice between the two predictors: `meals` (the reasoning is the same for `full` which disappeared from the list).

The `ell` variable has been kept in the second list (not found as collinear enough to be removed here) and appears moreover as a good predictor of `apicl_1999` in `api35`. Nevertheless its potential collinearity problem with `meals` encourages us not to keep it for the rest of the study. According to this discussion, we finally keep `meals`, `stype` and `grad.sch` which combines the advantages of being good predictors for the two target variables while not presenting major problems of collinearity between them.

The following synthetic database (called here `bdd_ex`) is now ready for the prediction of the missing API scores in `api29`, `api35` or both, using the function `OT_outcome` or `OT_joint`:

```
R> bdd_ex = step1$DB_READY[, c(1:3,10:12)]; head(bdd_ex,3)
```

```
 DB         Y    Z grad.sch    meals stype
  1 (600-800] <NA>      >10   [0-25]     H
  1 (600-800] <NA>      >10   [0-25]     H
  1 [200-600] <NA>     1-10 (75-100]     H
```

4.6 Data fusion using OT_outcome

The `OT_outcome` function provides individual predictions of Z in A (and/or Y in B) by considering the recoding problem involving optimal transportation of outcomes. In a first step, the aim of the function is so to determine γ from (2) while this estimate is used in a second step to provide the predictions. The main input arguments of the function and the output values are respectively described in Tables 5 and 6.

The arguments `datab`, `index_DB_Y_Z`, `quanti`, `nominal`, `ordinal`, and `logic` are not optional and must be carefully completed before each execution. A unique synthetic data.frame made of two overlayed databases (called A and B for example) (see Table 3) is expected as `datab` argument. If this data.frame corresponds to the output objects `DB_USED` or `DB_READY` of the `select_pred` or `merge_dbs` functions respectively, then the expected object has the required structure. Otherwise users must be sure that their data.frames are made up of two overlayed databases with at least 4 variables as described in the subsection Optimal transportation of outcomes applied to data recoding. The order of the variables have no importance in the raw database but will have to be specified a posteriori in the `index_DB_Y_Z` argument if necessary.

The subset of remaining predictors used for fusion may requires prior transformations according to the distance function chosen in input by the user. This process is fixed by setting the `dist.choice` argument. The distances actually implemented in the function are: the standard Euclidean ("E") and Manhattan ("M") distances, the Hamming distance ("H", for binary covariates), and the Gower distance ("G" sometimes preferred with mixed variables). Automatic transformations prior to the use of each distance function are summarized in Table 7.

The first version of the OT algorithm described in Gares et al. (2019) was tested on numeric coordinates extracted from a factor analysis of mixed data (FAMD) fitted on mixed raw covariates (Pages, 2002). This transformation is here available by setting the `FAMD.coord` argument to "YES". The minimal percentage of information explained by the FAMD is also fixable using the `FAMD.perc` argument. The `OT_outcome` functions proposes four types of models for the prediction of Y in B (and/or) Z in A, according to the values of the `method` and `maxrelax` arguments:

- When `maxrelax = 0` and `indiv.method = "sequential"` (default options), the fitted model corresponds to the OUTCOME algorithm described by $\hat{\mathcal{P}}_n^0$ in equation (6). Assuming that Y and Z in A follow the same distribution as Y and Z in B respectively (assumption 1, page 2), this related algorithm derives the joint distribution of Y and Z in A (respectively in B) in

Argument	OT_outcome	OT_joint	Description (default value)
datab	✓	✓	Data.frame in the expected structure
index_DB_Y_Z	✓	✓	Indexes of the ID,Y, and Z columns (1,2,3)
nominal	✓	✓	Column indexes of nominal variables (*)
ordinal	✓	✓	Col. indexes of ordinal variables (*)
logic	✓	✓	Col. indexes of boolean variables (*)
quanti	✓		Col. indexes of quantitative variables (*)
convert.num	✓	✓	Col. indexes of the quantitative variables to convert (* or =quanti in OT_joint)
convert.clss	✓	✓	Corresponding numbers of desired classes for conversion (*)
which.DB	✓	✓	Specify the target variables to complete: Both or only one (BOTH)
solvR	✓	✓	Choice of the solver to solve the optimization problem (glpk)
dist.choice	✓	✓	Distance function (Euclidean). See Table 7
percent.knn	✓	✓	Ratio of closest neighbors involved int the computations (1)
indiv.method	✓		Type of individual predictions (sequential) for OUTCOME and R-OUTCOME algorithms
maxrelax	✓	✓	Adding of a relaxation parameter (0)
lambda.reg		✓	Adding of regularization parameter (0)

Table 5: Main arguments of the OT_outcome and OT_joint functions. (*: NULL as default value)

a first step, and uses in a second step, a nearest neighbor procedure to predict missing values of Z in A (resp. Y in B). This algorithm calculates averaged distances between each subject from A (resp. B) and subgroups of subjects from B (resp. A) having same levels of Z in B (resp. Y in A). These calculations can be done using all subjects of each subgroups (by default, `percent.knn = 1`) or only restricted parts of them (`percent.knn< 1`).

- When `maxrelax` > 0 and `indiv.method = "sequential"`, assumption 1 is alleviated by relaxing the constraints on marginal distributions and an R-OUTCOME algorithm (with relaxation) is fitted. In this case, the `maxrelax` argument corresponds to the α_n parameter in (18).
- When `maxrelax = 0` and `indiv.method = "optimal"`, the second step of the original algorithm (nearest neighbor procedure) is replaced by a linear optimization problem: searching for the individual predictions of Z in A (resp. Y in B) by minimizing the total of the distances between each individual of A and individuals of each levels of Z in B.
- When `maxrelax` > 0 and `indiv.method = "optimal"`, the constraints on marginal distributions of the previous model are also relaxed and the function fits an R-OUTCOME algorithm with relaxation. For these three last situations, the corresponding R-OUTCOME algorithm is so described by $\hat{\mathcal{P}}_n^{0-R}$ (18).

When `maxrelax` > 0, it is recommended to calibrate the `maxrelax` parameter by testing different values according to the stability of individual predictions. In output, the `gamma_A` and `gamma_B` matrices correspond to the estimates of the joint distributions γ of (Y^A, Z^A) and (Y^B, Z^B) re-

Value	OT_outcome	OT_joint	Description
time_exe	✓	✓	Running time of the algorithm
gamma_A	✓	✓	Estimation of the joint distribution of (Y, Z) for the prediction of Z in A (*)
gamma_B	✓	✓	Estimation of the joint distribution of (Y, Z) for the prediction of Y in B (*)
profile	✓	✓	The list of detected profiles of covariates
res.prox	✓	✓	A list that provides all the information related to the estimated proximities between profiles and groups of profiles
estimator_ZA	✓	✓	Estimates of the probability distribution of Z conditional to X and Y in database A (*)
estimator_YB	✓	✓	Estimates of the probability distribution of Y conditional to X and Z in database B (*)
DATA1_OT	✓	✓	The database A fully completed (if required in input by the which.DB argument)
DATA2_OT	✓	✓	The database B fully completed (if required in input by the which.DB argument)

Table 6: Values of the OT_outcome and OT_joint functions. (*: NULL if not required)

spectively (equation (2)). Moreover, a posteriori estimates of conditional distributions probabilities $(Z^A|Y^A, X^A)$ and $(Y^B|Z^B, X^B)$ (see equation 10) are also provided in two lists called estimatorZA, and estimatorYB respectively and the completed databases A and B are stored in the DATA1_OT and DATA2_OT objects. In particular, the individual predictions are stored in the OTpred column of these data.frames. Moreover, the profile object is a data.frame that stores the profile of covariates encountered in the two data sources while the res_prox object stored all the distance computations that will be used in the validation step (users can consult details of the proxim_dist function of the pdf manual).

The computation of conditional probabilities implies to define the part of individuals considered as neighbors of each encountered profile of covariates. The argument prox.dist fixes this threshold for each profile, following the decision rule for A (and respectively for B): A subject i of A (or a statistical unit corresponding to a row of A) will be considered as neighbor of a given profile of shared variables C as soon as:

$$\text{dist}(subject_i, C) < \text{prox.dist} \times \max_{k=1,\ldots,n_A} \text{dist}(subject_k, C)$$

When the number of shared variables is small and all of them are categorical (optimal situation), it is suggested to set the prox.dist argument to 0. Finally, using the which.DB argument, users can choose to estimate the individual predictions of Y and Z in the two databases (which.DB = "BOTH") or only one of the both (which.DB = "A" for the predictions of Z in A or which.DB = "B" for the predictions Y in B).

From the data.frame bdd_ex built previously, we use the OT_outcome function to illustrate the prediction of the target variable apicl_2000 in the api35 dataset via an OUTCOME algorithm and using an Euclidean distance function:

```
R> outc1 = OT_outcome(bdd_ex, quanti = 1, ordinal = 2:6,
                      dist.choice = "E", indiv.method = "sequential",
```

```
                      which.DB = "B")

---------------------------------------
OT PROCEDURE in progress ...
---------------------------------------
Type                      = OUTCOME
Distance                  = Euclidean
Percent closest knn       = 100%
Relaxation parameter      = NO
Relaxation value          = 0
Individual pred process   = Sequential
DB imputed                = B
---------------------------------------
```

In `bdd_ex`, all variables except the first one (DB: the database identifier) are or can be considered as ordinal factors, and the `quanti` and `ordinal` arguments are filled in accordingly. For the prediction of `apicl_2000` in the `api35` dataset (the steps would be the same for the prediction of `apicl_1999` in `api29` by setting `which.DB = "A"` or `"BOTH"`), the optimal transportation theory determines a map γ that pushes, in the distribution of `apicl_1999` forward to the distribution of `apicl_2000` in the database `api35`. In this case, γ^B is an estimator of the joint distribution of `apicl_2000` and `apicl_1999` in `api35`. The estimation of γ^B is available in the `gamma_B` output object while all the profiles of predictors met in the two databases are listed in the `profile` object:

```
R> outc1$gamma_B
#                    G1        G2         G3        G4
#[200-600]  0.22248804 0.0000000 0.00000000 0.0000000
#(600-800]  0.02889318 0.2486188 0.15311005 0.0000000
#(800-1000] 0.00000000 0.0000000 0.09550874 0.2513812

R> outc1$profile[1,]         # the first profile

 ID grad.sch meals stype
P_1        3     1     3
```

The output object `estimatorYB` is a list that corresponds to the estimations of the conditional probabilities of `apicl_2000` in the `api35` dataset for a given profile of predictors. For example, the conditional probabilities related to the first profile of predictors are:

```
R> outc1$estimatorYB[1,,]    # conditional probabilities (1st profile)
       [,1]      [,2]      [,3]
G1 0.3333333 0.3333333 0.3333333
G2 0.3333333 0.3333333 0.3333333
G3 0.0000000 0.0000000 1.0000000
G4 0.0000000 0.0000000 1.0000000
```

According to these results, we can conclude that: for a subject from `api35` with a related profile of predictors `P_1` and whose levels of `apicl_1999` is '`G1`', the probability for `apicl_2000` to be predicted '`[200,600]`' is about 0.63. Finally, the individual predictions of `apicl_2000` in `api35` are available in the `DATA2_OT` object corresponding to the `OTpred` column:

```
R> head(outc1$DATA2_OT,3)  # The 1st 3 rows only

DB    Y  Z grad.sch meals stype    OTpred
 2 <NA> G1        2     4     1 [200-600]
 2 <NA> G3        2     3     1 (600-800]
 2 <NA> G2        2     3     2 (600-800]
```

The OUTCOME algorithm uses here a nearest neighbor procedure to assign the individual predictions from the estimation of γ which can be a drawback as described in Gares et al. (2019). The

R-OUTCOME algorithm proposes an enrichment that directly assigns the individual predictions of `apicl_2000` from the estimation of γ, without using the nearest neighbor approach. In this case, a linear optimization problem is solved to determine the individual predictions that minimize the sum of the individual distances in `api35` having the modalities of the target `apicl_2000` in `api29`. The corresponding `R` command is:

```
R> R_outc3 = OT_outcome(bdd_ex, quanti = 1, ordinal = 2:6,
                        dist.choice = "E" , indiv.method = "optimal",
                        which.DB = "B")
```

Moreover, the OUTCOME algorithm assumes that the distribution of `apicl_2000` is identically distributed in `api29` and `api35` (assumption 1 from the subsection 2.2) which can appear as a strong hypothesis to hold in many situations. To overcome this constraint, the `R_OUTCOME` algorithm also allows to relax the assumption 1 by adding a relaxation parameter (18) in the optimization system. This relaxation can be done by varying the `maxrelax` arguments of the function:

```
### R-OUTCOME algorithm: optimal assignments + relaxation parameter = 0.4
R> R_outc4 = OT_outcome(bdd_ex, quanti = 1, ordinal = 2:6,
                        dist.choice = "E",
                        indiv.method = "optimal",
                        maxrelax = 0.4, which.DB = "B")
```

The running times of these two models can take few minutes. The quality of these models will be compared in Tables 8, 9 and 10 (subsection Validation of the data fusion using verif_OT).

4.7 Data fusion using OT_joint

The `OT_joint` function provides individual predictions of Z in A (and/or Y in B) by considering the recoding problem as an optimal transportation problem of covariates and outcomes, that pushes the conditional distribution of $(Y^A|X^A)$ forward to the conditional distribution of $(Z^A|X^A)$ (and conversely $(Z^B|X^B)$ forward to the conditional distribution of $(Y|X)$). The aim is to determine γ from (7). Because joint distributions of outcomes and covariates are now mapped together (it was not the case with the OUTCOME family of algorithms), it is not required for target variables to be equally distributed (assumption 1).

The call of `OT_joint` was thought to propose globally the same structure as those of the function `OT_outcome` as described in Tables 5 and 6 with few differences described below. JOINT and R-JOINT algorithms are usable via `OT_joint` for solving the recoding problem depending on the values related to the `maxrelax` and `lambda_reg` arguments:

- When `maxrelax = 0` and `lambda.reg = 0` (default values), the fitted model corresponds to the JOINT algorithm described by $\hat{\mathcal{P}}_n$ in equation (7).
- When at least one of these two arguments differs from 0, the R-JOINT algorithm is called. Using R-JOINT, it is so possible to relax constraints on marginal distributions (`maxrelax > 0`) and/or add an eventual more or less strong $L1$ regularisation term among the constraints of the algorithm by filling the `lambda_reg` argument. `maxrelax` parameter correspond to parameter α_n in (16) and `lambda.reg` parameter correspond to parameter λ in (17).

When `maxrelax > 0` and/or `lambda.reg > 0`, it is recommended to calibrate these two parameters by testing many different values and so studying the stability of the related individual predictions. Nevertheless, by default or as a starting point, Gares et al. (2019) suggests the use of default values determined from simulated databases: 0.1 for the regularization parameter (`lambda.reg`) and 0.4 for the relaxation one (`maxrelax`). Finally, the output objects are similar to those previously described in the `OT_outcome` function.

The implementation of the function is very similar to that of `OT_outcome`, and applied to our example, the `R` code related to a JOINT algorithm is:

```
R> outj1 = OT_joint(bdd_ex, nominal = 1, ordinal = c(2:6),
                    dist.choice = "E"  , which.DB = "B")
```

Distance function	Euclidean	Manhattan	Gower	Hamming
Variable transformations				
Continuous	Standardized	Standardized	No	Not allowed
Boolean	Binary	Binary	No	Binary
Nominal	Disjunctive T	Disjunctive T	No	Disjunctive T
Ordinal	Discrete	Discrete	No	Disjunctive T
Incomplete information	Allowed*	Allowed*	Allowed*	Allowed*
`dist.choice` **argument**	"E"	"M"	"G"	"H"

Table 7: Internal variable transformations related to the choice of each distance function in `OT_outcome` and `OT_joint` . (*) If the number of covariates exceeds 1. T for table

```
---------------------------------------
OT JOINT PROCEDURE in progress ...
---------------------------------------
Type                  = JOINT
Distance              = Euclidean
Percent closest       = 100%
Relaxation term       = 0
Regularization term   = 0
Aggregation tol cov   = 0.3
DB imputed            = B
---------------------------------------

### Extract individual predictions from the OTpred column
R> head(outj1$DATA2_OT,3)  # The 1st 3 rows only

DB    Y  Z grad.sch meals stype    OTpred
 2 <NA> G1        2     4     1 [200-600]
 2 <NA> G3        2     3     1 (600-800]
 2 <NA> G2        2     3     2 (600-800]
```

For relaxing the constraints stated on marginal distributions, we use the `maxrelax` argument that corresponds to varying the α parameter of a R-JOINT algorithm (see 17) and a parameter of regularization λ can be simply added to the previous model as follows:

```
### R-JOINT algorithm (relaxation parameter = 0.4)
R> R_outj1 = OT_joint(bdd_ex, nominal = 1, ordinal = c(2:6),
                      dist.choice = "E",  maxrelax = 0.4,
                      which.DB = "B")

### R-JOINT algorithm (relaxation parameter = 0.4,
###                                   & regularization parameter = 0.1)
R> R_outj4 = OT_joint(bdd_ex, nominal = 1, ordinal = c(2:6),
                      dist.choice = "E",  maxrelax = 0.4,
                      lambda.reg = 0.1,
                      which.DB = "B")
```

As previously, these models will be compared in the next subsection.

4.8 Validation of the data fusion using verif_OT

Assessing the quality of individual predictions obtained through statistical matching techniques can be a complex task notably because it is legitimate to wonder about the true reliability of a joint

distribution estimated from two variables which are never jointly observed (Kadane, 1978; Rodgers, 1984). When the study aims to identify estimators of interests that improve the understanding of the relationships between variables in the different databases (called macro approach in D'Orazio et al. (2006)) without going until the creation of a complete and unique dataset, uncertainty analyses are usually proposed to estimate the sensitivity of the assessments (Rässler, 2012). On the contrary, if the objective is to create a synthetic dataset where the information is available for every unit, the use of auxiliary information or proxy variables must be privileged to assess the quality of the results (Paass, 1986). In the absence of complementary information, Rubin (1986) suggests to study the preservation of the relationships at best between the distributions of the target variables. In this way, the `verif_OT` function proposes tools to assess quality of the individual predictions provided by the algorithms previously introduced.

The first expected input argument of `verif_OT` is an '`otres`' object from the `OT_outcome` or `OT_joint` functions. In our example, this function is firstly applied to the `out_c1` model by following the `R` command:

```
### Quality criteria for outc1 (OUTCOME model)
R> verif_outc1   = verif_OT(outc1, group.class = TRUE, ordinal = FALSE,
                           stab.prob = TRUE, min.neigb = 5)
### First results related to outc1:
R> verif_outc1$nb.profil

27

R> verif_outc1$res.prox

      N   V_cram rank_cor
362.000    0.860    0.892
```

In output, the `nb.profil` value gives the number of profiles of predictors globally detected in the two databases (27). In the example, a profile will be characterized by a combination of values related to the three predictors kept: `grad.sch`, `meals` and `stype`.

The first step of the function is dedicated to the study of the proximity between the two target variables. Therefore, standard criteria (Cramer's V and Spearman's rank correlation coefficient) are used to evaluate the pairwise association between Y and Z, globally or in one of the two databases only, according to the predictions provided by the output of `OT_outcome` or `OT_joint`. Only the database `api35` was completed in the example (because `which.DB = "B"` as argument of `OT_outcome`) and stored in the `DATA2_OT` object, therefore, the criteria compare here the proximity distribution between the predicted values of `apicl_2000` (`Y`) and the observed value of `apicl_1999` (`Z`) in the database `api35` (`B`). Regarding independence or a small correlation between Y^A and Z^A (or Y^B and Z^B) must question about the reliability of the predictions especially when Y and Z are supposed to summarize a same information. In the example, whatever the criteria used, we notice via the `res.prox` object, a strong association between the two target variables in `api35` which reassures about the consistency of the predictions. The related confusion matrix between the predicted values of `apicl_2000` (`Y`) and the observed value of `apicl_1999` (`Z`) is stored in the `conf.mat` object.

Second, the function proposes an optional tool (by setting `group.clss= TRUE`) which evaluates the impact of grouping levels of one factor on the association of Y and Z. When users have initially no information about one of the two encodings of interest, this approach can be particularly useful to detect ordinal from nominal ones and its principle is as follow. Assuming that $Y \in \mathcal{Y}$, and $Z \in \mathcal{Z}$ ($\mathcal{Y}$ and $\mathcal{Z}$ are the respective levels) and that $|\mathcal{Y}| \geq |\mathcal{Z}|$. From Y, successive new variables $Y' \in \mathcal{Y}'$ are built, as soon as $\mathcal{Y}'$ is a partition of $\mathcal{Y}$ such as $|\mathcal{Y}'| = |\mathcal{Z}|$ (and inversely for Z if the levels of Z is the greatest). The related associations between Z and Y' (with now equal number of levels) are then studied using: Cramer's V, rank correlation, Kappa coefficient and confusion matrix and the results are stored in a table called `res.grp`. The corresponding outputs of the example are:

```
R> verif_outc1$conf.mat

           Z
```

```
predY       G1  G2  G3  G4 Sum
[200-600]   81   1   1   1  84
(600-800]   10  89  55   1 155
(800-1000]   0   0  34  89 123
Sum         91  90  90  91 362

R> verif_outc1$res.grp

grp levels Z to Y error_rate Kappa Vcramer RankCor
      G1/G2 G3/G4       13.3 0.794    0.83   0.877
      G1/G2/G3 G4       19.1 0.714    0.78   0.839
      G1 G3/G2/G4       28.2 0.593    0.68   0.624
      G1 G2/G3/G4       37.6 0.457    0.64   0.813
      G1 G4/G2/G3       43.4 0.374    0.59   0.115
      G1/G2 G4/G3       43.4 0.326    0.64   0.574
```

It appears from these results that grouping the levels `G2` and `G3` of `apicl_1999` (`Z`) strongly improves the association of this variable with `apicl_2000` (`Y`) (the error rate of the confusion matrix varies from 43.4 to 13.3). Moreover the structure of `conf.mat` confirms that the encoding of `apicl_1999` seems to be ordinal.

The third step of the quality process integrated in the function corresponds to the study of the Hellinger distance (Liese and Miescke (2007)). This distance function is used as a measure of the discrepancies between the observed and predicted distributions of Y ($\mathcal{L}(Y^A)$ versus $\mathcal{L}(\widehat{Y}^B)$) and/or ($\mathcal{L}(\widehat{Z}^A)$ versus $\mathcal{L}(Z^B)$). For Y and Z, the definition of the distance is respectively:

$$\text{dist}_{\text{hell}}(Y^A, \widehat{Y}^B) = \sqrt{\frac{1}{2}\sum_{y\in\mathcal{Y}}\left(\sqrt{\mu_{n,y}^{Y^A}} - \sqrt{\mu_{n,y}^{\widehat{Y}^B}}\right)^2)}$$

and

$$\text{dist}_{\text{hell}}(Z^B, \widehat{Z}^A) = \sqrt{\frac{1}{2}\sum_{z\in\mathcal{Z}}\left(\sqrt{\mu_{n,z}^{Z^B}} - \sqrt{\mu_{n,z}^{\widehat{Z}^A}}\right)^2)}$$

where $\mu_{n,y}^{\widehat{Y}^B}$ and $\mu_{n,z}^{\widehat{Z}^A}$ corresponds to the empirical estimators of $\mu^{\widehat{Y}^B}$ and $\mu^{\widehat{Z}^A}$ respectively.

The Hellinger distance varies between 0 (identical) and 1 (strong dissimilarities) while 0.05 can be used as an acceptable threshold below which two distributions can be considered as similar. When OUTCOME models are applied on datasets, this criterion shows that predictions respect assumption 1. It can also be used to determine the best relaxation parameter of an R-OUTCOME model. On the contrary, there is no need to interpret this criterion when an R-JOINT model is applied, because in this case, the assumption 1 is not required. Results related to this criterion are stored in the `hell` object:

```
R> verif_outc1$hell
                YA_YB ZA_ZB
Hellinger dist. 0.008    NA
```

With a p-value equals to 0.08, the assumption 1 is difficult to hold here and must challenge the user on the need to switch the algorithm or to improve the current one by adding relaxation and/or regularization parameters.

Finally, the `verif_OT` function uses the mean and standard deviance of the conditional probabilities $\mathbb{P}(Z = \hat{z}_i|Y = y_i, X = x_i)$ estimated by each model, as indicators of the stability of the individual predictions (provided that `stab.prob = TRUE`). It is nevertheless possible that conditional probabilities are computed from too few individuals (according to the frequency of each profile of shared variables met), to be considered as a reliable estimate of the reality. To avoid this problem, trimmed means and standard deviances are suggested by removing these specific probabilities from

the computation, using the `min.neigb` parameter. In output, the results related to this last study are stored in the `res.stab` object:

```
R> verif_outc1$eff.neig
Nb.neighbor Nb.Prob
          1      14
          2      18
          3      18
          4      28
          5      20
          6       6
          7      14
          8      16
          9      27
         10      20

R> verif_outc1$res.stab
         N min.N  mean    sd
2nd DB 284     5 0.968 0.122
```

The first result shows that 14 individual predictions among 362 (the total number of rows in `api35`) have been assigned to subjects that exhibits a unique combination of predictors and outcome. From these subjects, it would be obviously overkill to draw conclusions about the reliability of the predictions provided by the model. We therefore decide to fix here a threshold of 5 below which it is difficult to extract any information related to the prediction. From the remaining ones, we simply perform the average (0.968) which could be interpreted as follows: When the fitted model is confronted with a same profile of predictors and a same level of `apicl_1999`, more than 96 times of a hundred, it will return the same individual prediction.

We run a total of 11 models to determine the optimal one for the prediction of `apicl_2000` in `api35`. Every arguments from the syntax of the `OT_outcome` and `OT_joint` functions described on pages 15, 16 and 17, stay unchanged with the exception of `indiv.method`, `maxrelax`, and `lambda.reg` which vary. The values linked to each criterion of the `verif_OT` function are summarized in Tables 8, 9 and 10. The syntax related to `verif_OT` stay unchanged for each model (notably same `min.neigb` arguments). Note that comparisons of stability predictions between OUTCOME and JOINT models impose that the `prox.dist` argument of the `OT_outcome` function is fixed to 0.

Model	Type	Method	Relax	Regul	N	V_cram	rank_cor
outc1	OUTCOME	SEQUENTIAL	0.0	-	362	0.86	0.892
R_outc1	R-OUTCOME	SEQUENTIAL	0.4	-	362	0.94	0.923
R_outc2	R-OUTCOME	SEQUENTIAL	0.6	-	362	0.91	0.917
R_outc3	R-OUTCOME	OPTIMAL	0.0	-	362	0.87	0.911
R_outc4	R-OUTCOME	OPTIMAL	0.4	-	362	0.95	0.939
R_outc5	R-OUTCOME	OPTIMAL	0.6	-	362	0.92	0.932
outj1	JOINT	-	0.0	0.0	362	0.74	0.834
R_outj1	R-JOINT	-	0.4	0.0	362	0.95	0.935
R_outj2	R-JOINT	-	0.6	0.0	362	0.91	0.927
R_outj3	R-JOINT	-	0.8	0.0	362	0.91	0.927
R_outj4	R-JOINT	-	0.4	0.1	362	0.95	0.931

Table 8: Proximity between `apicl_2000` (Y predicted) and `apicl_1999` (Z observed) in `api35`: Summary table

From these results, we can conclude that:

- whatever the algorithm used (OUTCOME or JOINT), adding a relaxation parameter improves here the association between the target variables (see Table 8).
- according to the results of Table 9, the `outc1` and `R_outc3` models are not optimal because the observed values from the Hellinger criterion reflects a violation of assumption 1 (see Table 9). Therefore, for these two models, a relaxation parameter must be added or increased respectively.
- conversely, adding a relaxation parameter that is too high seems to affect the quality of the association. In this example, 0.4 seems to be a good value for this parameter whatever the algorithm used. Among the remaining models (`R_outc1`, `R_outc4`, `R_outj1`, and `R_outj4`), `R_outj1` and `R_outj4` seem to be those with the most stable predictive potential (Table 10). Moreover, `R_outc4` appears here as the best model from the tested OUTCOME algorithms.
- adding a regularization parameter to the `R_outj1` model caused a decrease in the stability of the predictions (see the value of `R_outj4`) and we thus conclude in favor of `R_outj1` as best model among those tested in the JOINT family of algorithms.
- Table 11 shows that the three remaining models counted between 86 an 91% of common predictions and these last result also reassures the user about the quality of the provided predictions.

Model	Type	Method	Relax	Hell(YA_YB)
outc1	OUTCOME	SEQUENTIAL	0.0	0.008
R_outc1	R-OUTCOME	SEQUENTIAL	0.4	0.085
R_outc2	R-OUTCOME	SEQUENTIAL	0.6	0.107
R_outc3	R-OUTCOME	OPTIMAL	0.0	0.002
R_outc4	R-OUTCOME	OPTIMAL	0.4	0.080
R_outc5	R-OUTCOME	OPTIMAL	0.6	0.102

Table 9: Hellinger distance: Summary table

Model	Type	Method	Relax	Regul	N	mean	sd
outc1	OUTCOME	SEQUENTIAL	0.0	-	284	0.968	0.122
R_outc1	R-OUTCOME	SEQUENTIAL	0.4	-	284	0.950	0.151
R_outc2	R-OUTCOME	SEQUENTIAL	0.6	-	284	0.954	0.145
R_outc3	R-OUTCOME	OPTIMAL	0.0	-	284	0.979	0.100
R_outc4	R-OUTCOME	OPTIMAL	0.4	-	284	0.987	0.080
R_outc5	R-OUTCOME	OPTIMAL	0.6	-	284	0.983	0.091
outj1	JOINT	-	0.0	0.0	284	0.911	0.116
R_outj1	R-JOINT	-	0.4	0.0	284	0.942	0.128
R_outj2	R-JOINT	-	0.6	0.0	284	0.953	0.171
R_outj3	R-JOINT	-	0.8	0.0	284	0.934	0.199
R_outj4	R-JOINT	-	0.4	0.1	284	0.926	0.097

Table 10: Stability of the predictions: Summary table (`min.neigb = 5`)

The confusion matrices related to `R_outc4` and `R_outj1` are described in Table 12 and seems to confirm that the encoding of `apicl_2000` in three groups, could simply correspond to the grouping of levels $G2$ and $G3$ of the `api_cl1999` variable.

Finally, note that the running time of each model took less than 15 seconds with `R` version 4.0.3 for Windows (10 Pro-64bits/ Process Intel 2.66 GHz).

Model	outc1	R_outc1	R_outc4	outj1	R_outj1
R_outc1	0.84	-	-	-	-
R_outc4	0.83	0.95	-	-	-
outj1	0.72	0.81	0.83	-	-
R_outj1	0.83	0.91	0.94	0.84	-
R_outj4	0.84	0.96	0.97	0.84	0.92

Table 11: Ratio of common predictions between two models

(a)	**apicl_1999**				(b)	**apicl_1999**			
apicl_2000	$G1$	$G2$	$G3$	$G4$	**apicl_2000**	$G1$	$G2$	$G3$	$G4$
$[200-600]$	91	15	0	0	$[200-600]$	91	14	1	0
$(600-800]$	0	75	90	0	$(600-800]$	0	76	89	0
$(800-1000]$	0	0	0	91	$(800-1000]$	0	0	0	91

Table 12: Confusion matrices in the api35 dataset for the models (a) R_outc4 and (b) R_outj1

5 Conclusion and perspectives

To our knowledge, OTrecod is the first R package that takes advantage of the optimal transportation theory (Monge (1781)) in the context of data fusion and the comparative study of methods described in Gares and Omer (2020) underlines the promising performances of these approaches.

For more details and examples about the functions, users are invited to consult the "ARTICLES" section of the related website https://otrecoding.github.io/OTrecod/.

5.1 Drawbacks

The functions of OTrecod only apply to a specific data fusion context where there is no overlapping part between the two raw data sources. This package does not deal with record linkage which is already the focus of extensive works (Sariyar and Borg, 2010). This package is not adapted when the target variables (outcomes) of the two databases are continuous with an infinite number of values (like weights in grams with decimals for example). Moreover, if the data fusion requires only one shared variable to work, the quality of the fusion depends on the presence of a subset of good shared predictors in the raw databases. Finally, if the function OT_outcome allows all types of predictors, the current version of the function OT_joint imposes categorical matching variables only (scale variables are allowed): this restriction should be relaxed in a next version.

5.2 Perspectives

A number of more advanced research still need further investigation:

1. the possibility of extending the OT algorithm for recoding variables to multidimensional frameworks.
2. the stability of the algorithm when the matching variables are incomplete and the non-response processes are missing at random or not.
3. the contribution of calibration techniques in the quality process assessment (Deming and Stephan, 1940)
4. the creation of a tuning function which defines a grid search to find the optimal combination of parameters related to the OT algorithms, could be added in the future versions of the package.

6 Acknowledgments

The authors would especially thank Pierre Navaro (CNRS UMR 6625) for their advices during the implementation of the **OTrecod** package. This research has received the help from **Region Occitanie** Grant RBIO-2015-14054319 and Mastodons-CNRS Grant.

References

J. C. Adamek. Fusion: combining data from separate sources. Marketing Research, 6(3):48, 1994.

W. Bergsma. A bias-correction for cramér's v and tschuprow's t. Journal of the Korean Statistical Society, 42(3):323–328, 2013.

L. Breiman. Random forests. Machine learning, 45(1):5–32, 2001.

L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. Classification and regression trees. CRC press, 1984.

F. Castanedo. A review of data fusion techniques. The Scientific World Journal, 2013:704504, 2013.

N. Cibella. How to choose the matching variables, report wp2, ess-net. Statistical Methodology Project on Integration of Surveys and Administrative Data, EUROSTAT, 2010.

M. Cohen. Statistical matching and microsimulation models, improving information for social policy decisions, the use of microsimulation modeling, technical papers, ii. Washington, DC: National Academy, 1991.

M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, Advances in Neural Information Processing Systems, volume 26. Curran Associates, Inc., 2013. URL `https://proceedings.neurips.cc/paper/2013/file/af21d0c97db2e27e13572cbf59eb343d-Paper.pdf`.

S. Das and D. S. Tripathy. OMICsPCA: An R package for quantitative integration and analysis of multiple omics assays from heterogeneous samples, 2020. R package version 1.8.0.

W. E. Deming and F. F. Stephan. On a least squares adjustment of a sampled frequency table when the expected marginal totals are known. The Annals of Mathematical Statistics, 11(4):427–444, 1940.

M. D'Orazio and M. M. D'Orazio. Package 'statmatch'. Statistical matching or data fusion, version, 1(0), 2019.

M. D'Orazio, M. Di Zio, and M. Scanu. Statistical matching: Theory and practice. John Wiley & Sons, 2006.

D. E. Farrar and R. R. Glauber. Multicollinearity in regression analysis: the problem revisited. The Review of Economic and Statistics, pages 92–107, 1967.

J. Forrest, D. de la Nuez, and R. Lougee-Heimer. Clp user guide. IBM Research, 2004.

V. Gares and J. Omer. Regularized optimal transport of covariates and outcomes in data recoding. Journal of American Statistical Association, 2020.

V. Gares, C. Dimeglio, G. Guernec, R. Fantin, B. Lepage, M. R. Kosorok, and N. Savy. On the use of optimal transportation theory to recode variables and application to database merging. The International Journal of Biostatistics, 1(ahead-of-print), 2019.

K. A. Grajski, L. Breiman, G. V. Di Prisco, and W. J. Freeman. Classification of eeg spatial patterns with a tree-structured methodology: Cart. IEEE transactions on biomedical engineering, 33(12): 1076–1086, 1986.

D. Hall and J. Llinas. An introduction to multisensor data fusion. Proceedings of the IEEE, International Conference on Intelligent Robots and Systems, 85:6–23, 1997.

T. Hastie, R. Mazumder, J. D. Lee, and R. Zadeh. Matrix completion and low-rank svd via fast alternating least squares. Journal of Machine Learning Research, 16(104):3367–3402, 2015. URL http://jmlr.org/papers/v16/hastie15a.html.

C. Hernandez-Ferrer, C. Ruiz-Arenas, A. Beltran-Gomila, and J. R. González. Multidataset: an r package for encapsulating multiple data sets with application to omic data integration. BMC bioinformatics, 18(1):36, 2017.

F. Hitchcock. The distribution of a product from several sources to numerous localities. Journal of mathematics and physics / Massachusetts Institute of Technology., 20:224–230, 1941.

T. Hothorn, P. Buehlmann, S. Dudoit, A. Molinaro, and M. Van Der Laan. Survival ensembles. Biostatistics, 7(3):355–373, 2006a.

T. Hothorn, K. Hornik, and A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. Journal of Computational and Graphical Statistics, 15(3):651–674, 2006b.

J. Josse and F. Husson. missmda: A package for handling missing values in multivariate data analysis. Journal of Statistical Software, 70(1):1–31, 2016. doi: 10.18637/jss.v070.i01. URL https://www.jstatsoft.org/index.php/jss/article/view/v070i01.

J. B. Kadane. Some statistical problems in merging data files. 1978 Compendium of Tax Research, pages 159–171, 1978.

L. Kantorovich. On the transfer of masses. Doklady Akademii Nauk SSSR, 37:7–8, 1942.

L. A. Klein. Sensor and data fusion: a tool for information assessment and decision making, volume 138. SPIE press, 2004.

S. Kotz, N. Balakrishnan, and N. L. Johnson. Continuous multivariate distributions, Volume 1: Models and applications. John Wiley & Sons, 2004.

F. Liese and K.-J. Miescke. Statistical decision theory. In Statistical Decision Theory, pages 1–52. Springer, 2007.

R. Little and D. Rubin. Statistical analysis with missing data, third edition. Wiley series in probability and mathematical statistics. Probability and mathematical statistics. Wiley, 2019.

A. Makhorin. Gnu linear programming kit, reference manual. Free software foundation, 4, 2011.

I. Mayer, A. Sportisse, J. Josse, N. Tierney, and N. Vialaneix. R-miss-tastic: a unified platform for missing values methods and workflows, 2019. URL https://arxiv.org/abs/1908.04822.

G. Monge. Mémoire sur la Théorie des Déblais et des Remblais. Histoire de l'Académie royale des sciences de Paris, pages 666–704, 1781.

B. Muzellec, J. Josse, C. Boyer, and M. Cuturi. Missing data imputation using optimal transport. In ICML, 2020.

G. Paass. Statistical match: evaluation of existing procedures and improvements by using additional information. Microanalytic Simulation Models to Support Social and Financial Policy, pages 401–420, 1986.

J. Pages. Analyse factorielle multiple appliquée aux variables qualitatives et aux données mixtes. Revue de statistique appliquée, 4:5–37, 2002.

R Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, 2016. URL https://www.R-project.org/. ISBN 3-900051-07-0.

S. Rässler. Statistical matching: A frequentist theory, practical applications, and alternative Bayesian approaches, volume 168. Springer Science & Business Media, 2012.

W. L. Rodgers. An evaluation of statistical matching. Journal of Business & Economic Statistics, 2(1):91–102, 1984.

F. Rohart, B. Gautier, A. Singh, and K.-A. Lê Cao. mixomics: An r package for 'omics feature selection and multiple data integration. PLoS computational biology, 13(11):e1005752, 2017.

D. B. Rubin. Statistical matching using file concatenation with adjusted weights and multiple imputations. Journal of Business & Economic Statistics, 4(1):87–94, 1986.

M. Sariyar and A. Borg. The recordlinkage package: Detecting errors in data. The R Journal, 2(2): 61–67, 2010.

M. Scanu. Recommendations on statistical matching, report wp2, ess-net. Statistical Methodology Project on Integration of Surveys and Administrative Data, 2010.

D. Schuhmacher, B. Bähre, C. Gottschlich, V. Hartmann, F. Heinemann, and B. Schmitzer. transport: Computation of Optimal Transport Plans and Wasserstein Distances, 2020. URL https://cran.r-project.org/package=transport. R package version 0.12-2.

D. J. Stekhoven and P. Bühlmann. Missforest–non-parametric missing value imputation for mixed-type data. Bioinformatics (Oxford, England), 28:112–8, Jan 2012.

C. Strobl, A.-L. Boulesteix, A. Zeileis, and T. Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. BMC Bioinformatics, 8(25), 2007. URL http://www.biomedcentral.com/1471-2105/8/25.

C. Strobl, A.-L. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis. Conditional variable importance for random forests. BMC Bioinformatics, 9(307), 2008. URL http://www.biomedcentral.com/1471-2105/9/307.

C. Strobl, T. Hothorn, and A. Zeileis. Party on! The R Journal, 1(2):14–17, Dec. 2009. URL http://journal.r-project.org/archive/2009/RJ-2009-013/index.html.

S. Theußl, F. Schwendinger, and K. Hornik. Roi: The r optimization infrastructure package. Research Report Series / Department of Statistics and Mathematics 133, WU Vienna University of Economics and Business, Vienna, October 2017. URL http://epub.wu.ac.at/5858/.

S. Theußl, F. Schwendinger, and K. Hornik. Roi: An extensible r optimization infrastructure. Journal of Statistical Software, 2020.

S. Van Buuren and K. Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. Journal of Statistical Software, Articles, 45(3):1–67, 2011.

S. Van Buuren, H. C. Boshuizen, and D. L. Knook. Multiple imputation of missing blood pressure covariates in survival analysis. Statistics in medicine, 18(6):681–694, 1999.

B. Vantaggi. Statistical matching of multiple sources: A look through coherence. Int. J. Approx. Reasoning, 49:701–711, 11 2008. doi: 10.1016/j.ijar.2008.07.005.

A. Zeileis, T. Hothorn, and K. Hornik. Model-based recursive partitioning. Journal of Computational and Graphical Statistics, 17(2):492–514, 2008. doi: 10.1198/106186008X319331. URL https://doi.org/10.1198/106186008X319331.

Z. Zhu, T. Wang, and R. J. Samworth. High-dimensional principal component analysis with heterogeneous missingness, 2019. URL https://arxiv.org/abs/1906.12125.

7 SUPPLEMENTS

```
### BASIC R CODE FOR SUMMARY TABLES  8, 9, 10, and 11 ---------

## Validation of each model: Repeat the following R command for each model by
## changing outc1:
R> verif_outc1   = verif_OT(outc1, group.class = TRUE, ordinal = FALSE,
                            stab.prob = TRUE, min.neigb = 5)

## Association between Y and Z: Summary Table 8
R> res.prx = rbind(
           outc1   = verif_outc1$res.prox    , R_outc1 = verif_R_outc1$res.prox,
           R_outc2 = verif_R_outc2$res.prox  , R_outc3 = verif_R_outc3$res.prox,
           R_outc4 = verif_R_outc4$res.prox  , R_outc5 = verif_R_outc5$res.prox,
           outj1   = verif_outj1$res.prox    , R_outj1 = verif_R_outj1$res.prox,
           R_outj2 = verif_R_outj2$res.prox  , R_outj3 = verif_R_outj3$res.prox,
           R_outj4 = verif_R_outj4$res.prox )

R> res.prx = data.frame(Model = c("outc1","R_outc2","R_outc3","R_outc4",
                                  "R_outc5","R_outc6","outj1", "R_outj1",
                                  "R_outj2","R_outj3","R_outj4"),
                        Type  = c("OUTCOME",rep("R-OUTCOME",5),"JOINT",
                                  rep("R-JOINT",4)),
                        Relax = c(0,0.4,0.6,0,0.4,0.6,0,0.4,0.6,0.8,0.4),
                        Regul = c(rep(0,10),0.1), res.prx)

R> row.names(res.prx) = NULL; head(res.prx,3)

   Name      Type Relax Regul   N V_cram rank_cor
  outc1   OUTCOME   0.0   0.0 362   0.86    0.892
R_outc1 R-OUTCOME   0.0   0.0 362   0.87    0.911
R_outc2 R_OUTCOME   0.4   0.0 362   0.93    0.933
#-----

## Hellinger distance: Summary Table 9
R> res.helld = rbind(
            outc1   = verif_outc1$hell  , R_outc1 = verif_R_outc1$hell,
            R_outc2 = verif_R_outc2$hell, R_outc3 = verif_R_outc3$hell,
            R_outc4 = verif_R_outc4$hell, R_outc5 = verif_R_outc5$hell,
            outj1   = verif_outj1$hell  , R_outj1 = verif_R_outj1$hell,
            R_outj2 = verif_R_outj2$hell, R_outj3 = verif_R_outj3$hell,
            R_outj4 = verif_R_outj4$hell )

res.helld = data.frame(res.prx[,1:4], res.helld)
row.names(res.helld) = NULL; res.helld
#-----

## Stability of the prediction: Summary Table 10
# same R code as for Summary Table 8, changing res.prox by res.stab
#----

## Ratio of common predictions: Table 11

R> stoc = list(outc1$DATA2_OT$OTpred  , R_outc1$DATA2_OT$OTpred,
               R_outc4$DATA2_OT$OTpred, outj1$DATA2_OT$OTpred  ,
```

```
              R_outj1$DATA2_OT$OTpred, R_outj4$DATA2_OT$OTpred)

R> corpred = matrix(ncol = 6, nrow = 6)
R> for (i in 1:6){
      for (j in 1:6){
        corpred[i,j] = round(sum(diag(table(stoc[[i]],stoc[[j]])))/362,2)
      }
   }; corpred
#-----
```