Supporting Information

Estimating Missing Unit Process Data in Life Cycle Assessment Using a Similarity-based Approach

Ping Hou[1,2], Jiarui Cai[1], Shen Qu[1], Ming Xu[1,3]*

[1]School for Environment and Sustainability, University of Michigan, Ann Arbor, MI, 48109, USA.
[2]Michigan Institute for Computational Discovery & Engineering, University of Michigan, Ann Arbor, MI, 48104, USA.
[3]Department of Civil and Environmental Engineering, University of Michigan, Ann Arbor, MI, 48109, USA.
*Correspondence to: mingxu@umich.edu.

Number of pages: **7**
Number of figures: **N/A**
Number of Tables: **3**

Text S1. Distance measuring methods tested

A distance metric is a function that defines a distance between two observations. Given an *mx*-by-*n* data matrix *X*, which is treated as *mx* (1-by-*n*) row vectors $x_1, x_2, \ldots, x_{mx}$, and an *my*-by-*n* data matrix *Y*, which is treated as *my* (1-by-*n*) row vectors $y_1, y_2, \ldots, y_{my}$, the various distances between the vector x_s and y_t are defined as follows:

1. Euclidean distance

The Euclidean distance is the straight-line distance between two points in Euclidean space. The Euclidean distance is a special case of the Minkowski distance, where *q*=2.

$$d_{st} = (x_s - y_t)(x_s - y_t)^T = \left(\sum_{i=1}^{n} |x_{si} - y_{ti}|^2\right)^{1/2}$$

2. Standardized Euclidean distance

$$d_{st} = (x_s - y_t)V^{-1}(x_s - y_t)^T$$

where *V* is the *n*-by-*n* diagonal matrix whose *j*th diagonal element is $(S(j)^2)$, where *S* is a vector of scaling factors for each dimension.

3. City block distance (Manhattan distance)

The city block distance between two points is the sum of the absolute differences of their coordinates. The city block distance is a special case of the Minkowski distance, where *q*=1.

$$d_{st} = \sum_{i=1}^{n} |x_{si} - y_{ti}|$$

4. Chebychev distance

The Chebychev distance is a special case of the MInkowski distance, where *q*=∞.

$$d_{st} = \max_i\{|x_{si} - y_{ti}|\}$$

5. Cosine distance

$$d_{st} = \left(1 - \frac{x_s {y_t}^T}{\sqrt{(x_s {x_s}^T)(y_t {y_t}^T)}}\right)$$

6. Correlation distance

$$d_{st} = 1 - \frac{(x_s - \overline{x_s})(y_t - \overline{y_t})^T}{\sqrt{(x_s - \overline{x_s})(x_s - \overline{x_s})^T}\sqrt{(y_t - \overline{y_t})(y_t - \overline{y_t})^T}}$$

where

$$\bar{x_s} = \frac{1}{n}\sum_i x_{si}$$

$$y_t = \frac{1}{n}\sum_i y_{ti}$$

7. Hamming distance

$$d_{st} = (\#(x_{si} \neq y_{ti})/n)$$

8. Jaccard distance

$$d_{st} = \frac{\#[(x_{si} \neq y_{ti}) \cap (x_{si} \neq 0) \cap (y_{ti} \neq 0)]}{\#[(x_{si} \neq 0) \cup (y_{ti} \neq 0)]}$$

9. Spearman distance

$$d_{st} = 1 - \frac{(r_s - \bar{r_s})(r_t - \bar{r_t})^T}{\sqrt{(r_s - \bar{r_s})(r_s - \bar{r_s})^T}\sqrt{(r_t - \bar{r_t})(r_t - \bar{r_t})^T}}$$

where

r_{si} is the rank of x_{si} taken over $x_{1i}, x_{2i}, \ldots, x_{mx,i}$

r_{ti} is the rank of y_{ti} taken over $y_{1i}, y_{2i}, \ldots, y_{my,i}$

r_s and r_t are the coordinate-wise rank vectors of x_s and y_t, i.e., $r_s = (r_{s1}, r_{s2}, \ldots, r_{sn})$ and $r_t = (r_{t1}, r_{t2}, \ldots, r_{tn})$

$$\bar{r_s} = \frac{1}{n}\sum_i r_{si} = \frac{(n+1)}{2}$$

$$\bar{r_t} = \frac{1}{n}\sum_i r_{ti} = \frac{(n+1)}{2}$$

10. Minkowski distance qq

$$d_{st} = \left(\sum_{i=1}^{n} |x_{si} - y_{ti}|^q\right)^{1/q}$$

For the special case of q=1, the Minkowski distance gives the city block distance. For the special case of q=2, the Minkowski distance gives the Euclidean distance. For the special case of q=∞, the Minkowski distance gives the Chebychev distance.

Table S1. Average Mean Percentage Error (MPE) when missing 1% data calculated using different distance functions

Method	Definition	MPE
Eculidean distance	The straight-line distance between two points in Euclidean space. equivalent to Minkowsk distance when q=2	80.99%
Standardized Eculidean distance	Each coordinate difference between observations is scaled by dividing by the corresponding element of the standard deviation	105.96%
City block distance	Also called Manhattan distance, the sum of the absolute differences of their coordinates, equivalent to Minkowski distance when q=1	80.90%
Chebychev distance	Maximum coordinate difference, equivalent to Minkowski distance when q=∞	86.08%
Cosince distance	One minus the cosine of the included angle between points	1393.67%
Correlation distance	One minus the correlation between points	1208.16%
Hamming distance	Percentage of coordinates that differ	100.00%
Jaccard distance	One minus the Jaccard coefficient, which is the percentage of nonzero coordinates that differ	24972.57%
Spearman	One minus the sample Spearman's rank correlation between observations	1194.69%

Note: Minkowski distance is a metric which can be considered as a generalization of the Euclidean distance (q=2), the City block distance (q=1), and the Chebychev distance (q=∞). When q is smaller the MPE is getting smaller. Therefore, through adjusting the q value in Minkowski distance, we can find the best q value that can have the best estimation performance.

Text S2. Normalization

Normalization sometimes is needed to represent data in similar order of magnitudes. There are different ways of normalization, such as Z-score and min-max. In this paper, we define another way of matrix normalization. We first pre-multiplied original matrix A by a diagonal matrix, L, in which the diagonal elements are the inverse of the maximum values in each row of matrix A.

$$L = \begin{bmatrix} \frac{1}{\max a_{1,:}} & 0 & 0 \\ 0 & \dots & 0 \\ 0 & 0 & \frac{1}{\max a_{n,:}} \end{bmatrix}$$

then,

$$B = LA = \begin{bmatrix} \frac{1}{\max a_{1,:}} & 0 & 0 \\ 0 & \dots & 0 \\ 0 & 0 & \frac{1}{\max a_{n,:}} \end{bmatrix} \begin{bmatrix} a_{11} & \dots & a_{n1} \\ \dots & \dots & \dots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} = \begin{bmatrix} \frac{a_{11}}{\max a_{1,:}} & \dots & \frac{a_{n1}}{\max a_{1,:}} \\ \dots & \dots & \dots \\ \frac{a_{m1}}{\max a_{n,:}} & \dots & \frac{a_{mn}}{\max a_{n,:}} \end{bmatrix}$$

We then post-multiplied B by R, a diagonal matrix in which the diagonal elements are the inverse of the maximum values in each column of matrix B.

$$R = \begin{bmatrix} \frac{1}{\max b_{:,1}} & 0 & 0 \\ 0 & \dots & 0 \\ 0 & 0 & \frac{1}{\max b_{:,m}} \end{bmatrix}$$

then,

$$C = BR = \begin{bmatrix} b_{11} & \dots & b_{n1} \\ \dots & \dots & \dots \\ b_{m1} & \dots & b_{mn} \end{bmatrix} \begin{bmatrix} \frac{1}{\max b_{:,1}} & 0 & 0 \\ 0 & \dots & 0 \\ 0 & 0 & \frac{1}{\max b_{:,m}} \end{bmatrix} = \begin{bmatrix} \frac{b_{11}}{\max b_{:,1}} & \dots & \frac{b_{1n}}{\max b_{:,m}} \\ \dots & \dots & \dots \\ \frac{b_{m1}}{\max b_{:,1}} & \dots & \frac{b_{mn}}{\max b_{:,m}} \end{bmatrix}$$

After the matrix normalization, the resulting matrix C has the maximum value as 1 and the minimum value as 0 for all the rows and columns. By doing so, the values in the whole matrix are normalized in the similar orders of magnitude. Another benefit of such transformation is that MPEs calculated based on the matrix C are the same as those based on the original matrix A since this transformation is basically the multiplication of constant values, which can be recorded for denormalizing the data after estimation. The row normalization is to reduce the order of magnitude difference in intermediate and elementary flows, which is equivalent to converting the units of the flows. While the column normalization is to remove the scale difference of the process units, which is equivalent to converting the functional units.

Based on this definition, we tried three different normalization strategies: 1) normalization based on the complete matrix; 2) normalization based on the training set to avoid introducing future information in the test set; and 3) without normalization. Table S2 shows the comparison of the average MPE using the three different strategies.

In general, estimation with normalization based on the training set has the highest average MPE. This is because, in ecoinvent, the order of the magnitude of the test data can be very different from that of the training data. If there are magnitude of difference between the test data and the training data, test data after normalization can be extremely small or large, while the maximum of the normalized training data is always 1. Therefore, normalizing the test data based on the information from the training data can be problematic and skew the test data.

Much to our surprise, estimation without normalization has the lowest average MPE. We believe this is because normalization, while making the data more regular, can actually lose information that might be useful in the estimation.

Therefore, based on the above comparison and analysis, we did the estimation without normalization.

Table S2 Average Mean Percentage Error (MPE) using three different normalization strategies

Average MPE	Normalization based on the whole matrix	Normalization based on the training set	Without normalization
1% missing	3.14×10^{-14}%	17.95%	2.09×10^{-13}%
5% missing	13.43%	100%	2.85×10^{-12}%
10% missing	59.45%	100%	39.32%
20% missing	97.55%	100%	91.39%

Text S3.

An example of estimating 1% data missing in a process named “petrol, unleaded//[RER] market for petrol, unleaded” with MPE=2.94%. Details see the supporting Excel file sheet 1 - sheet 3.

Text S4. Computational time

We examined the computational time required to implement this method. We used 2.67 GHz Intel Xeon X5650 processors with 4GB RAM memory. In the case of missing 1% data with one processor, it takes approximately 25 minutes to estimate the missing data for all processes. The estimation itself takes the greatest computational time (94.1%), while other procedures including the similarity calculation and data preparation only need approximately 5.9% of the computational time. The estimation step dominates the computational time because it involves many matrix multiplications, i.e., 2,545 times of matrix multiplication for estimating missing data in each process, in total 6.5 million times of matrix multiplication. When more data are missing, the time spent on similarity calculation would decrease since the dataset for calculating similarity is smaller. However, the time for estimation would increase significantly since the dataset for estimation becomes larger.

In order to improve the computational efficiency, we used high-performance computing (HPC) to run the calculation in parallel. We used 10 processors (2.67 GHz Intel Xeon X5650 processors with 4GB RAM memory) regarding to 10 different q to calculate simultaneously. **Table S2** shows the computational time required for calculating the whole database. Overall the computational resource needed for implementing our method in ecoinvent 3.1 is manageable. For the case study, it only requires 3 minutes with one processor to run the calculation. More time might be needed to find the indexes of the intermediate flows and elementary flows in the unit process matrix.

Table S3. Computational time required for completing the estimation

Missing data scenarios	Time required
1% missing	1.25 hours
5% missing	5.62 hours
10% missing	11.94 hours
20% missing	57.22 hours

Text S5. Case study

A case study of process “Diesel, combusted in industrial boiler” in US LCI database. Details see the supporting Excel file sheet 4.