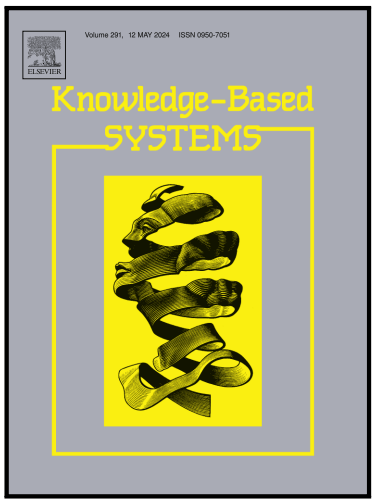


# Statistical Matching using Autoencoders-Canonical Correlation Analysis, Kernel Canonical Correlation Analysis and Multi-output Multilayer Perceptron

Hugues Annoy , Alessandro Beretta , Cédric Heuchenne

PII: S0950-7051(25)01665-X
DOI: https://doi.org/10.1016/j.knosys.2025.114626
Reference: KNOSYS 114626

To appear in: *Knowledge-Based Systems*

Received date: 11 August 2024
Revised date: 1 October 2025
Accepted date: 4 October 2025

Please cite this article as: Hugues Annoy , Alessandro Beretta , Cédric Heuchenne , Statistical Matching using Autoencoders-Canonical Correlation Analysis, Kernel Canonical Correlation Analysis and Multi-output Multilayer Perceptron, *Knowledge-Based Systems* (2025), doi: https://doi.org/10.1016/j.knosys.2025.114626

**Highlights**

- One new statistical matching technique based on Autoencoders is proposed.
- This novel statistical matching technique handle both numerical and categorical features.
- This techniques handle sample weights.
- Experimental comparisons are done with two different data sets using several criteria.

# Statistical Matching using Autoencoders-Canonical Correlation Analysis, Kernel Canonical Correlation Analysis and Multi-output Multilayer Perceptron*

Hugues Annoye (Corresponding Author)[a,b,d], Alessandro Beretta[a,e], Cédric Heuchenne[a,c,b,f]

[a] *CAPE, UCLouvain Saint-Louis Bruxelles*
[b] *ISBA, Université catholique de Louvain*
[c] *HEC, Université de Liège*
[d] *Email: hugues.annoye@uclouvain.be*
[e] *Email: beretta.alessandro@me.com*
[f] *Email: cedric.heuchenne@uclouvain.be*

**Abstract**

A lot of data are gathered every day, whether via surveys or other sources. For many people, the need for variables from different data sources is a key factor and leads to the need of methods to combine them. A recognized practice to combine data sets in this field is statistical matching. In this paper, we investigate and extend to statistical matching an Autoencoders-Canonical Correlation Analysis (A-CCA). A-CCA is an extension of KCCA, that reduces the need for kernels, with the added benefit of a dimensionality reduction. It can be regarded as an extension of Deep Canonical Correlation Analysis (DCCA), providing enhanced flexibility that makes it well suited for statistical matching. This method is designed to deal with various variable types, sampling weights and incompatibilities among categorical variables. We compare the performance of this method with other methods based on Kernel Canonical Correlation Analysis (KCCA) or Multi-output Multilayer Perceptron (MMLP), using 2017 Belgian Statistics on Income and Living Conditions (SILC). We divide this data set in two parts and we act as if they were coming from two different sources.

*Keywords:*
Statistical matching, Canonical Correlation Analysis, Kernel Canonical Correlation Analysis, Multi-output Multilayer Perceptron, Autoencoders



*This work was funded by Innoviris in the Prospective Research for Brussels program (project 2019-PRB-8) and European Regional Development Fund (ERDF, project 2021BE16RFPR001-T-11-05). We gratefully acknowledge data in the form of the Survey on Income and Living Conditions (2016 and 2017) from Eurostat and the Household Budget Survey (2016) from Statistics Belgium.

**Abstract**

A lot of data are gathered every day, whether via surveys or other sources. For many people, the need for variables from different data sources is a key factor and leads to the need of methods to combine them. A recognized practice to combine data sets in this field is statistical matching. In this paper, we investigate and extend to statistical matching an Autoencoders-Canonical Correlation Analysis (A-CCA). A-CCA is an extension of KCCA, that reduces the need for kernels, with the added benefit of a dimensionality reduction. It can be regarded as an extension of Deep Canonical Correlation Analysis (DCCA), providing enhanced flexibility that makes it well suited for statistical matching. This method is designed to deal with various variable types, sampling weights and incompatibilities among categorical variables. We compare the performance of this method with other methods based on Kernel Canonical Correlation Analysis (KCCA) or Multi-output Multilayer Perceptron (MMLP), using 2017 Belgian Statistics on Income and Living Conditions (SILC). We divide this data set in two parts and we act as if they were coming from two different sources.

## 1. Introduction

In business applications as well as in the scientific world, the accessibility of data remains a problem due to the fact that data are not available from a single source. *Statistical matching* (D'Orazio et al., 2006b), also referred to as *data fusion*, *synthetical matching* or *statistical record-linkage*, can be seen as a solution to this issue. Indeed, these techniques introduced in Anderson (1957) can be used to merge two data sets when their record linkage is impossible either because an identifier of individuals such as the social security number is not available or because, as it is the case with independent sample surveys conducted on large populations, the samples do not overlap.

In the study of the consumption expenditure in function of the income of the household the use of statistical matching is common (Tonkin and Webber, 2012; Donatiello et al., 2014; Serafino and Tonkin, 2017; López-Laborda et al., 2020). Indeed, the national statistical offices of many countries collect these informations, but usually not in a single survey (for example, in Europe we have the Household Budget Survey (HBS) and the Statistics on Income and Living Conditions (SILC)). Then, the purpose of statistical matching is to create a unique data set where income and expenditure are displayed together. National institutions also use statistical matching methods as in Saverio et al. (2008), where they match two Italian surveys (Labour Force and Time Use) to have one synthetic data set and avoid the costs of doing another survey in which both information are jointly collected. For a more detailed history, see Rässler (2002) while a detailed literature review of statistical matching can be found in Kim and Shao (2013); Van Buuren (2018); Fosdick et al. (2016); Conti et al. (2017).

The purpose of this paper is to present new statistical matching procedures based on autoencoder and Canonical Correlation Analysis (CCA). We compare this method with Kernel Canonical Correlation Analysis (KCCA) and Multi-output Multilayer Perceptron (MMLP), as well as with more classical econometric methods using distance hot-deck (HD) and multivariate linear/multinomial logistic regressions (REG). This A-CCA method offers an advantage in reducing dimensionality compared to KCCA, which can be highly advantageous.

This new approach, called Autoencoder Canonical Correlation Analysis (A-CCA), utilizes the autoencoder, an artificial neural network, to create a lower dimensional representation of the data on which CCA is applied to achieve statistical matching. Autoencoders have already been used in conjunction with CCA, primarily to learn deep multi-view representations. Wang et al. (2016) consider two matched sequences of data (e.g. images or audio data), where one of the sequences is incomplete. In this case, they propose deep canonically correlated autoencoders (DCCAE), which are similar to our A-CCA technique, but used for classification purposes. Kaloga et al. (2020) propose an extension of this in the form of multiview variational graph autoencoders for CCA (MVGCCA). Xiu et al. (2022) apply this idea to design a nonlinear process monitoring method, and for the same task, Xiu and Li (2023) use KCCA while Xiu et al. (2024) propose Joint Sparse Constrained Canonical Correlation Analysis (JSCCCA).

An autoencoder is an unsupervised neural network consisting of an encoder that compresses data efficiently by utilizing the underlying structure therein, and a decoder, which decompresses the data into a representation that resembles the original version as closely as possible. It was first introduced in Rumelhart et al. (1986) and most often finds its application in the fields of image compression, denoising and generation. Therefore, the background for using it for the specific intent of statistical matching is scarce if not nonexistent. However, autoencoders have been employed for the same purpose of compression and revealing underlying structures and dependencies in data in other fields. An example can be found in Luo et al. (2018), where autoencoders are employed to learn the underlying semantic dependencies in speech. Their aim is to match linguistic inputs and responses in a way that generates coherent dialogue. However, in contrast to our approach, the autoencoders are treated separately from their matching module, which is composed of a multi-layer perceptron.

We compare that methodology with KCCA, MMLP and more classical methods (HD and REG). KCCA was first used in statistical matching by Mitsuhiro and Hoshino (2020) and we extended it in Annoye et al. (2024) to take into account both the sampling weights and the problem of incompatibility between categorical variables, as pointed out by D'Orazio et al. (2006a). That type of matching assigns new values using kernelized means of the observed ones. KCCA is a machine learning technique developed by Lai and Fyfe (2000) and Akaho (2001) as an extension of the classical Canonical Correlation Analysis (CCA) introduced in Hotelling (1936). It is a technique used to detect nonlinear relationships thanks to the kernel trick. It maps data into higher dimensional spaces where a classical CCA is performed.

The article is structured in the following way. In section 2, we present background information about statistical matching including KCCA. Then, section 3 provides the methodology

using A-CCA. Afterward, we provide in section 4, an application using the SILC 2017 survey data for Belgium; we treat them as if they were coming from two separate sources and, using a cross-validation technique, compare the performance of the proposed statistical matching methods. Finally, in Section 5, we present an application in which we merge the HBS 2016 and SILC 2016 surveys for Belgium, like in Tonkin and Webber (2012); Serafino and Tonkin (2017); Annoye et al. (2024).

## 2. Background

We consider two independent sample surveys that are encoded into two data sets $A$ and $B$ containing $n_A$ and $n_B$ individuals, respectively. Each data set can be decomposed in three matrices : a matrix of common variables denoted $\mathbf{X}$ and two matrices of non-common variables denoted $\mathbf{Y}$ and $\mathbf{Z}$. Let us denote the number of columns of each matrices by $J_q$, for $q \in \{x, y, z\}$, each column corresponding to a variable. An important point is that for each data set there is one of the two non-common variable matrices missing. For the rest of the paper we will assume, that it is $\mathbf{Z}^A$ (resp. $\mathbf{Y}^B$) for $A$ (resp. $B$) that is missing.

The dimensions of all these matrices can be found in Figure 1. $\mathbf{x}_i^A$ will denote the $i$-th individual in $\mathbf{X}^A$ while $\mathbf{w}^A$ (resp. $\mathbf{w}^B$) is used for the vectors of individual weights of data set $A$ (resp. $B$). We suppose that in the rest of this work, these weights are standardized such that their sum is equal to one and all weights are bounded between zero and one. Finally, we indicate by $\mathbf{W}^A$ and $\mathbf{W}^B$ the diagonal matrices that correspond to these two vectors.

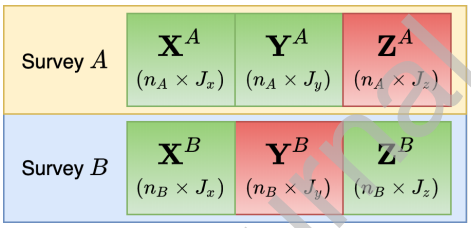


Figure 1: Sample survey data.

The goal of the methodologies described in this paper is to create a unique data set, where $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ are jointly displayed. For simplification purposes, in the rest of the paper, we will only describe the creation of the synthetic data set $(\mathbf{X}^B, \mathbf{Y}^B, \mathbf{Z}^B)$. Of course, the same procedure can be used to create $(\mathbf{X}^A, \mathbf{Y}^A, \mathbf{Z}^A)$.

### 2.1. Sampling Weights

As in Annoye et al. (2024), all the algorithms in this paper take the sampling weights into account. Sampling weights are provided to allow our analysis of such a data base to be as close as possible to the population. This is due to the fact that many surveys are conducted via stratification, and two different individuals do not have the same probability of being drawn when the sampling is conducted.

These weights will be numbers that are proportional to the inverse of the probability of selection that we will correct for non-response. That will also force our sample to reproduce some characteristics of the population of interest.

### 2.2. Kernel Canonical Correlation Analysis (KCCA)

In Annoye et al. (2024), we developed an algorithm based on Kernel Canonical Correlation Analysis (KCCA), which takes into account both the sampling weights and the incompatibility problem between the categorical variables.

Recall that, KCCA is an extension of the classical Canonical Correlation Analysis (CCA), but where the matrices $\mathbf{X}$ and $\mathbf{Y}$ are projected into a feature space :

$$\mathbf{\Phi}_x(\mathbf{X}) = (\boldsymbol{\phi}_x(\mathbf{x}_1), \boldsymbol{\phi}_x(\mathbf{x}_2), \cdots, \boldsymbol{\phi}_x(\mathbf{x}_n)) \in H_x$$
and
$$\mathbf{\Phi}_y(\mathbf{Y}) = \left(\boldsymbol{\phi}_y(\mathbf{y}_1), \boldsymbol{\phi}_y(\mathbf{y}_2), \cdots, \boldsymbol{\phi}_y(\mathbf{y}_n)\right) \in H_y,$$

where $H_x$ and $H_y$ are Hilbert spaces. To simplify the notations, we suppose in the rest of the paper that mapped data are centered. One can easily extend to the non-centered case with the technique used for the Kernel Principal Component Analysis (KPCA) in Schölkopf et al. (1998).

$\mathbf{U} = \langle \mathbf{a} | \mathbf{\Phi}_x(\mathbf{X}) \rangle$ and $\mathbf{V} = \left\langle \mathbf{b} \middle| \mathbf{\Phi}_y(\mathbf{Y}) \right\rangle$, where $\mathbf{a} \in H_x$ and $\mathbf{b} \in H_y$, are inner products in the corresponding Hilbert spaces. In KCCA, the objective is to find $\mathbf{a}$ and $\mathbf{b}$ that will render maximal the correlation between $\mathbf{U}$ and $\mathbf{V}$. To ensure the uniqueness of the solution, some variance contraints have to be added.

As it is given in Akaho (2001), we have :

$$\mathbf{a} = \sum_{i=1}^{n} \alpha_i \boldsymbol{\phi}_x(\mathbf{x}_i) \qquad \mathbf{b} = \sum_{i=1}^{n} \beta_i \boldsymbol{\phi}_y(\mathbf{y}_i)$$

$$\mathbf{U} = \sum_{i=1}^{n} \alpha_i \langle \boldsymbol{\phi}_x(\mathbf{x}_i) | \boldsymbol{\phi}_x(\mathbf{X}) \rangle \qquad \mathbf{V} = \sum_{i=1}^{n} \beta_i \left\langle \boldsymbol{\phi}_y(\mathbf{y}_i) \middle| \boldsymbol{\phi}_y(\mathbf{Y}) \right\rangle,$$

where $\alpha_i$ and $\beta_i$ are scalars. Thanks to Mercer's theorem we can avoid explicit forms for $\boldsymbol{\phi}_x$ and $\boldsymbol{\phi}_y$. In fact, the inner product $\left\langle \boldsymbol{\phi}_x(\mathbf{x}_i) \middle| \boldsymbol{\phi}_x(\mathbf{x}_j) \right\rangle$, resp. $\left\langle \boldsymbol{\phi}_y(\mathbf{y}_i) \middle| \boldsymbol{\phi}_y(\mathbf{y}_j) \right\rangle$, can be replaced by a symmetric positive definite kernel $(\mathbf{K}_x)_{ij} = K_{h_x}(\mathbf{x}_i, \mathbf{x}_j)$, resp. $(\mathbf{K}_y)_{ij} = K_{h_y}(\mathbf{y}_i, \mathbf{y}_j)$. $\mathbf{K}_x$ and $\mathbf{K}_y$ are called the Gramian matrices.

If we want to take into account the sampling weights as explained in Annoye et al. (2024), we can solve the following generalized eigenvalue ($\lambda$) problem to calculate $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n)^T$ and $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_n)^T$:

$$\begin{pmatrix} 0 & \mathbf{K}_x \mathbf{W} \mathbf{K}_y \\ \mathbf{K}_y \mathbf{W} \mathbf{K}_x & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} = \lambda \cdot \begin{pmatrix} \frac{1}{2}\left((\mathbf{K}_x + \gamma I_n)\mathbf{W}\mathbf{K}_x + \mathbf{K}_x \mathbf{W}(\mathbf{K}_x + \gamma I_n)\right) & 0 \\ 0 & \frac{1}{2}\left((\mathbf{K}_y + \gamma I_n)\mathbf{W}\mathbf{K}_y + \mathbf{K}_y \mathbf{W}(\mathbf{K}_y + \gamma I_n)\right) \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix},$$

where $I_n$ is the $n \times n$ identity matrix.

We can choose the different hyperparameters (the bandwidth parameters of the kernels $K_{h_x}(\cdot,\cdot)$ and $K_{h_y}(\cdot,\cdot)$, and the regularization parameter $\gamma$) using a cross-validation procedure.

### 2.3. Statistical matching using KCCA

To impute the missing values, we fitted a KCCA model based on $\mathbf{X}^A$ and $\mathbf{Y}^A$ to obtain $\boldsymbol{\alpha}$. We can use it to impute missing values in $\mathbf{Y}^B$. To this end, we calculate $\mathbf{U}^A = \mathbf{K}_x^A\boldsymbol{\alpha}$ and $\mathbf{U}^B = \mathbf{K}_x^B\boldsymbol{\alpha}$, where $\left(\mathbf{K}_x^A\right)_{ij} = K_{h_x}\left(\mathbf{x}_i^A, \mathbf{x}_j^A\right)$ and $\left(\mathbf{K}_x^B\right)_{ij} = K_{h_x}\left(\mathbf{x}_i^B, \mathbf{x}_j^A\right)$. Hereafter, we obtain $\widehat{\mathbf{Y}}^B = \boldsymbol{\Omega}\mathbf{Y}^A$, a weighted mean of the variables in $\mathbf{Y}^A$, where

$$\begin{aligned}\boldsymbol{\Omega} &= (\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \cdots, \boldsymbol{\omega}_{n_B})^T \\ \boldsymbol{\omega}_i &= \left(\frac{w_1^A\varpi_{i1}}{\sum_{j=1}^{n_A} w_j^A\varpi_{ij}}, \frac{w_2^A\varpi_{i2}}{\sum_{j=1}^{n_A} w_j^A\varpi_{ij}}, \cdots, \frac{w_{n_A}^A\varpi_{in_A}}{\sum_{j=1}^{n_A} w_j^A\varpi_{ij}}\right)^T, \end{aligned} \quad (1)$$

where $\varpi_{ij} = K_h\left(u_i^B, u_j^A\right)$ for $i \in \{1, \cdots, n_B\}$ and $j \in \{1, \cdots, n_A\}$, and $u_i^B$ (resp. $u_j^A$) is an element of $\mathbf{U}^B$ (resp. $\mathbf{U}^A$) and the corresponding bandwidth $h$ is chosen by cross-validation.

## 3. Proposed statistical matching technique

In this section, we present two statistical matching methods. Subsection 3.1 introduces a novel approach that combines autoencoders with canonical correlation analysis (CCA). Subsection 3.2 describes a method based on a multi-output multilayer perceptron (MMLP). Finally, Subsection 3.3 explains how categorical variables are handled in both methods.

### 3.1. Autoencoders and Canonical Correlation Analysis (A-CCA)

As explained in Appendix Appendix A or in Kuss and Graepel (2003), KCCA can be seen as two Kernel Principal Component Analysis (KPCA) followed by a classical CCA. Thus, we can replace the KPCAs with autoencoders, which are strongly related to Principal Component Analysis (PCA) and have the advantage of dimensionality reduction.

#### 3.1.1. Autoencoders

The autoencoder was first introduced by Rumelhart et al. (1986) and is a self-supervised feed-forward neural network. The purpose of an autoencoder is to efficiently compress and reconstruct data. It consists of an encoder that reduces the dimension of the data into a latent space by harnessing any underlying data structures and patterns, and a decoder that constructs a representation from the latent space that resembles the original data as closely as possible.

Assume that we want to apply an autoencoder to an arbitrary data set $\mathbf{Q}$. Consider two sets $\mathbb{F}^J$ and $\mathbb{G}^{\ddot{J}}$, where $J$ denotes the number of columns in $\mathbf{Q}$ and $\ddot{J}$ represents the number of columns in the latent space, with $\ddot{J} \leq J$. Denote the encoder by $\mathfrak{e} \in \mathcal{E}$, where $\mathcal{E}$ is a set of functions mapping $\mathbb{F}^J \to \mathbb{G}^{\ddot{J}}$, and the decoder by $\mathfrak{d} \in \mathcal{D}$, where $\mathcal{D}$ is a set of functions mapping $\mathbb{G}^{\ddot{J}} \to \mathbb{F}^J$. Applying this process to data set $\mathbf{Q}$ yields

$$\begin{aligned}\mathfrak{e}: \quad & \mathbb{F}^J \to \mathbb{G}^{\ddot{J}}: \quad \mathbf{Q} \mapsto \boldsymbol{\varphi}_Q \\ \mathfrak{d}: \quad & \mathbb{G}^{\ddot{J}} \to \mathbb{F}^J: \quad \boldsymbol{\varphi}_Q \mapsto \hat{\mathbf{Q}},\end{aligned}$$

where $\boldsymbol{\varphi}_Q$ denotes the latent space representation of $\mathbf{Q}$ and $\hat{\mathbf{Q}}$ is the reconstructed counterpart. The encoder and decoder are trained to minimize the reconstruction error $\mathcal{L}(\mathbf{Q}, \hat{\mathbf{Q}})$, which we compute as the sum of the weighted mean-squared reconstruction error of each variable $\mathbf{q}_j$, $j = 1, \cdots, J$,

$$\mathcal{L}^{\text{MSE}}\left(\mathbf{q}_j, \hat{\mathbf{q}}_j\right) = \sum_{i=1}^{n} w_i(q_{ij} - \hat{q}_{ij})^2, \quad (2)$$

where $\hat{q}_{ij}$ is the reconstruction of $q_{ij}$.

#### 3.1.2. A-CCA

We fit two separate autoencoders to $\mathbf{X}^A$ and $\mathbf{Y}^A$ and obtain their latent space representations $\boldsymbol{\varphi}_X^A$ and $\boldsymbol{\varphi}_Y^A$. Canonical correlation analysis is applied on these representations in order to find canonical vectors $\mathbf{a}$ and $\mathbf{b}$ such that

$$\max_{\mathbf{a},\mathbf{b}} \mathbf{a}^T{\boldsymbol{\varphi}_X^A}^T\mathbf{W}\boldsymbol{\varphi}_Y^A\mathbf{b}$$

under the constraints

$$\mathbf{a}^T{\boldsymbol{\varphi}_X^A}^T\mathbf{W}\boldsymbol{\varphi}_X^A\mathbf{a} = 1 \text{ and } \mathbf{b}^T{\boldsymbol{\varphi}_Y^A}^T\mathbf{W}\boldsymbol{\varphi}_Y\mathbf{b} = 1.$$

The variables $\mathbf{U}^A = \boldsymbol{\varphi}_X^A\mathbf{a}$ and $\mathbf{V}^A = \boldsymbol{\varphi}_Y^A\mathbf{b}$, which maximize the correlation between $\boldsymbol{\varphi}_X^A\mathbf{a}$ and $\boldsymbol{\varphi}_Y^A\mathbf{b}$, are the canonical variables. As described in Annoye et al. (2024), this procedure can also be extended to multiple canonical variables.

Combining autoencoders with CCA, as we do here, is closely related to Deep Canonical Correlation Analysis (DCCA; Andrew et al., 2013). A key difference, however, is that in our A–CCA framework, the dimensionality of the latent space is treated as a hyperparameter that can differ between $\mathbf{X}$ and $\mathbf{Y}$, whereas in DCCA it is typically constrained to be the same for both.

*Statistical matching: imputation of the missing values.* We compute $\mathbf{U}^A = \boldsymbol{\varphi}_X^A\mathbf{a}$ and $\mathbf{U}^B = \boldsymbol{\varphi}_X^B\mathbf{a}$, where $\boldsymbol{\varphi}_X^A$ ($\boldsymbol{\varphi}_X^B$) is the latent space representation of $\mathbf{X}^A$ ($\mathbf{X}^B$).

Hereafter, we obtain $\widehat{\mathbf{Y}}^B = \boldsymbol{\Omega}\mathbf{Y}^A$, a weighted mean of the variables in $\mathbf{Y}^A$, where $\boldsymbol{\Omega}$ is defined as in Equation 1. We use a cross-validation to choose the corresponding bandwidth.

A summary of the process can be found in Algorithm 1.

### 3.2. Multi-output Multilayer Perceptron (MMLP)

A Multi-output Multilayer Perceptron (MMLP) is also implemented to compare the performance of the two procedures described above. MMLP is one of the simpler neural network and we use the donor data set to construct the network and to predict the non-common variable of the receiver data set using the common variables. We use elastic-net regularization.

### 3.3. Dealing with categorical variables.

To deal with incompatibility problems in the categorical variables, we use the method proposed in Annoye et al. (2024). As pointed out by D'Orazio et al. (2006a) and Annoye et al. (2024), we want to avoid incompatibilities when predicting categorical variables. For example, if we predict a city of residence we

**Algorithm 1** A-CCA

**Input:** $\mathbf{X}^A$ and $\mathbf{Y}^A$, the centered common and non-common variables in data set $A$ and $\mathbf{X}^B$, the centered common variables in data set $B$. $\mathbf{w}^A$ and $\mathbf{w}^B$, the weights in both data sets.

**Output:** A data set $B$ with the centered $\mathbf{X}^B$ and $\mathbf{Y}^B$.

1: Fit an autoencoder to $\mathbf{X}^A$ to obtain $\boldsymbol{\varphi}_X^A$.
2: Fit an autoencoder to $\mathbf{Y}^A$ to obtain $\boldsymbol{\varphi}_Y^A$.
3: Calculate $\mathbf{a}$ and $\mathbf{b}$ the canonical correlation of $\boldsymbol{\varphi}_X^A$ and $\boldsymbol{\varphi}_Y^A$ using $\mathbf{w}^A$.
4: Use the fitted autoencoder on $\mathbf{X}^A$ to obtain $\boldsymbol{\varphi}_X^B$ with $\mathbf{X}^B$.
5: Calculate $\mathbf{U}^A = \boldsymbol{\varphi}_X^A\mathbf{a}$ and $\mathbf{U}^B = \boldsymbol{\varphi}_X^B\mathbf{a}$ the canonical correlation variables based on the latent space.
6: **for** $i = 1, \cdots, n^B$ **do**
7:  **for** $j = 1, \cdots, n^A$ **do**
8:  Calculate $\varpi_{ij} = K_h\left(u_i^B, u_j^A\right)$ where $K_h$ is a Gaussian kernel.
9:  **end for**
10: **end for**
11: Calculate $\widehat{\mathbf{Y}}^B = \boldsymbol{\Omega}\mathbf{Y}^A$, where:

$$\boldsymbol{\Omega} = (\omega_1, \omega_2, \cdots, \omega_{n_B})^T$$

$$\omega_i = \left(\frac{w_1^A \varpi_{i1}}{\sum_{j=1}^{n_A} w_j^A \varpi_{ij}}, \frac{w_2^A \varpi_{i2}}{\sum_{j=1}^{n_A} w_j^A \varpi_{ij}}, \cdots, \frac{w_{n_A}^A \varpi_{in_A}}{\sum_{j=1}^{n_A} w_j^A \varpi_{ij}}\right)^T.$$

endalgorithmic

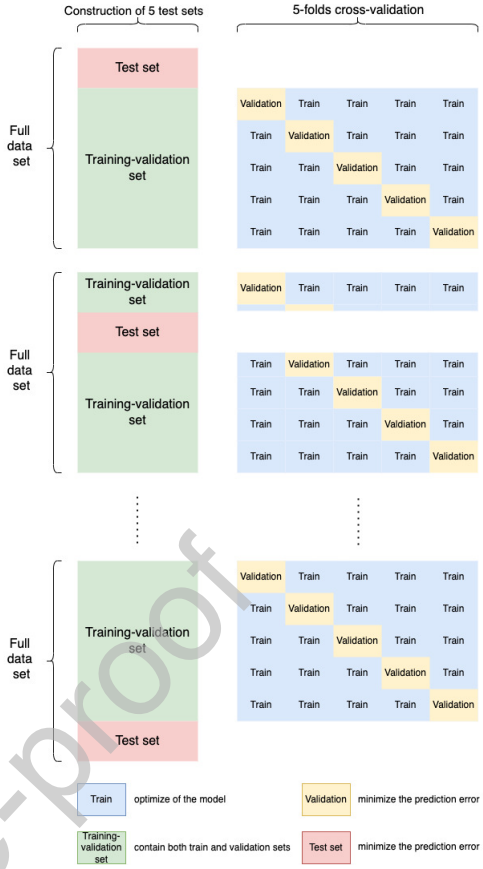


Figure 2: Representation of the data partitioning.

want it to be compatible with the country of residence. We want our technique of statistical matching to be able to do this for as many variables as possible in an automatic way. This is the case for the two-step algorithm proposed with A-CCA (algorithm 2) but this two-step procedure is also upheld for KCCA and MMLP.

## 4. Experiment results

In this section, we compare the performance of the three methods discussed previously (KCCA, A-CCA and MMLP) not only with each other but also with more traditional methods of statistical matching like distance hot-deck (HD) and multivariate linear/multinomial logistic regressions (REG). We will use the same data set as in Annoye et al. (2024), i.e. 2017 Belgian Statistics on Income and Living Conditions (SILC) limited to the variables provided in the Tables 1 for the common variables and 2 for the non common once.

We will divide this data set in five folds. For each of the five folds, we will remove the non-common variables and use the complete four other folds to predict them as if they are missing. As mentioned before, we use 5-fold cross-validation procedure on this four folds to tune the hyperparameters of the different algorithms. A representation of the partitioning of the data set can be found in Figure 2.

As explained in Section 3.3 to deal with categorical constraint, we implement a two-step procedure to all the methodologies except the distance hot-deck one. We use as minimization criteria for the first step the sum of the weighted Misclassification Rates (wMCR) of the categorical non-common variables, which have been transformed into dummies :

$$\text{wMCR}\left(\mathbf{y}^d, \hat{\mathbf{y}}^d\right) = \sum_{i=1}^{n} w_i I\left(y_i^d \neq \hat{y}_i^d\right), \tag{3}$$

where $n$ denotes the total number of observations, $w_i$ is the individual sampling weight bounded between zero and one, $I(.)$ is a characteristic function and $\hat{y}_i^d$ denotes the imputed value of a true dummy variable $y_i^d$. For the second step, we use the Root weighted standardized Mean Squared Errors (RwsMSE) of the continuous non-common variables:

$$\text{RwsMSE}\left(\mathbf{y}^c, \hat{\mathbf{y}}^c\right) = \sqrt{\sum_{i=1}^{n} w_i \frac{\left(y_i^c - \hat{y}_i^c\right)^2}{\hat{\sigma}^2}}, \tag{4}$$

where $n$ and $w_i$ are the same as in Equation 3 while $\hat{y}_i^c$ denotes the imputed value of a true continuous variable $y_i^c$ with $i = 1, \cdots, n$, and

$$\hat{\sigma}^2 = \sum_{i=1}^{n} w_i \left[y_i^c - \left(\sum_{i=1}^{n} w_i y_i^c\right)\right]^2.$$

Concerning the choice of the different hyperparameters used by the machine learning algorithms (KCCA, A-CCA and MMLP), we used a 5-fold cross-validation, as described hereafter.

| Variable | Description | Type |
|---|---|---|
| RB090 | Gender | Categorical |
| RX010 | Age | Continuous |
| DB040 | Region | Categorical |
| PE040 | Educational level attained (ISCED level) | Categorical |
| PB190 | Marital status | Categorical |
| PB220A | Country of citizenship | Categorical |
| PB210 | Country of birth | Categorical |
| PL031 | Activity status | Categorical |
| PL031 | Number of hours worked per week | Categorical |
| PL140 | Type of work contract | Categorical |
| PL111 | Economic activities in employment | Categorical |
| PL040 | Status in employment | Categorical |
| PL051 | Occupation status (ISCO-08) | Categorical |
| HX060 | Type of the household | Categorical |
| HX040 | Size of the household | Continuous |
| HS110 | Car ownership | Categorical |

Table 1: Common variables

| Variable | Description | Type |
|---|---|---|
| PY010N | Employee cash or near cash income (Net income) | Continuous |
| PY030G | Employer's social insurance contribution (Income) | Continuous |
| PY100N | Old-age benefits (Net income) | Continuous |
| RX050 | Low work intensity status | Categorical |
| PB200 | Consensual union | Categorical |
| PH030 | Limitation in activities because of health problems | Categorical |

Table 2: Non-common variables

***KCCA***. In this procedure, we consider canonical variables of two dimensions. For the other hyperparameters, we use a grid search. In Table 3, we list the different hyperparameters for KCCA. We also provide the intervals used to find their optimal values but some of these intervals are already the result of several trials.

***A-CCA***. In both Steps 1 and 2, the autoencoders for the common and the non-common variables contain two hidden layers with rectified linear unit (ReLU) activation functions in both the encoder and decoder. The general structure of these autoencoders is displayed in Figure 3, where $\mathbf{m}$ is used to denote the weight vectors of the neurons and $\kappa_1^E, \kappa_2^E, \kappa_1^D, \kappa_2^D$ denote the number of neurons in each of the hidden layers. Elastic net regularization with parameters $L_1$ (for the Lasso part) and $L_2$ (for the ridge part) is used to avoid over-fitting. The advantage of using Elastic Net regularization in our context is that it allows us to combine Lasso and Ridge regularization, thereby combining their positive effects: neural weights selection (Lasso) and shrinkage. Moreover, because we allow the values of $L_1$ and $L_2$ penalties to be close to zero, we can test cases that nearly consist of only Lasso or Ridge regularization in our algorithm, and we can also balance both with our cross-validation.

To observe the influence of the different hyperparameters, we initially conducted lots of trials using large intervals with wide

| Step | Hyperparameter | Interval |
|---|---|---|
| 1 | Bandwidth $h$ of the kernel (prediction) | $5 \cdot 10^{-3} - 3.5 \cdot 10^{-2}$ |
| | Bandwidth $0.5 \cdot h_x^{-2}$ of the kernel in $\mathbf{K}_x$ | $4 \cdot 10^{-4} - 1.2 \cdot 10^{-3}$ |
| | Bandwidth $0.5 \cdot h_y^{-2}$ of the kernel in $\mathbf{K}_y$ | $4 \cdot 10^{-4} - 1.2 \cdot 10^{-3}$ |
| | Regularisation parameter $\gamma$ | $1 \cdot 10^{-5} - 3 \cdot 10^{-5}$ |
| 2 | Bandwidth $h$ of the kernel (prediction) | $1 \cdot 10^{-2} - 7 \cdot 10^{-2}$ |
| | Bandwidth $0.5 \cdot h_x^{-2}$ of the kernel in $\mathbf{K}_x$ | $1.4 \cdot 10^{-3} - 2.2 \cdot 10^{-3}$ |
| | Bandwidth $0.5 \cdot h_y^{-2}$ of the kernel in $\mathbf{K}_y$ | $1 \cdot 10^{-4} - 1.6 \cdot 10^{-3}$ |
| | Regularisation parameter $\gamma$ | $1 \cdot 10^{-5} - 3 \cdot 10^{-5}$ |

Table 3: Hyperparameters for KCCA

steps for the various hyperparameters values. Afterward, we fixed certain hyperparameters for which the errors of Equation (3) (for phase 1) and of Equation (4) (for phase 2) were relatively stable; for the others, we narrowed their ranges to better capture the areas where the minimum of these errors was supposed to be. Then, depending on the hyperparameter and guided by time constraints, we performed either a random or a grid search within each of these ranges. The possible dimensions of the latent spaces are linked to the number of variables, including the transformation of the categorical variables into dummies. The learning rate has be shown to be around the default value ($10^{-3}$) used in the Keras package. In Table 4, we list all the hyperparameters of the A-CCA that are tuned by 5-fold cross-validation. Although beyond the scope of this work, other methods besides this trial-and-error approach could be considered for selecting hyperparameters value. In the autoencoder part, the batch size is equal to 256, the maximum number of epochs is 500 and the optimization algorithm stops if its reconstruction error has not improved over the last 10 epochs. Finally, we consider two-dimensional canonical variables, as for KCCA.

A key difference with DCCA is that our approach minimizes the prediction error, whereas DCCA typically maximizes canonical correlations. In prediction tasks, certain dependencies between the predicted non-common variables are not explicitly modeled; these are partially captured by the latent space of our autoencoders and the kernel applied at the final stage of our algorithm. The presence of the kernel motivates the inclusion of a CCA component, which allows us to avoid using a kernel with too many dimensions.

***MMLP***. In both Steps 1 and 2, the Multi-output Multilayer Perceptron contains three hidden layers with rectified linear unit (ReLU) activation functions. Elastic net regularization with parameters $L_1$ and $L_2$ is used to avoid overfitting. The batch size is equal to 256 and the maximum number of epochs is 500. The optimization algorithm stops if the loss functions has not improved over the last 5 epochs. In Table 5, we list the hyperparameters of the MMLP that are tuned by 5-fold cross-validation and the intervals used to find their optimal values.

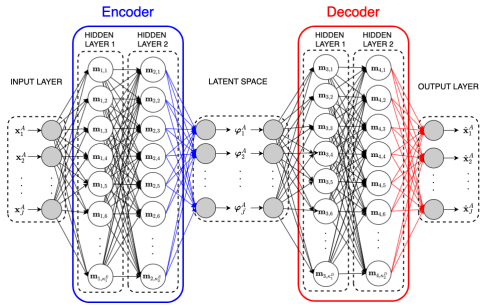


Figure 3: Architecture of an autoencoder with 2 hidden layers applied to $\mathbf{X}^A$

| Search | Step | Hyperparameter | Interval |
|---|---|---|---|
| Grid | 1 & 2 | Bandwidth $h$ of the kernel | 0.1 - 0.3 |
| Random | 1 | Dimensions of the latent space ($\ddot{J}_{\mathbf{X}}$) | 1 - 60 |
| | | Dimensions of the latent space ($\ddot{J}_{\mathbf{Y}}$) | 1 - 12 |
| | 2 | Dimensions of the latent space ($\ddot{J}_{\mathbf{X}}$) | 1 - 72 |
| | | Dimensions of the latent space ($\ddot{J}_{\mathbf{Y}}$) | 1 - 3 |
| | 1 & 2 | Neurons in the hidden layers ($\mathbf{X}$) | 50 - 600 |
| | | Neurons in the hidden layers ($\mathbf{Y}$) | 50 - 600 |
| | | Learning rate | $10^{-4.5}$ - $10^{-2}$ |
| | | $L_1$ penalty | $10^{-6}$ - 0.5 |
| | | $L_2$ penalty | $10^{-6}$ - 0.5 |

Table 4: Hyperparameters for A-CCA

### 4.1. Implementation

We employ R software with Keras and Tensorflow packages to conduct all our experiments. Some portions of the code are developed directly in C++ to improve computational speed. Other programming languages like Python or Matlab could have also been utilized as they offer packages similar to those utilized with R.

### 4.2. Results

We will compare our different statistical matching methods using weighted Misclassification Rates (wMCR), for the categorical variables and the weighted standardized Mean Absolute Error (wsMAE) and Root weighted standardized Mean Squared Error (RwsMSE), for the continuous variables. The definition of wsMAE is the following;

$$\text{wsMAE}\left(\mathbf{y}^c, \hat{\mathbf{y}}^c\right) = \sum_{i=1}^{n} w_i \left| \frac{y_i^c - \hat{y}_i^c}{\hat{\sigma}} \right|,$$

where $n$ is the total number of observations, $w_i$ is the sampling weight (bounded between zero and one), $\hat{y}_i^c$ is the imputed value of a true continuous variable $y_i^c$ with $i = 1, \cdots, n$, and

$$\hat{\sigma}^2 = \sum_{i=1}^{n} w_i \left[ y_i^c - \left( \sum_{i=1}^{n} w_i y_i^c \right) \right]^2.$$

| Step | Hyperparameter | Interval |
|---|---|---|
| 1 & 2 | Neurons in the hidden layers | 50 - 600 |
| | Learning rate | $10^{-4.5}$ - $10^{-2}$ |
| | $L_1$ penalty | $10^{-6}$ - 0.5 |
| | $L_2$ penalty | $10^{-6}$ - 0.5 |

Table 5: Hyperparameters for MMLP

To have a single measure of the quality of the matching, we will calculate the mean of the RwsMSE over all variables (with the categorical variables transformed into dummies).

We also use two multivariate coefficients of determination. A first one based on the generalized variance (i.e. determinant of covariance matrix) and defined by Cohen et al. (2002):

$$\text{mult-}R_1^2 = 1 - \frac{\det\left(R_{Y\hat{Y}}\right)}{\det\left(R_Y\right)\det\left(R_{\hat{Y}}\right)},$$

where $R_Y$ (resp. $R_{\hat{Y}}$) is the correlation matrix of the variables in $Y$ (resp. $\hat{Y}$) and $R_{Y\hat{Y}}$ is the full correlation matrix of the variables in $Y$ and $\hat{Y}$. The second one is the one defined by Jones (2019) and based on the geometric interpretation of a $R^2$ that can be seen as total squared-Euclidean distance of $y_i$ with the mean $\bar{y}$:

$$\text{mult-}R_2^2 = 1 - \frac{SSE}{SST},$$

where $\text{SST} = \sum_{i=1}^{n} w_i \left[d\left(y_i, \bar{y}\right)\right]^2$, $\text{SSE} = \sum_{i=1}^{n} w_i \left(d\left(y_i, \hat{y}_i\right)\right)^2$ and $d(p,q) = \sqrt{\sum_{j=1}^{J}\left(p_j - q_j\right)^2}$.

Finally, it is important for us to have a quantity that will assess the quality of the bivariate distributions. We choose to use the Cramér–von Mises criterion that can be defined by :

$$\text{CVM} = \int_{\mathbb{R}}\int_{\mathbb{R}} \left[\hat{F}_{n_A}(x,y) - \hat{G}_{n_B}(x,y)\right]^2 \mathrm{d}H_{n_A+n_B}(x,y),$$

where $\hat{F}_{n_A}(x,y)$, $\hat{G}_{n_B}(x,y)$ and $H_{n_A+n_B}(x,y)$ are the bivariate empirical distributions of two variables $X$ and $Y$ in data set $A$, $B$ and $A+B$, respectively.

In practice, we approximated the double integrals by a sum and we have an empirical Cramér–von Mises :

$$\widetilde{\text{CVM}} = \frac{n_A + n_B}{2} \sum_{j\in\{A,B\}} \sum_{i=1}^{n_j} w_i^j \left[\hat{F}_{n_A}\left(x_i^j, y_i^j\right) - \hat{G}_{n_B}\left(x_i^j, y_i^j\right)\right]^2, \tag{5}$$

where $w_i^A$ (resp. $w_i^B$) denotes the sampling weights bounded between zero and one of data set $A$ (resp. $B$).

We use all the aforementioned criteria to evaluate the quality of the results obtained with the different statistical matching techniques. We calculate them in each of the five folds and take their averages over the five folds. The results can be found in Table 20.

Due to its random nature HD underperformed all the others method in terms of one-variable measures (wsMAE, RwsMSE, wMC) and in terms of overall error measures (Total RwsMSE). All the other methods are tightly bunched and with value of the

error of the continuous variables twice as small as the one of HD.

The best performance in terms of overall error measure is achieved by REG, followed by the three others methods.

This is also the case if we look at the two multivariate extensions of the $R^2$. However, the conclusions are different if we focus on the empirical Cramér–von Mises ($\widetilde{\text{CVM}}$) criteria which, as a reminder, measures the distance between the bivariate distributions. In Table 20, when we compare the average of the $\widetilde{\text{CVM}}$ criteria for all pairs of non-common variables[1], we can see that the best method is HD followed by A-CCA. In contrast, REG is by far the worst method with this measure. The low values of HD are not surprising. Indeed, HD simply reproduces the dependencies of the non-common variables of the donor data set.

However, with the average of the $\widetilde{\text{CVM}}$ on all couples of one common and one non-common variable, the best method among the three machine learning techniques is A-CCA, followed by KCCA. All three perform better than REG and HD and A-CCA has the best performance regarding the average of the $\widetilde{\text{CVM}}$ criteria between all combinations containing both a non-common and a common variable.

On figures 4–6 we plot the density of the original distributions of the continuous variables and the ones imputed by the different methodologies. We only furnish the ones for the first fold, the graphs being equivalent for the others. As seen with the numerical results, the distributions obtained from A-CCA seem to be very close to the original ones, as the ones obtained from KCCA; in particular, both seem to outperform REG or MMLP.

In conclusion, the results indicate that A-CCA and MMLP combine the advantages of both REG and HD as we have already seen for KCCA in Annoye et al. (2024). They outperform REG in preserving the joint distribution, as measured by the empirical Cramér–von Mises criteria. At the same time, they are also better than HD in terms of prediction error. Furthermore, we have A-CCA that is lightly better than KCCA in preserving the joint distribution.

Additional studies on the standard deviation of the results and on the impact of the number of common variables on them can be found in Sections 4.3 and 4.4.

### *4.3. Standard deviation of the results*

Here, we provide the estimated standard deviation of the results obtained from the 5 folds (test sets, see Figure 2) and for which the averages are available in Table 20. We can see that the worst results generally have the largest variance. If we construct normal-based 95% confidence intervals for the prediction measures (mult-$R^2$s, RwsMSEs, wMCRs, wsMAEs), all the techniques have overlapping intervals except HD. On the other hand, with $\widetilde{\text{CVM}}$s that measure how the multivariate distributions are preserved, the REG and MMLP methods have intervals that are completely different compared to the other techniques. This confirms the results described in Section 4.2. We also provide the minimum and maximum of each criterion over the 5 folds in Tables 22 and 23.

[1]For a given fold, we compute $\widetilde{\text{CVM}}$, where data set A in eq. 5 (resp. B) contain the true data (resp. the one obtained through statistical matching).

### *4.4. Test with different number of common variables*

In this section, we apply our different methods to the SILC data set with 5, 10 or 20 common variables. The variables used as common (resp. non-common) are those listed in Table 6 (resp. Table 2). HD is not displayed because this method can handle only a small number of common categorical variables to create the donation classes.

As expected, the results improve as the number of common variables increases, regardless of the method used. However, our proposed methods using CCA (see Tables 7 and 8) perform well even with five variables, achieving good results in terms of CVMs. In contrast, Average $\widetilde{CVM}$s of the regression-based method are nearly 20 times larger than those of our methods, see Table 10 (and 5 times larger for MMLP in Table 9).

The evolution of mult-$R^2$s and RwsMSE is quite similar across all methods.

| Variable | Description | Type |
|---|---|---|
| RB090 | Gender | Categorical |
| RX010 | Age | Continuous |
| DB040 | Region | Categorical |
| PE040 | Educational level attained (ISCED level) | Categorical |
| PB190 | Marital status | Categorical |
| PB220A | Country of citizenship | Categorical |
| PB210 | Country of birth | Categorical |
| PL031 | Activity status | Categorical |
| PL031 | Number of hours worked per week | Categorical |
| PL140 | Type of work contract | Categorical |
| PL111 | Economic activities in employment | Categorical |
| PL040 | Status in employment | Categorical |
| PL051 | Occupation status (ISCO-08) | Categorical |
| HX060 | Type of the household | Categorical |
| HX040 | Size of the household | Continuous |
| HS110 | Car ownership | Categorical |
| DB100 | Degree of urbanization | Categorical |
| HH030 | Number of rooms available to the houshold | Continuous |
| HX070 | Tenure status | Categorical |
| HH040 | Damage due to water intrusion | Categorical |

Table 6: Common variables

### *4.5. Complimentary analysis*

Generally, the user of the data will perform some analysis such as regression or clustering. In this section, we will perform some of these analyses on the different matched data sets coming from the different techniques we used in the previous subsection.

| Variable | Measure | 5 | 10 | 20 |
|---|---|---|---|---|
| PY010N | wsMAE | 0.53 | 0.36 | 0.23 |
| PY010N | RwsMSE | 0.98 | 0.8 | 0.59 |
| PY030G | wsMAE | 0.54 | 0.38 | 0.27 |
| PY030G | RwsMSE | 0.98 | 0.82 | 0.63 |
| PY100N | wsMAE | 0.25 | 0.2 | 0.2 |
| PY100N | RwsMSE | 0.68 | 0.58 | 0.58 |
| RX050 | wMCR | 0.16 | 0.14 | 0.13 |
| PB200 | wMCR | 0.24 | 0.23 | 0.18 |
| PH030 | wMCR | 0.36 | 0.32 | 0.33 |
| Total (cont.) | RwsMSE | 0.88 | 0.73 | 0.6 |
| Total (cat.) | RwsMSE | 1 | 0.96 | 0.92 |
| Total | RwsMSE | 0.97 | 0.91 | 0.85 |
| Total | mult-$R_1^2$ (Cohen, 2013) | 0.97 | 0.96 | 0.98 |
| Total | mult-$R_2^2$ (Jones, 2019) | 0.12 | 0.39 | 0.63 |
| Total | Aver. $\widetilde{\text{CVM}}$ Bivariate Non-Common | 1.77 | 1.39 | 1 |
| Total | Aver. $\widetilde{\text{CVM}}$ Bivariate Mixed | 0.85 | 0.71 | 0.52 |

Table 7: Average of results over the 5 folds for KCCA, wsMAE, weighted standardized Mean Absolute Error; RwsMSE, Root weighted standardized Mean Squared Error; wMCR, weighted Misclassification Rate; Total, all variables; Total (cont.), all continuous variables; Total (cat.), all categorical variables; Aver. $\widetilde{\text{CVM}}$ Bivariate Non-Common, average of the $\widetilde{\text{CVM}}$ criteria for all couples of non-common variables; Aver. $\widetilde{\text{CVM}}$ Bivariate Mixed, average of the $\widetilde{\text{CVM}}$ criteria for all couples of one common and one non-common variable.

| Variable | Measure | 5 | 10 | 20 |
|---|---|---|---|---|
| PY010N | wsMAE | 0.54 | 0.38 | 0.27 |
| PY010N | RwsMSE | 0.98 | 0.83 | 0.65 |
| PY030G | wsMAE | 0.55 | 0.4 | 0.29 |
| PY030G | RwsMSE | 0.99 | 0.84 | 0.68 |
| PY100N | wsMAE | 0.25 | 0.2 | 0.22 |
| PY100N | RwsMSE | 0.69 | 0.58 | 0.62 |
| RX050 | wMCR | 0.16 | 0.14 | 0.15 |
| PB200 | wMCR | 0.24 | 0.2 | 0.17 |
| PH030 | wMCR | 0.36 | 0.33 | 0.33 |
| Total (cont.) | RwsMSE | 0.89 | 0.75 | 0.65 |
| Total (cat.) | RwsMSE | 1 | 0.96 | 0.93 |
| Total | RwsMSE | 0.98 | 0.91 | 0.87 |
| Total | mult-$R_1^2$ (Cohen, 2013) | 0.97 | 0.97 | 0.98 |
| Total | mult-$R_2^2$ (Jones, 2019) | 0.11 | 0.35 | 0.56 |
| Total | Aver. $\widetilde{\text{CVM}}$ Bivariate Non-Common | 1.94 | 1.23 | 1.4 |
| Total | Aver. $\widetilde{\text{CVM}}$ Bivariate Mixed | 0.94 | 0.62 | 0.72 |

Table 8: Average of results over the 5 folds for A-CCA, wsMAE, weighted standardized Mean Absolute Error; RwsMSE, Root weighted standardized Mean Squared Error; wMCR, weighted Misclassification Rate; Total, all variables; Total (cont.), all continuous variables; Total (cat.), all categorical variables; Aver. $\widetilde{\text{CVM}}$ Bivariate Non-Common, average of the $\widetilde{\text{CVM}}$ criteria for all couples of non-common variables; Aver. $\widetilde{\text{CVM}}$ Bivariate Mixed, average of the $\widetilde{\text{CVM}}$ criteria for all couples of one common and one non-common variable.

#### *4.5.1. Regression*

In this subsection, we regress all of our three continuous non-common variables against all of the common variables. We will perform this with the three methods developed in this paper, two comparison methods, regression and hot-deck. In 11, we present the RwsMSE for each of our regressions. KCCA and A-CCA have similar results and are the two best methods, the closest to the one obtained on the true data set. These results are much better than those obtained with Hot-deck. As already pointed out by Annoye et al. (2024), Regression performs similarly or slightly worse than KCCA, but this is due to the fact that we are using RwsMSE, a prediction measure that does not quantify the dependence between the non-common variables.

#### *4.5.2. Regression with k-nearest neighbors*

In Table 12 we compared our different methods with a $k$-nearest neighbors algorithm to predict the three categorical non-common variables with all the continuous variables, using the misclassification rate (MCR).

On the data constructed with the two new methods proposed in this article, we obtained similar results to those obtained for KCCA in Annoye et al. (2024) : the regression with $k$-nearest neighbors performs with our methods as good as on the true data and even outperforms that one for the variable PB200.

#### *4.5.3. Clustering with k-means*

We performed here a clustering to group data in our different data sets (true and predicted). To compare the results with those of KCCA (Annoye et al., 2024), we use the same number of centroid, $k = 13$ that was calculated by maximizing the Calinski–Harabasz. In Table 24, we provide the inverse of the Calinski–Harabasz.

$$\text{CH}^{-1}(k) = \frac{\text{W}(k)/(n-k)}{\text{B}(k)/(k-1)},$$

$$\text{B}(k) = \sum_{i=1}^{k} n_i \|c_i - c\|^2,$$

$$\text{W}(k) = \sum_{i=1}^{k} \sum_{x \in C_i} w_x \|x - c_i\|^2,$$

where $n$ is the number of individuals, $n_i$ is the sum of the weights of individuals in the cluster $C_i$ (such as $\sum_{i=1}^{k} n_i = n$), $c_i$ is the centroid of the cluster $C_i$ and $c$ is the overall centroid. This quantity is smaller on the imputed data sets constructed with KCCA, A-CCA and REG than on the true one.

To determine the quality of the classifications constructed on our different data set, we use Purity and Entropy :

$$\text{Purity} = \frac{1}{n} \sum_{i=1}^{k} \max_{1 \le j \le l} n_i^j,$$

$$\text{Entropy} = \frac{-1}{n \log_2(l)} \sum_{i=1}^{k} \sum_{j=1}^{l} n_i^j \log_2\left(\frac{n_i^j}{n_i}\right),$$

where $n_i^j$ is the sum of the weights of samples in cluster $i$ that belong to a priori known class $j \in \{1, \cdots, l\}$ constructed with the help of non-continuous variables. Entropy must be minimized while purity must be maximized. This latter is also bounded between 0 and 1.

In Table 24, we can observe that the highest (lowest) level of purity (entropy) is obtained, depending on the variable, when applied to KCCA, A-CCA, MMLP or REG data set. All these

| Variable | Measure | 5 | 10 | 20 |
|---|---|---|---|---|
| PY010N | wsMAE | 0.48 | 0.33 | 0.27 |
| PY010N | RwsMSE | 0.91 | 0.77 | 0.77 |
| PY030G | wsMAE | 0.49 | 0.36 | 0.29 |
| PY030G | RwsMSE | 0.93 | 0.79 | 0.82 |
| PY100N | wsMAE | 0.27 | 0.19 | 0.21 |
| PY100N | RwsMSE | 0.7 | 0.56 | 0.59 |
| RX050 | wMCR | 0.16 | 0.12 | 0.12 |
| PB200 | wMCR | 0.21 | 0.18 | 0.15 |
| PH030 | wMCR | 0.29 | 0.29 | 0.29 |
| Total (cont.) | RwsMSE | 0.85 | 0.71 | 0.72 |
| Total (cat.) | RwsMSE | 0.85 | 0.81 | 0.8 |
| Total | RwsMSE | 0.85 | 0.79 | 0.79 |
| Total | mult-$R_1^2$ (Cohen, 2013) | 0.94 | 0.98 | 0.99 |
| Total | mult-$R_2^2$ (Jones, 2019) | 0.21 | 0.44 | 0.41 |
| Total | Aver. $\widetilde{\widehat{\text{CVM}}}$ Bivariate Non-Common | 11.06 | 4.65 | 2.26 |
| Total | Aver. $\widetilde{\widehat{\text{CVM}}}$ Bivariate Mixed | 5.44 | 2.44 | 1.22 |

Table 9: Average of results over the 5 folds for MMLP; wsMAE, weighted standardized Mean Absolute Error; RwsMSE, Root weighted standardized Mean Squared Error; wMCR, weighted Misclassification Rate; Total, all variables; Total (cont.), all continuous variables; Total (cat.), all categorical variables; Aver. $\widetilde{\widehat{\text{CVM}}}$ Bivariate Non-Common, average of the $\widetilde{\widehat{\text{CVM}}}$ criteria for all couples of non-common variables; Aver. $\widetilde{\widehat{\text{CVM}}}$ Bivariate Mixed, average of the $\widetilde{\widehat{\text{CVM}}}$ criteria for all couples of one common and one non-common variable.

| Variable | Measure | 5 | 10 | 20 |
|---|---|---|---|---|
| PY010N | wsMAE | 0.54 | 0.3 | 0.21 |
| PY010N | RwsMSE | 0.96 | 0.72 | 0.58 |
| PY030G | wsMAE | 0.55 | 0.33 | 0.23 |
| PY030G | RwsMSE | 0.98 | 0.74 | 0.6 |
| PY100N | wsMAE | 0.23 | 0.19 | 0.19 |
| PY100N | RwsMSE | 0.64 | 0.56 | 0.55 |
| RX050 | wMCR | 0.11 | 0.08 | 0.08 |
| PB200 | wMCR | 0.17 | 0.17 | 0.12 |
| PH030 | wMCR | 0.25 | 0.23 | 0.23 |
| Total (cont.) | RwsMSE | 0.86 | 0.67 | 0.58 |
| Total (cat.) | RwsMSE | 0.74 | 0.72 | 0.71 |
| Total | RwsMSE | 0.77 | 0.71 | 0.68 |
| Total | mult-$R_1^2$ (Cohen, 2013) | 0.98 | 0.99 | 1 |
| Total | mult-$R_2^2$ (Jones, 2019) | 0.15 | 0.49 | 0.64 |
| Total | Aver. $\widetilde{\widehat{\text{CVM}}}$ Bivariate Non-Common | 38.61 | 25.52 | 16.2 |
| Total | Aver. $\widetilde{\widehat{\text{CVM}}}$ Bivariate Mixed | 18.39 | 14.13 | 9.18 |

Table 10: Average of results over the 5 folds for REG; wsMAE, weighted standardized Mean Absolute Error; RwsMSE, Root weighted standardized Mean Squared Error; wMCR, weighted Misclassification Rate; Total, all variables; Total (cont.), all continuous variables; Total (cat.), all categorical variables; Aver. $\widetilde{\widehat{\text{CVM}}}$ Bivariate Non-Common, average of the $\widetilde{\widehat{\text{CVM}}}$ criteria for all couples of non-common variables; Aver. $\widetilde{\widehat{\text{CVM}}}$ Bivariate Mixed, average of the $\widetilde{\widehat{\text{CVM}}}$ criteria for all couples of one common and one non-common variable.

methods seem comparable, in terms of clustering, and have results that can be slightly better than those obtained from the true data set; in our example, they always outperform the clustering technique applied to the HD data set.

| Dependant variable | True | KCCA | A-CCA | MMLP | REG | HD |
|---|---|---|---|---|---|---|
| PY010N | 0.47 | 0.49 | 0.50 | 0.56 | 0.50 | 0.94 |
| PY0100N | 0.52 | 0.52 | 0.53 | 0.55 | 0.55 | 0.66 |
| PY030G | 0.50 | 0.53 | 0.53 | 0.58 | 0.53 | 0.94 |

Table 11: Results of the different regressions : RwsMSE, Root weighted standardized Mean Squared Error.

| | True | KCCA | A-CCA | MMLP | REG | HD |
|---|---|---|---|---|---|---|
| PH030 | 0.28 | 0.29 | 0.30 | 0.26 | 0.26 | 0.30 |
| RX050 | 0.10 | 0.12 | 0.13 | 0.10 | 0.15 | 0.53 |
| PB200 | 0.37 | 0.15 | 0.15 | 0.14 | 0.38 | 0.53 |

Table 12: Results for the different k-NN regressions : wMCR, weighted Misclassification Rate.

### *4.6. Larger data set*

We compare the results of performing the same exercise on a larger version of SILC that includes 185 non-common variables while maintaining the same number of common variables. On the big data set, A-CCA has the best results in terms of RwsMSE; KCCA seems to be the best for $R^2$ and third for all other measures; MMLP provides the best CVM among our proposed methods. Not surprisingly, the results are less good in this second exercise. This is due to the fact that we are using the same number of common variables to predict a much larger number of non-common variables.

As already pointed out in Annoye et al. (2024), the success of HD in terms of CVM has to be mitigated by the fact that the results on the other measures are very poor and the imputation is taken from a unique individual, so it increases some disclosure risk. All other proposed methods outperform REG imputation.

Moreover, if we have larger data sets, KCCA can be quite computationally expensive, with kernel matrices that have an $O\left(n^2\right)$ complexity in memory and computation. A-CCA, which uses autoencoders that are neural networks, can thus be quite effective on larger data sets due to stochastic gradient descent (SGD) and mini-batch learning.

### *4.7. Data sets that benefit from A-CCA*

In this subsection, we conduct some simulations to observe the differences between our two best techniques, i.e., A-CCA and KCCA.

A-CCA is particularly interesting because, when the relationships between the two groups of variables—common and non-common—are mostly linear but include some terms that can be captured by non-linear activation functions, we can expect A-CCA to outperform KCCA.

This advantage arises from the greater flexibility of A-CCA, which allows to more effectively capture non-linear relationships through its activation functions, while still accurately reproducing linear relationships between the projected spaces due to the CCA component.

On the other hand, KCCA, which can be seen as a combination of KPCA followed by CCA, is particularly useful for purely non-linear cases but does not share these advantages. This is because KPCA, which leverages the kernel trick, is inherently designed for non-linear scenarios, operating in a reproducing kernel Hilbert space.

Additionally, A-CCA provides dimensionality reduction in the projected space from the very first step, a feature not present in KCCA.

All of this makes A-CCA highly advantageous for automation, as it enables the method to adapt effectively to both linear and non-linear relationships between variables.

#### *4.7.1. Data set with linear relations between variables*

In this subsection, we will see that in the particular case in which only linear relationships are present, A-CCA seems to perform better than KCCA.

As in the previous subsections, we generate two data sets of length 1000. The common variables are simulated as follows:

$$\begin{aligned}
&\mu_1 = (0, 4, 16)^T \\
&\Sigma_1 = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 10 & 40 \\ 2 & 40 & 165 \end{pmatrix} \\
&(X_1, X_2, X_3) \sim \mathcal{N}(\mu_1, \Sigma_1) \\
&X_4 \sim \mathcal{N}(0, 10) \\
&\mu_2 = (0, 1, 2)^T \\
&\Sigma_2 = \begin{pmatrix} 1 & 0.5 & 0.5 \\ 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 1 \end{pmatrix} \\
&(X_5, X_6, X_7) \sim \mathcal{N}(\mu_2, \Sigma_2),
\end{aligned} \tag{6}$$

while the non-common variables are generated as follows:

$$\begin{aligned}
&Y_1 \sim \mathcal{N}(0, 1) \\
&Y_2 = 3 \cdot X_2 + 4 \cdot Y_1 + \epsilon_2, \quad \epsilon_2, \sim \mathcal{N}(0, 1) \\
&Y_3 = 5 \cdot X_3 + 2 \cdot X_4 + Y_1 + \epsilon_3, \quad \epsilon_3 \sim \mathcal{N}(0, 1) \\
&Y_4 = 5 \cdot X_2 + 2 \cdot X_1 - Y_1 + \epsilon_4, \quad \epsilon_4 \sim \mathcal{N}(4, 11) \\
&Y_5 = 5 \cdot X_1 + 2 \cdot X_3 + Y_3.
\end{aligned} \tag{7}$$

Next, we train our method on the training set and create imputations for the test set. We then compare the results by evaluating the imputed data against the generated data using the model described in Equation (7).

The results, presented in Table 13, show that A-CCA performs better. This can be explained by greater flexibility of A-CCA, which allows to more easily capture the linear relationships between the projected spaces in the case of linear dependencies.

Indeed, an autoencoder is an extension of PCA using neural networks, which has the potential to capture non-linearity. It can also produce good results in linear cases, thanks to the flexibility of its neural networks.

On the other hand, KCCA, which can be viewed as a combination of KPCA followed by CCA, is particularly useful for non-linear cases. However, in linear cases such as the one studied here, it tends to yield poorer results.

#### *4.7.2. Data set with linear and non-linear relations between variables and log-normal distribution*

In this subsection, we generate two data sets of length 1000– one for training and one for testing–using the following rules.

| Variable | Measure | KCCA | A-CCA |
|---|---|---|---|
| $Y_1$ | wsMAE | 7.50 | 3.62 |
| $Y_1$ | RwsMSE | 9.46 | 4.55 |
| $Y_2$ | wsMAE | 0.13 | 0.14 |
| $Y_2$ | RwsMSE | 0.16 | 0.17 |
| $Y_3$ | wsMAE | 0.07 | 0.08 |
| $Y_3$ | RwsMSE | 0.09 | 0.10 |
| $Y_4$ | wsMAE | 0.17 | 0.18 |
| $Y_4$ | RwsMSE | 0.22 | 0.23 |
| $Y_5$ | wsMAE | 0.05 | 0.07 |
| $Y_5$ | RwsMSE | 0.07 | 0.09 |
| Total | RwsMSE | 2.00 | 1.03 |
| Total | Multivariate $R_1^2$ (Cohen, 2013) | 1.00 | 1.00 |
| Total | Multivariate $R_2^2$ (Jones, 2019) | 0.99 | 0.99 |

Table 13: Results for KCCA and A-CCA when the data sets have linear relationships. RwsMSE, Root weighted standardized Mean Squared Error;Total, all variables.

| Variable | Measure | KCCA | A-CCA |
|---|---|---|---|
| $Y_1$ | wsMAE | 0.36 | 0.34 |
| $Y_1$ | RwsMSE | 0.59 | 0.48 |
| $Y_2$ | wsMAE | 0.35 | 0.34 |
| $Y_2$ | RwsMSE | 0.51 | 0.44 |
| $Y_3$ | wsMAE | 0.09 | 0.08 |
| $Y_3$ | RwsMSE | 0.28 | 0.14 |
| $Y_4$ | wsMAE | 0.17 | 0.15 |
| $Y_4$ | RwsMSE | 0.38 | 0.23 |
| $Y_5$ | wsMAE | 0.12 | 0.14 |
| $Y_5$ | RwsMSE | 0.32 | 0.21 |
| Total | RwsMSE | 0.41 | 0.30 |
| Total | Multivariate $R_1^2$ (Cohen, 2013) | 1.00 | 1.00 |
| Total | Multivariate $R_2^2$ (Jones, 2019) | 0.91 | 0.97 |

Table 14: Results for KCCA and A-CCA when the data sets have linear relationships but non-linear terms. In this case $Y_1$ is generated using a log-Normal distribution. wsMAE, weighted standardized Mean Absolute Error; RwsMSE, Root weighted standardized Mean Squared Error;Total, all variables.

The common variables are simulated as follows:

$$
\begin{aligned}
&Y^* \sim \text{Lognormal}\,(0,1)\\
&X_1 \sim \mathcal{N}(0,1)\\
&X_2 \sim \mathcal{N}(0,1) + Y^*\\
&X_3 = 4\cdot X_2 + 2\cdot X_1 + \epsilon, \epsilon \sim \mathcal{N}(0,1)\\
&X_4 \sim \mathcal{N}(0,10)\\
&\mu_2 = (0,1,2)^T\\
&\Sigma_2 = \begin{pmatrix} 1 & 0.5 & 0.5\\ 0.5 & 1 & 0.5\\ 0.5 & 0.5 & 1 \end{pmatrix}\\
&(X_5, X_6, X_7) \sim \mathcal{N}(\mu_2, \Sigma_2)
\end{aligned}
\tag{8}
$$

while the non-common variables are simulated as follows:

$$
\begin{aligned}
&Y_1 = Y^*\\
&Y_2 = 3\cdot X_2 + 4\cdot \log(Y_1) + \epsilon_2, \quad \epsilon_2, \sim \mathcal{N}(0,1)\\
&Y_3 = 5\cdot X_3 + 2\cdot X_4 + \log(Y_1) + \epsilon_3, \quad \epsilon_3 \sim \mathcal{N}(0,1)\\
&Y_4 = 5\cdot X_2 + 2\cdot X_1 - \log(Y_1) + \epsilon_4, \quad \epsilon_4 \sim \mathcal{N}(0,2)\\
&Y_5 = 5\cdot X_1 + 2\cdot X_3 + Y_3.
\end{aligned}
\tag{9}
$$

Next, we train our method on the training set and create imputations for the test set. We then compare the results by evaluating the imputed data against the generated data using the model described in Equation (9). The results are presented in Table 14. As expected, the total RwsMSE is lower for A-CCA than for KCCA.

Furthermore, if we perform the same exercise, where the common variables are simulated as before according to Equations (8), while the non-common variables are generated according to Equations (9), but with all the $\log(Y_1)$ replaced by $\log(Y_1 + 1)$, the results are even better as presented in Table 15. This is because adding one inside the logarithm brings us closer to linearity.

#### *4.7.3. Data set with linear and non-linear relations between variables and beta distribution*

In this subsection, we generate two data sets of length 1000– one for training and one for testing–using the following rules.

| Variable | Measure | KCCA | A-CCA |
|---|---|---|---|
| $Y_1$ | wsMAE | 0.36 | 0.34 |
| $Y_1$ | RwsMSE | 0.59 | 0.48 |
| $Y_2$ | wsMAE | 0.26 | 0.26 |
| $Y_2$ | RwsMSE | 0.43 | 0.34 |
| $Y_3$ | wsMAE | 0.08 | 0.08 |
| $Y_3$ | RwsMSE | 0.28 | 0.14 |
| $Y_4$ | wsMAE | 0.17 | 0.14 |
| $Y_4$ | RwsMSE | 0.37 | 0.22 |
| $Y_5$ | wsMAE | 0.09 | 0.12 |
| $Y_5$ | RwsMSE | 0.31 | 0.19 |
| Total | RwsMSE | 0.40 | 0.27 |
| Total | Multivariate $R_1^2$ (Cohen, 2013) | 1.00 | 1.00 |
| Total | Multivariate $R_2^2$ (Jones, 2019) | 0.91 | 0.97 |

Table 15: Results for KCCA and A-CCA when the data sets have linear relationships but non-linear terms. In this case $Y_1$ is generated using a log-Normal distribution. wsMAE, weighted standardized Mean Absolute Error; RwsMSE, Root weighted standardized Mean Squared Error;Total, all variables.

The common variables are simulated as follows:

$$
\begin{aligned}
&Y^* \sim 0.2\cdot \text{Beta}\,(2,2)\\
&X_1 \sim \mathcal{N}(0,1)\\
&X_2 \sim \mathcal{N}(0,1) + Y^*\\
&X_3 = 4\cdot X_2 + 2\cdot X_1 + \epsilon, \epsilon \sim \mathcal{N}(0,1)\\
&X_4 \sim \mathcal{N}(0,10)\\
&\mu_2 = (0,1,2)^T\\
&\Sigma_2 = \begin{pmatrix} 1 & 0.5 & 0.5\\ 0.5 & 1 & 0.5\\ 0.5 & 0.5 & 1 \end{pmatrix}\\
&(X_5, X_6, X_7) \sim \mathcal{N}(\mu_2, \Sigma_2)
\end{aligned}
\tag{10}
$$

while the non-common variables are simulated as follows:

$$
\begin{aligned}
Y_1 &= Y^* \\
Y_2 &= 3 \cdot X_2 + 4 \cdot \log(Y_1 + 1) + \epsilon_2, \quad \epsilon_2, \sim \mathcal{N}(0,1) \\
Y_3 &= 5 \cdot X_3 + 2 \cdot X_4 + \log(Y_1 + 1) + \epsilon_3, \quad \epsilon_3 \sim \mathcal{N}(0,1) \\
Y_4 &= 5 \cdot X_2 + 2 \cdot X_1 - \log(Y_1 + 1) + \epsilon_4, \quad \epsilon_4 \sim \mathcal{N}(0,2) \\
Y_5 &= 5 \cdot X_1 + 2 \cdot X_3 + Y_3 \\
Y_6 &= 3 \cdot X_5 + 2 \cdot X_4 + \log(Y_1 + 1) + \epsilon_6; \quad \epsilon_6 \sim \mathcal{N}(0,1).
\end{aligned} \tag{11}
$$

Next, we train our method on the training set and create imputations for the test set. We then compare the results by evaluating the imputed data against the generated data using the model described in Equation (11).The results are presented in Table 16. As expected, the total RwsMSE is lower for A-CCA than for KCCA. Moreover, KCCA fails to reconstruct $Y_1$ effectively because a highly concentrated variable does not contribute notably to the kernel compared to other variables. However, with A-CCA, thanks to the flexibility of the autoencoder, the results for this variable are much better.

| Variable | Measure | KCCA | A-CCA |
|---|---|---|---|
| $Y_1$ | wsMAE | 4.26 | 0.16 |
| $Y_1$ | RwsMSE | 5.12 | 0.22 |
| $Y_2$ | wsMAE | 0.59 | 1.90 |
| $Y_2$ | RwsMSE | 0.73 | 2.39 |
| $Y_3$ | wsMAE | 0.10 | 0.19 |
| $Y_3$ | RwsMSE | 0.13 | 0.27 |
| $Y_4$ | wsMAE | 0.28 | 0.54 |
| $Y_4$ | RwsMSE | 0.35 | 0.68 |
| $Y_5$ | wsMAE | 0.17 | 0.17 |
| $Y_5$ | RwsMSE | 0.22 | 0.24 |
| $Y_6$ | wsMAE | 0.08 | 0.23 |
| $Y_6$ | RwsMSE | 0.11 | 0.31 |
| Total | RwsMSE | 1.11 | 0.69 |
| Total | Multivariate $R_1^2$ (Cohen, 2013) | 1.00 | 1.00 |
| Total | Multivariate $R_2^2$ (Jones, 2019) | 0.98 | 0.92 |

Table 16: Results for KCCA and A-CCA when the data sets have linear relationships but non-linear terms. In this case $Y_1$ is generated using a beta distribution. RwsMSE, Root weighted standardized Mean Squared Error;Total, all variables.

## 5. Application

In this section we will present an application of the proposed methods to integrate two real data sets, namely the Household Budget Survey (HBS) data sets and the Statistics on Income and Living Conditions (SILC) for Belgium in 2016. It is the same example as in Annoye et al. (2024) with the same objective imputing the consumption variables (grouped into 10 macro-categories, see Table 18) from HBS into SILC, using 12 common variables (see Table 17). We will use the two-step procedure of Annoye et al. (2024) as explained in Section 3.3. First, we impute some binary variables that indicate for each expenditure if the amount is equal to zero or not. In the next step, we impute the continuous variables taking into account the corresponding dummy variables that we constructed in the first step. Because in contrast to the previous section we do not have the true values of the non-common variables in SILC, the receiver data set, we cannot calculate all the different quantities of Section 4. The only exception is $\widetilde{\text{CVM}}$ criteria, where we also do not know the bivariate distributions of the common and non-common variables in SILC but we can approximate by the one in HBS, the donor data set.

We provide in Table 19 the averages of the $\widetilde{\text{CVM}}$ criteria for all couples of one common and one non-common variable as well as all couples of non-common variables[2]. We do not provide the results for HD because it will obviously overrate their quality because we would calculate with that formula the difference between two similar distributions calculated on the same donor data set.

We know from Annoye et al. (2024) that KCCA has good results for this application. In this similar experiment, we see that KCCA and A-CCA provide the best results for non-common variables while MMLP and A-CCA provide the best result in the mixed case. REG is as expected the worst performing method.

For the mixed case, we think that the very low digit for MMLP can come from overfitting since the CVM measure is based on the donor data set and the dependency between the common and non-common variables is reproduced from the donor data set by a flexible model for each non common variable.

In conclusion, A-CCA seems to be again the best method. The results are consistent with the ones in the section 4 (at least for non-common variables).

| **Variable** | **Type** |
|---|---|
| Gender | Categorical |
| Age | Continuous |
| Region | Categorical |
| Educational level attained (ISCED level) | Categorical |
| Marital status | Categorical |
| Activity status | Categorical |
| Type of work contract | Categorical |
| Status in employment | Categorical |
| Size of the household | Continuous |
| Number of Children | Continuous |
| Monthly imputed rent | Continuous |
| Total net income | Continuous |

Table 17: Common variables

## 6. Conclusion

In this paper, we proposed a new statistical matching method using Autoencoders Canonical Correlation Analysis (A-CCA) and we compare it with two other statistical matching machine learning techniques : one using Multi-output Multilayer Perceptron (MMLP), the more simple method and one using Kernel Canonical Correlation Analysis (KCCA), the best method in Annoye et al. (2024). We compare them with more traditional

[2]Data set A and B from Equation 5 which defines $\widetilde{\text{CVM}}$, are HBS data set (donor) and SILC data set (receiver), respectively.

| Variable | Type |
|---|---|
| Food products and non-alcoholic beverages | Continuous |
| Alcoholic beverages, tobacco, narcotics | Continuous |
| Clothing and footwear | Continuous |
| Housing, water, electricity, gas and other fuels | Continuous |
| Furnishing, household equipment and routine maintenance of the house | Continuous |
| Health | Continuous |
| Transport | Continuous |
| Communications | Continuous |
| Recreation and culture | Continuous |
| Education | Continuous |
| Restaurants and hotels (horeca) | Continuous |
| Miscellaneous goods and services | Continuous |

Table 18: Non-common variables

| | Non-Common | Mixed |
|---|---|---|
| **KCCA** | 431.37 | 500.89 |
| **A-CCA** | 444.02 | 467.45 |
| **MMLP** | 464.81 | 453.92 |
| **REG** | 962.63 | 918.48 |

Table 19: Fusion SILC-HBS. Average of the empirical Cramér-von Mises criteria ($\widetilde{\text{CVM}}$), as in eq. 5, for the different statistical matching techniques. Non-Common, average of the $\widetilde{\text{CVM}}$ criteria for all couples of non-common variables; Mixed, average of the $\widetilde{\text{CVM}}$ criteria for all couples of one common and one non-common variable.

methods such as distance hot-deck (HD) and multivariate and multinomial regression (REG). All the methodologies take the sampling weights into account. We deal with mixed data and incompatibilities between categorical variables by using a two-step procedure.

First, we use the 2017 Belgian Statistics on Income and Living Conditions (SILC) data set to assess the quality of our methodology. We divided it into five folds. Using four folds, we predict in the fifth fold a set of variables as if they were missing and we do that for all the folds. With this application we have seen that the best method is A-CCA, followed by KCCA. These two methods harness the different advantages of both REG and HD methods: having small prediction errors and preserving the joint distributions. We provide a second application, where we impute the consumption variables from the Household Budget Survey (HBS) into SILC data set. Once again, we observe that A-CCA and MMLP seem to perform very well.

For further research, the creation of a bootstrap version of these procedures seems worthwhile, given the computational cost of methods such as A-CCA and KCCA, as well as the study of the integration of multiple data sets via an iterative procedure. A final promising avenue is to see if this method can be modified to generate fictitious data sets to deal with confidential data issues.

**Algorithm 2** Two-step algorithm with A-CCA

**Input:** $\mathbf{X}^A$ and $\mathbf{Y}^A$, the centered common and non-common variables in data set $A$ and $\mathbf{X}^B$, the centered common variables in data set $B$. $\mathbf{w}^A$ and $\mathbf{w}^B$, the weights in both data sets.

**Output:** A data set $B$ with the centered $\mathbf{X}^B$ and $\widehat{\mathbf{Y}}^B$.

**for** $i = 1, \cdots, n^B$ **do**
  **for** $j = 1, \cdots, n^A$ **do**
    Calculate $\Theta_{ij} = \mathbf{1}\left(\left(\mathbf{X}^A_{ca}\right)_i = \left(\mathbf{X}^B_{ca}\right)_j\right)$.
  **end for**
  Set $k = 1$
  **while** $\Theta_{i\cdot} = \overrightarrow{0}$ **do**
    **for** $j = 1, \cdots, n^A$ **do**
      Calculate $\Theta_{ij} = \mathbf{1}\left(\left(\mathbf{X}^A_{ca}\right)_i \overset{d_{ca}-k}{=} \left(\mathbf{X}^B_{ca}\right)_j\right)$ where $\overset{d_{ca}-k}{=}$ denotes the fact that the equality must be satisfied on $d_{ca}-k$ components.
    **end for**
    Set $k = k + 1$.
  **end while**
**end for**
Fit an autoencoder on $\mathbf{X}^A$ to obtain $\boldsymbol{\varphi}^A_X$ and another on $\mathbf{Y}^A_{ca}$ to obtain $\boldsymbol{\varphi}^A_Y$.
Calculate $\mathbf{a}_{ca}$ and $\mathbf{b}_{ca}$ the canonical correlation of $\boldsymbol{\varphi}^A_X$ and $\boldsymbol{\varphi}^A_Y$ using $\mathbf{w}^A$.
Use the fitted autoencoder on $\mathbf{X}^A$ to obtain $\boldsymbol{\varphi}^B_X$ with $\mathbf{X}^B$.
Calculate $\mathbf{U}^A_{ca} = \boldsymbol{\varphi}^A_X\mathbf{a}_{ca}$ and $\mathbf{U}^B_{ca} = \boldsymbol{\varphi}^B_X\mathbf{a}_{ca}$ the canonical correlation variables based on the latent space.
**for** $i = 1, \cdots, n^B$ **do**
  **for** $j = 1, \cdots, n^A$ **do**
    Calculate $\varpi_{ij} = K_h\left(u^B_i, u^A_j\right)$ where $K_h$ is a Gaussian kernel.
  **end for**
**end for**
**for** $i = 1, \cdots, n^B$ **do**
  Draw a $\left(\widehat{\mathbf{Y}}^B_{ca}\right)_i$ from $\mathbf{Y}^A_{ca}$ with the following probability $\tilde{\boldsymbol{\omega}}_i$.
**end for**
Create the matrices $\widetilde{\mathbf{X}^A} = \left(\mathbf{X}^A, \mathbf{Y}^A_{ca}\right)$ and $\widetilde{\mathbf{X}^B} = \left(\mathbf{X}^B, \widehat{\mathbf{Y}}^B_{ca}\right)$.
Fit an autoencoder on $\widetilde{\mathbf{X}^A}$ to obtain $\boldsymbol{\varphi}^A_{\widetilde{X}}$ and another on $\mathbf{Y}^A_{co}$ to obtain $\boldsymbol{\varphi}^A_{Y_{co}}$.
Calculate $\mathbf{a}_{co}$ and $\mathbf{b}_{co}$ the canonical correlation of $\boldsymbol{\varphi}^A_{\widetilde{X}}$ and $\boldsymbol{\varphi}^A_{Y_{co}}$ using $\mathbf{w}^A$.
Use the fitted autoencoder on $\widetilde{\mathbf{X}^A}$ to obtain $\boldsymbol{\varphi}^B_{\widetilde{X}}$ with $\widetilde{\mathbf{X}^B}$.
Calculate $\widetilde{\mathbf{U}^A} = \boldsymbol{\varphi}^A_{\widetilde{X}}\mathbf{a}_{co}$ and $\widetilde{\mathbf{U}^B} = \boldsymbol{\varphi}^B_{\widetilde{X}}\mathbf{a}_{co}$ the canonical correlation variables based on the latent space.
**for** $i = 1, \cdots, n^B$ **do**
  **for** $j = 1, \cdots, n^A$ **do**
    Calculate $\varpi_{ij} = K_h\left(\widetilde{u}^B_i, \widetilde{u}^A_j\right)$ where $K_h$ is a Gaussian kernel.
  **end for**
**end for**
Calculate $\widehat{\mathbf{Y}}^B_{co} = \boldsymbol{\Omega}\mathbf{Y}^A_{co}$, where:

$$\boldsymbol{\Omega} = (\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \cdots, \boldsymbol{\omega}_{n_B})^T$$

$$\boldsymbol{\omega}_i = \left(\frac{w^A_1\varpi_{i1}}{\sum_{j=1}^{n_A} w^A_j\varpi_{ij}}, \frac{w^A_2\varpi_{i2}}{\sum_{j=1}^{n_A} w^A_j\varpi_{ij}}, \cdots, \frac{w^A_{n_A}\varpi_{in_A}}{\sum_{j=1}^{n_A} w^A_j\varpi_{ij}}\right)^T.$$

$\widehat{\mathbf{Y}}^B = \left(\widehat{\mathbf{Y}}^B_{ca}, \widehat{\mathbf{Y}}^B_{co}\right)$. endalgorithmic

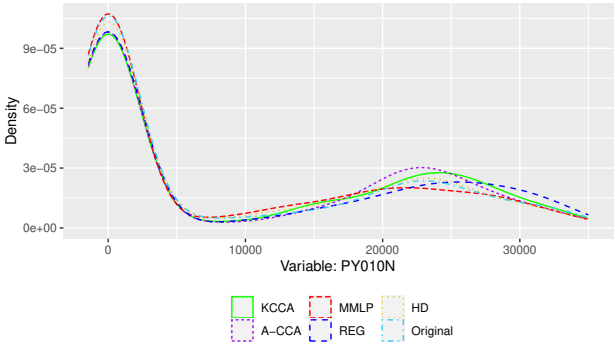


Figure 4: Density plot of variable PY010N for the first fold

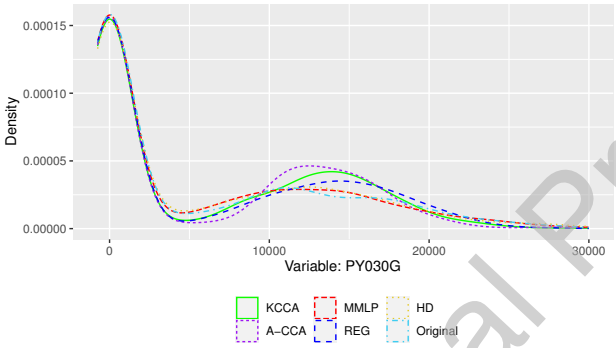


Figure 5: Density plot of variable PY030G for the first fold

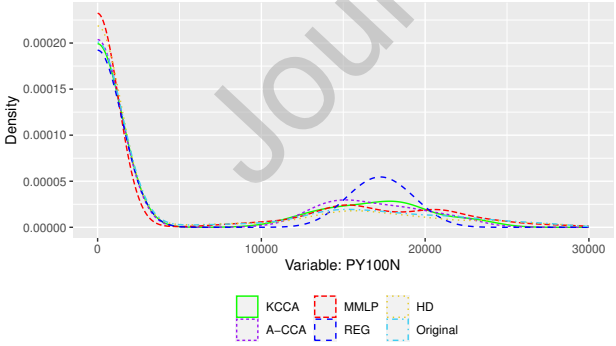


Figure 6: Density plot of variable PY100N for the first fold

| Variable | Measure | KCCA | A-CCA | MMLP | HD | REG |
|---|---|---|---|---|---|---|
| PY010N | wsMAE | 0.23 | 0.26 | 0.24 | 0.64 | 0.22 |
| PY010N | RwsMSE | 0.59 | 0. 65 | 0.64 | 1.46 | 0.58 |
| PY030G | wsMAE | 0.27 | 0.28 | 0.27 | 0.64 | 0.24 |
| PY030G | RwsMSE | 0.63 | 0.67 | 0.68 | 1.48 | 0.61 |
| PY100N | wsMAE | 0.20 | 0.22 | 0.20 | 0.29 | 0.20 |
| PY100N | RwsMSE | 0.58 | 0.61 | 0.57 | 0.77 | 0.57 |
| RX050 | wMCR | 0.13 | 0.14 | 0.12 | 0.16 | 0.12 |
| PB200 | wMCR | 0.14 | 0.13 | 0.13 | 0.19 | 0.13 |
| PH030 | wMCR | 0.33 | 0.33 | 0.29 | 0.37 | 0.26 |
| Total (cont.) | RwsMSE | 0.60 | 0.64 | 0.63 | 1.23 | 0.58 |
| Total (cat.) | RwsMSE | 0.95 | 0.95 | 0.90 | 1.03 | 0.87 |
| Total | RwsMSE | 0.87 | 0.87 | 0.83 | 1.08 | 0.80 |
| Total | Multivariate $R_1^2$ (Cohen, 2013) | 0.99 | 0.99 | 0.99 | 0.97 | 0.99 |
| Total | Multivariate $R_2^2$ (Jones, 2019) | 0.63 | 0.57 | 0.58 | -1.05 | 0.64 |
| Total | Average $\widetilde{\text{CVM}}$ Bivariate Non-Common | 1.18 | 1.03 | 3.06 | 0.67 | 9.85 |
| Total | Average $\widetilde{\text{CVM}}$ Bivariate Mixed | 0.67 | 0.57 | 1.67 | 0.82 | 5.42 |

Table 20: Results (average over five folds) of the different statistical matching techniques applied to the SILC data set only. wsMAE, weighted standardized Mean Absolute Error; RwsMSE, Root weighted standardized Mean Squared Error; wMCR, weighted Misclassification Rate; Total, all variables; Total (cont.), all continuous variables; Total (cat.), all categorical variables; Bivariate Non-Common, average of the $\widetilde{\text{CVM}}$ criteria for all couples of non-common variables; Bivariate Mixed, average of the $\widetilde{\text{CVM}}$ criteria for all couples of one common and one non-common variable.

| Variable | Measure | KCCA | A-CCA | MMLP | HD | REG |
|---|---|---|---|---|---|---|
| PY010N | wsMAE | 0.04 | 0.05 | 0.04 | 0.13 | 0.04 |
| PY010N | RwsMSE | 0.14 | 0.12 | 0.11 | 0.48 | 0.14 |
| PY030G | wsMAE | 0.05 | 0.06 | 0.05 | 0.14 | 0.05 |
| PY030G | RwsMSE | 0.12 | 0.11 | 0.10 | 0.53 | 0.14 |
| PY100N | wsMAE | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 |
| PY100N | RwsMSE | 0.03 | 0.04 | 0.03 | 0.04 | 0.05 |
| RX050 | wMCR | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 |
| PB200 | wMCR | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 |
| PH030 | wMCR | 0.01 | 0.01 | 0.00 | 0.01 | 0.01 |
| Total (cont.) | RwsMSE | 0.09 | 0.08 | 0.07 | 0.34 | 0.09 |
| Total (cat.) | RwsMSE | 0.02 | 0.02 | 0.02 | 0.01 | 0.02 |
| Total | RwsMSE | 0.03 | 0.02 | 0.03 | 0.09 | 0.03 |
| Total | mult-$R^2_1$ (Cohen, 2013) | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Total | mult-$R^2_2$ (Jones, 2019) | 0.17 | 0.15 | 0.14 | 1.30 | 0.17 |
| Total | Average $\widetilde{\text{CVM}}$ Bivariate Non-Common | 0.44 | 0.28 | 1.19 | 0.25 | 2.95 |
| Total | Average $\widetilde{\text{CVM}}$ Bivariate Mixed | 0.25 | 0.15 | 0.66 | 0.17 | 1.55 |

Table 21: Standard deviation of the results over the 5 folds for the different statistical matching techniques applied to the SILC data set only. wsMAE, weighted standardized Mean Absolute Error; RwsMSE, Root weighted standardized Mean Squared Error; wMCR, weighted Misclassification Rate; Total, all variables; Total (cont.), all continuous variables; Total (cat.), all categorical variables; Bivariate Non-Common, average of the $\widetilde{\text{CVM}}$ criteria for all couples of non-common variables; Bivariate Mixed, average of the $\widetilde{\text{CVM}}$ criteria for all couples of one common and one non-common variable.

| Variable | Measure | KCCA | A-CCA | MMLP | HD | REG |
|---|---|---|---|---|---|---|
| PY010N | wsMAE | 0.16 | 0.18 | 0.17 | 0.41 | 0.15 |
| PY010N | RwsMSE | 0.51 | 0.57 | 0.55 | 1.04 | 0.48 |
| PY030G | wsMAE | 0.18 | 0.17 | 0.18 | 0.40 | 0.16 |
| PY030G | RwsMSE | 0.55 | 0.58 | 0.59 | 1.03 | 0.52 |
| PY100N | wsMAE | 0.19 | 0.20 | 0.19 | 0.27 | 0.19 |
| PY100N | RwsMSE | 0.55 | 0.54 | 0.53 | 0.71 | 0.53 |
| RX050 | wMCR | 0.12 | 0.13 | 0.11 | 0.15 | 0.09 |
| PB200 | wMCR | 0.13 | 0.12 | 0.12 | 0.18 | 0.12 |
| PH030 | wMCR | 0.32 | 0.32 | 0.28 | 0.35 | 0.25 |
| Total (cont.) | RwsMSE | 0.54 | 0.57 | 0.56 | 0.95 | 0.51 |
| Total (cat.) | RwsMSE | 0.94 | 0.93 | 0.88 | 1.02 | 0.86 |
| Total | RwsMSE | 0.84 | 0.84 | 0.80 | 1.01 | 0.77 |
| Total | mult-$R_1^2$ (Cohen, 2013) | 0.98 | 0.98 | 0.99 | 0.97 | 0.99 |
| Total | mult-$R_2^2$ (Jones, 2019) | 0.33 | 0.30 | 0.35 | -2.87 | 0.33 |
| Total | Average $\widetilde{\text{CVM}}$ Bivariate Non-Common | 0.70 | 0.85 | 1.39 | 0.44 | 6.79 |
| Total | Average $\widetilde{\text{CVM}}$ Bivariate Mixed | 0.41 | 0.48 | 0.79 | 0.66 | 3.78 |

Table 22: Minimum of the results over the 5 folds for the different statistical matching techniques applied to the SILC data set only. wsMAE, weighted standardized Mean Absolute Error; RwsMSE, Root weighted standardized Mean Squared Error; wMCR, weighted Misclassification Rate; Total, all variables; Total (cont.), all continuous variables; Total (cat.), all categorical variables; Bivariate Non-Common, average of the $\widetilde{\text{CVM}}$ criteria for all couples of non-common variables; Bivariate Mixed, average of the $\widetilde{\text{CVM}}$ criteria for all couples of one common and one non-common variable.

| Variable | Measure | KCCA | A-CCA | MMLP | HD | REG |
|---|---|---|---|---|---|---|
| PY010N | wsMAE | 0.25 | 0.30 | 0.28 | 0.73 | 0.24 |
| PY010N | RwsMSE | 0.83 | 0.85 | 0.82 | 2.11 | 0.83 |
| PY030G | wsMAE | 0.29 | 0.35 | 0.32 | 0.73 | 0.27 |
| PY030G | RwsMSE | 0.85 | 0.85 | 0.84 | 2.17 | 0.85 |
| PY100N | wsMAE | 0.21 | 0.23 | 0.21 | 0.31 | 0.21 |
| PY100N | RwsMSE | 0.62 | 0.65 | 0.62 | 0.82 | 0.62 |
| RX050 | wMCR | 0.14 | 0.15 | 0.14 | 0.17 | 0.13 |
| PB200 | wMCR | 0.16 | 0.14 | 0.14 | 0.20 | 0.14 |
| PH030 | wMCR | 0.35 | 0.33 | 0.30 | 0.38 | 0.27 |
| Total (cont.) | RwsMSE | 0.75 | 0.77 | 0.73 | 1.68 | 0.74 |
| Total (cat.) | RwsMSE | 0.99 | 0.97 | 0.94 | 1.04 | 0.90 |
| Total | RwsMSE | 0.90 | 0.90 | 0.87 | 1.18 | 0.84 |
| Total | mult-$R^2_1$ (Cohen, 2013) | 0.99 | 0.99 | 1.00 | 0.98 | 1.00 |
| Total | mult-$R^2_2$ (Jones, 2019) | 0.72 | 0.67 | 0.69 | -0.05 | 0.75 |
| Total | Average $\widetilde{\text{CVM}}$ Bivariate Non-Common | 1.77 | 1.52 | 4.59 | 0.96 | 12.05 |
| Total | Average $\widetilde{\text{CVM}}$ Bivariate Mixed | 0.98 | 0.84 | 2.55 | 1.09 | 6.67 |

Table 23: Maximum of the results over the 5 folds for the different statistical matching techniques applied to the SILC data set only. wsMAE, weighted standardized Mean Absolute Error; RwsMSE, Root weighted standardized Mean Squared Error; wMCR, weighted Misclassification Rate; Total, all variables; Total (cont.), all continuous variables; Total (cat.), all categorical variables; Bivariate Non-Common, average of the $\widetilde{\text{CVM}}$ criteria for all couples of non-common variables; Bivariate Mixed, average of the $\widetilde{\text{CVM}}$ criteria for all couples of one common and one non-common variable.

| | True | KCCA | A-CCA | MMLP | REG | HD |
|---|---|---|---|---|---|---|
| $CH^{-1}$ (13) | $9.38 \cdot 10^{-5}$ | $7.89 \cdot 10^{-5}$ | $6.78 \cdot 10^{-5}$ | $2.26 \cdot 10^{-4}$ | $2.01 \cdot 10^{-5}$ | $3.35 \cdot 10^{-3}$ |
| Purity DB040 | 0.58 | 0.58 | 0.58 | 0.58 | 0.59 | 0.51 |
| Purity HX060 | 0.44 | 0.45 | 0.46 | 0.47 | 0.46 | 0.42 |
| Purity PB220A | 0.90 | 0.91 | 0.91 | 0.90 | 0.90 | 0.88 |
| Purity PE040 | 0.46 | 0.47 | 0.46 | 0.46 | 0.52 | 0.37 |
| Purity PL031 | 0.73 | 0.75 | 0.77 | 0.82 | 0.78 | 0.65 |
| Purity RB090 | 0.65 | 0.68 | 0.65 | 0.58 | 0.66 | 0.64 |
| Purity RX050 | 0.82 | 0.83 | 0.86 | 0.83 | 0.88 | 0.80 |
| Entropy DB040 | 0.81 | 0.81 | 0.81 | 0.81 | 0.78 | 0.90 |
| Entropy HX060 | 0.72 | 0.71 | 0.71 | 0.70 | 0.70 | 0.75 |
| Entropy PB220A | 0.29 | 0.30 | 0.30 | 0.29 | 0.30 | 0.40 |
| Entropy PE040 | 0.70 | 0.67 | 0.70 | 0.69 | 0.64 | 0.76 |
| Entropy PL031 | 0.37 | 0.34 | 0.32 | 0.26 | 0.29 | 0.58 |
| Entropy RB090 | 0.89 | 0.85 | 0.89 | 0.95 | 0.90 | 0.92 |
| Entropy RX050 | 0.40 | 0.39 | 0.33 | 0.38 | 0.29 | 0.43 |

Table 24: Results on one fold of the different $k$-means with $k = 13$, $CH^{-1}$, inverse of the weighted Calinski-Harabasz index.

| Variable | Measure | KCCA | A-CCA | MMLP | HD | REG |
|---|---|---|---|---|---|---|
| Total (cont.) | RwsMSE | 1.12 | 0.86 | 0.92 | 1.38 | 8.67 |
| Total (cat.) | RwsMSE | 0.85 | 0.85 | 0.84 | 1.10 | 0.71 |
| Total | RwsMSE | 0.89 | 0.85 | 0.85 | 1.14 | 1.92 |
| Total | Multivariate $R_2^2$ (Jones, 2019) | 0.24 | 0.1 | -0.45 | -1.1 | 0.24 |
| Total | Average $\widetilde{\text{CVM}}$ Bivariate Non-Common | 111.13 | 108.39 | 102.03 | 0.61 | 138.74 |
| Total | Average $\widetilde{\text{CVM}}$ Bivariate Mixed | 55.18 | 53.77 | 50.64 | 0.46 | 71.39 |

Table 25: Results (average over five folds) of the different statistical matching techniques applied to the full SILC data set only. wsMAE, weighted standardized Mean Absolute Error; RwsMSE, Root weighted standardized Mean Squared Error; wMCR, weighted Misclassification Rate; Total, all variables; Total (cont.), all continuous variables; Total (cat.), all categorical variables; Bivariate Non-Common, average of the $\widetilde{\text{CVM}}$ criteria for all couples of non-common variables; Bivariate Mixed, average of the $\widetilde{\text{CVM}}$ criteria for all couples of one common and one non-common variable;

## Appendix A. Link between KPCA and KCCA

Finding the kernel principal component is equivalent to solve the following problem:

$$\arg\min_{\mathbf{A}_x\in\mathbb{R}^{n\times\varrho}} \left\|\mathbf{\Phi}_x^T - \mathbf{\Upsilon}_x\mathbf{\Upsilon}_x^T\mathbf{\Phi}_x^T\right\|^2 = \arg\min_{\mathbf{A}_x\in\mathbb{R}^{n\times\varrho}} \left\|\mathbf{\Phi}_x^T - \mathbf{\Phi}_x^T\mathbf{A}_x\mathbf{A}_x^T\mathbf{K}_x\right\|^2,$$

subject to $\mathbf{\Upsilon}_x^T\mathbf{\Upsilon}_x = \mathbf{\Phi}_x^T\mathbf{A}_x\mathbf{A}_x^T\mathbf{\Phi}_x = I_d$. As explained in Kuss and Graepel (2003), KCCA can be seen as two Kernel Principal Component Analysis (KPCA) followed by a classical CCA.

Indeed, the $\varrho$ first principal components $\boldsymbol{\upsilon}_i$ belong to $\mathcal{L}(\mathbf{\Phi}_x)$ the effective features space, thus we have that $\mathcal{L}(\mathbf{\Upsilon}_x) \subseteq \mathcal{L}(\mathbf{\Phi}_x)$ where $\mathbf{\Upsilon}_x$ is the matrix that combines the $\varrho$ $\boldsymbol{\upsilon}_i$ and so $\mathbf{\Upsilon}_x = \mathbf{\Phi}_x^T\mathbf{A}_x$. $\mathbf{A}_x$ can be calculated as $\mathbf{V}_x\mathbf{\Lambda}^{\frac{-1}{2}}$ where $\mathbf{V}_x$ and $\mathbf{\Lambda}$ come from the eigenvalue decomposition $\mathbf{K}_x = \mathbf{V}_x\mathbf{\Lambda}\mathbf{V}_x^T$. Indeed, as explained in Washizawa (2012) :

$$\begin{aligned}
&\left\|\mathbf{\Phi}_x^T - \mathbf{\Upsilon}_x\mathbf{\Upsilon}_x^T\mathbf{\Phi}_x^T\right\|^2 \\
&= \left\|\mathbf{\Phi}_x^T - \mathbf{\Phi}_x^T\mathbf{A}_x\mathbf{A}_x^T\mathbf{K}_x\right\|^2 \\
&= \mathrm{Trace}\left(\left(\mathbf{\Phi}_x^T - \mathbf{\Phi}_x^T\mathbf{A}_x\mathbf{A}_x^T\mathbf{K}_x\right)^T\left(\mathbf{\Phi}_x^T - \mathbf{\Phi}_x^T\mathbf{A}_x\mathbf{A}_x^T\mathbf{K}_x\right)\right) \\
&= \mathrm{Trace}\left(\left(\mathbf{\Phi}_x - \mathbf{K}_x\mathbf{A}_x\mathbf{A}_x^T\mathbf{\Phi}_x\right)\left(\mathbf{\Phi}_x^T - \mathbf{\Phi}_x^T\mathbf{A}_x\mathbf{A}_x^T\mathbf{K}_x\right)\right) \\
&= \mathrm{Trace}\left(\mathbf{K}_x - \mathbf{K}_x\mathbf{A}_x\mathbf{A}_x^T\mathbf{K}_x - \mathbf{K}_x\mathbf{A}_x^T\mathbf{A}_x\mathbf{K}_x + \mathbf{K}_x\mathbf{A}_x\mathbf{A}_x^T\mathbf{K}_xA_x\mathbf{A}_x^T\mathbf{K}_x\right) \\
&= \left\|\mathbf{K}_x\mathbf{A}_x\mathbf{A}_x^T\mathbf{K}_x^{\frac{1}{2}} - \mathbf{K}_x^{\frac{1}{2}}\right\|^2. \qquad \text{(A.1)}
\end{aligned}$$

As a reminder, $\mathbf{K}_x$ being a Gram matrix (and so symmetric positive definite matrix), we have the following eigenvalue decompositions:

$$\begin{aligned}
\mathbf{K}_x &= \mathbf{V}_x\mathbf{\Lambda}\mathbf{V}_x^T \\
\mathbf{K}_x^{-1} &= \left(\mathbf{V}_x\mathbf{\Lambda}\mathbf{V}_x^T\right)^{-1} = \mathbf{V}_x\mathbf{\Lambda}^{-1}\mathbf{V}_x, \\
\mathbf{K}_x^{\frac{1}{2}} &= \mathbf{V}_x\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{V}_x^T.
\end{aligned}$$

where $\mathbf{V}_x$ is an orthogonal matrix. Thus by the Eckart-Young theorem, also called Schmidt approximation theorem (Ben-Israel and Greville, 1973), equation (A.1) is minimized when :

$$\mathbf{K}_x\mathbf{A}_x\mathbf{A}_x^T\mathbf{K}_x^{\frac{1}{2}} = \sum_{i=1}^{d}\sqrt{\lambda_i}v_iv_i^T$$

where $d$ is the rank of $\mathbf{K}_x$.

From that, we conclude that:

$$\mathbf{A}_x\mathbf{A}_x^T = \mathbf{K}_x^{-1} = \sum_{i=1}^{d}\lambda_i^{-1}v_iv_i^T = \mathbf{V}_x\mathbf{\Lambda}^{-1}\mathbf{V}_x^T$$

Thus $\mathbf{A}_x$ can be calculated as $\mathbf{V}_x\mathbf{\Lambda}^{\frac{-1}{2}}$ where $\mathbf{V}_x$ and $\mathbf{\Lambda}$ come from the eigenvalue decomposition $\mathbf{K}_x = \mathbf{V}_x\mathbf{\Lambda}\mathbf{V}_x^T$. The coordinates of the $\boldsymbol{\phi}_x$ with respect to the principal components are $\mathbf{C}_x = \mathbf{K}_x\mathbf{A}_x$, which we will name kernel principal component scores.

If we apply a classical CCA on them:

$$\begin{pmatrix} 0 & \mathbf{C}_x^T\mathbf{C}_y \\ \mathbf{C}_y^T\mathbf{C}_x & 0 \end{pmatrix}\begin{pmatrix}\boldsymbol{\psi}\\ \boldsymbol{\eta}\end{pmatrix} = \lambda\begin{pmatrix} \mathbf{C}_x^T\mathbf{C}_x & 0 \\ 0 & \mathbf{C}_y^T\mathbf{C}_y \end{pmatrix}\begin{pmatrix}\boldsymbol{\psi}\\ \boldsymbol{\eta}\end{pmatrix},$$

and we recall that KCCA corresponds to the following eigenvalue problem:

$$\begin{pmatrix} 0 & \mathbf{K}_x\mathbf{K}_y \\ \mathbf{K}_y\mathbf{K}_x & 0 \end{pmatrix}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} = \lambda\begin{pmatrix} \mathbf{K}_x\mathbf{K}_x & 0 \\ 0 & \mathbf{K}_y\mathbf{K}_y \end{pmatrix}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix},$$

we obtain the following link with KCCA:

$$\boldsymbol{\alpha}_j = \mathbf{A}_x\boldsymbol{\psi}_j,$$

$$\boldsymbol{\beta}_j = \mathbf{A}_y\boldsymbol{\eta}_j,$$

and, then, the kernel canonical variables are:

$$\boldsymbol{u}_j = \mathbf{K}_x\boldsymbol{\alpha}_j = \mathbf{K}_x\mathbf{A}_x\boldsymbol{\psi}_j,$$

$$\mathbf{v}_j = \mathbf{K}_y\boldsymbol{\beta}_j = \mathbf{K}_y\mathbf{A}_y\boldsymbol{\eta}_j.$$

## Compliance with ethical standards

***Conflict of interest***. The authors declare that they have no conflict of interest.


## Acknowledgement

We want to thank Ida-Marie Jensen for her contributions and ideas for this article.


## DECLARATION OF INTERESTS

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

[Une image contenant texte, tableau blanc Description générée automatiquement]

## References


Akaho, S., 2001. A kernel method for canonical correlation analysis. arXiv preprint arXiv:0609071 , 1–7arXiv:0609071.

Anderson, T.W., 1957. Maximum likelihood estimates for a multivariate normal distribution when some observations are missing. Journal of the American Statistical Association 52, 200–203. doi:https://doi.org/10.1080/01621459.1957.10501379.

Andrew, G., Arora, R., Bilmes, J., Livescu, K., 2013. Deep canonical correlation analysis, in: Dasgupta, S., McAllester, D. (Eds.), Proceedings of the 30th International Conference on Machine Learning, PMLR, Atlanta, Georgia, USA. pp. 1247–1255. URL: https://proceedings.mlr.press/v28/andrew13.html.

Annoye, H., Beretta, A., Heuchenne, C., 2024. Statistical matching using kernel canonical correlation analysis and super-organizing map. Expert Systems with Applications , 123134doi:https://doi.org/10.1016/j.eswa.2023.123134.

Ben-Israel, A., Greville, T., 1973. Generalized matrix inverses: Theory and practice.

Cohen, J., Cohen, P., West, S.G., Aiken, L.R.S., 2002. Applied multiple regression/correlation analysis for the behavioral sciences. Routledge. doi:https://doi.org/10.4324/9780203774441.

Conti, P.L., Marella, D., Scanu, M., 2017. How far from identifiability? a systematic overview of the statistical matching problem in a non parametric framework. Communications in Statistics - Theory and Methods 46, 967–994. doi:https://doi.org/10.1080/03610926.2015.1010005.

Donatiello, G., D'Orazio, M., Frattarola, D., Rizzi, A., Scanu, M., Spaziani, M., 2014. Statistical matching of income and consumption expenditures. International Journal of Economic Sciences 3, 50.

D'Orazio, M., Di Zio, M., Scanu, M., 2006a. Statistical matching for categorical data: Displaying uncertainty and using logical constraints. Journal of Official Statistics - Stockholm 22, 137.

D'Orazio, M., Di Zio, M., Scanu, M., 2006b. Statistical matching: Theory and practice. John Wiley & Sons. doi:https://doi.org/10.1002/0470023554.

Fosdick, B.K., DeYoreo, M., Reiter, J.P., et al., 2016. Categorical data fusion using auxiliary information. Annals of Applied Statistics 10, 1907–1929. doi:https://doi.org/10.1214/16-AOAS925.

Hotelling, H., 1936. Relations between two sets of variates. Biometrika 28 (3-4), 321–377. doi:https://doi.org/10.2307/2333955.

Jones, T., 2019. A coefficient of determination for probabilistic topic models. arXiv preprint arXiv:1911.11061 arXiv:1911.11061.

Kaloga, Y., Borgnat, P., Chepuri, S.P., Abry, P., Habrard, A., 2020. Multiview variational graph autoencoders for canonical correlation analysis. arXiv preprint arXiv:2010.16132 arXiv:2010.16132.

Kim, J.K., Shao, J., 2013. Statistical methods for handling incomplete data. Chapman and Hall/CRC, New York. doi:https://doi.org/10.1201/b13981.

Kuss, M., Graepel, T., 2003. The geometry of kernel canonical correlation analysis. Technical Report. Max Planck Institute for Biological Cybernetics.

Lai, P.L., Fyfe, C., 2000. Kernel and nonlinear canonical correlation analysis. International Journal of Neural Systems 10, 365–377. doi:https://doi.org/10.1142/s012906570000034x.

López-Laborda, J., Marín-González, C., Onrubia-Fernández, J., 2020. Estimating engel curves: a new way to improve the silc-hbs matching process using glm methods. Journal of Applied Statistics 47, 1–18. doi:https://doi.org/10.1080/02664763.2020.1796933.

Luo, L., Xu, J., Lin, J., Zeng, Q., Sun, X., 2018. An auto-encoder matching model for learning utterance-level semantic dependency in dialogue generation. arXiv preprint arXiv:1808.08795 doi:https://doi.org/10.18653/v1/D18-1075, arXiv:1808.08795.

Mitsuhiro, M., Hoshino, T., 2020. Kernel canonical correlation analysis for data combination of multiple-source datasets. Japanese Journal of Statistics and Data Science 3, 1–18. doi:https://doi.org/10.1007/s42081-020-00074-z.

Rässler, S., 2002. Statistical matching: A frequentist theory, practical applications, and alternative Bayesian approaches. volume 168. Springer Science & Business Media. doi:https://doi.org/10.1007/978-1-4613-0053-3.

Rumelhart, D.E., Hinton, G., Williams, R., 1986. Learning Internal Representations by Error Propagation. MIT Press, Cambridge, MA, USA. pp. 318–362.

Saverio, G., Romano, M.C., Gianni, C., Di Zio, M., Marcello, D., Federica, P., Mauro, S., TORELLI, N., 2008. Time Use and Labour Force: a proposal to integrate the datathrough statistical matching. Technical Report. Istat-Produzione libraria e centro stampa.

Schölkopf, B., Smola, A., Müller, K.R., 1998. Nonlinear component analysis as a kernel eigenvalue problem. Neural computation 10, 1299–1319.

Serafino, P., Tonkin, R., 2017. Statistical matching of European Union statistics on income and living conditions (EU-SILC) and the household budget survey. Technical Report. Eurostat: Statistical Working Papers. Luxembourg: Publications Office of the European Union. doi:https://doi.org/10.2785/933460.

Tonkin, R., Webber, D., 2012. Statistical matching of eu-silc and household budget survey to compare poverty estimates using income, expenditures and material deprivation, in: EU-SILC International Conference, Vienna, pp. 6–7. doi:https://doi.org/10.2785/4151.

Van Buuren, S., 2018. Flexible Imputation of Missing Data. Chapman and Hall/CRC, New York. doi:https://doi.org/10.1201/9780429492259.

Wang, W., Arora, R., Livescu, K., Bilmes, J., 2016. On deep multi-view representation learning: Objectives and optimization. arXiv preprint arXiv:1602.01024 arXiv:1602.01024.

Washizawa, Y., 2012. Subset basis approximation of kernel principal component analysis. Principal Component Analysis , 67.

Xiu, X., Li, Y., 2023. Learning sparse kernel cca with graph priors for nonlinear process monitoring. IEEE Sensors Journal 23, 7381–7389. doi:10.1109/JSEN.2023.3245832.

Xiu, X., Miao, Z., Yang, Y., Liu, W., 2022. Deep canonical correlation analysis using sparsity-constrained optimization for nonlinear process monitoring. IEEE Transactions on Industrial Informatics 18, 6690–6699. doi:10.1109/TII.2021.3121770.

Xiu, X., Pan, L., Yang, Y., Liu, W., 2024. Efficient and fast joint sparse constrained canonical correlation analysis for fault detection. IEEE Transactions on Neural Networks and Learning Systems 35, 4153–4163. doi:10.1109/TNNLS.2022.3201881.