University of Michigan

Match Probabilities

Authors:
Xiaomeng Du
Timothy NeCamp
Dana Turjeman

Supervisor:
Prof. Jian Kang

January 30, 2017

1 Introduction

1.1 Background

Utilizing websites and mobile apps, many retail companies collect detailed information on their online users[1]. Despite the vast amount of data that is widely available to these firms regarding users' online activities, demographics, and purchases, there are certain categories of information (e.g., attitudinal, behavioral, or reaction to hypotheticals) that firms would also like to collect.

In order to gather such information, firms use surveys to get valuable and clear measures regarding the population of interest. Such surveys allow both marketers and researchers to make statistical inferences for both managerial and academic purposes. That is, there is a pressing and increasing need to leverage the big amounts of information, along with data from surveys that include a variety of measures.

1.1.1 Collecting Sensitive Information via a Survey

A central challenge is that statistical inferences obtained from a survey might be biased if respondents believe their identity might be revealed. In particular, people may be dishonest when requested to provide sensitive information, such as criminal activity, sexual history, or even income. It has been shown (See [Ong and Weiss, 2000], for a review) that assuring complete *confidentiality* through promising not to reveal any identifying information would be enough to ensure that reliable measures are obtained. However, with an increasing awareness to data breaches, people might be reluctant to respond or might respond in a selective or deliberately inaccurate manner if they are not assured of complete *anonymity* (i.e., there is no way to identify them given available data and information technology). Therefore, conducting an anonymous survey can be a better option if there is a need in collecting sensitive information.

1.1.2 Problem in Fusing Anonymous Data

The problem arises when the marketer or researcher (hereinafter - user) wants to fuse the vast amount of data she already has (hereinafter - big data base), with data from an anonymous survey. Because the big data base has identifying information (e.g. social security numbers, addresses), a one to one match between survey respondents and big data individuals would violate anonymity. Hence, in order to assure that the data fusion process will not reveal the identity of any one survey respondent S_i, to any individual B_j from the big data base, the user needs to know the match probabilities. This is where our method comes in hand.

1.2 Goal

Given a non-anonymous database and anonymous survey, our package will return a table of matching probabilities; a match probability is the probability that a respondent from the anonymous survey is identified as an individual from the larger database. In other words, **our goal is to develop a clear measure of the probability that any one person in the survey can be identified as being any specific person from the database**.

1.3 Literature Review

There has been much previous work on data fusion and statistical matching. Most methods, such as [Gönen and Margolin, 2014], utilize notions of distances to match data sets and avoid model-based approaches. This is because these methods tend to only be concerned with finding the correct match (i.e. minimizing the distance), as opposed to analyzing the matching probabilities. For the problem at hand, calculating the match probabilities, a model-based approach is necessary. Our model assumptions are similar to those of [Gilula et al., 2006], however, our goals are different, and

[1] http://www.emarketer.com/Article/How-Marketers-Using-Data/1012360

hence new methodology needed to be developed.

1.4 Novelty

The ideas behind the model were inspired by [Gilula et al., 2006]. However, the calculation of probabilities, as presented here, is completely novel. We are unaware of any previous literature or software packages that have obtained matching probabilities as such. We suspect that, due to an increasing awareness to data breaches over the past years, there is a need for assurance of anonymity, as opposed to mere confidentiality. Additionally, to the best of our knowledge, there were no pre existing software packages implementing the method in [Breiman and Friedman, 1997], hence, the calculation of the parameter estimates in the software package we present here is also novel.

2 Method

2.1 Notation

Before stating the algorithm it is necessary to define notation. N_S is the number of survey respondents, N_B is the number of big data base individuals, $V_{B_{qx1}}$ are the variables that are available only in the big data base, while $V_{C_{px1}}$ are the variables common in both the survey and big data base (they are further subcategorized into those within the survey, V_{C_S}, and those within the big data base, V_{C_B}, as their granulation may differ - for example, age can be presented as *data of birth* in the big data base, and as *age in years*, in the anonymous survey). V_S would be the variables that only pertain to the survey, however, those are not utilized in our algorithm.

2.2 Algorithm

We make a linear model assumptions, specifically:

$$V_B = \theta^T V_C + \epsilon \quad \text{where: } V_{B_{qx1}}, \theta_{pxq}, V_{C_{px1}}, \text{ with } \epsilon \sim MVN(0, \Sigma_{qxq})$$

The algorithm to calculate the probability that a survey respondent i, denoted S_i, will be matched with individual j from the big data base, denoted B_j, is as follows:

1. Compute parameter estimates to estimate likelihood $f(V_B|V_C, \theta, \Sigma)$ using V_B and V_{C_B}:
   (a) Calculate $\hat{\theta}_{OLS}$ by performing OLS on each column of V_B.
   (b) Calculate B^*, following [Breiman and Friedman, 1997] and using eigenvalue and eigenvector decomposition of
   $$\hat{E}(V_B V_B^T)^{-1} \hat{E}(V_B V_{C_B}^T) \hat{E}(V_{C_B} V_{C_B}^T) \hat{E}(V_{C_B} V_B^T)$$
   (c) Calculate $\hat{\theta}^* = \hat{\theta}_{OLS} B^*$
   (d) Calculate $\hat{\Sigma} = \hat{cov}(V_B - \hat{V_B})$ where $\hat{V_B} = \hat{\theta}^{*T} V_{C_B}$
2. Compute probability of matching with estimated likelihood:
   (a) Calculate $P(S_i = B_j) = \frac{f(V_{B,j}|V_{C_{S_i}}\ \hat{\theta}^*\ \hat{\Sigma})}{\sum_{k=1}^{N} f(V_{B,k}|V_{C_{S_i}}\ \hat{\theta}^*\ \hat{\Sigma})}$ for all i and j.
   (b) Return matrix M with $M_{i,j} = P(S_i = B_j)$

There are several important aspects of this algorithm. Part 1 of the algorithm is used to estimate the likelihood of V_B given V_C. Note that though we make a linear assumption, our response V_B, is

a vector making our situation different from typical OLS. One strategy to overcome this difficulty would be to allow each column of $\hat{\theta}$ to be what would be found from performing separate linear regressions on each corresponding dimension of V_B. Doing so, however, would ignore relationships between dimensions of V_B. Hence, as done in [Breiman and Friedman, 1997], we calculate an additional coefficient averaging matrix, B^*, which accounts for information between dimensions. Using $\hat{\theta}_{OLS}$ from performing separate linear regressions and B^*, we are able to obtain a better (lower standard error) estimate of θ.

In Part 2 of the algorithm, we are able to calculate the probability of matching a survey correspondent to an individual from the big data base by allowing the probability to be proportional to the likelihood. Note that this would be the same probability obtained by assuming the data arises from a mixture of Gaussians (similar to Linear Discriminant Analysis with N_s classes and no aprior preference towards any class). Also, our probabilities behave as expected, the higher the likelihood, the higher the probability of matching.

2.3 Time/Space Complexity Analysis

Online data-bases can be quite large. This may create a problem with use of our method both in terms of computational time and storage. Though our package is an R package, we were able to exploit Rcpp to help alleviate potential computation time and storage issues.

When doing matrix operations, such as centering our data, R creates multiple copies of the matrix each time it is altered. If a matrix is large, this can eat up a large amount of storage. By doing most matrix operations through Rcpp functions, we are able to use a reference to the matrix and alter it without creating a new copy. This is especially helpful when the old matrix is no longer needed.

In addition to helping with storage, coding the most computationally expensive aspects of our algorithm (i.e. performing regression, eigen decomposition to find B^*, iterating over multiple for-loops to find every probability for each S_i and B_j) saved significant time. By looking at the time comparison between our package (which uses Rcpp) and our same algorithm coded using only R functions in Figure 1, we see a significant improvement in speed. We also note that our computational time seems to grow linearly as N_B grows.

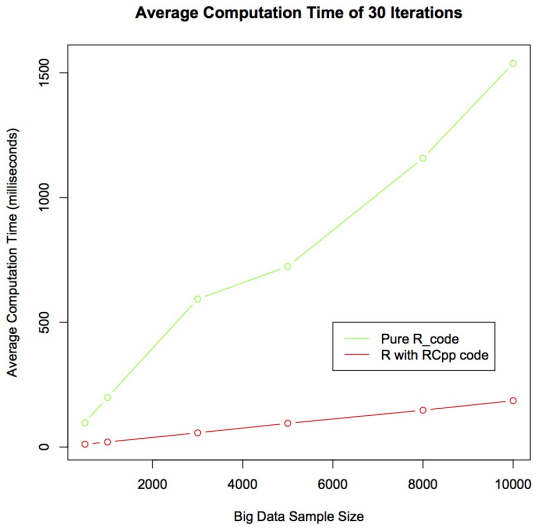


Figure 1: Timing Comparisons

3 Examples and Simulation Studies

Our package contains four functions. The main function calculates the matching probability table, and the rest of the functions use this table to generate summaries of the table. Note also that our package necessitates installation of Rcpp and RcppEigen.

3.1 Illustration of Software Usage

3.1.1 Main Functionality - Calculating Match Probabilities Matrix

The main method has the form: `match_probability(Matrix big_data, Matrix survey_data, Matrix common_var_index)`. The inputs are two datasets with common variables (`big_data`,

`survey_data`) and an $n \times 2$ matrix (`common_var_index`) that contains the matching index of common variables of the two datasets.

Figure 2 illustrates a possible structure of the data. The dark-blue section of user ids stands for the individuals that are in the big_data. The dark-green section of user ids stands for the individuals that are in the survey_data. There is no overlap between the two datasets due to the anonymity of the survey, however, a sample of survey respondents can be a subsample of the big_data given. The purpose of our function is to calculate the matching probabilities of each of the individuals in survey_data to each of the individuals in big_data.

	Bigger Data Base Variables					Common Variables			Survey Variables		
user id	zip	bodytype	messages/month	weight	height (cm)	dob/age	gender	ethnicity	income (K$)	mrg time	prev_affr
B_1	81233	6	204	139	196	7-Nov-78	M	1			
B_2	41229	2	219	174	167	9-Apr-78	M	3			
B_3	92408	1	148	109	168	26-Sep-73	F	2			
B_4	76472	5	62	162	177	16-Mar-78	M	1			
B_5	29879	2	68	119	173	12-Dec-57	F	1			
B_6	67589	8	148	150	185	4-May-62	M	3			
B_7	11058	4	187	144	169	18-Apr-64	F	1			
B_8	85471	4	261	144	192	11-Nov-91	M	2			
B_9	73367	5	263	181	167	1-Apr-83	M	1			
:											
B_N_B	87397	8	30	125	170	24-Dec-48	M	3			
S_1						35-39	M	1	177	0	5
S_2						20-24	M	2	93	14	5
S_3						40-44	F	3	81	16	5
:											
S_N_S						More than 70	M	1	159	28	4

Figure 2: Illustration of the data

There are two kinds of variables in big_data: those that are also in the survey_data (the light green section), and those that are unique (the light blue section). We will use all the unique variables as responses and all the common variables as predictors in our method. This may require the users to clean the dataset ahead to make sure only the wanted variables are included and are formatted correctly (i.e. numeric). For example, in this case, we want to delete the first column(zip) of the big dataset, and change the measurement of the age variable. As for survey_data, there are two kinds of variables as well: the common variables (the light green section), and the unique variables (the red section). We will only use the common variables in survey_data. The common_var_index of this example is as following:

big_data	survey_data
5	1
6	2
7	3

The first and second columns are the index of common variables in big_data, V_{C_B}, and survey_data, V_{C_S}, respectively . Each row of the table links together the big_data variable with its corresponding survey_data variable. The number of columns is two, while the number of rows is the total number of common variables. The output is a matching probability table with the number of columns equal to N_B, and number of rows equals to N_S. Each row contains the probability of the individual in survey_data being matched with the individuals in big_data. To demonstrate, the figure below is an example of running the function with a simulated dataset. We have 10,000 rows in big_data, and 50 rows in survey_data. There are 10 common variables, and 3 response variables. Thus the common_var_index has 10 rows. Part of the matching probability table is shown as follows:

	big_individual_1	big_individual_2	big_individual_3	big_individual_4
survey_respondent_1	0.031715583	2.03E-20	0	8.6E-98
survey_respondent_2	7.99E-18	0.058655739	0	4.47E-34
survey_respondent_3	0	0	0	0

The whole table contains 50 rows and 10000 columns. Each cell contains the matching probability of one individual in survey_data to one individual in big_data. For example, cell (1,1) contains $P(S_i = B_j)$=0.0317.

3.1.2 Illustration and Summarizing Functions

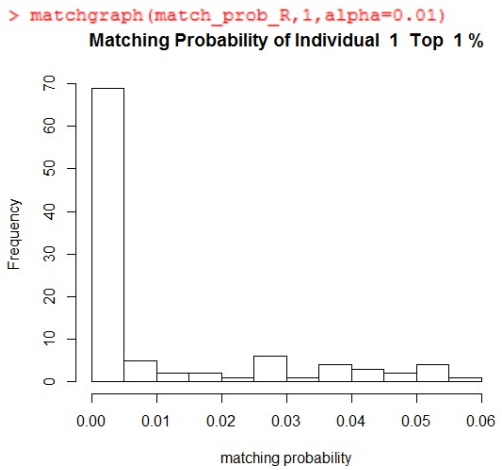
The other functions contained in the package provide summaries of the probability matrix.

1. Function `topmatch(probability_matrix,survey_ind,n=5)`

   This function returns the top n matching probability of the x'th individual in survey_data. Inputs: probability_matrix: matching probability table, survey_ind: index of row number of the desired individual, n: the number of top probabilities, by default equals to 5. Example output:

```
> topmatch(match_prob_R,1)
[1] 0.05594499 0.05448572 0.05388017 0.05273519 0.05163958
```

2. Function `matchgraph(proability_matrix,survey_ind,min_prob=NULL,upper_quantile=NULL)`
   This function returns the histogram of matching probabilities of a certain survey individual. Inputs: probability_matrix: matching probability table, survey_ind: index of row number of the desired individual, min_prob: all probabilities that are larger than it will be counted. If min_prob = NULL, all probabilities will be counted, upper_quantile: value must be between 0 and 1. if assigned, the probabilities above the upper quantile will be counted. Note that min_prob and upper_quantile cannot be assigned values at the same time. Example output and plot:



   This output is the histogram showing the distribution of the top 1% of all matching probabilities for survey respondent 1.

3. Function `overallmatch(probability_matrix,min_prob=NULL,n=10)`

This function shows the overall matching probabilities and users can use it to decide whether there is a high chance of matching. Inputs: probability_matrix: matching probability table, min_prob: all probabilities that are larger than it will be counted. If min_prob=NULL, all probabilities will be counted, n: the number of top probabilities, by default equals to 10. Example output:

```
> overallmatch(match_prob_R)
 [1] 0.4233608 0.3053277 0.2681727 0.2681137 0.2598611 0.1757246 0.1723406
 [8] 0.1716733 0.1674603 0.1630427
```

This output shows the top 10 matching probabilities over all survey individuals. It can tell us the best matches of the two datasets. Here the biggest matching probability is 0.423.

3.2 Advantages

The goal of our package is to provide a way to calculate the matching probability table of two datasets that share common variables. We have successfully achieved this goal and improved time efficiency by using `c++` functions to accelerate the calculation of eigenvalues, eigenvectors and regression coefficients. We also offer other methods of generating output, including visualization of the matching probability distribution of each individual. We also create a way of calculating matching probabilities by applying the procedure of [Breiman and Friedman, 1997].

4 Summary

Overall, our package successfully implements a novel approach to data fusion. The package can be used in order to calculate matching probabilities and also provides ways to summarize and visualize the results. Hence, the goal of our package has been achieved.

4.1 Limitations and Future Work

There are some limitations in our current implementation. For one, our algorithm can only be used if the variables of interest are continuous. We would like to extend our algorithm to work with other data types and distributions. Additionally, our algorithm is based on some model assumptions we hope to relax in the future. For example, allowing the relationship between V_B and V_C to go beyond linear would be beneficial. Lastly, our algorithm does not incorporate any of the variables that are included only in the survey (V_S). We assume independence between V_B and V_S given V_C. This assumption may not be valid and we would like to relax it in future projects.

References

[Breiman and Friedman, 1997] Breiman, L. and Friedman, J. H. (1997). Predicting multivariate responses in multiple linear regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 59(1):3–54.

[Gilula et al., 2006] Gilula, Z., McCulloch, R. E., and Rossi, P. E. (2006). A direct approach to data fusion. *Journal of Marketing Research*, 43(1):73–83.

[Gönen and Margolin, 2014] Gönen, M. and Margolin, A. A. (2014). Localized data fusion for kernel k-means clustering with application to cancer biology. In *Advances in Neural Information Processing Systems*, pages 1305–1313.

[Ong and Weiss, 2000] Ong, A. D. and Weiss, D. J. (2000). The impact of anonymity on responses to sensitive questions1. *Journal of Applied Social Psychology*, 30(8):1691–1708.