Proposal for FORTE 2021-2027
An algorithm to match identities from two heterogeneous data sets

Laurent Bergé

January 5, 2022

Abstract

This document illustrates the strategy to deal with the matching of Patstat inventors to persons from the Swedish national registry. After a brief presentation of the author's experience, the three main steps of the algorithm are detailed. The document ends with a full example of the outcome of the algorithm and finally reports the expected output of this proposal.

Contents

1 Presentation of the author and brief description of the algorithm

I am an associate professor of economics at the university of Bordeaux working in the field of the economics of innovation. My work is empirical and involves working with large data sets, in particular I have a good knowledge of the Patstat data since my research is often linked to patents. My background also includes mathematics and statistics : I have a bachelor in pure mathematics and spent 7 months in a statistics postdoc at the university Paris-Descartes. This implies that I have always been at ease with programming and developing statistical algorithms. I now present some relevant experience before introducing the algorithm to match the data sets of the FORTES project.

1.1 Relevant experience

Relevant programming experience. I have over 10 years of experience in developing R packages. In total, I am the author 7 R packages, one of which tied to a patent, and 6 others open source. They cover statistical methods, big data management, graphics and econometrics. In particular, I have created the package named fixest, which performs fixed-effect econometrics estimations and is currently the fastest one across platforms. Although this package is an R package, the core of the algorithm is written in C++ which, since it is a low level language, is very fast. From this experience, I have built a good understanding of how to optimize code, both for speed and memory usage. I will mobilize these skills to make the algorithm linked to FORTES as fast and memory efficient as possible.

Relevant data science experience. I have extensive experience in dealing with the problem of matching (potentially large) data sets from heterogenous sources. Recently I have been working to match the following data sets which are relevant to the current project:

- Microsoft Academic Graph (MAG) data, which is a large data base on publications, to Ph.D. data granted from US institutions. It involved matching 1,572,328 MAG authors to 930,496 Ph.D. recipients.
- Patent data from the EPO from French assignees to companies from the French national registry of companies. It involved matching 249,531 patents to 8,741,085 French companies.
- US patent data from the USPTO to LinkedIn, manually extracted, data. It involved matching 18,066 LinkedIn profiles to 1,913,545 US and Canada inventors.

One important lesson I have drawn from my experience is that *no algorithm, however sophisticated, can create information.* The most important step is then to extract relevant information from the set of information we observe, and this involves dealing with

numerous yet important details, often specific to the problem at hand. Many algorithms will vary on that front.

1.2 The algorithm

The objective of the algorithm is to match inventors from Patstat patents to persons from the Swedish national registry (and later extended to other countries). The algorithm offered to deal with this problem has three main steps:

1. match on names to obtain inventor-person pairs who are potentially the same person (Section 2),

2. create informative bilateral variables for each inventor-person pair: these variables will help decide which pair is a true match (Section 3),

3. run a Bayesian algorithm to categorize all inventor-person pairs into two groups: one group of true matches and one of wrong matches (Section 4).

The outcome of the algorithm is: a) a probability for each pair to be a true match, and b) the distribution of the covariates for each of the two groups. The last point is important since it is what makes the algorithm very transparent: at the blink of an eye it is possible to understand the classification.

Related literature. In the context of patents, many algorithms have been developed to disambiguate the inventors. For instance Trajtenberg et al. (2009) , Raffo and Lhuillery (2009), Carayol and Cassi (2009), and Li et al. (2014) provide different methodologies to disambiguate inventors, with applications to USPTO or EPO patents. The task of disambiguation, although sharing similar traits, is different from the task of matching two data sets.[1]

On the task of matching two data sets, a recent article from Abramitzky et al. (2021) review the pros and cons of automated methods to link data bases. The current algorithm has some flavor of their algorithm but it differs in many details.

Outline. The next three sections describe the three main steps of the algorithm. Throughout, the MAG-PhD matching previously presented is used to illustrate the algorithm. Section 5 provides an example of final outcome of the algorithm. Section 6 ends by reporting the expected output from this proposal.

[1] In the task of disambiguation there is only one data set, and we can see it as the matching of a data set with itself.

Table 1: Example of raw name fields values from the MAG and PhD data sets.

Raw name fields	
PhD data	MAG data
kadlubar, fred frank	o a plumb
tarng, ming-yueh	rachel g fruchter
halaby, charles nicholas	frank holmquist
mitchell, owen robert, jr	i theodore landau
smith, cornelius c	richard b du boff
de la fuente, francisco arturo	martin j van der maaten

2 Matching on names

Names are special: they are the major blocking factor. In other words, if the names are different,[2] then the persons are different, no matter the value of any other variable. Indeed, even if two identities report working in the same company and living in the same building, if the two have different names, then we can confidently consider that these identities in fact correspond to two different persons. The reverse is not true.[3] Since names are extremely important, it is critical to extract the most information out of the raw names to ensure that we don't miss potential matches and we don't include wrong matches.

I will illustrate this section with an existing project where I merged Microsoft Academic Graph (MAG) scientists to US Ph.D. data. The name information was inside text fields that took a different form for each data set.

2.1 Cleaning

The first, and essential, step is to extract the names out of the raw name fields. Table 1 reports a sample of raw name fields from the two data sets. As we can see, the formatting is very different between these two data sets. The objective is then to extract the name information and harmonize the formatting between the two.

Matching on full name strings should not be done. Before coming to the cleaning procedure, the first lesson that we can get is that one should never use the raw name fields to perform the matching, for instance using a text-based distance measure like Jaro-Winckler. Indeed, think of the following example:

Ex.	Name 1	Name 2	Jaro-Winkler
1	john addison lambeth	john lambeth	low
2	john a lambeth	john b lambeth	high

[2]Note that the names need not be identical to be matched, the strategy for name mathching is developed in Section 2.2.

[3]If two identities have the same name, even if they don't have any other variable in common, they can refer the same person–more information is needed to really decide.

In the first example, these two names might well refer to the same person, the missing second name does not mean that the author does not have one and it may well be 'Addison'. But here the Jaro-Winkler will score low because the strings are very different.

In the second example, the two persons are different for sure since mistakes on the second name initial never happens.[4] However since it just corresponds to one misspell, the Jaro-Winkler will score high.

To circumvent this problem I will try extract the maximum information out of the string fields, namely: the first, the second, the third and the last name. And will match separately on each of these fields (details on the merging strategy are in Section 2.2).

Extracting the most out of names. I have developed an algorithm to extract the names out of the raw fields. After cleaning the name field appropriately, here are some specificities of the algorithm:

- it catches the particles, so they are assigned to the last name appropriately,
- it catches titles (e.g. prof dr for the Germans, jr, ii, etc), so they are not considered as names,[5]
- it guesses whether a word is a first or a last name–this is especially useful for women who often have several names. In case of doubt the author is duplicated with this word used consecutively as a first and a last name. When several last names are present, the author is duplicated for each of the last names and their combination.[6]

Table 2 reports the results of this algorithm.

Note that in the previous example, the name extraction in the PhD data set was easier and less error-prone since the last name was separated with a comma. On the other hand, the names from the MAG data set required to guess the last names, when several were present, hence generating more potential mismatches – these potential mismatches will then be resolved in the last step of the algorithm, in Section 4.

Cleaning specific to the Swedish case. The Swedish alphabet contains special characters that are not ASCII, like ö, ä or å. Instead of containing these characters, Swedish inventors names can be spelled out in patents in an ASCII-form with varying translations of these characters: ex: Björn can be transformed into Bjorn or Bjoern, Åmål into Amal or Aamaal, etc. To cope with this spelling problem, one way is to enforce a strong conversion of these letters, by transforming all "ö" into "o" and all "oe" into "o" to ensure that "ö" translates uniformly into "o". Taking the case of changing "oe" into "o", the cost of the transformation is that valid "oe" spellings would be mistaken for "ö" and hence

[4] Of course mistakes on the second name initial can happen, but the rate is negligible;

[5] These titles may in turn be or not be used for matching. I tend to consider that they do no provide enough relevant information to be used for matching. But they can be used.

[6] This point is especially relevant for the FORTES study, focusing on women.

Table 2: Example of results of the algorithm extracting names.

		Extracted information				
Data set	Raw	First	Second	Third	Last	Extra
MAG	sandra frazier byrne	sandra			frazier byrne	
	–	sandra			byrne frazier	
	–	sandra			frazier	
	–	sandra			byrne	
	gwen van servellen	gwen			van servellen	
	m david egger	m	david		egger	
	donald richard ort	donald	richard		ort	
	mary jo puckett cliatt	mary	jo		puckett cliatt	
	–	mary	jo		cliatt puckett	
	–	mary	jo		cliatt	
	–	mary	jo		puckett	
	pierre frederic emmanuel monsan	pierre	frederic	emmanuel	monsan	
Ph.D.	ashe, warren kelly	warren	kelly		ashe	
	edmonds, james doyle, jr	james	doyle		edmonds	jr

introduce noise in the spelling of the names. In terms of consequences in the algorithm, this means that there will be more individuals that are mistakenly matched.[7] The problem is not so important thanks to the last part of the algorithm which tells apart wrong from true matches.

2.2 Matching

The end product of the previous step is a set of names (first, second, third and last) homogeneous across the two data sets to match. Now the objective is to merge the names from the two data sets to obtain a set of identity pairs which are potential matches. There are two main steps: 1) the matching, 2) the post treatment.

2.2.1 Matching

This matching on names is always done on the largest set of available information. For instance, Laurent R Bergé will never be matched to Laurent Lucien Bergé, but can be matched to i) Laurent Berge, ii) L Bergé or iii) L R Bergé. That is, we use the maximum set of information in the two data sets to decide which matches are plausible or not plausible. I also allow one, but only one, of the items (i.e. first, second, third or last name) to contain one misspell.[8] I don't use the Jaro-Winkler measure, just one misspell and also restrict the two names to have at least the first two letters in common since mistakes on the first two letters are too obvious to really be plausible.

[7] The importance of this problem can be assessed by looking at the frequency of valid "oe" spellings in statistics Sweden data.

[8] One misspell means either: i) an inversion, ii) an addition, or iii) a removal.

Table 3: Match based on the largest set of information available.

Data 1	Data 2	Matched?	Strength
L R Bergé	Laurent Berge	✓	−
L R Bergé	Lucien Berge	✓	−
L R Berge	Laurent Simon Berge		
Laurent Berge	L Robert Berge	✓	−
Laurent Berge	Lucien Robert Berge		
Laurent Berge	Laurent Begre	✓	−
Laurent R Berge	Laurent Berge	✓	+
Laurent Robert Berge	Laurent R Berge	✓	++

Note on fuzzy matching. I don't think increasing the flexibility in terms of misspells is a good idea. In general misspells in names should be the exception and not the rule. It is quite unreasonable to expect that more than 1% of the names are misspelled. Having both a first and last name misspelled at the same time may appear, but at a marginal rate, so should not matter since we would lose at most a few patents (in the FORTES case). Of course it is possible to be "looser" in terms of name matching. But this would come at a dear price. To gain the few patents in which the names were strongly misspelled (having 2+ misspells), we would introduce an enormous amount of false positives, strongly degrading the quality of the matched data set. In the case in which we are interested in precision, and this is an important point to make valid inferences later on in the analysis, then adding too much fuzzy matching incurs a negative trade-off and should be used only sparingly.

From these rules, Table 3 reports exactly how the algorithm works with several examples. We can also see that that the matching strength will differ between the cases, this information can later be used to resolve the matches.

Raw merge: MAG-PhD example. Still using the MAG-PhD matching exercise example, Figure 1 reports how the identities from the two data sets were merged on names. In total, 157 different merges were performed, resulting in a data set of 991,454 observations, linking 244,211 unique MAG-authors to 351,159 unique PhD-authors. As we can see, the majority of the matches are rather low quality: just the first initial and the last name account for 31% of the matches, followed by the initial with a fuzzy matched last name (16%). However, the third category which accounts for 15% of the cases is rather a strong match with full first and last names plus the initial of the second name being identical between the matched MAG and PhD authors.

Matching specific to the Swedish case. It seems to happen with some non negligible frequency that the first given name may not be the one used by the person. For example *Lars Mikael Anders Andersson* may be addressed as *Mikael Andersson* and not Lars Andersson. If ignored, this feature may lead to omit inventors who would possibly be

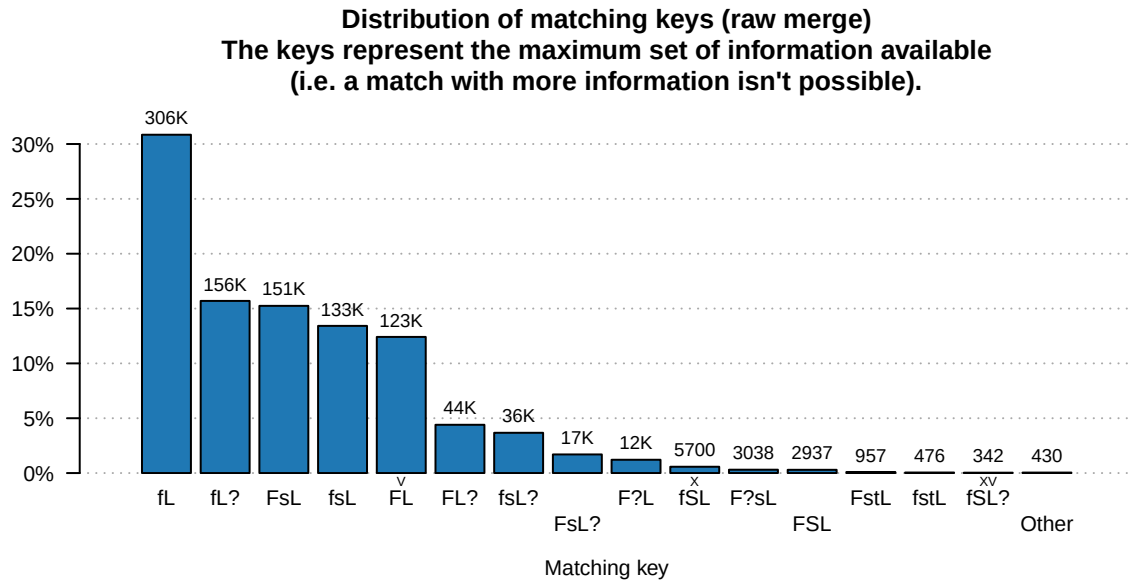

Figure 1: Distribution of merging keys for the initial name merging: MAG-PhD example.

Notes: F/S/T/L stand for first, second, third and last name. Lower case letters means just the initial was used. A question mark after an upper case letter means that the variable to which refers the letter was fuzzy-matched (= one misspell).

Table 4: Meaning of the name-keys.

	Key	Meaning	Key	Meaning	Key	Meaning
First name	F	full	f	initial	F?	fuzzy matched
Second name	S	full	s	initial	S?	fuzzy matched
Third name	T	full	t	initial	T?	fuzzy matched
Last name	L	full			L?	fuzzy matched

true matches. To avoid this issue, in the matching procedure we add an extra step where we would match inventors displaying only one first name to either the second or the third given names in the Swedish registry.

2.2.2 Post-treatment

Out of this raw merge, I apply two post treatments:

1. removing misspells that are not misspells ('natural' misspells),
2. removing weak matches when strictly stronger matches are available.

Removing natural misspells. Many names that could be seen as misspelled are not misspelled, but rather are just different names. In my previous work, I used registry-like information to deduce which names are natural misspells. For example, I used the data on PhDs for which we know by construction that each observation relates to a different person,[9] to find out 'natural misspells': i.e. pairs of first names or last names that would be fuzzy matched although they just refer to different persons. In the context of FORTES, I would use the data from the Swedish national registry[10] to deduce natural misspells.

The procedure is as follows: I merge the data set with itself using the following keys:

- first name + initial of the second name + fuzzy matched last name,
- fuzzy matched first name + initial of the second name + last name.

So this is a rather strong matching requirement. The Table 5 reports, in the case of the PhD data, the top 'natural misspells'. This means that there are 1676 different pairs of persons with the same first name, same middle initial and one is named Johnson and the other Johnston. What does this mean for us? It means that if we allow Johnson to be fuzzy matched to Johnston we will get only wrong matches. Same for John and Joan, etc. In the context of the PhD data, I've taken the cut-off of 40, meaning the top 2% natural misspells, and removed all matches from the previous data base that were fuzzy matched using these top 2% natural misspells. It pruned 107,660 observations.

Of course these natural misspells are country specific so the same exercise has to be performed for Sweden.

Removing weak matches. It is frequent that one identity from one data set can be matched to several identities from the target data set. In this second step, when this is the case, I remove matches that are strictly weaker. Still using the MAG-PhD example, Table 6 reports good and weak matches. A match is removed only if there exists a match

[9] Setting aside the rare persons who have more than one PhD.

[10] In the Swedish national registry data, thanks to person identifiers, we know for sure when different observations (hence different names) refer to different persons.

Table 5: Natural misspells: frequency of persons that would be wrongly matched due to the fuzzy matching on names.

Last names		Freq.	First names		Freq.
johnson	johnston	1676	mary	mark	1337
lin	liu	414	yong	young	1317
kelley	kelly	383	john	joan	1200
cheng	chen	370	john	jon	716
davies	davis	366	roberta	robert	520
clarke	clark	349	ching	chung	415

Table 6: Removing weak matches.

Data 1-author Name	Data 2-Match – Stronger Name	Key	Data 2-Match – Weaker Name	Key
john dauns	dauns, john	FL	dans, j clarice	fL?
morton raban	raban, morton	FL	rabin, morton herbert	FL?
marida hollos	hollos, marida clara	FL	hollow, m kevin, sister	fL?
richard l verrier	verrier, richard leonard	FsL	vernier, richard	FL?
john w mamer	mamer, john willian	FsL	mader, j woodson, jr	fsL?

Notes: Table 4 reports the meaning of the keys.

that is strictly better: for example, if the last name is misspelled and there exists one match on the full last name (no misspell); or when the first name contains only the initial and there exists another match on the full first name. On the MAG-PhD example, there were 242,244 strictly weak matches.

Final merge: MAG-PhD example. After the name matching and the post-processing, we end up with a data set of 641,550 observations linking 242,262 unique MAG authors to 301,008 unique PhD authors.

3 Creating the bilateral variables

At this stage, we have a bilateral data set of person-inventor pairs, all with the same names. Each of these pairs is a potential match, the question is now: which pair is a true match and which is a wrong match? To take that decision we need information, hence we construct a set of bilateral variables which are relevant to discern the matches. In the case of the FORTES project, the relevant information are:

- name matching strength,
- name probability,
- person address,
- company.

There may be more relevant variables from the national registry data set. However, the person address and the company information are the most relevant ones. The name matching strength and name probability, which I will detail below, are useful variables to inform the match but less critical than the address and company.

I now describe some potential strategies to create variables relating to that information.

3.1 Name related variables

Name match strength. By construction, the probability to be a wrong match is higher for names that have been fuzzy matched or matched with only a small set of information. I categorize the matching keys into three groups:[11]

low: fL, fL?, fsL, fsL?, F?L, or FL?.

high: no fuzzy matched in this category. 1) at least one full first, second or third name + at least one initial more. 2) three initials (fstL).

mid: the rest.

The idea of this variable is that there will be much more wrong candidates for matches of low strength and this should be reflected in the decision making. Note that in the context of FORTES, names in which only the initial of the first name is present should be very rare since the name information is usually complete both in the Swedish national registry and in the patent data set. So the low group will be mostly composed of fuzzy matched names.

Name probability. The idea is that there will be much more wrong candidates for very common names (John Johnson) and this should be used in the algorithm. Hence I create the probability of having a given first + last name combination. One way to do so is to extract the unique list of first, second and last names from the Swedish national registry and the patent data set.[12] From this list I construct the probability of having any first name and last name so that p_i^F, resp. p_j^L, represents the probability of having the first name i, resp. last name j. The definition of p_i^F is $p_i^F = n_i^F/N$ with n_i^F the number of observations having first name i and N the total number of observations.

For each person in the patent data set of first name i and last name j, I create the probability of having their name, defined as $p_i^F p_j^L$. I don't go beyond first and last name because the second and third names may not be used for matching while the first and last names are always used. I finally scale $p_i^F p_j^L$ by dividing it by the maximal value (so that the value can be understood as a percentage of the maximal frequency).

This group is then cut in four categories based on the quantiles: low, mid-low, mid-high and high corresponding to the 1st, 2nd, 3rd and 4th quartiles. This variable can be

[11] Table 4 reports the meaning of the keys.

[12] I take the unique to avoid double counting the same persons patent data set.

Table 7: Example of Swedish inventor addresses in Patstat.

Raw inventor addresses from Patstat
itw sverige ab, eura division,as,33373 bredaryd
gruvvagen 25,792 33 mora
vastmannagatan 53,s-113 25 stockholm
huawei technologies sweden ab,skalholtsgatan 9,16440 kista
edlunda 14,185 99 vaxholm

finally combined to the name matching strength to create a categorical variable combining both the name probability and the matching strength information.

3.2 Same address

The address of residence of the inventors and of the persons in the Swedish registry are extremely useful to decide whether two identities are the same.

Cleaning the addresses. The first task is to extract the *true* physical address from the *raw* address fields. Addresses can be formatted in very different ways, may contain misspells or non-related information, and from these we ougth to extract the relevant information: the true address. For instance, Table 7 reports some inventor addreses from Patstat and give a small glimpse of the variety. Another important point is that, to obtain a variable of good quality, the level of precision of the addresss should be at the street and not only at the city. This precision increases the difficulty of this task.

One way to extract the addresses from the raw fields is to *match* the addresses to a registriy of existing addresses (using postal data for instance, and if available). After a first pass extracting the street/number/postal code/city from the addresses,[13] we correct the potentially badly extracted fields by matching the full address to the registry and using all the valid fields. In case of doubt, for instance a mispelled street matched to two possible streets, the addresses could be duplicated. We would end up with a link, for each Patstat and Swedish national registry addresses to a physical address which is uniquely identified.

From there, we would create a variable which tells whether two identities live at the same address. This information is binary in essence: either two persons live at the same address, either not. There is no shade of gray. In practice, this is more complex: for example reporting the address of the employer instead of the residence address is common practice in patents, we need to take that into account.

Missing addresses. We can see in Table 7, a small random sample of inventor addresses, that some addresses are the employer's instead of the employee's. What should

[13]This address extraction can be done with a set of pattern matching steps. I have a routine to deal with Patstat addresses already.

we do in these cases? In practice this means that the address information has been withold and is in effect missing. We should not consider that the addresses are different but simply that we cannot tell from the inventor address where the inventor lives. Considering the missing addresses as different addresses would lead to wrong inference and the algorithm should take that feature explicitly into account.

Multiple addresses. Note that since persons can move and we have that information at our disposal from the Swedish national registry, the full set of a person's addresses should be used to be matched to the inventor's unique address.

The address bilateral variable. To conclude, we will create a "same address" variable at the inventor-person pair which takes the following values:

same: the inventor reports a valid address and it is the same as at least one address of the person in the Swedish national registry.

different: the inventor reports a valid address and it is different from all addresses of the person in the Swedish national registry.

missing: the inventor reports the employer's address.[14]

3.3 Same company

Now we describe the creation of the variable giving whether two identities have been working at the company at the same time. Telling whether two companies are the same is a tricky business. The main problems are that: 1) companies have many name variations, and 2) companies can change name over time.

Sources of information. In Patstat, we assume that the companies in which the inventors are working are the patent assignees. In the case in which there are multiple assignees, we will conservatively assume that the inventor is working at the same time in all the companies listed in the patent.

In the Swedish national registry, we will make use of all the companies in which a person work in a given year.

Creating company identifiers. The first step is to create unique company identifiers. To goal is to attach, to each patent-company and person-company, a company identifier that uniquely identifies companies across the two data sets. How to identify companies? We will apply a matching algorithm similar to this one in which we will make use of the most relevant information as possible. They will include in the case of patents: company

[14]Note that we need an extra step to deduce if an address is that of the employer's.

name, company address, inventors from the company,[15] patent fields. As in the current algorithm first there is a matching on the company names, then bilateral variables are created and we end with a Bayesian algorithm to find out which are the true matches. We end up with a set of company identifiers associated to many identity variations (especially for multi nationals). In a second step, we would merge the companies from the Swedish national registry to the companies identities from the patents. The information used for the matching would be the company names and addresses. The final outcome is a set of unique company identifiers covering the two data sets.

A note on matching company names. Contrary to person names, company names don't really have a compelling structure. There exists many ways to match two company names. In practice, after implementing some cleaning, in particular finding out acronyms or removing non-informative qualifiers, the decision rule to judge that two company names are the same are either: i) one name is fully included in the other with a maximum of one misspell, or ii) the proximity between the two character strings is of 0.9 or higher with the Jaro-Winclker measure. Of course, there can be many alternative ways: for instance one could use word inclusion with a higher weight for rarer words.

Creating the variable. Once we have the set of company identifiers, we create the company distance variable. This variable will also make use of the information on years on both the patents an the Swedish national registry. We define the following values:

perfect match: the two identities work at the same company *at the same time*.

loose match: the two identities work at the same company but not during the same years. The minimum lag in years is 3.[16]

loosest match: the two identities work at the same company but not during the same years. The minimum lag in years is strictly greater than 3.

no match: the two identities never work at the same company.

4 A Bayesian algorithm to tell good and wrong matches apart

This is the final step: from a set of potentially matching person-inventor pairs and a set of informative bilateral variables we now devise an algorithm finding the probability of a pair to be a good match.

Starting from the person-inventor pairs found in Section 2.2, we want to estimate the probability that a pair is a true match, given the distribution of the covariates that

[15] To avoid the risk of using inventors on both sides of the equation, we would use only inventors for whom we are sure they are the same (by using name match + address) to create this variable.

[16] We can choose this number depending on the distribution in the data.

we defined in the previous section. We proceed using a classification algorithm. This algorithm will create two groups of observations that are coherent in terms of observables.

In words, the algorithm does the following. We consider two groups: a group of true matching pairs and another group of wrong matches. First we initialize the algorithm by providing an informed initialization, e.g. where we put all person-inventor pairs who have the same residence address in the same group–this will be the group of true matches. Then the algorithm updates the distribution of all covariates given the groups. From these distributions, it computes the probability of each pair to belong to each group. From these probabilities, the distributions of each group are recomputed; followed by an update of the probabilities to belong to each group. *Et caetera*, until reaching an equilibrium.

The advantage of such an algorithm is that we end up not only with a probability to belong to each group (which is the main focus), but also with the distribution of covariates for each group. This means that we can see in the blink of an eye if the classification makes sense. See Section 5 for an example of end result.

We now come to describing the algorithm in itself.

4.1 The algorithm

Assume the following data generating process:

1. the class of observation (an observation is an inventor-person pair) i is drawn from a Multinomial of parameter ρ: $c_i \sim \mathcal{M}(\rho)$.

2. Let m be observation i's class, then K different variables are independently generated according to $X_{ik} \sim \mathring{\ell}_k(\theta_{km})$, where X_{ik} is the k's generated value, $\mathring{\ell}_k$ is the law of variable k and θ_{km} is the parameters of this law (note that it is specific to the class m). I'll note $\ell_k(X_{ik}, \theta_{km})$ the density of this law for X_{ik} and the parameters.

In our case, we only observe the set of variables X. We want to find out the parameters that maximize the following likelihood:

$$\mathcal{L}(X|\rho, \theta) = \prod_i \left\{ \sum_m P(c_i = m|\rho) \prod_k \ell_k(X_{ik}, \theta_{km}) \right\}.$$

We can't maximize that directly, so we'll employ a variational EM approach (or a mixture model). There are some details on why we do that in Section A.

Solving the problem. Let τ_{im} be the probability that observation i belongs to class m. Then the algorithm is as follows:

Step 1. Maximize:

$$\sum_i \sum_m \tau_{im} \left\{ \ln \rho_m + \sum_k \ln \ell_k(X_{ik}, \theta_{km}) \right\},$$

for all θ and ρ given τ.

Step 2. Update τ as follows:

$$\tau_{im} = \frac{\exp(\alpha_{im})}{\sum_{m'} \exp(\alpha_{im'})},$$

with $\alpha_{im} \equiv \ln \rho_m + \sum_k \ln \ell_k (X_{ik}, \theta_{km})$. Repeat Step 1 and Step 2 until reaching an equilibrium.

Some details on why we are doing this is given in Section A. Bottom line: the likelihood (or in this case the lower bound) is strictly increasing at each step (Dempster et al., 1977).

4.2 Application of the algorithm

As seen in the algorithm, we need to define a distribution for each variable, the variables being the ones defined in Section 3. I propose to use the following distributions:

- Name probability combined to match quality (a categorical variable with 12 values) $\Rightarrow$ multinomial law
- Variable catching the address proximity (a categorical variable with 3 values) $\Rightarrow$ multinomial law
- Variable catching the company proximity (a categorical variable with 4 values) $\Rightarrow$ multinomial law

So far, there are only categorical variables but the algorithm is not limited to that and can accept continuous distributions (as shown in the next section). However it seems appropriate to consider the address and company variables as categorical.

4.3 The initialization

The algorithm requires an initialization, that is: an initial guess of which pairs belong to which group. An initialization could be all pairs that have the same residence address and all pairs that work at the same company. If the data is well behaved, the initialization should not matter and assigning randomly each observation to a group should also lead to the same equilibrium.[17] The next section gives an example of the outcome of the algorithm.

[17]However, the convergence of random initializations to a single equilibrium is not guaranteed since it depends on the specificity of the data. In case of multiple convergence points, a simple yet general rule is to keep the equilibrium that maximizes the likelihood value. In any case, since the algorithm is non random, if we initialize non randomly (as proposed) we always reach the same equilibrium.

4.4 Extension of the algorithm: post-processing to make use of patent information

Now that we have obtained a set of identifiers associated to patents, we can use the information contained in patents to further merge identities. The idea is that inventors who were not matched to a person from statistics Sweden could be matched if extra information were provided. For each person-identifier we dispose of a pool of patents, we can repeat the same algorithm but this time we merge non-matched inventors to statistics Sweden persons that were matched to at least one patent.

The algorithm is the same (Section, 2, 3 and 4), the only step that differ is the step of creating the bilateral variables in which we would add patent-related variables such as:

- a binary variable taking value 1 if the identities share at least one identical team member,
- variables reporting the proximity in topics: this could be the share of IPC codes[18] or text-based measures on the abstracts for instance.

The last step of the algorithm will then decide which person-inventor pairs should be matched based on this new set of information.

5 The MAG-PhD data: example of results

In this section, we use the MAG-PhD matching exercise to illustrate the algorithm.

Setup. The variables and distributions used to run the Bayesian algorithm are as follows:

- Name probability combined to match quality (a categorical variable with 12 values) $\Rightarrow$ multinomial law
- Title proximity (continuous variable between 0 and 1) $\Rightarrow$ Beta law (it's a really nice law, very flexible)
- Lag PhD-First publication (discrete variable between -40 and +34) $\Rightarrow$ Gaussian law
- Same institution (equal to 0 or 1) $\Rightarrow$ multinomial law

As we can see, the variables used in that setup are different from the ones that would be used in the FORTES project. It also shows that continuous variables (like text similarity measures) can easily be included in the algorithm.

[18] IPC: international patent classification. These codes are used to reported technical fields.

Table 8: Example of results from the EM.

type	pub full name	phd full name	match quality	lag	same instit	title proxi.	name prob.	prob true match
FsL	elbert j day	day, elbert jackson	high	1	0	0.56	6.1e-05	0.96
fsL	e j day	day, elbert jackson	low	19	0	0.11	0.021	6.4e-08
FsL	howard c elliott	elliott, howard c, jr	high	-5	1	0.44	0.002	0.99
fL	h elliott	elliott, howard c, jr	low	22	0	0	0.021	2e-12
fL	h elliott	elliott, howard c, jr	low	24	0	0	0.021	6.2e-14
fL	h elliott	elliott, howard c, jr	low	26	0	0	0.021	1.4e-15
FsL	paul e cable	cable, paul elvin	high	1	0	0.29	0.00051	0.85
fL?	p cabler	cable, paul elvin	low	16	0	0	1e-04	1.1e-07
FsL	james l kassner	kassner, james lyle, jr	high	8	0	0.38	0.00044	0.45
FsL	wayne h finley	finley, wayne h	high	4	1	0.059	0.00031	0.98
FsL	wayne h finley	finley, wayne h	high	11	0	0.059	0.00031	0.021
FsL	theodore r adkins	adkins, theodore roosevelt, jr	high	-3	0	0.25	0.00024	0.77
FL	david platt	platt, david	mid	-2	1	0.8	0.0039	0.97
FL	david platt	platt, david	mid	2	0	0.2	0.0039	0.04
fL	d platt	platt, david	low	10	0	0	0.012	3.7e-06
FL	lawrence rosen	rosen, lawrence	mid	-5	1	0.8	0.0023	0.93
FL	lawrence rosen	rosen, lawrence	mid	0	0	0	0.0023	9.6e-05
FL	lawrence rosen	rosen, lawrence	mid	9	0	0	0.0023	5.2e-06
FL?	lawrence r rouen	rosen, lawrence	low	10	0	0	1e-05	5e-06
FL	lawrence rosen	rosen, lawrence	mid	11	0	0	0.0023	1.2e-06
FL	lawrence rosen	rosen, lawrence	mid	19	0	0	0.0023	1.4e-10
fL?	l rossen	rosen, lawrence	low	26	0	0	0.00059	1.4e-14
FsL	herbert s schwartz	schwartz, herbert s	high	5	0	0.11	0.0027	0.16
FsL	herbert s schwartz	schwartz, herbert s	high	15	0	0	0.0027	3.6e-07
fsL	a r ducharme	du charme, arthur ross, jr	low	3	0	0.12	0.0017	0.26
FsL	james e bailey	bailey, jr, james edmund	high	4	0	0.29	0.025	0.37
fsL	j e bailey	bailey, jr, james edmund	low	3	0	0	0.11	1e-04
FsL	james e bailey	bailey, jr, james edmund	high	10	0	0	0.025	3.8e-05
FSL	john kent folmar	folmar, john kent	high	3	0	0.1	0.00024	0.64
FSL	john kent folmar	folmar, john kent	high	15	0	0	0.00024	2e-06

Results. The results of the classification are illustrated by Figure 2. The second class is the class of the 'true' matches. We can see that the algorithm brushes a very coherent picture. In this group, people tend to have a first publication very close to the PhD year, often have the same institution and some proximity in the titles. We also can see that it is mostly populated by strong match and low probability names.

When matching these data sets, I ended up with 17% of the sample being strictly more probable to belong to the true match group, which represented 107,591 pairs. Table 8 shows results from the EM algorithm for a random sample of pairs. The last column gives the probability to belong to the class of true matches given the observables.

6 Expected outputs

The expected outputs are:

- a working paper detailing the matching algorithm and all the choices that were made
- the result of the algorithm, including:
  - a unique identifier associated to a person, across data sets

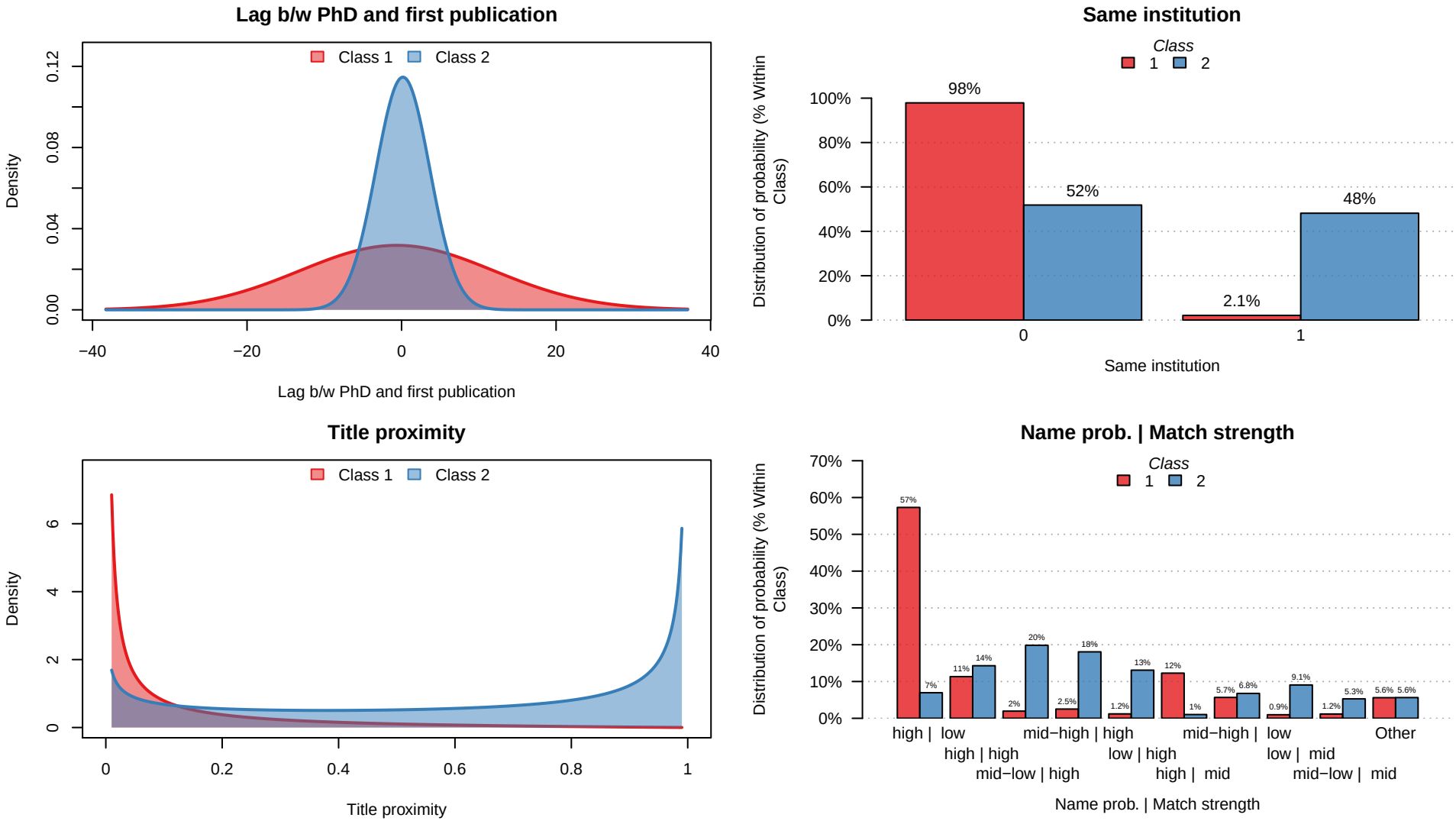

Figure 2: Results of the classification. Distribution of the parameters for all variables and the two groups.

  - the probability of being a good match for each person-inventor pairs with the same names
  - the distributions of the matching variables for each group

- several R programs, including:
  - `funNameExtract`: a function extracting the names (Section 2.1)
  - `funNameMatch`: a function to match the names (Section 2.2)
  - `funCleanAdd`: a function to clean the addresses (Section 3.2)
  - `funCleanComp`: a function to match companies (Section 3.3)
  - `funBayesEM`: a function performing the Bayesian algorithm (Section 4)
  - `funMain`: a main function encompassing all the previous ones

- a Shiny application (equivalent to an HTML web page) to perform the matching given some input data in a "point-and-click" way

Figure 3 provides a representation of the pieces of software with the user inputs and the output for each function. Except for the `funBayesEM` function, there will be two versions of the functions. One that is tailored to the Swedish data, taking into account all its specific features to provide the best possible results. This function will require the user to provide a very specific set of data. The second version of the function is a generic one, which will work with more general data but is not tailored to the Swedish case. These functions will try to provide the best results in a generic way but are necessarily inferior to a bespoke algorithm, specific to the input data.

Adapting to non-swedish cases. Since the functions are modular, it is possible to directly modify by hand the inputs/outputs of the functions and to adapt them to the case at hand without modifying the flow of the algorithm. For instance, if another data set has some specific features on the names, the user can manually modify the output of the cleaning procedure (`funNameExtract`), e.g. by adding new rules, before feeding the new data into the matching algorithm (`funNameMatch`). The final algorithm, `funBayesEM`, only requires a set of bilateral variables which can be modified/added at will by the user.

Note that some adaptations may require to modify the source code of the functions: this will be made easier by the fact that the code will be commented and written in a piece-wise fashion so that the code should need to be modified only in some specific places.[19]

[19]Note that a fair programming knowledge is indispensable to make complex adaptations. This cannot be avoided.

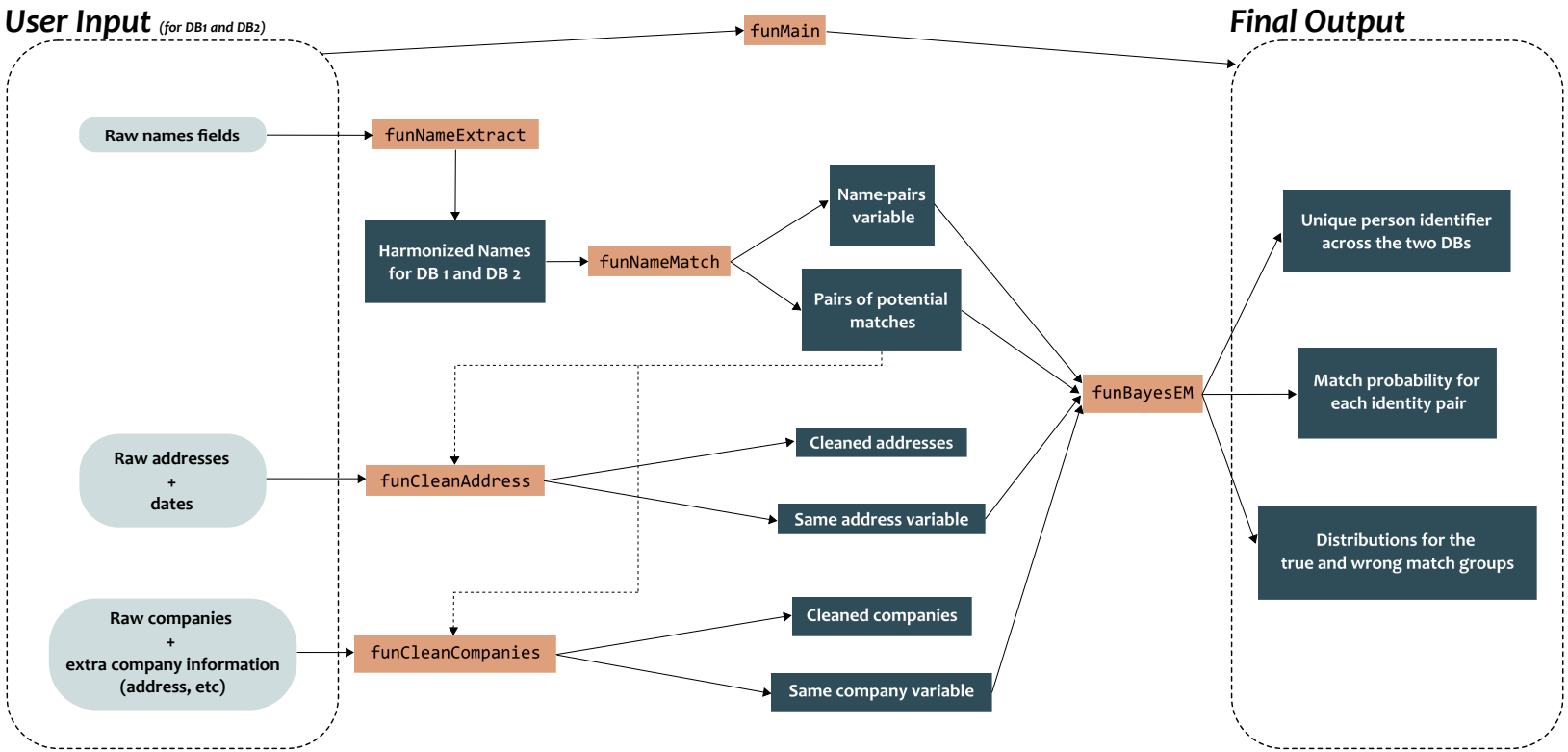

Figure 3: Software input-output map.

Performance. The author has extensive experience in a) dealing with large data sets[20] and b) optimizing code,[21] such that the algorithm should be easily applied to a data set of the size of Patstat and the Swedish national registry and achieve high performance in terms of speed.

References

Abramitzky, R., L. Boustan, K. Eriksson, J. Feigenbaum, and S. Pérez (2021): "Automated linking of historical data," *Journal of Economic Literature*, 59, 865–918.

Carayol, N., and L. Cassi (2009): "Who's who in patents. A Bayesian approach.," in *Les Cahiers du GREThA*.

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

Li, G.-C., R. Lai, A. D'Amour, D. M. Doolin, Y. Sun, V. I. Torvik, Z. Y. Amy, and L. Fleming (2014): "Disambiguation and co-authorship networks of the US patent inventor database (1975–2010)," *Research Policy*, 43, 941–955.

Raffo, J., and S. Lhuillery (2009): "How to play the "Names Game": Patent retrieval comparing different heuristics," *Research Policy*, 38, 1617–1627.

Trajtenberg, M., G. Shiff, and R. Melamed (2009): "The "Names Game": Harnessing Inventors, Patent Data for Economic Research," *Annals of Economics and Statistics*, 93/94, 79–108.

A Variational EM

We want to estimate the following data generating process:

- The class of observation i is generated according to a multinomial $c_i \sim \mathcal{M}(1, \rho = (\rho_1, \ldots, \rho_M))$ of parameter ρ.
- Let m be individual i's class, then K different variables are independently generated according to $X_{ik} \sim \ell_k(\theta_{km})$, where X_{ik} is the k's generated value ℓ_k is the law of variable k and θ_{km} is the parameters of this law (note that it is specific to the class m).

[20] For example the MAG data set is several hundreds of GB large.

[21] For example, the R package fixest is the fastest one, across platforms, to estimate econometrics models with fixed effects.

We observe X, that is all the variables, for each individual. We don't observe the classes. We want to estimate the parameters that maximize the likelihood of observing X:

$$\mathcal{L}(X|\rho,\theta) = \prod_i \left\{ \sum_m P(c_i = m|\rho) P(X_i|\theta_m) \right\}$$

However, the maximization problem defined in the previous equation is intractable: we can't maximize it directly over θ and ρ. The problem comes from the sum over m inside the product over i.

We will use a variational EM approach where we just need to remove the classes (then removing the sum over m). Let z represent the variational counterpart of the class c.

The lower bound writes:

$$\begin{aligned}
\mathscr{L} &= \sum_z q(z|\tau) \ln \frac{P(X,z|\rho,\theta)}{q(z|\tau)} \\
&= \sum_z q(z|\tau) \ln P(z|\rho,\theta) \\
&\quad + \sum_z q(z|\tau) \ln P(X|z,\rho,\theta) \\
&\quad - \sum_z q(z|\tau) \ln q(z|\tau)
\end{aligned}$$

where $q(z|\tau)$ represents a distribution of z, such that $\sum_z q(z|\tau) = 1$.

We take z as independent across observations and generated according a to multinomial law for which the parameters, τ, are observation-specific. The lower bound is easy to optimize, let's look at all parts.

The term in $\ln P(z|\rho,\theta)$. This term can be simplified into:

$$\sum_z q(z|\tau) \ln P(z|\rho,\theta) = \sum_i \sum_m \tau_{im} \ln \rho_m.$$

The term in $\ln P(X|z,\rho,\theta)$. This term can be simplified into:

$$\sum_z q(z|\tau) \ln P(X|z,\rho,\theta) = \sum_i \sum_m \tau_{im} \ln \prod_k \ell_k(X_{ik}|\theta_{km}).$$

The term in $\ln q(z|\tau)$. This term can be simplified into:

$$\sum_z q(z|\tau) \ln q(z|\tau) = \sum_i \sum_m \tau_{im} \ln \tau_{im}.$$

Summing up. The lower bound can then be written as:

$$\mathscr{L} = \sum_i \sum_m \tau_{im} \left\{ \ln \rho_m + \ln \prod_k \ell_k(X_{ik}|\theta_{km}) - \ln \tau_{im} \right\}$$

A.1 The V-EM algorithm

A.1.1 The M-step

We need to find the parameters ρ and θ that maximize the lower bound given τ.

The ρ. We need to find ρ such that the following is maximum:

$$\begin{aligned} \rho &= \arg\max_{\rho} \sum_i \sum_m \tau_{im} \ln \rho_m \\ s.t. &\quad \sum_m \rho_m = 1 \end{aligned}$$

The solution is:

$$\rho_m = \frac{\sum_i \tau_{im}}{n}$$

with n the number of observations.

The θ_{km}. We need to find the set of parameters θ_{km} that maximize:

$$\begin{aligned} \theta_{km} &= \arg\max_{\theta_{km}} \sum_i \sum_m \tau_{im} \ln \ell_k \left(X_{ik}|\theta_{km}\right) \\ s.t. &\quad \text{constraints on } \theta_{km} \end{aligned}$$

The solution of this problem is straightforward.

A.1.2 The V-E step

Here we need to find τ that maximizes the lower bound given ρ and θ:

$$\begin{aligned} \tau &= \arg\max_{\tau} \sum_i \sum_m \tau_{im} \left\{\alpha_{im} - \ln \tau_{im}\right\} \\ s.t. &\quad \sum_m \tau_{im} = 1, \, \forall i \end{aligned}$$

with $\alpha_{im} \equiv \ln \rho_m + \ln \prod_k \ell_k \left(X_{ik}|\theta_{km}\right)$ a constant quantity with respect to τ. The solution of this problem is:

$$\tau_{im} = \frac{\exp\left(\alpha_{im}\right)}{\sum_{m'} \exp\left(\alpha_{im'}\right)}.$$