This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 676060

31/03/'18

COR*-IDS progress report: Inter-generational Linkage

Sam Jenkinson, Koen Matthys, Hideko Matsuo (KU Leuven)

Project deliverable

Expected Result 7.1: *Concepts and techniques for individual record linkage providing algorithms and relevant syntaxes to construct intergenerational life course trajectories from original or IDS-converted multi-level and multi-source data bases.*

Expected Results 7.3/7.4: *Methodologies and techniques for the longitudinal analysis: harmonization and transformation of historical data sets.*

1. Introduction

This report serves as update on the work towards the Longpop expected results of ESR7: 7.1, 7.3, 7.4 (see above). All of these expected results are being met by the work towards the construction and planned release of a COR* intermediate data structure (IDS) for researchers to use longitudinal historical demographic analysis. This is being done based on the original input of the COR* multi-level, multi-source historic demographic database in 2010 (Matthijs and Moreels 2010).

Our deliverable research output targets the intended expected results. It will provide concepts and techniques for individual record linkage, through the replication and improvement of the initial individual and intergenerational record linkage. This will also provide the relevant syntax and algorithms for use in R. It will also involve work towards, and the reporting on, the methods and techniques used for the harmonisation and transformation of the COR* database into IDS format.

This particular report of 7.3. focuses on the intergenerational linkages in the original COR*-2010 database and the creation of tables representing these relationships. The problems concerning the relationships within the COR*-2010 database have been noted

already in the previous deliverable report of 7.1. *COR technical note for the construction of intermediate data structure (IDS)* (September 2017). These concern the potential inconsistency of information documented in the population register, but also the way it has been recorded within the database with relationship codes across multiple columns. There is, additionally, some level of interpretation by those involved in data collection and data entry, also creating further levels of complexity and concern.

2. Indiv-indiv table

A fundamental part of COR*-IDS is the creation of a relationships table, "indiv-indiv" (Alter & Mandemakers 2014). This table accounts for the relationships between every individual included in the database in an entity attribute format. It includes both the type and duration of the said relationship, including both biological and non-biological relationships. This consists of the timestamp type, whether it has a specific duration or is time invariant, and also the type of relationship from one individual to another, e.g. mother, father, daughter, husband, wife, lodger, etc.

3. Intergenerational linkage obtained for COR*-2010 (IDmoeder & IDvader)

The COR*-2010 database is a pre-existing database. This means that much of the intergenerational linkage has been done already, resulting into the identification of two parental variables: IDmoeder (IDmother) and IDvader (IDfather). These are constructed variables and are largely based on the population register, but also to some extent, on the birth, death and marriage certificates.

Van Baelen (2007) documents the steps implemented to derive family relationships within the COR*-2010 sample. Two primary approaches were used to obtain kinship relationship: exact family relationships; and the use of family names. The first approach makes use of the schema of relationship codes (see Table 1) to derive up to 60 potential types of inter- and intra-generational relationships on the basis of the information contained in the population register. The second approach is based on the calculation of the indicator through the information of family name and the geographical location of the individual. These two primary approaches, compared and applied in the COR*-2010, successfully obtained 3,142 and 2,760 mother and fathers relationships with their children.

Table 1: Schema of relationships documented in COR*-2010 (originally in Dutch)

	00: referentiepersoon (reference person)
0: 0de generatie (generation)	01: vader of moeder (father or mother)
	02: schoonvader of schoonmoeder (in-law father or mother)
	03: stiefvader of stiefmoeder (step-father or step-mother)
	04: oom of tante (uncle or aunt)
	05: voogd (guardian)
	06: grootvader/grootmoeder (grand-father or grand-mother)
	07: stiefschoonvader/moeder (step-in-law father or step-in-law mother)
	08: stiefgrootvader/moeder (step-in–law grandfather or step-in-law grandmother)
	09: overgrootvader/moeder (great grandfather, great grandmother)
1: 1ste generatie (generation)	11: gehuwde partner (married partner)
	12: ongehuwde partner (unmarried partner)
	13: broer of zus (brother or sister)
	14: schoonbroer of schoonzus (brother-in-law or sister-in-law)
	15: stiefbroer of stiefzus (step-brother or step-sister)
	16: halfbroer of halfzus (half-brother or half-sister)
	17: neef of nicht (cousin/e) (cousin, niece)
	18: halfschoonbroer of halfschoonzus (half-in-law brother, half-in-law sister)
	19: halfneef/nicht (half-cousin, half-niece)
2: 2de generatie (generation)	21: zoon of dochter (son or daughter)
	22: zoon of dochter uit een vorig huwelijk (son or daughter from a former marriage)
	23: onwettige zoon of dochter (non-recognized son or daughter)
	24: schoonzoon of schoondochter (in-law son or in-law daughter)
	25 stiefzoon of stiefdochter (half-son, half-daughter)
	27: bevoogd kind (patronized child)
	28: neef of nicht (counsin, niece)
	29: halfneef/nicht (half-cousin, half-niece)
3: 3de generatie (generation)	31: kleinzoon of kleindochter (grandson, granddaughter)
	32: stiefkleinzoon of stiefkleindochter (half grandson, grand daughter)
	33: schoonkleinzoon of schoonkleindochter (in-law grandson, in-law granddaughter)

	34: achterkleinkind (great-grandchild)
4: andere familie relatie (other family relationship)	40: onbekende familierelatie (unknown family relationship) 41: andere familierelatie (other family relationship)
5: geen familierelatie (non-family relationship)	51: collectief huishouden (collective household) 52: andere relatie (other relationship) 53: buur (neighbour) 54: ambtenaar burgerlijke stand (civil servant, registrar) 55: bekende (known) 56: personeel burgerlijk gasthuis (personnel from hospital)
6: onbekende relatie (unknown relationship)	60: onbekende relatie (unknown relation)

We also envisage a few issues regarding the identification of relationships (i.e. fathers and mothers), deriving from these two methods. The first approach is not suitable for non-standardized households (e.g. non-married parents and illegitimate children), since these proportions are considered relatively high until the mid-19th century. These households are also subject to the absence of many family members in the household (i.e. siblings, aunts/uncles, grandfather/mother), rejecting the possibility to identify any additional members in the household. Also, note should be made that relationship variables present in the population register are collected during the fieldwork based on the interpretation of collectors following the relationship codes noted above. This may introduce biases in recording these relationships. Also, the second approach assumes that children always receive the father's name, although this may not always be the case, for instance among illegitimate children. The relative chance of living in the same area among the same family members is also dependent on the type of resident area (e.g. urban as opposed to sub-urban or village). And furthermore, they are less applicable to non-Antwerp origins (i.e. migrants). At least to mention a few, the above indicates the potential demographic groups who are possibly not linked, based on the two primary approaches implemented in COR*-2010. The magnitude of such cases is not known at this moment and subject to further investigation. Considering the socio-economic demographic population characteristics of Antwerp arrondissement throughout the 19th century, this limitation may suggest that linkages are restricted to sub-group of population residing in such standardized household (e.g. married parents with children, also grandparents).

In light of our concerns on information contained within the population register and our inability to be clear on whether these variables were constructed from this or the certificates, it is necessary to perform a number of consistency checks on these parental linkages. Firstly, we will cross-check the information contained within these variables against what is contained in both the birth and marriage certificates, focusing on how similar the names are, using a Levenshtein similarity metric (see section 5)(Levenshtein 1965). Once this is done, the second step will be to move on to create the indiv-indiv table for direct biological relationships, starting with fathers and mothers as contained in the both birth and marriage certificates. Finally, once this is completed, we will be able to calculate Levenshtein values for siblings and grandparents/grandchildren, giving us an indiv-indiv table covering the direct biological family as well as three generations of the same families.

4. Evaluating the accuracy of IDmoeder and IDvader in COR*-2010

Within COR*-2010, there is a table called *"*Vastuniek*"* which is informally known as "*the golden standard"*, because most information in this table was rigorously updated and cleaned from all sources within the COR*-2010 database. This table contained two variables: IDmoeder; and IDvader. These two variables contain the identification numbers (IDNR) of an individual's mother and father. See table below for information on these variables.

Table 2: Vastuniek Intergenerational Linkage COR*-2010

	#N	Missing	Unique
IDmoeder	11,412	22,171	3142
IDvader	10,495	23,088	2760

In order to use these variables for the indiv-indiv-table as part of COR*-IDS we need to ascertain the accuracy of these intergenerational linkages. To this end we have implemented the following:

1. Pull the name level (first, middle and second) information for each individuals mother and father from the Vastuniek table;
2. Compare this, using a Levenshtein similarity tool (described below) to the parental information contained for each individual within the birth certificates file; and
3. The assessment of Levenshtein similarity will be done based on the similarity of names between those contained within IDmoeder and IDvader from the Vastuniek

table, and the certificates using a Levenshtein similarity measure, which will be discussed later. The assessment of names, as opposed to other variable similarity is chosen because of few common variables on parental information contained within *"*Vastuniek*"* and the birth certificates. For this reason, only names can be used. Based on the scores that will be obtained by this method, those which our analysis shows to be identical, or likely the same person (i.e. score equals one) will then be used to create the indiv-indiv table. Those who are deemed too dissimilar will then be analysed further across other sources and a decision will be made. This will focus on whether to use what is contained within the *"Vastuniek"* table*,* or to create new IDNR using the individual information contained for the parents within the birth certificates.

5. Methods: Levenshtein similarity

To be able to compare the names, we need to measure the similarity, or dissimilarity, between the two names (the parental names from Vastuniek and also the parental names from the birth certificates). To do this, we use a Levenshtein edit distance calculation which measures the similarity of two names (henceforth referred to as character strings or simply strings) (Levenshtein 1965).

Levenshtein similarity (LS) is a measure provided by the R record linkage package of the likeness between two character strings including an original and a comparison (Borg 2016). It measures the number of insertions, deletions or substitutions required to make each character string the same.

Eq.1 (Borg 2016**)**

$$1 - \frac{d(str1, str2)}{Max(A, B)}$$

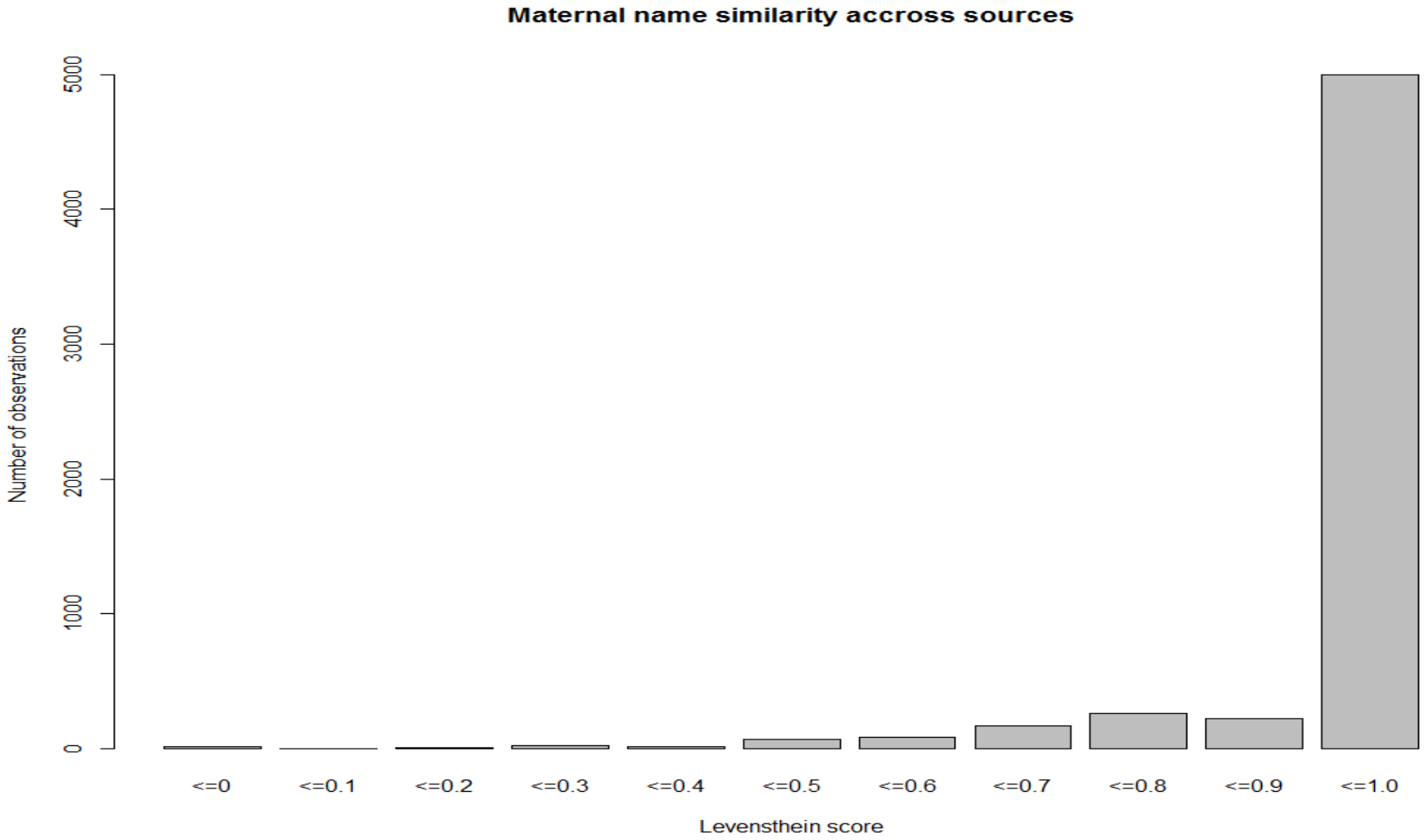
The equation for measuring Levenshtein similarity can be seen above; *d* represents the Levenshtein distance function, *str1* and *str2* the two strings (or names) are compared, and A and B are the lengths of the strings respectively (Borg 2016, p.49). The function gives

a score with 1 being the maximum and anything below that representing increasing dissimilarity (or decreasing similarity).

The first and second name variables within the sample for mothers and fathers were combined to create one variable including first name/second name. A cleaning function was then performed on the name variables to remove unnecessary capitalisation and extra white space either before, between or following the names. Once this was done, the Levenshtein-similarity was calculated between the two names. Following this, the outcome scores were categorised in groups, as can be seen in Figures 1, Figure 2, and Table 3.

Only those who have both a parental ID in the Vastuniek file under IDmoeder/IDvader, and also have a birth certificate record, are included here. This gives a total of 5,883 observations who have a mother and 5,676 who have a father. These comparison will be used to produce the indiv-indiv table.

Figures 1 & 2: Distribution of Levenshtein scores values by mothers and fathers



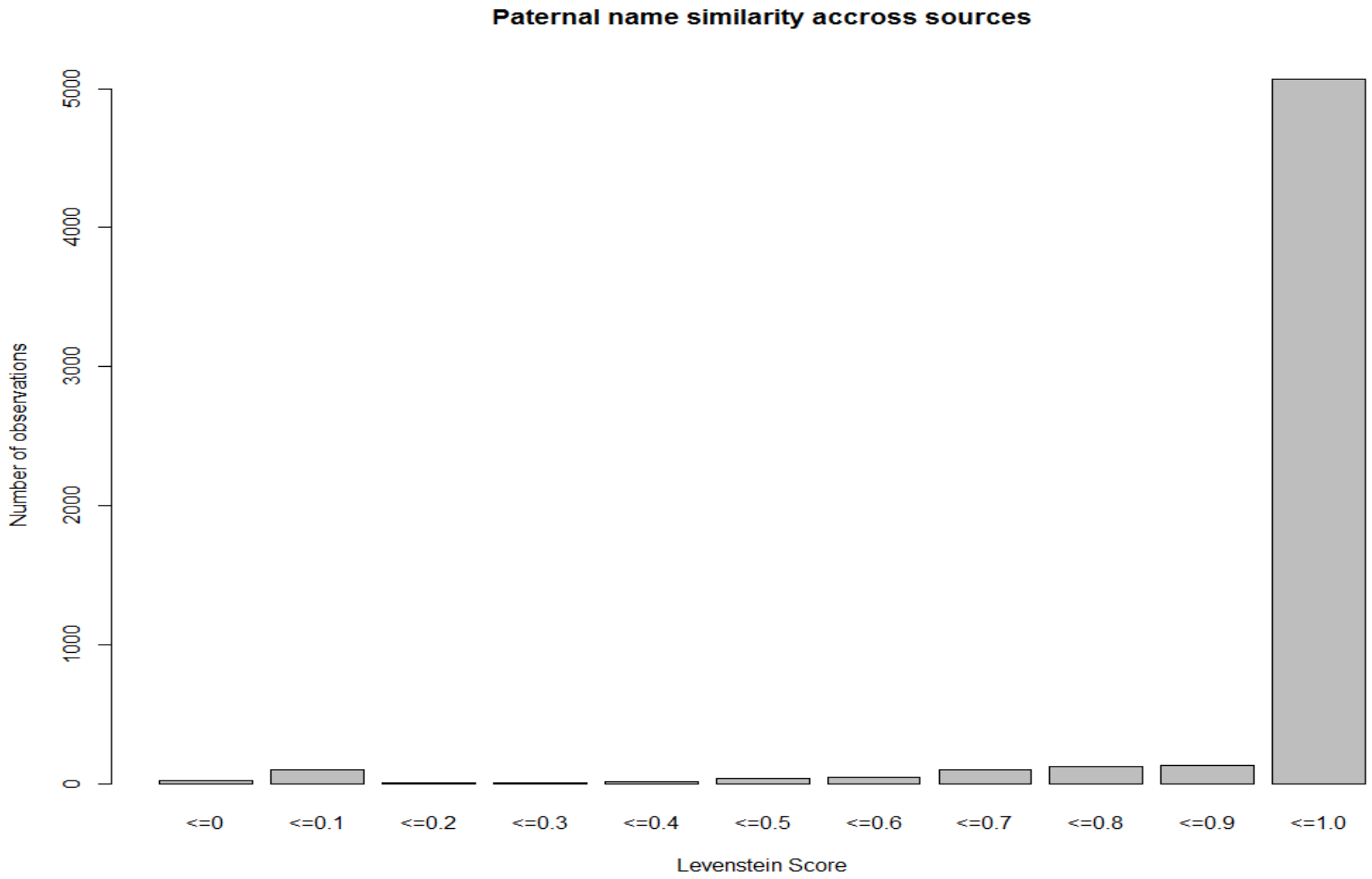

Paternal name similarity accross sources
Number of observations
0
1000
2000
3000
4000
5000
<=0
<=0.1
<=0.2
<=0.3
<=0.4
<=0.5
<=0.6
<=0.7
<=0.8
<=0.9
<=1.0
Levenstein Score

Table 3: Levenshtein score category

	<0	0<0.1	0.1<0.2	0.2<0.3	0.3<0.4	0.4<0.5	0.5<0.6	0.6<0.7	0.7<0.8	0.8<0.9	0.9<1.0	Total
Mothers	19	1	7	27	17	72	83	168	261	228	5000	5,883
Fathers	24	104	8	5	16	42	47	102	124	134	5070	5,676

For the sample of fathers, 89% (5,070/5,676) of records match completely with 10.7% having some dissimilarity between names. For mothers the story is slightly worse with 84% (5,000/5,883) being complete matches and 15% having different levels of dissimilarity. Tables 3 (above) and Tables 4-10 (below) depict the number of observations falling into each category and also the type of problems encountered as a result of name differences, including examples. Table 3 shows the number of observations falling within each category which captures the level of similarity. The categories start from those having a Levenshtein-score below 0, the most dissimilar, before moving on in increments of 0.1 until reaching a maximum of 1, indicating complete similarity, or identical records. Each group contains a range, for example any observation falling in the <0.1 category can have a score greater than 0, but below 0.1.

Of those who are not identical across sources, there appear to be a large number which are highly likely to be the same individuals. Examples of this can be seen with those with the lowest similarity score in Table 4, which shows this scoring the lowest levels of Levenshtein-similarity. All but one observation appears to exist, because of an error where "e" has been replaced "<U+653C><U+3E39>", but also something similar for the letter K, though it is not clear how this has come about. In total for fathers this affects 39 observations (all of the lowest category of scores but 1). For mothers this number is 36, and again explains the majority falling into the two lower categories.

Another problem is where both first names and surnames are missing from birth certificates (Table 5). This shows those with "NA, NA" indicating neither first nor surname name has been registered on the birth certificate. This is a problem which only affects fathers, occurring for no observations of mothers, and is therefore likely a sign of illegitimacy. In total, this effects exactly 101 cases where both first and second name are missing for father, but a name is present in Vastuniek. These will need to be investigated further, using the marriage certificates, in order to decide upon whether to use the linkage obtained from the Vastuniek table and population register information.

Other cases have one name missing. This can be from either source, contrary to the above which only affects birth certificates. Examples can be seen in Table 6 where either a first name or surname entry is not present. For fathers, this effects only 7 cases and for mothers only 6. Examples can be seen in Table 6.

Moving to Table 7, those falling into the 0.8 and 0.9 are in many ways the most difficult to be confident in and will require further analysis and cross checking with other sources. This is because, whilst they are often highly similar names, the differences resulting from a 1 or 2 letter difference can be quite meaningful. An example is IDNR 9481 which on the birth certificate is listed as “joannes franciscus vanbauwel” and in the Vastuniek file as “josephus franciscus vanbauwel”. Another example is IDNR “jacobus franciscus gilliams” and “Joannes franciscus gilliams”. These are small differences in terms of the Levenshtein similarity of names, but quite meaningful differences in being interpreted as different names rather than mistakes. For these, we will compare names recorded in the marriage certificates and complete further analysis before including in the indiv-indiv table.

For the rest a number of different problems are apparent, but appear most frequently for records in categories representing Levenshtein-scores from 0.3 to 0.7. For a number of people, the differences are largely inconsistencies in the presence or omission of middle names across sources. We have decided to treat individuals with the same first and last name in both sources, but middle names either missing or conflicting, as the same person. This accounts for 75 cases for fathers and 121 of cases for mothers and can be seen in Table 8.

Amongst observations which fall between the categories 0.3 to 0.7 an even larger number differences appear to result because of the ordering and absence/presence of different middle and first names. For this we plan to check against the marriage certificates and possibly population register, to get an idea of further similarity, but also which name should be taken into the database. For fathers this accounts for 79 observations and for mothers 93. These can be seen in Table 9.

The rest, as can be seen in Table 10, for both mothers and fathers require further investigation. This is because the names are very dissimilar, or because there are variations which require cross checking. Many of these for mothers are likely to be changes in name following marriage. Many of the observations contain similar surnames which are a combination of the old name with a new name. We will cross check these with marriage certificates before including in the indiv-indiv table.

This means that, including those with the “<U+653C><U+3E39>” and also those with the same and last name, the total which we are treating as the same individual across records at this point is 5,157 mothers and 5,184 fathers representing 87.66% and 91.33% respectively. This leaves 12.34% of mothers and 8.67% of father who are either incorrect or need further investigation before they can be included in the indiv-indiv table which will be created.

Those which are deemed not to be the same, will then require a decision as to whether to choose as the parent. This will either involve using the IDNR of the link provided by the Vastuniek table, or alternatively by using the information contained within the birth certificate to create a new IDNR, using the attributes contained within these records.

6. Further work in the second and third quarter of 2018

Those which we have identified as the same individuals across records will be used to create the indiv-indiv table. For the rest, we will do the following:

a) Cross-check records against marriage certificates and use those which are the same within the indiv-indiv table;
b) Investigate remaining conflicts between Vastuniek and the certificates, including potentially creating new IDNR and attributes for any individuals contained within the certificates and not present/conflicting with Vastuniek;
c) Create sibling and grand-parental linkages based upon these initial maternal and paternal links;
d) Implement controls to identify non-linkages possibly among members from non-standardized households and migrants; and
e) Construct individual attribute table in line with the metadata (submitted in 2017).

Table 4: Levenshtein similarity including error messages

Child IDNR	Father IDNR	Birth Certificate	Vastauniek	Levenshtein Similarity
10137	5007	NA NA	joannes hers<U+653C><U+3E39>e	-0,867
10488	9365	jacobus detiege	jacobus deti<U+653C><U+3E38>ge	0,000
10678	9365	jacobus detiege	jacobus deti<U+653C><U+3E38>ge	0,000
18076	18075	gommarus rong<U+653C><U+3E39>	gommarius ronge	-0,133
20446	9365	jacobus detiege	jacobus deti<U+653C><U+3E38>ge	0,000
20448	9365	jacobus detiege	jacobus deti<U+653C><U+3E38>ge	0,000
20449	9365	jacobus detiege	jacobus deti<U+653C><U+3E38>ge	0,000
21320	18075	gommarus rong<U+653C><U+3E39>	gommarius ronge	-0,133
27690	9365	jacobus detiege	jacobus deti<U+653C><U+3E38>ge	0,000
3866	3669	balthazar franciscus ginn<U+653C><U+3E39>	franciscus ginee	0,000
3952	3669	balthazar franciscus gen<U+653C><U+3E39>	franciscus ginee	-0,038
5009	5007	joannes hersee	joannes hers<U+653C><U+3E39>e	-0,067
5010	5007	joannes hersee	joannes hers<U+653C><U+3E39>e	-0,067
5011	5007	joannes hersee	joannes hers<U+653C><U+3E39>e	-0,067
5012	5007	joannes hersee	joannes hers<U+653C><U+3E39>e	-0,067
5598	5007	joannes hersee	joannes hers<U+653C><U+3E39>e	-0,067
8838	5007	joannes hersee	joannes hers<U+653C><U+3E39>e	-0,067
8972	5007	joannes hersee	joannes hers<U+653C><U+3E39>e	-0,067
9202	5007	joannes hersee	joannes hers<U+653C><U+3E39>e	-0,067
9413	9406	stanislas korv<U+653C><U+3E39>	stanislas korve	0,000
9414	9406	stanislas korv<U+653C><U+3E39>	stanislas korve	0,000
9950	5007	joannes hersee	joannes hers<U+653C><U+3E39>e	-0,067
9951	5007	joannes hersee	joannes hers<U+653C><U+3E39>e	-0,067
9952	5007	joannes hersee	joannes hers<U+653C><U+3E39>e	-0,067

Table 5: Missing names from birth certificates

Child IDNR	Fathers IDNR	Birth Certificate	Vastuniek	Levenshtein Similarity
1081	1079	NA NA	carolus ludovicus kortals	<=0.1
10928	10926	NA NA	norbertus hens	<=0.1
11059	11057	NA NA	josephus franciscus degroof	<=0.1
11256	21102	NA NA	joannes gerardus franciscus verloove	<=0.1
124	121	NA NA	matheus korewyn	<=0.1
12775	12773	NA NA	ludovicus alphonsus regina kortout	<=0.1
1396	1395	NA NA	cornelius ludovicus kornelissens	<=0.1
14264	14423	NA NA	carolus jacobus versaaren	<=0.1
1430	5183	NA NA	josephus korynen	<=0.1
14462	14472	NA NA	carolus koppens	<=0.1
14463	14465	NA NA	ludovicus dejonge	<=0.1
14518	16608	NA NA	franciscus eduardus dekauwer	<=0.1
1493	1491	NA NA	henricus haudering	<=0.1
1657	1655	NA NA	petrus carolus snepvangers	<=0.1

Table 6: Fathers partially missing names

Child IDNR	Fathers IDNR	Birth Certificate	Vastuniek	Levenshtein Similarity
1476	6745	NA koronel	benjamin koronel	<=0.5
18093	18092	nestorius nolf	NA nolf	<=0.4
18094	18092	nestorius nolf	NA nolf	<=0.4
18200	321	NA korrewyn	henricus korrewyn	<=0.6
21136	3094	NA korrewyns	antonius korewyn	<=0.4
23276	23255	damaso logaria korpuz	NA korpuz	<=0.4
23284	23255	damaso logaria korpuz	NA korpuz	<=0.4

Table 7: Levenshtein similarity categories 0.8 & 0.9

Child IDNR	Mothers IDNR	Birth Certificate	Vastuniek	Levenshtein Similarity
10	7	isabella wolles	isabella wallis	<=0.9
10422	4351	rosalia wuyts	rosalia truyts	<=0.9
10937	10933	catharina kornelissens	catharina kornelis	<=0.9
11	7	isabella wolles	isabella wallis	<=0.9
11557	11555	anna maria kortoos	anna maria kortens?	<=0.9
12263	20510	josephina philomena deruyter	josephina philomena ruyters	<=0.9
12741	6453	maria catharina vandenynde	anna catharina vandenynde	<=0.9
12782	2235	elisabeth catharina driessens	elisabeth maria catharina driessens	<=0.9
12785	8882	maria elisabeth kortebeek	maria elisabeth korteberch	<=0.9
12995	7136	maria norbertina philomena helena korbeels	maria norbertina felicia helena korbeels	<=0.9
13715	13702	dorothea wouman	dorothea waumans	<=0.9
141	140	maria anna verhees	maria anna verbers	<=0.9
142	140	maria anna verhees	maria anna verbers	<=0.9
143	140	maria anna verhees	maria anna verbers	<=0.9
14326	6459	magdalena sophia bruyndonks	maria magdalena sophia bruyndonks	<=0.9

Table 8: IDNR with same first and last names but with inaccurate middle names

Child IDNR	Fathers IDNR	Birth Certificate	Vastuniek
10498	28323	abraham josephus tossanus mansart	abraham mansart
12464	12414	josephus augustus augustinus selis	josephus augustus selis
12466	12414	josephus augustus augustinus selis	josephus augustus selis
12467	12414	josephus augustus augustinus selis	josephus augustus selis
12468	12414	josephus augustus augustinus selis	josephus augustus selis
12469	12414	josephus augustus augustinus selis	josephus augustus selis
13098	12414	josephus augustus augustinus selis	josephus augustus selis
13099	12414	josephus augustus augustinus selis	josephus augustus selis
14790	14780	petrus joannes dingemans	petrus dingemans
14792	14780	petrus joannes dingemans	petrus dingemans
14921	14920	walterus vanrooi	walterus cornelius vanrooi
4251	10019	josephus korluy	josephus franciscus korluy
4558	4592	josephus korynen	josephus franciscus korynen
4577	4592	josephus korynen	josephus franciscus korynen
5229	5226	joannes franciscus kloek	joannes kloek
5230	5226	joannes franciscus kloek	joannes kloek
68	4592	josephus korynen	josephus franciscus korynen

Table 9: Misplaced and inconsistent first and middle names

Child IDNR	Fathers IDNR	Birth Certificate	Vastuniek	Levenshtein Similarity
10092	9481	joannes franciscus vanbauwel	josephus franciscus vanbauwel	<=0.9
10430	6786	franciscus adrianus kornelissen	adrianus franciscus kornelissen	<=0.7
12180	12178	alphonsus kornet	marcellus alphonsius maria josephus kornet	<=0.4
13337	13318	jacobus vanhoidonk	adrianus jacobus vanhooidonk	<=0.7
13396	10912	josephus franciscus kornelis	franciscus josephus kornelis	<=0.5
13428	10912	josephus franciscus kornelis	franciscus josephus kornelis	<=0.5
14117	14115	carolus verlinden	egidius carolus verlinden	<=0.7
14147	6484	baptistus korremans	joannes baptista korremans	<=0.7
14200	14195	carolus vandenynde	guilielmus carolus vandenynde	<=0.7
14217	5548	franciscus wilms	joannes franciscus wilms	<=0.7
14711	14656	ludovicus martin	joannes ludovicus martin	<=0.7
1481	1450	ferdinandus josephus korvilain	josephus ferdinandus korvilain	<=0.4
15576	10912	josephus franciscus kornelis	franciscus josephus kornelis	<=0.5
15596	10912	josephus franciscus kornelis	franciscus josephus kornelis	<=0.5
15996	233	joannes vandenbrand	antonius joannes vandenbrand	<=0.7
16503	11558	hubertus guilielmus belletable	ludovicus fredericus hubertus guilielmus belletable	<=0.6
1777	1772	franciscus kortoos	joannes franciscus kortoos	<=0.7
18048	18049	guilielmus martinus soetewy	martinus guilielmus soetewy	<=0.5
18050	18049	guilielmus martinus soetewy	martinus guilielmus soetewy	<=0.5
18145	192	nicolaus kormon	joannes nicolaus kormon	<=0.7
18234	233	joannes vandenbrand	antonius joannes vandenbrand	<=0.7

Table 10: Incorrect and needing further investigation

Child IDNR	Fathers IDNR	Birth Certificate	Vastuniek
10152	8108	joannes kalay	joannes korluy
10155	10153	joannes demaar	joannes baptistus demaas
10156	10153	joannes demaar	joannes baptistus demaas
10157	10153	joannes demaar	joannes baptistus demaas
10158	10153	joannes demaar	joannes baptistus demaas
1022	1018	evaristus nachtergaale	evanitas nachtegaale
1023	1018	evaristus nachtergaale	evanitas nachtegaale
1158	1157	ernestus genovena josephus goossens	emilius georgius josephus goossens
11618	11625	prosperius achillus napoleon kornesse	prosperius achillus napoleon honorius cornelius hubertus kornesse
11700	11580	arnoldus vanderlemmer	petrus vandenlemmer
11702	11580	arnoldus antonius vandenlemmer	petrus vandenlemmer
13224	13222	petrus sikard	petrus josephus likard
13749	6673	amatus petrus henricus somers	armandus henricus somers
15496	6673	amatus petrus henricus somers	armandus henricus somers
16263	16262	gasparius dekonink	josephus dekonink
17401	20315	petrus wouters	jacobus vangiel
17645	17644	abraham josephus tossanus mansart	abraham josephus mansard
17646	17644	abraham josephus tossanus mansart	abraham josephus mansard
1766	1757	prosperius joannes staas	eduardus ludovicus kornet
18403	20754	jacobus korty	joannes korty
19032	10153	joannes demaar	joannes baptistus demaas
2103	2101	petrus joannes delveaux	josephus delveaux
2427	2335	matheus korrewyn	guilielmus korrewyn
2445	2335	matheus korrewyn	guilielmus korrewyn
253	251	augustus ludovicus josephus kornou	augustus kornaar
27105	18367	david truyens	marcus truyens

27246	27244	achillus albertus eduardus korn<U+653C><U+3E39>	ariel albertus eduardus korme
27571	22893	egidius janssens	ludovicus wyten
28276	2248	cornelius kornemuse	joannes baptistus kornemuse
2864	2862	carolus victor maria josephus mendiaux	carolus victor mondiaux
2865	2862	carolus victor maria josephus mendiaux	carolus victor mondiaux
29190	29187	joannes derksen	jacobus mateisen
29191	29187	joannes derksen	jacobus mateisen
29192	29187	joannes derksen	jacobus mateisen
31467	25417	joannes franciscus hendriks	franciscus demees
3702	3686	petrus joannes regenmortel	joannes vanregemortel
3702	3686	petrus joannes regemortel	joannes vanregemortel
3783	3780	albertus vandenynde	joannes cornelius vandenynde
38	36	josephus korrieri	antonius franciscus korrieri
4394	4387	joannes franciscus kornelissen	jacobus amarandus kornelissen
4402	4398	cornelius kornelissens	petrus joannes kornelissens
4404	4398	cornelius kornelissens	petrus joannes kornelissens
4405	4398	cornelius kornelissens	petrus joannes kornelissens
4407	4398	cornelius kornelissens	petrus joannes kornelissens
4446	9487	machutus korrynen	michael korynen

7. References

Alter, G. and K. Mandemakers (2014). The intermediate data structure (IDS) for longitudinal historical microdata, version 4. *Historical life course studies,* 1, 1-26.

Borg, A (2016). Package 'Record Linkage'

https://cran.r-project.org/web/packages/RecordLinkage/RecordLinkage.pdf

Levenshtein, V.I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Cybernetics and control theory,* 10, 8, 707-10. (Translated from Doklady Akademii Nauk SSRR, 163, 4, 845-48, August 1965).

Matthijs, K. and S. Moreels (2010). The Antwerp COR-database: a unique Flemish source for historical-demographic research. *The History of the Family,* 15, 109-115.

Sariyar, M. and A. Borg (2010). The record linkage package: detecting errors in data. Contributed research articles in *R Journal,* 2/2, December 2010. Access at http://journal.r-project.org/archive/2010-2/RJournal_2010-2_Sariyar+Borg.pdf

Van Baelen, H. (2007). Constructie van een historisch-demografisch longitudinale database: methodologie van de demographica flandria selecta. *Manuscript*. Centrum voor Sociologisch Onderzoek (CeSO).

Appendix: R syntax for obtaining Levenshtein scores

########## Checking IdVader & IdMOeder #############

Aim: To check the accuracy of the mother and father variables of included in the "Vastauniek" table of the COR database

we need to compare this to the information included within the birth register.

Vastauniek includes two variables for mother, and father; IDvader, IDmoeder.

This needs to be checked against the information from the birth register "Geboorteddood".

Every person included in this register includes one IDNR and then variables for the mothers first name/surname and fathers

first name and surname. These variables are "vrnaamva", "famnaamva" & "vrnaammo" & "famnaammo".

#libraries to load#

library(readr)

library("Hmisc", lib.loc="~/R/R-3.3.1/library")

#

Set working directory

setwd("C:/Users/u0110986/Dropbox/Data/COR/Mother Father checks")

Step 1: Subset Birth register to variables required

which var requited?

IDNr of childm, name of child, date of birth,

"IDNR", "vrnaam", "famnaam", "gbdag", "gbmaand", "gbjaar", "vrnaamva", "famnaamva", "vrnaammo", "famnaammo"

#1.0 load birth register to R

b<-read.table("Birth Register full.csv", sep=";", header=TRUE)

Birth_Register_full <- read_delim("C:/Users/u0110986/Dropbox/Data/COR/Mother Father checks/Birth Register full.csv",

```
                    ";", escape_double = FALSE, trim_ws = TRUE)
#Step 2. Select var required from this database#
# fathers indiv raw var
b1 <- c("IDNR", #ID
        "vrnaam",  #first name
        "famnaam", #family name
        "gbdag", #birth day
        "gbmaand", #birth month
        "gbjaar", #birth year
        "gesl", #sex
        "vrnaamva", #fathers first name
        "famnaamva", #fathers surname
        "vrnaammo", #mothers name
        "famnaammo") #mothers surname
b1 <- Birth_Register_full[b1]

#step 2 Preparing Vastauniek table with parents IDNR and names
#loading file
v <- read_delim("C:/Users/u0110986/Dropbox/Data/COR/Mother Father
checks/inwoner_vastuniek04050607.csv",
                    ";", escape_double = FALSE, trim_ws = TRUE)
#Which Var to keep?
# same as below, but IDNRmo and IDvaeder
v1var <- c("IDNR", #ID
    "vrnaam",  #first name
    "famnaam", #family name
    "gbdag", #birth day
    "gbmaand", #birth month
    "gbjaar", #birth year
    "geslacht", #sex
    "IDvader", #Id of father
```

```
      "IDmoeder", #ID of mothers
      "source") #source
v1 <- v[v1var]

#to make more simple replace old files with new
v <- v1
rm(v1)
b <- b1
rm(b1)

# step3. Now have two files. But V does not have parent names, only IDs which are needed for a name check.
# these are in V attached to the IDNR of ID moedervader. Need to subset these mothers/fathers from the IDNR
#collumn and then add back in with the name value changed to mothers name/fathers name. Luckily half the work is done
# from the original indiv indiv files.

#loading indiv-indiv
i <- read_delim("C:/Users/u0110986/Dropbox/Data/COR/Mother Father checks/indiv_indiv.csv",
           ";", escape_double = FALSE, trim_ws = TRUE)

#will need to be done seperately for mothers and fathers.
#fathers first

#subsetting indiv-indiv to keep only fathers
iv <- i[ which(i$relation=='father'),]

#subsetting v by fathers
# Need to change INDR1 (fathers ID) to IDNR
iv$IDNR <- iv$IDNR1
```

```
# can't remember how to merge keeping only values present in one file
# will try merge function as I think it only keeps individuals present in both

iv1 <- merge(iv,v, by="IDNR")

#worked. Not to subset to keep only variables which are required
#IDNR2 is son. So this needs to stay as will be changed to IDNR when merging with original V to
#add names
v2var <- c("IDNR", #ID
        "IDNR2",
        "vrnaam",  #first name
        "famnaam") #famename
v2 <- iv1[v2var]

#change vrnaam/famnaam to vrnaamva & "famnaamva"
v2$vrnaamva1 <- v2$vrnaam
v2$famnaamva1 <- v2$famnaam
v2var <- c("IDNR", #ID
        "IDNR2",
        "vrnaamva1",  #first name
        "famnaamva1") #famename
v2 <- v2[v2var]
#fathers finished#

#Mothers
#subsetting indiv-indiv to keep only fathers
im <- i[ which(i$relation=='mother'),]

#subsetting v by fathers
```

```
# Need to change INDR1 (fathers ID) to IDNR

im$IDNR <- im$IDNR1


im1 <- merge(im,v, by="IDNR")


#worked. Not to subset to keep only variables which are required

#IDNR2 is son. So this needs to stay as will be changed to IDNR when merging with original V to

#add names

m2var <- c("IDNR", #ID

        "IDNR2",

        "vrnaam",  #first name

        "famnaam") #fam name

m2 <- im1[m2var]


#change vrnaam/famnaam to vrnaamva & "famnaamva"

m2$vrnaammo1 <- m2$vrnaam

m2$famnaammo1 <- m2$famnaam

m2var <- c("IDNR", #ID

        "IDNR2",

        "vrnaammo1",  #first name

        "famnaammo1") #famename

m2 <- m2[m2var]


# both mother and father completed

# need to merge both into original file by IDNR2 making a column for mothers and fathers name

# create blank column in original file for mother father names

# make matching IDNR and IDNR2 columns

# merge

# then make matching mother father names column (1/2 even)
```

```
# are columns same? Overall?

# how similar?


#changing IDNR variables in order to remerge. IDNR will become mother IDNR and Father IDNR. IDNR 2 will

#become IDNR in order to merge


#m2 IDNR var

 m2$moIDNR <- m2$IDNR

 m2$IDNR <- m2$IDNR2


 #v2 IDNR

 v2$vaIDNR <- v2$IDNR #new father IDNR

 v2$IDNR <- v2$IDNR2  #IDNR which will match that in birth register


#Subset only to keep correct IDNR var plus names

m2var <- c("IDNR", #ID

        "moIDNR",

        "vrnaammo1",  #first name

        "famnaammo1") #famename

m2 <- m2[m2var]


v2var <- c("IDNR", #ID

        "vaIDNR",

        "vrnaamva1",  #first name

        "famnaamva1") #famename

v2 <- v2[v2var]


#OK so problem is when merging we will lose cases which are unmatched. Can try to keep all

#below
```

```
#mothers names and birth register
a <- merge(b,m2, by="IDNR", all = TRUE) #keeps all (some odd observations in here)
a1 <- merge(b,m2, by="IDNR") # keeps only those matched 5883
#created some odd observations

#fathers
c <- merge(b,v2, by="IDNR", all = TRUE) #12236 obs
c1 <- merge(b,v2, by="IDNR") #5676 obs (in both vast and b)

# question is now. To do comparison on all just those with records in both? I think just those
# records in both. And do Mothers and Fathers separately. Just a quick 1 if column matches. How much?

#remove all but the comparison files C1 (fathers), a1 (mothers)
rm(a, b, Birth_Register_full, c, i, im, im1, iv, iv1, m2, v, v2)

#DO c COMPARISSONS FIRST AS FATHERS LOOK MORE STRAIGHT FORWARD
#WHICH VAR NEED COMPARING?
#1. "vrnaamva" vs "vrnaamva1",  #first name
#1. "famnaamva" vs "famnaamva1"
c1$vrnaamva.identical<- (c1$vrnaamva==c1$vrnaamva1)
c1$famnaamva.identical<- (c1$famnaamva==c1$famnaamva1)

#How many are wrong? Fathers first name
summary(c1$vrnaamva.identical)
# Mode     FALSE    TRUE    NA's
# logical     805    4770     101
# 14% not identical. Why? Will export to view after
summary(c1$famnaamva.identical)
# Mode     FALSE    TRUE    NA's
```

```
# logical    1287    4288     101
# 22% not identical

#How many mothers are wrong?
#DO c COMPARISSONS FIRST AS FATHERS LOOK MORE STRAIGHT FORWARD
#WHICH VAR NEED COMPARING?
#1. "vrnaamva" vs "vrnaamva1",  #first name
#1. "famnaamva" vs "famnaamva1"
a1$vrnaammo.identical<- (a1$vrnaammo==a1$vrnaammo1)
a1$famnaammo.identical<- (a1$famnaammo==a1$famnaammo1)
summary(a1$vrnaammo.identical)
#   Mode   FALSE    TRUE
#logical     826    5057
#14% not identical
summary(a1$famnaammo.identical)
#    Mode   FALSE    TRUE
# logical    1739    4144
# 29% not identical

#Fathers
names(c1)
c1var <- c("IDNR",
        "vaIDNR",
        "vrnaam",
        "famnaam",
        "famnaamva",
        "famnaamva1",
        "famnaamva.identical",
        "vrnaamva",
        "vrnaamva1",
        "vrnaamva.identical")
```

```
c2 <- c1[c1var]


#mothers
names(a1)
a1var <- c("IDNR",
        "moIDNR",
        "vrnaam",
        "famnaam",
        "famnaammo",
        "famnaammo1",
        "famnaammo.identical",
        "vrnaammo",
        "vrnaammo1",
        "vrnaammo.identical")
a2 <- a1[a1var]


## OK this makes it easier to view the errors. A quick scan shows most to be stray capitalization
# white spaces and slightly different spellings.
# one option now is to remove these and check again, if necessary or do some kind of a phonetic
# root matching, if it is necesary. Though personally I do not see the point. This looks OK.
# Can use code form indiv indiv previously to make these family links into records


#need to add original research person's name from Vastaniek file. Too difficult to go back through.
#Load vasta uniek from before.


#re-load v
v <- read_delim("C:/Users/u0110986/Dropbox/Data/COR/Mother Father checks/inwoner_vastuniek04050607.csv",
           ";", escape_double = FALSE, trim_ws = TRUE)
```

```
#subset to IDs and names
vvar <- c("IDNR",
          "vrnaam",
          "famnaam")
v1 <- v[vvar]

# rename vrnaam & famnaam to take account of being vastauniek and not that in B cert
library(reshape)
v1$vrnaamv <- v1$vrnaam
v1$famnaamv <- v1$famnaam
vvar <- c("IDNR",
         "vrnaamv",
         "famnaamv")
v1 <- v1[vvar]
a3 <- merge(a2,v1, by="IDNR")
c3 <- merge(c2,v1, by="IDNR")

#Now change order to make easier to read
vvar <- c("IDNR",
          "vaIDNR",
          "vrnaam",
          "famnaam",
          "vrnaamv",
          "famnaamv",
          "famnaamva",
          "famnaamva1",
          "famnaamva.identical",
          "vrnaamva",
          "vrnaamva1",
          "vrnaamva.identical")
c4 <- c3[vvar]
```

```
#mothers
names(a3)
vvar <- c("IDNR",
          "moIDNR",
          "vrnaam",
          "famnaam",
          "vrnaamv",
          "famnaamv",
          "famnaammo",
          "famnaammo1",
          "famnaammo.identical",
          "vrnaammo",
          "vrnaammo1",
          "vrnaammo.identical")
a4 <- a3[vvar]

#remove everything but c4/a4
rm(a2,a3,c2,c3,v,v1)

#make all names lowercase
cleaning = function(df){

  cols = colnames(df)
  df   = df

  for(col in cols){

    if(is.character(df[, col]) == TRUE ){
      df[,col]   =   as.factor(gsub("\\.|", "", df[,col]))   #removes stray ".", works better if stray characters removed 1st
```

```
    df[,col]  =  tolower(df[,col])                #converts to lowercase
    df[,col]  =  trimws(df[,col],                 #removes leading and trailing whitespace
                 which = c("both"))

    df[,col][df[,col] == ""]  = NA                #replaces stray "" with NA works better if blanks removed last
    df[,col][df[,col] == " "] = NA                #replaces stray " " with NA works better if blanks removed last
    df[,col]  =  as.character(df[,col])           #converts chr to factor

   }
  }

  name = data.frame(df)
  return(name)

}
a5 = cleaning(a4) # stray white space removed, lower case, NA's inputted.
c5 = cleaning(c4) # #N stayed same

#remove lines not required
rm(a4,c4,a1,c1)

#Re-run comparisson checks

#A5 file
# First vrnaam vs vrnaamv

#First name post cleaning function
a5$vrnaam <- as.character(a5$vrnaam)
a5$vrnaamv <- as.character(a5$vrnaamv)
```

```
a5$vrnaam.identical<- (a5$vrnaam==a5$vrnaamv)
summary(a5$vrnaam.identical)
#    Mode   FALSE    TRUE    NA's
# logical     785    4947     151
#NA's are when both names are missing in each dataset
#785 appear to be just variations of the same name (some exceptions)

#surname
a5$famnaam <- as.character(a5$famnaam)
a5$famnaamv <- as.character(a5$famnaamv)
a5$famnaam.identical<- (a5$famnaam==a5$famnaamv)
summary(a5$famnaam.identical)

#    Mode   FALSE    TRUE    NA's
# logical     569    5283      31

c5$famnaam <- as.character(c5$famnaam)
c5$famnaamv <- as.character(c5$famnaamv)
c5$famnaam.identical<- (c5$famnaam==c5$famnaamv)
summary(c5$famnaam.identical)
# Mode      FALSE    TRUE    NA's
# logical     535    5112      29

#mothers names
a5$famnaammo <- as.character(a5$famnaammo)
a5$famnaammo1 <- as.character(a5$famnaammo1)
a5$vrnaammo <- as.character(a5$vrnaammo)
a5$vrnaammo1 <- as.character(a5$vrnaammo1)

a5$vrnaammo.identical<- (a5$vrnaammo==a5$vrnaammo1)
a5$famnaammo.identical<- (a5$famnaammo==a5$famnaammo1)
```

```
summary(a5$vrnaammo.identical)
# Mode      FALSE    TRUE    NA's
# logical     820    5057       6
summary(a5$famnaammo.identical)
# Mode   FALSE    TRUE
# logical     898    4985
#first names similar figure. Surnames halved
# variations of same name

#Fathers names
c5$famnaamva <- as.character(c5$famnaamva)
c5$famnaamva1 <- as.character(c5$famnaamva1)
c5$vrnaamva <- as.character(c5$vrnaamva)
c5$vrnaamva1 <- as.character(c5$vrnaamva1)

c5$vrnaamva.identical<- (c5$vrnaamva==c5$vrnaamva1)
c5$famnaamva.identical<- (c5$famnaamva==c5$famnaamva1)

summary(c5$vrnaamva.identical)
# Mode      FALSE    TRUE    NA's
# logical     795    4772     109
# variations of similar names

summary(c5$famnaamva.identical)
#    Mode   FALSE    TRUE    NA's
# logical     493    5082     101

#write to CSV and send to Hideko
#the ones with a number and V are vastauniek
```

```
write.csv(a5, "C:/Users/u0110986/Dropbox/Data/COR/Mother Father checks/mothers.csv", row.names=FALSE)

write.csv(c5, "C:/Users/u0110986/Dropbox/Data/COR/Mother Father checks/fatherss.csv", row.names=FALSE)


#Tasks

#Next step is to limit the indiv indiv table already created to just those with identical names

#This will be done by subsetting both A5 and C5 by those who are identical

#Next step is to check those not identical.


#identifying similar records


#step 1. Checking similarity with a score

library(RecordLinkage)

library(Hmisc)

#Fathers 1st names

c5$vrnaamva_match <- levenshteinSim(c5$vrnaamva, c5$vrnaamva1)

c5$famnaamva_match <- levenshteinSim(c5$famnaamva, c5$famnaamva1)


#mothers

a5$vrnaammo_match <- levenshteinSim(a5$vrnaammo, a5$vrnaammo1)

a5$famnaammo_match <- levenshteinSim(a5$famnaammo, a5$famnaammo1)


#some descriptie stats on how similar

summary(c5$vrnaamva_match) #first name match

#  Min.    1st Qu.  Median    Mean 3rd Qu.    Max.    NA's

# 0.07143 1.00000 1.00000 0.96040 1.00000 1.00000     109

summary(c5$famnaamva_match) #family name match

#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's

# -2.4000  1.0000  1.0000  0.9707  1.0000  1.0000     101

summary(a5$vrnaammo_match) #first name match
```

```
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#   0.1111  1.0000  1.0000  0.9472  1.0000  1.0000       6
summary(a5$famnaammo_match) #family name match
# Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
# -1.833   1.000   1.000   0.947   1.000   1.000

# Limit to those over a certain point perhaps?
plot(a5$famnaammo_match)
describe(a5$famnaammo_match)

# continuation
#Merge both name collumns re-run what done before accross both names

#merge both names the " " leaves a space, which I guess is fine.
#Mothers below
a5$mname_bcert = as.factor(paste0(a5$vrnaammo," ",a5$famnaammo))
a5$mname_vast = as.factor(paste0(a5$vrnaammo1," ",a5$famnaammo1))
#fathers
c5$fname_bcert = as.factor(paste0(c5$vrnaamva," ",c5$famnaamva))
c5$fname_vast = as.factor(paste0(c5$vrnaamva1," ",c5$famnaamva1))

#change to character
a5$mname_bcert = as.character(a5$mname_bcert)
a5$mname_vast = as.character(a5$mname_vas)
c5$fname_bcert = as.character(c5$fname_bcert)
c5$fname_vast = as.character(c5$fname_vast)

#levenstein comparisson
a5$mothers_match <- levenshteinSim(a5$mname_bcert, a5$mname_vast)
c5$fathers_match <- levenshteinSim(c5$fname_bcert, c5$fname_vast)
```

```
#summaries
#mothers
summary(a5$mothers_match)
hist(a5$mothers_match)
# Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
# -0.3077  0.9655  1.0000  0.9450  1.0000  1.0000
summary(c5$fathers_match)
hist(c5$fathers_match)
#     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
# -0.8667  1.0000  1.0000  0.9451  1.0000  1.0000

# comments
# majority are identical. Fairly similar for both sexes
# will export and view in excel

write.csv(a5, "C:/Users/u0110986/Dropbox/Data/COR/Mother Father checks/mothers.csv",
row.names=FALSE)
write.csv(c5, "C:/Users/u0110986/Dropbox/Data/COR/Mother Father checks/fatherss.csv",
row.names=FALSE)

#need putting in cats. Maybe use R commander to
library("Rcmdr", lib.loc="~/R/R-3.3.1/library")
library("RcmdrMisc", lib.loc="~/R/R-3.3.1/library")


a5$mothers_match_cat <- with(a5, bin.var(mothers_match, bins=3,
                                method='natural', labels=c('1','2','3')))

summary(a5$mothers_match_cat)
describe(a5$mothers_match_cat)
#gives good idea of dist. But makes more sense to just to it at the numbers
```

```
a5$mothers_match_cat <- cut(a5$mothers_match, breaks=c(-0.309, 0, 0.1, 0.2, 0.3, 0.4, 0.5,
0.6, 0.7, 0.8, 0.9, 1.1),
                    labels=c("<=0",
                        "<=0.1",
                        "<=0.2",
                        "<=0.3",
                        "<=0.4",
                        "<=0.5",
                        "<=0.6",
                        "<=0.7",
                        "<=0.8",
                        "<=0.9",
                        "<=1.0"))
c5$fathers_match_cat <- cut(c5$fathers_match, breaks=c(-0.9, 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6,
0.7, 0.8, 0.9, 1.1),
                    labels=c("<=0",
                        "<=0.1",
                        "<=0.2",
                        "<=0.3",
                        "<=0.4",
                        "<=0.5",
                        "<=0.6",
                        "<=0.7",
                        "<=0.8",
                        "<=0.9",
                        "<=1.0"))
summary(c5$fathers_match_cat)
plot(c5$fathers_match_cat)

write.csv(a5, "C:/Users/u0110986/Dropbox/Data/COR/Mother Father checks/mothers.csv",
row.names=FALSE)
```

```
write.csv(c5, "C:/Users/u0110986/Dropbox/Data/COR/Mother Father checks/fatherss.csv",
row.names=FALSE)
```