Assembling the pages

A sorting-based approach to historical record linkage

Author:
Jelte van Boheemen
3248526

Bachelor's thesis Artifical Intelligence
July 6, 2016
Universiteit Utrecht
10 ECTs

First Supervisor:
dr. ir. Gerrit Bloothooft

Second Supervisor:
dr. Marijn Schraagen

Table of Contents

1 – Introduction

1.1 - Adopting names

Throughout history people have formed communities to live and work together. Within these communities one had to distinguish one person from another. In small communities this is a nearly trivial task; in a village consisting of ten families a person might have simply been known as *Jan*. As communities grew larger and communication between communities increased there was a call for a more unique identifier. Even *Jan from the village across the river* would soon prove insufficient. In the Netherlands, as in many other countries, people adopted a patronymic (based on the father's first name), toponymic (based on location) surname or based their surname on their profession or a distinctive feature. *Jan* for example might have become *Jan Willems* (his father's name was *Willem*), *Jan van Gent* (he came from *Gent), Jan Smit* (working as a smith) or *Jan de Lange* (for a Jan of considerable height). Children adopt the surname of their father. If a woman gets married, she adopts the surname of her husband, while keeping her own surname as a *maiden name.* In 1611, the *Eeuwig Edict* (Dutch for *eternal edict)* decreed that in the southern provinces of the Netherlands births, marriages and deaths should be recorded by the local churches. The manner in which these events were recorded varied among churches. A decree by Napoleon and the start of civil registration in 1796-1811 forced the entire population of the Netherlands to register a fixed surname, while in civil certificates, important events of people were recorded: birth, marriage and death. This standard allowed for distinguishing persons in larger communities and across different communities for purposes of e.g. taxation and military draft.

1.2 - (Historical) record linkage

Being able to distinguish one entity, a person in this case, from another generates a problem: if several references to an entity are found (e.g. a specific name is used in two separate documents), how can one tell which references point to which entities? The process of discovering relations between records (containing references to entities) is called *record linkage*. This term refers to the practice of establishing *links* between two or more records and was first coined by Halbert Dunn in 1949 in a dramatic and poetic way:

> "*Each person in the world creates a Book of Life. This Book starts with birth and ends with death. Its pages are made up of the records of the principal events in life. Record linkage is the name given to the process of assembling the pages of this book into a volume.*" [1]

A link can signify any relation, but usually means 'concerns the same entity'. This task seems easy if both records are exactly same and contain unique information, but in practice differences exist between records which are meant to indicate the same entity.

Humans are very good at this linking task: differences between records are easily bridged. This comes from the fact that humans posses knowledge of the world. This knowledge is acquired though experience and learning. A computer does not have this knowledge, and will have to be instructed. For example, without background information a computer might not know that *Nel* can be an abbreviated form of the names *Petronella* or *Cornelia*. A human may have learned this abbreviation and can therefore predict future uses. A computer without any specific techniques to deal with differences would simply see two strings of characters that are not the same. Computers do have an advantage in record linkage: they are able to inspect many records in a fraction of the

time humans need. When dealing with 20 records, linking is no hard task for a human. When dealing with millions upon millions of records, large groups of humans working in a coordinated fashion would still need an unsurmountable amount of time to establish all links. The goal is therefore to design software in such a way that computerized linking of records is possible.

Research of record linkage often combines aspects of computer science, linguistics and sociology. This embeds the subject in the broader field of Artificial Intelligence, which often combines these research topics. Linguistic techniques can be utilized to process information without knowledge of the context. For example, techniques that measure how similar two words are can be used to decide if two references are meant to indicate the same entity. Sociological and historical information about the population of which records are being linked can also help in this decision. For example, with the knowledge of how surnames developed (briefly discussed earlier in the introduction) in a certain population, a difference in surnames between two records might be explained.
With an increasing number of systems being digitized, the amount of records present in different databases grows fast. Research and development of accurate and efficient methods to link these records is therefore important, incorporating computer scientific research.

Nominal record linkage links records that are based on names of entities. It is used in a wide range of fields, such as medical research [2]**,** counter-terrorism [3] and historical research. In this research, a method for historical record linkage has been developed and evaluated. The goal of historical record linkage is by definition of Schraagen [4]: "*to discover relations between historical entities in a database, for any specific definition of relation, entity and database.*" For the purpose of this research an historical entity will be a person, the database will be a collection of civil certificates and the relation will be '*is the same person'*.

Linking historical documents brings its own package of problems compared to linking modern data. Data comes from a large number of sources, such as regional or municipal archives, often not in a standardized form. Up to a certain point all certificates were written by hand, making transcription and digitizing prone to errors. A unique identifier for each person such as the *Burgerservicenummer (BSN)* or *Social Security Number (SSN)* for the USA is also absent. Datasets can also be very large, even for a country with a small population like the Netherlands, since every person that is alive or was alive or died in a certain period can be referred in multiple certificates. Establishing links between historical records that do not match exactly, or have do not contain sufficient information, is therefore far from trivial.

Record linking strategies mostly fall into one of two categories: *deterministic and probabilistic [5]*. Deterministic strategies classify a possible link as either a link or not a link. These strategies employ a set of rules to decide on this classification. These rules often are established by domain experts or follow from previous studies. Since the rules are tailored to fit the data being investigated, they are very domain-specific, and a new set will have to be developed for each specific problem. Probabilistic strategies compute a probability of being a correct link for a number of candidates. These probabilities are based on a number of *features*, which can be compared to rules in deterministic strategies. Since the method of computing probabilities remains the same throughout all datasets, these strategies are less domain-specific then deterministic strategies. For each problem, a set of features will still have to be developed or selected. Groundwork for these strategies has been laid in the 1969 by Fellegi and Sunter [6].

Both deterministic (e.g. in [6]) and probabilistic (e.g. in [7]) strategies have successfully been employed in the past, and are continuing to be researched and used. In recent years, research into machine-learning strategies is also being conducted. These strategies use part of the data to construct a training set, which the method then uses to learn and improve itself. Examples of these

strategies are the neural network-approaches used in [8], [9].

1.3 - A sorting-based approach

This research evaluates an approach based on the sorting of records: the sorting-based method. The focus of this method lies heavily upon structuring the data, eliminating the need for a complex linking-algorithm. The goal is to find an ordering in which records that 'belong together' are direct neighbors.

The sorting based-method mainly utilizes names of historical entities, making it a nominal historical record-linkage method. It is claimed to be possible to link records without using names. For example, Huijsmans proposes the G3-method in [11].This method uses date of birth, place of birth and gender (*geboortedatum*, *geboorteplaats*, *geslacht* in Dutch). While the G3-method is interesting as an alternative to name-based methods and the fields qualify for the sorting-based approach[1], it has not been evaluated. The goal of this research is to evaluate a sorting-based approach, and for this purpose some comparison with results of existing methods is desirable. Most methods utilize names, making a name-based method a better choice.

Since there can be differences in records that concern the same person, the sorting task is far from trivial. For example, if *Jan Jansen* is misspelled as *Jan Janssen* in a particular record, they might not end up sorted next to each other. A way to make different records equal when this is justified has to be developed. This research proposes the use of manipulations of fields. Manipulations are operations that involve changing fields in such a way that allow for two different fields to be made equal. Examples of these manipulations are changing a name to its standard form and only considering the leading 4 characters of each name. Caution has to be taken when executing this process: if records are homogenized too much (because fields are manipulated to be the same), records that do not reference the same person might be sorted together. On the other hand: if the manipulations are incapable to bridge differences in records that should be bridged, two records that do reference the same person might not be sorted together.
There are clear limits for manipulations. Fields, before or after manipulation, have to either contain a string (e.g. a name) or a number (estimated birth date) to allow sorting. A technique that is often used to establish links between fuzzy candidates, *edit distance* (e.g. *Levenshtein-distance* [13], *Jaro-Winkler distance*[14]*)* compares two strings and outputs a score expressing their similarity. Since these techniques only work between fields of two records, they are unfit to act as a manipulation (which requires an input field and outputs a string or number).

After the data is manipulated and sorted, a simple linking-algorithm is responsible finding links in the data. The rules used in this deterministic algorithm have to be tailored to the specific dataset and the sorting-based nature of the method. A deterministic approach was chosen to keep the linking of records as simple as possible, emphasizing the importance of a solid ordering of records.

This approach is expected to have a very low computational time: efficient ordering algorithms exist, and the amount of candidate records to be inspected when linking is but a fraction of the full dataset.

Chapter two describes the data used in this research and preprocessing steps that were executed. In chapter three the method for sorting and linking, as well as assembling life courses, is described. The results of the method are presented and discussed in chapter four. Conclusions and propositions for further research are presented in chapter five.

1 In section 3.1 properties that fields should have to qualify for the sorting-based method are presented.

2 - Material

2.1 - Source

The data used in this research comes from the *Wie-Was-Wie* database[2], containing civil certificates for birth, marriage and death. This database is the successor to GENLIAS, of which all certificates are included and expanded upon. These certificates have been transcribed manually from the original documents (stored in regional, municipal and provincial archives), largely by volunteers. These digitized records were then entered into the database.
Civil registration was started in 1811 in the Netherlands, with parts of southern provinces such as Zeeuws-Vlaanderen in the province of Zeeland having a head start in 1796 and 1798. For privacy reasons, only documents of certain age are released into the public database: certificates of birth become available after 100 years, marriage after 75 years and death after 50 years. To keep the scope of this research sufficiently small, only certificates drafted in one province has been used. Zeeland was chosen because a large number of certificates has been transcribed, making the digital database close to complete. The resulting dataset contains 1.558.389 certificates: 698.397 births, 192.231 marriages and 666.088 deaths.

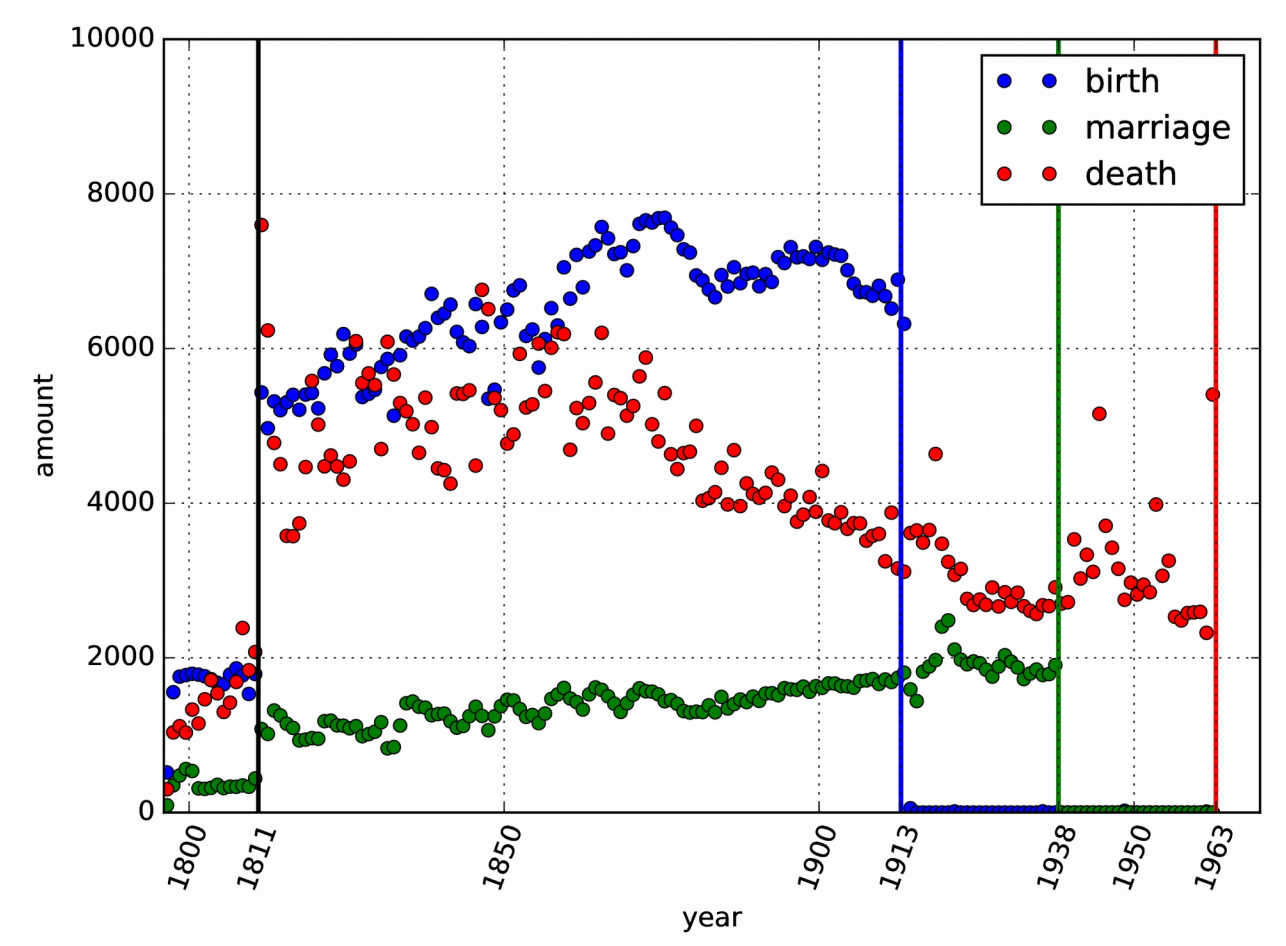

Figure 1: Number of certificates by year between 1795 and 1965. Colored vertical lines represent cut-off point for corresponding type of certificate.

2 *www.wie-was-wie.nl*

Figure 1 shows the distribution of records over the years. For the entire province of Zeeland, birth ranges from 1811-1913, marriage from 1811-1938 and death from 1811-1963. Records before 1811 are the result of civil registration starting earlier in Zeeuws-Vlaanderen. Records after the respective cutoff years are expected to be errors: the dates were transcribed wrong or the record should not have been adopted into the public database yet. All records, even those which are suspected to contain errors, are used in this research.

ID (serial number)	First name	Family name	Age	Role
4118505	wilhelmina louiza henrietta	van doorn	70	deceased
4118506	hendrik	du buisson becius	-	partner
4118507	abraham	van doorn	-	father
4118508	petronella	van dishoek	-	mother

Table 1: A death certificate.

A digitized certificate consists of a serial number in the database and the type, date and place of the event. All records contain the full names of the person concerned (*ego:* child, groom, bride or deceased) and the names of his/her parents and partner if available. Age at the time of marriage or death is available for the ego. Table 1 shows these fields for all persons present on a single death certificate. Several other fields of information are present, on the certificate or derived from the certificate, but none of these are relevant to this research.

Table 2, on the following page, shows an overview of different fields that are relevant to this research. Generally, if one name of a person (first name or family name) is missing, the other is also missing. Only in a small amount of cases the first name is available when the family name is not. Information about the father is missing more often than information of the mother. This difference is most prevalent at birth. In these cases the father is either unknown or wished to be excluded from the record, e.g. when the child was born out of wedlock.
Information about the partner is never present for children being born because they obviously do not have a partner. Since partner is defined as the person one is married to, no missing information about partners at time of marriage is found. Partner names are missing for a large number of records of death. Part of these records concern children dying in their infancy. In the other cases, a person died without anyone being able to identify his/her (deceased) spouse.

34% of all death certificates concern people people under the age of 1, indicating high child mortality in at least parts of the dataset. If a person survived his or her first year, the average age at death is 48. The average age of marriage is 26 for brides and 29 for grooms, including remarriages after a spouse dies or a divorce.

Field	% missing	Field	% missing
first name ego		family name ego	
birth	0.42%	birth	0.16%
marriage	0.03%	marriage	0.08%
death	1.65%	death	0.63%
first name father		family name father	
birth	4.51%	birth	4.50%
marriage	2.94%	marriage	2.92%
death	11.15%	death	11.04%
first name mother		family name mother	
birth	0.04%	birth	0.05%
marriage	1.06%	marriage	1.07%
death	8.57%	death	8.58%
first name partner		family name partner	
birth	100%	birth	100%
marriage	0%	marriage	0%
death	59.33%	death	59.30%
estimated birth date		sex	
birth	0%	birth	0.02%
marriage	0.05%	marriage	0%
death	0.98%	death	0.26%

Table 2: Percentage of records where a certain field is missing, split by type of record. The computation of estimated birth date is discussed in section 2.2.

2.2 - Preprocessing

Large portions of the database were preprocessed to filter out common errors. These processing steps were executed by the LINKS-project of the NWO CATCH-programme[3], and are outlined below.

An important preprocessing step was the creation of two separate tables for the data: registrations and persons. Fields from the original certificates were split into these tables based on if they give information about a recorded event as a whole or about a specific person involved. A record in the new table of registrations contains all the information about the event: serial number, type, date of draft, date of event and place of draft. Every person mentioned on the original certificate has a separate record in the table of persons. For example, a birth certificate generates three separate records: one for the child, one for the mother and one for the father. They are assigned their own serial number in the table of persons. These three records share the serial number of the registration, so information of relations between persons is not lost.
Fields containing names were scanned for elements such as prefixes ('*de Jonge'*) and titles ('*Meester Jonge'*). These elements were moved to their respective fields. Multi-part first names were split, making for multiple fields containing all the parts, with concatenations remaining intact ('*Jan-Willem'*).

A range of new fields was added to include an estimation of dates of life-events for persons. For example, for a record of a groom being 27 years old at time of marriage and the date of marriage being known (29-4-1869), a range between a minimal and a maximal date is set for his date of birth (1841-04-29 to 1842-04-29). When more information is missing, the span of this range increases: for example, a deceased person with no mention of age is estimated to be between 0 and 110 years of age. Conversions from dates to days since year 0 have also been added for all dates. This conversion results in a single number, making comparison and computation easier.

Some certificates contain fields for remarks or notes. These fields have been scanned for viable information which was then moved to its respective fields.

For the current research, information of the father and mother was added to the record of the ego, eliminating the need to inspect multiple records when only the ego should be linked to other egos.

Table 3 shows the same certificate as table 1, with preprocessing steps applied. Note that prefixes *du* and *van* are exluded from the family names. These prefixes were moved to their own field.

ID	ef	el	mf	ml	ff	fl	pf	pl	min. bd	max. bd	role
4118505	wilhelmina	doorn	petronella	dishoek	abraham	doorn	hendrik	buisson becius	656920	657285	deceased
4118506	hendrik	buisson becius	-	-	-	buisson becius	wilhelmina	doorn	633179	677739	partner
4118507	abraham	doorn	-	-	-	doorn	petronella	dishoek	620761	652537	father
4118508	petronella	dishoek	-	-	-	dishoek	abraham	doorn	639023	652537	mother

*Table 3: A death certificate after preprocessing. Key used for fields: **ef** is first name ego, **el** is family name ego, **mf** is first name mother, **ml** is family name mother, **ff** is first name father, **fl** is family name father, **pf** is first name partner, **pl** is family name partner, **bd** is estimated birth date (in days since year 0). These codes will be used throughout this thesis.*

3 *http://www.iisg.nl/hsn/projects/links.html*

3 - Method

As outlined in the introduction, the aim of this research is to evaluate a sorting-based approach to record linkage. The method to collect links between records from the data is therefore split into two phases: first manipulating fields and sorting the records, and then checking for matching records in the sorted data.

3.1 - Selecting fields & sorting

The tables in the database contain a large number of fields. The more fields are considered in the linking-method, the higher the chance of non-matching information between fields. To minimize this chance, and the method's computation time, a minimum set of fields that produces correct links has to be selected.

When selecting fields various factors have to be taken into account:

- Is the field present in all three types of records? (birth, marriage, death)
- Is the field immutable or can it change between records? (e.g. living location can change, while birthdate cannot)
- Is the field complete or empty for most records? (see table 2 in section 2.1)
- Are manipulations possible for the field's content that allow for different sorting and/or linking results?

Obvious candidates for selected fields are first name and family name of ego (which together make up the full name). In case of a multi-part first name, only the first one was used. For family names, prefixes were not used. The reason for excluding any extra name-information is that it is possible for a person to use only part of his names on a record. A person with two first name (e.g. *Jan Pieter*) might have both his names recorded at marriage, while at death the bereaved will identify him simply as *Jan*.

When a child is born, the full names of both parents are recorded (the father's family name being the same as the ego's family name). When a person marries or dies, these names are also recorded, making them present on every type of record. The fact that these fields are constant and present in all three types of record makes them a strong candidate for selection.
As shown in table 2 in section 2.1, the full name of the partner is not present at birth and is often missing at death. Since the aim is to link all types of records, the full name of the partner was not selected.

Children often got the same name as a sibling that died before they were born. In order to distinguish between multiple siblings which have the same name, some field identifying their age had to be selected. The estimated date of birth, the average of the minimum and maximum date of birth (as described in section 2.2), is a strong candidate for this field. A single number of days can act as a sorting key and can easily be compared to other numbers. Since it is possible to use the field late in the sorting-order, even badly estimated birth dates will be sorted close to each other (granted other fields match). Some of the tested procedures ignore this estimated birth date when establishing links to test whether this field is necessary for establishing correct links.

The resulting five names (full name of ego, full name of mother, first name of father), together with date of birth, have been shown to lead to correct links very often [12].

To distinguish between two persons having a unisex name, the sex of ego has been selected as an additional field. For example, for two siblings having grandmother *Alexandra*, they both might be named after her: *Alex*. This obviously only happens when the first sibling dies before the second is born. With child mortality being high in parts of the dataset, this is expected to occur regularly.

The role and serial number (as provided in the preprocessed database) of the ego were selected for aid in establishing and recording links. These fields were not manipulated or considered when sorting records. The role was however considered when evaluating possible links.

An optimal sorting then had to be found for this selection of fields. Sorting occurs alphabetically for fields containing text (names) and numerically for fields containing numbers (estimated birth date), both in ascending order. Records are sorted on the fields from first to last, so the order of the fields influences the resulting ordering. Because of this fact, fields were ordered based on ascending importance (fields that should match are placed first) and descending uncertainty (fields that might contain wrong or missing information are placed last).
The chosen sorting-order therefore is:

full name ego | *sex ego* | *full name mother* | *full name father* | *estimated birth date*

Full names consist of the fields family name and first name (in that order). The full name and sex of the ego were placed in the first position, since it is essential that these fields match. The full name of the mother was placed before the full name of the father because the former is missing in less records, as shown in section 2.1. The estimated birth date was placed at the very end of the sorting-order since it is a mapping from age (often only in years or months) to a range of dates. Both this mapping and the fact that an average is computed for this range lead to uncertainty in this field. This sorting-order is evaluated against two alternative sorting-orders in section 4.3.

el	**ef**	**sex**	**ml**	**mf**	**fl**	**ff**	**bd**	**role**
doorn	wilhelmina	f	dishoek	petronella	doorn	abraham	657103	10
doorn	wilhelmina	f	dishoek	pieternella	doorn	abraham	657040	4
doorn	wilhelmina	f	doorn	jacoba	doorn	null	671562	1
doorn	wilhelmina	f	doorn	jacoba	doorn	null	671566	10
doorn	wilhelmina	f	null	null	doorn	null	647143	10
doorn	wilhelmina	f	schorer	adriana	doorn	hendrik	661241	1
doorn	wilhelmina	f	vlag	maria	doorn	leendert	697569	4
doorn	wilhelmina	f	vlag	maria	doorn	leendert	697661	1

Table 4: A few records after sorting. Code for roles: 1 is child, 4 is bride and 10 is deceased. The roles are displayed only as additional information, they are not used in sorting the records.

Table 4 shows all records present in the database for *Wilhelmina van Doorn*. There is no single pair of records present in this small sample that shares every field, so with exact matching no links would be found. Section 3.2 describes manipulations so that fields like *Petronella* and *Pieternella* could be matched. Section 3.3 describes how the linking algorithm processes this sorted output.

3.2 - Manipulating fields

Matching on precise information would work flawless in the perfect situation where every record reflects exactly the (correctly spelled) information of the people involved, and where the transcription and digitalization of these records occurs without errors. In historical records this is far from the truth. In the sorting-based approach, fields have been manipulated in a way that allows for matching regardless of certain errors or different variations in spelling of the same person's name. The manipulations involve making the field 'fuzzier' and thus being able to be matched with more values for that field. The more information is omitted, the more links are expected to be found. For example, *Johan* and *Johannes* do not match. Prefixes (discussed in the following section) of the same names do match *(Joha* and *Joha*). If manipulations allow too many records to match, the amount of incorrect links will grow. For example, reducing both *Jacobus* and *Johannes* to *J* will results in the field matching when it should not.

In sections 3.2.1 and 3.2.2 a number of manipulations are selected. These manipulations have been evaluated, on their own or in combination with other manipulations, in a *procedure*. A procedure is a complete iteration of the record-linkage method. This iteration uses a unique combination of manipulations of fields. Rules for the linking-algorithm are also variable, these are discussed in section 3.3.

3.2.1 - Family name

Two manipulations have been evaluated for family names: prefixing and phoneticizing.

Prefixing

Prefixing consists of considering only the first part of the family name in the sorting of records and in the linking algorithm. The term *prefix* is also used to denote the element before the family name (*Jan de Jonge)*, but for the purpose of this thesis prefixing denotes considering only the leading characters of the main part of a name (*Jan de Jonge*). For example, in Dutch *Jansen* and *Janse* are both common family names, and might be used as a variation of the same name. The prefixed name consists of N characters, and the value for N can be varied. When N is decreased, less combinations of prefixed names are possible. Since less combinations exists, more records are then expected to be linked.

Values for N have been selected. Dutch family names typically consist of 4 to 10 characters. For the dataset used in this research, the average length of a person's family name is 6.6. This average length predicts that N should be smaller then 6.6, since above this number most family names can not be prefixed. Figures 2 and 3 plot *selectivity* and frequency distributions for a range of values for N.

Selectivity is the number of records that have a unique combination of selected fields divided by the total number of records. For example, a selectivity of .25 for the fields family name of ego and mother indicates that 25% of the total amount of records has a unique combination of these two names. The other 75% of records consist of at least two occurrences of a combination of these names. When selecting a good N-value it is important that N is smaller then the point at which selectivity stabilizes. At this stabilizing point there is a minimal expected gain in links as the amount of combinations of tested fields approaches the number of full name-combinations. Figure 2 shows that at N=6 the selectivity stabilizes, and between N=4 and N=5 the growth reduces significantly.

These were calculated for the combination of family names of ego and mother on all marriage certificates. The father was excluded since he shares the family name with ego. Marriage certificates have been used since information of both parents is available and complete (in records of birth information of the father is missing more often).

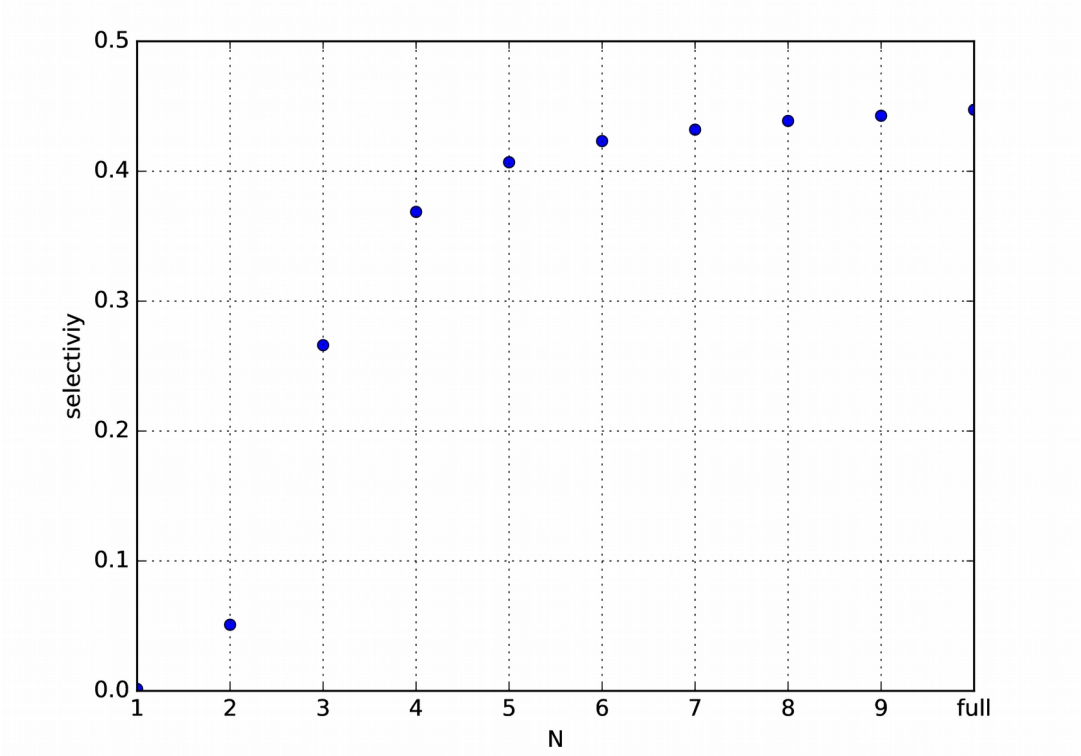

Figure 2: Selectivity for N leading characters of family names, calculated for the combination of names of grooms/brides and their mothers.

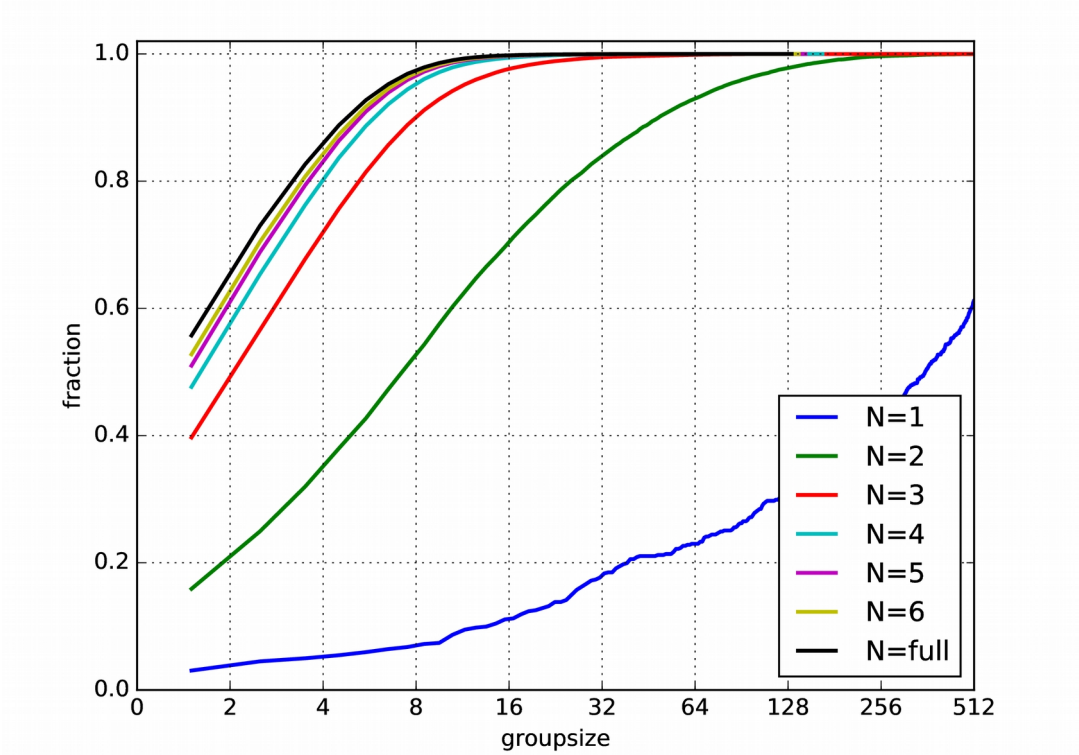

Figure 3: Cumulative frequency distribution for N leading characters of family names, calculated for grooms/brides and their mothers. Groupsize denotes the largest amount of records a combination of fields is found in. E.g. for a combination of family names of ego and mother that occurs only once in all records of marriage, the groupsize is 1. If the combination is found on two records, the groupsize is 2.

Figure 3 shows the cumulative frequency distribution for a number of leading characters of family names. The figure shows that for N=1 and N=2 the curves stabilize very late. For N=3 the most common combination of family names occurs around 32 times, signified by the curve stabilizing at 1.0 at that point. For N=4 the curve stabilizes between groupsize 8 and 16, which is significantly lower than for N=3. For N>4 the curves show very little difference in shape and stabilizing point. Combining this information with the fact that growth of selectivity is reduced between N=4 and N=5, procedures utilizing prefixed forms with N=4 and N=5 have been tested.

Prefixing with N=4 reduced the amount of distinct family names from 51.380 to 12.525. Using N=5 resulted in 25.922 possible family names.

Phoneticizing

Prefixing will not catch variation in the leading N characters of the name. There are however cases where these two variations of a name identify the same person. Standardizing both names and comparing these two standardized forms can unify known variations of the same name. Unfortunately, a solid standard for Dutch family names is not available. As an alternative, family names were transformed to a semi-phonetic form. One might encounter two records ordered together with family names *Claassen* and *Klaassen*. In Dutch, the two names are pronounced the same. A reference table containing names and their semi-phonetic form, calculated using the rules developed in [15], was used to perform the manipulations. This table maps 562.720 family names to

343.406 semi-phonetic forms. For the records used in this research (children, grooms, brides and deceased) this manipulation reduced the amount of distinct family names from 51.380 to 39.335. For 18.690 out of 1.752.283 records the family name was not available in the reference table, in these cases the non-manipulated family name was used.

Prefixing of this form has also been tested, as there can be spelling variations in the last part of the name that are phonetically identical (*'Schoonaard' and 'Schoonaerd').* A good N-value again was decided based on selectivity and frequency distribution of family name and the mother's family name in marriage certificates.

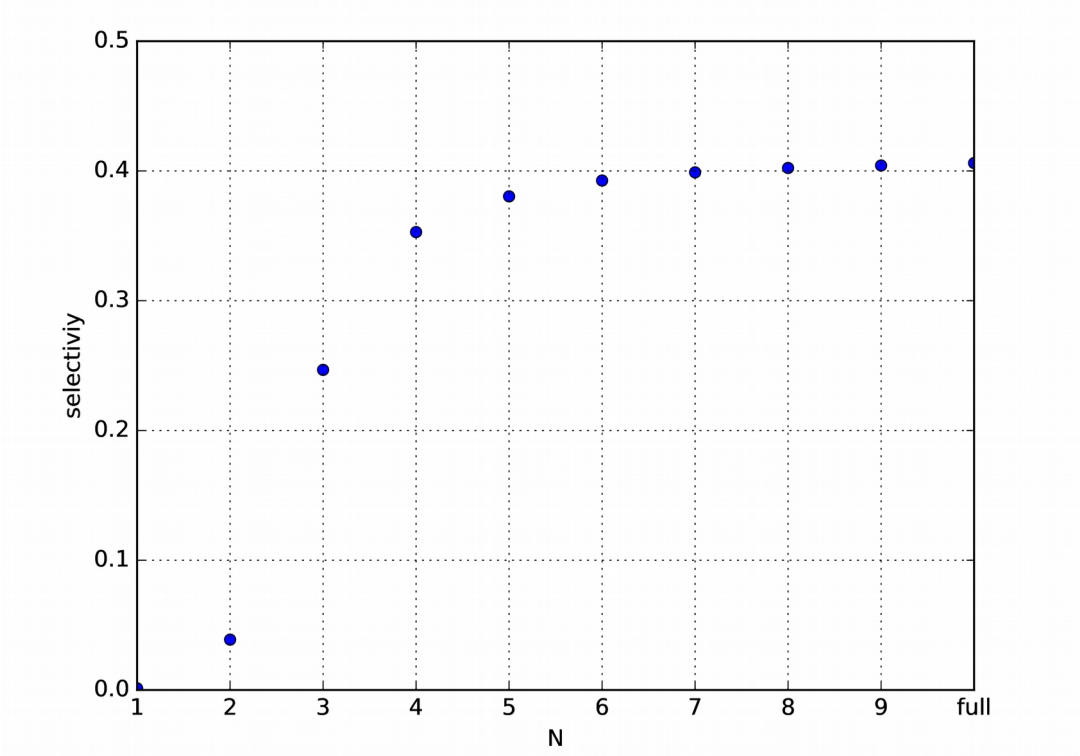

Figure 4: Selectivity for N leading characters of semi-phonetic form of family names, calculated for grooms/brides and their mothers.

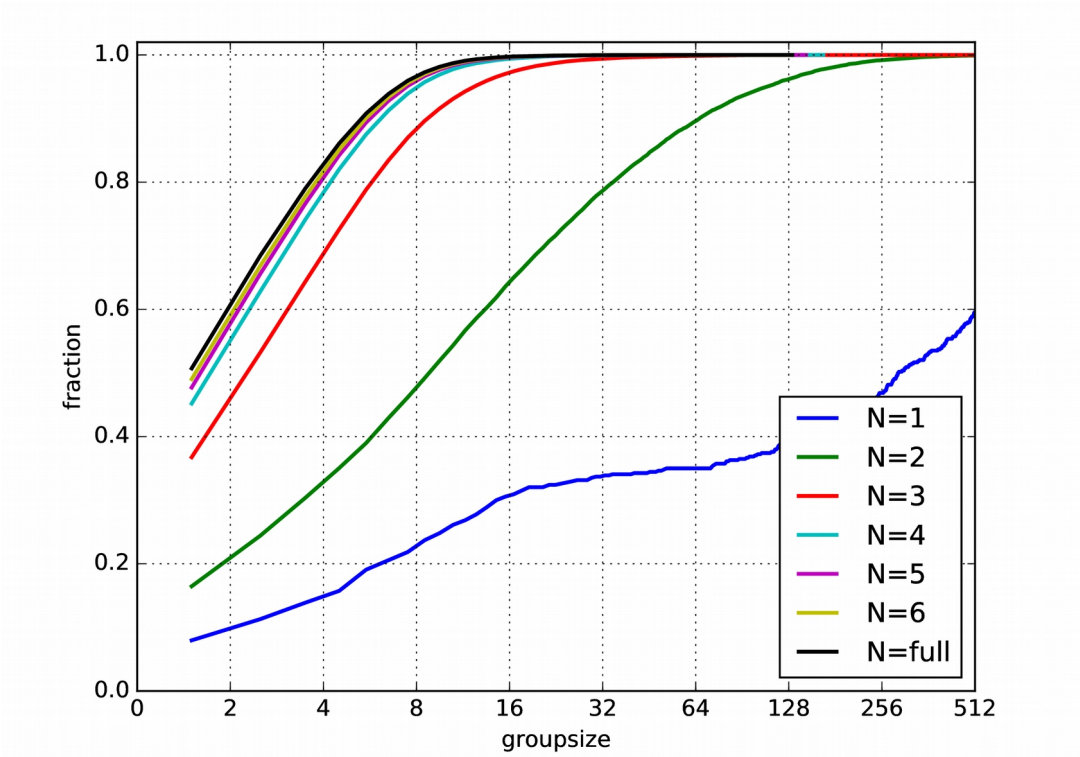

Figure 5: Cumulative frequency distribution for N leading characters of semi-phonetic form of family names

Figures 4 and 5 show curves similar to prefixing the normal form of family names. Therefore, procedures utilizing prefixing with N=4 and N=5 for the semi-phonetic form of family names were selected for testing. Prefixing with these values further reduced the amount of distinct family names to 12.782 and 24.798 respectively.

3.2.2 - First name

Prefixing

Similarly to family names, first names can be prefixed to catch variation in spelling in the last part of the name. A good N-value was selected based on selectivity and the frequency distribution of combinations of the first names of brides or grooms and their mothers. For first names the father is also included.

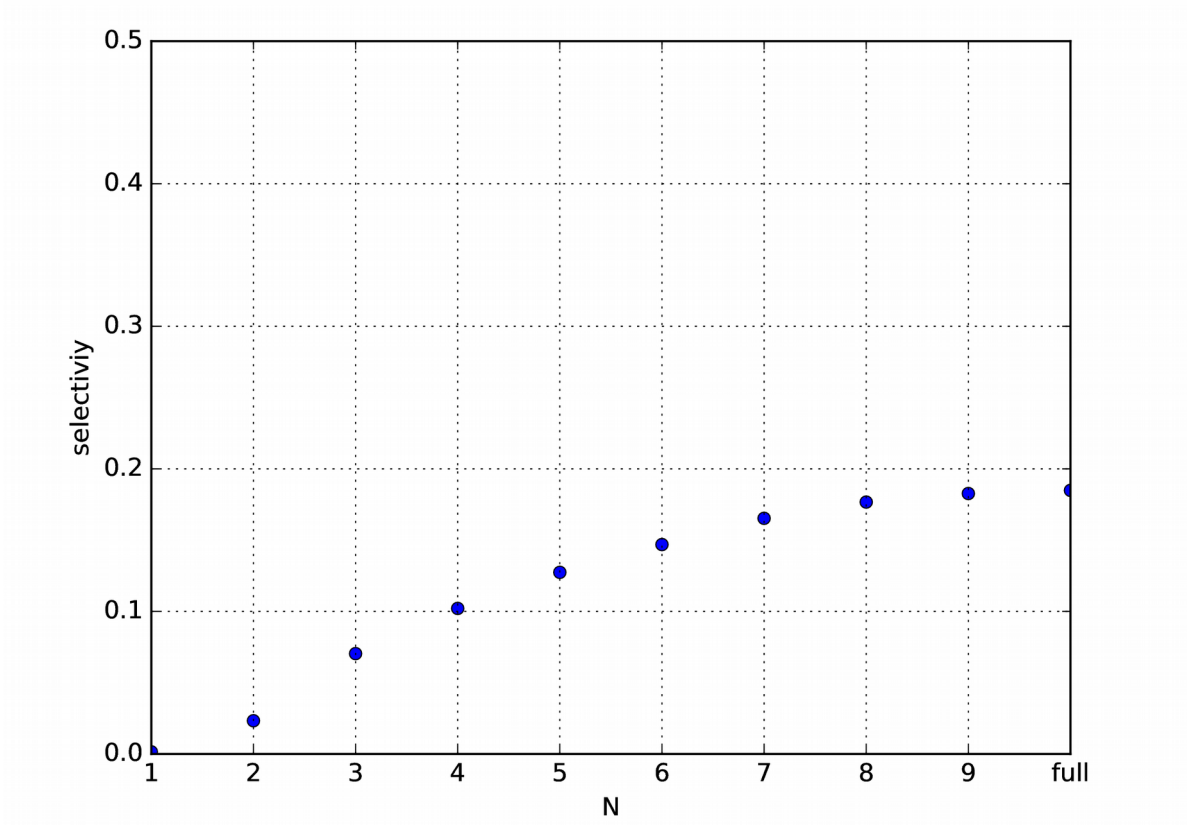

Figure 6: Selectivity for N leading characters of first names, calculated for grooms/brides and their mothers and fathers.

Compared to family names, selectivity is overall much lower for first names. This is explained by the fact that three instead of two names are considered (the family name of ego and father are the same). There are also less distinct first names then there are distinct family names (12.587 as opposed to 51.380) present in the dataset. Figure 6 shows that selectivity continues to grow and does not fully stabilize before the full name. Selectivity can therefore not be used in determining a good value for N, so the N-values were based on the cumulative frequency distribution alone.

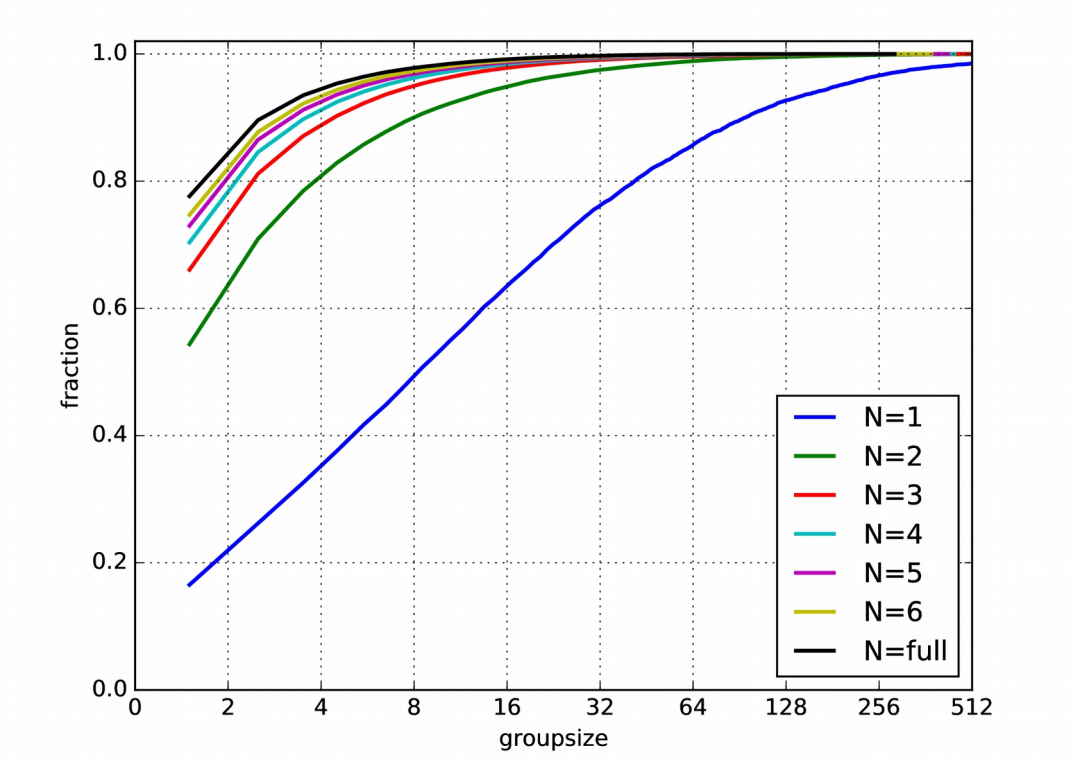

Figure 7: Cumulative frequency distribution for N leading characters of first names, calculated for grooms/brides and their mothers and fathers.

Figure 7 shows the cumulative frequency distribution for a number of leading characters of first names. Applying the same reasoning used for family names, Figure 7 suggests N=3 or N=4. N<3 show overall lower curves that stabilize late. Between N=3 and N=4 performance stabilizes. Based on this information, procedures utilizing prefixed first names with N=3 and N=4 have been tested.

Prefixing with N=3 reduces the amount of distinct first names from 12.587 to 1.690 Using N=4 resulted in 4.230 different first names.

Standardizing

As opposed to family names, there is a gender independent name-standard for first names available for Dutch. For example, the standard form of *Petronella* is *Petrus*. Female first names are often derived from their male counterparts (e.g. *Johan* and *Johanna*). The inclusion of gender as a field will have to distinguish between siblings sharing a standard name.
These standard forms are based on name-variants harvested in a record linkage method allowing up to one non-matching name. From these non-matching names, valid name-variants have been deduced [12]. The results were adopted into a reference table that maps 129.725 first names to 926 standard names. Using this table, first names were standardized, reducing the amount of distinct first names from 12.578 to 4.065. This means that for 3.139 names, no standard was present in the reference table. For 538.338 out of 1.752.283 records the non-manipulated first name is the same as it's standard form. This is caused by either the name not having a standard form (in which case the normal form is used) or the name already being the standard form.
Procedures utilizing the resulting standardized first names have been tested.

3.3 - Linking

Record linking-strategies generally use a technique to reduce the amount of candidate links that have to be inspected for each record. For example, only records sharing the initial of the first name might be checked. For the sorting-based method this selection is very simple: only the next x candidates are inspected for each record, with x being the number of established links for the record plus 1. Assuming the sorting is correct, only boundaries between persons (blocks of records) will have to be found in the linking phase. To easily identify all links concerning a single person, links were assigned a block number. Links were established in the following way:

For all records in sorted list:

1. Inspect record at index = i. c = i+1.
2. Check records at i and c for a link.
3. If (2) produces a link, store link and block number, repeat (2) with c = c+1.
4. If (2) does not produce a link, repeat (1) for i+n, where n is the amount of links found.

A link between record A and record B with threshold T is defined as:

1. The first name of ego must always match exactly.
2. Fields containing names match with <= T non-matching fields. T is set by the procedure. For example, in a procedure with T = 1, two records that differ only in the first name of the mother a link is accepted. With T = 0 the link is rejected.
3. Sex of A and B match.
4. If the procedure checks for date of birth, the difference between days since year 0 for birth of A and birth of B is less than 400. If one of the birth dates is missing, there is no match between A and B.
5. If the roles of A and B are both child or both deceased, the date of event (birth or death) must match exactly. In this case, the records are suspected to be duplicate entries.

The difference between birth dates was set at 400 to catch some errors in the original certificates. If an age and date of event is present, for example a groom was 25 when he marries on 14-6-1869, the range of birth date will be spread over 365-366 days (14-6-1843 to 14-6-1844). Expressed as days since year 0, this becomes 672940 to 673306, a range of 366 days (indicating a leap year).
Setting a limit of 366 days would mean no mistakes or errors in date of event or age is allowed when establishing links. An additional 34-35 days have been set, based on an educated guess, to allow for some error.
It is a possibility that two records of the same type exist for one person. For marriage this is obvious: a person can marry after being widowed. For birth and death it is possible that there are multiple records of the same event. For example, a person dying in one municipality and buried in another might have two death-certificates in two different archives. A strict rule (rule 5) is necessary to determine these cases. If this rule is not used, two siblings dying in their first year might be matched as the same person.

If a block of links (representing a person) contains more then one link, only the links between the record at index i and the records at higher indices are recorded and counted. For some purposes the mutual links between records at higher indices should also be considered. Blocks of links can be

'unpacked' to generate these links. For example, for a block of links concerning records A, B and C the algorithm records links A-B, A-C. After unpacking, the links are A-B, A-C, B-C. Links are considered symmetrical: if A-B then B-A. Therefore, only one link between two records has to be recorded and counted. Amount of links reported in this thesis are counted without being unpacked, unless stated otherwise.

3.4 - Procedures

Based on sections 3.1 and 3.2 a number of procedures have been constructed. These procedures all involve a single manipulation.

A procedure has a code of the form X-Y(-Z) :

X corresponds to the category of the procedure.
Y is the code given to a specific manipulation.
Z is the N-value for prefixing manipulations.

The categories X are:

1. Estimated birth dates must match within a specified range, no non-matching names are allowed.
2. Birth dates are ignored, no non-matching names are allowed.
3. Estimated birth dates must match within a specified range, one non-matching name is allowed (first name of ego must match)

The following procedures have been tested for all categories (X=1, X=2 and X=3):

X-1:	no manipulated fields
X-2:	standardized first names
X-3-4:	prefixed family names, N=4
X-3-5:	prefixed family names, N=5
X-4-3:	prefixed first names, N=3
X-4-4:	prefixed first names, N=4
X-5:	semi-phonetic family names
X-5-4:	semi-phonetic family names, N=4
X-5-5:	semi-phonetic family names, N=5

Based on the results of these procedures, discussed in sections 4.1 and 4.2, procedures combining manipulations have been constructed.

c1:	standardized first names,	prefixed(N=4) family names
c2:	standardized first names,	prefixed(N=4) semi-phonetic family names
c3:	prefixed(N=3) first names,	prefixed(N=4) family names
c4:	prefixed(N=3) first names,	prefixed(N=4) semi-phonetic family names

Motivation for the choice of manipulations is provided in section 4.3.

3.5 - Life courses

Being able to bring records related to an individual together allows for a range of applications. One of these is to reconstruct the life course of this individual from his/her collection of records. If this can be done for every individual in a population, a detailed reconstruction of this population is formed. While such a full reconstruction is beyond the scope of this research, it is worthwhile to investigate how well the current method lends itself to constructing it.

Evaluating the sorting-based method involved checking life courses (blocks of links) for consistency. However, these life courses so far involve only principal events (birth, marriages, death). Ideally, all events where the person in question (ego) is involved would be part of this life course.
The life course reconstruction then consists of two steps: gathering all events involving a person into a life course, and then analyzing and evaluating these life courses.

3.5.1 - Necessary records

Events that the sorting-based method did not include in a person's life course are:

- Birth of a child.
- Marriage of a child.
- Death of a child.
- Death of a partner.

Records detailing these events could not be linked because the necessary fields are often missing. When a child is born, only the names of the parents are recorded. A record of the parent of the child being born thus only has the information (deduced by inspecting records of the same certificate): name of ego, name of partner, name of the child born (parents being ego and partner). Additionally, the birth date of the child can be deduced. For children getting married or dying the same information is available, with age replacing the exact birth date of the child. This age is used in conjunction with the date of the event, to establish a range of possible birth dates. When a partner dies, only the names of ego and partner are available.

Combining these facts, a linking procedure will have to be performed which uses the fields name of ego and name of partner. Checking for birth dates is not possible since the birth date (or age, allowing a estimated birth date) of ego is not present on any of these records. Since the name of the partner is included, the resulting blocks of links will represent all events that took place in the marriage between ego and partner. The birth of ego is not included (since a partner is not present at birth, these records cannot be linked). The death of ego will sometimes be included. When a person remarries however, usually only the last partner will be mentioned on the death certificate, excluding the death of ego from life courses detailing his or her other marriages.

3.5.2 - Including birth and death

To include birth and death of ego into the life course, the results of earlier linking (based on ego, parents and possibly birth date: the *ego/parents/birth date-procedure*) will have to be used. A table will be constructed, including the serial number of each record (present in the original database), and an assigned ID. This assigned ID is the number of the block in which the record is linked. Only consistent blocks will be allowed to assign an ID. Records sharing an assigned ID then belong to the

same person.

For each block of links produced by the *ego-partner procedure*, a base record is selected. This record is the marriage between the ego and the partner. If a block does not contain such a record, it is excluded from further analysis. The reason for requiring a marriage between ego and partner is explained in the next section. With the serial number of this base record, the assigned ID of the person involved can be found in the table constructed for this purpose. Notice that only if a marriage is linked in both the ego/parents and ego/partner procedures this step will succeed. Now, each block of ego/partner-links is enriched with all records of birth and death that share the same assigned ID. In other words, the birth and death of ego are added to the block. It is possible that the death of ego was already present, but a later check accounts for this fact.

The resulting blocks represent the life course of the ego, and the events of his/her marriage with the partner. It is possible that several of these life courses exist for one person (in case of remarriage).

3.5.3 - Checking for consistency

The complete life courses are analyzed and checked for consistency. This is done through a number of steps:

1. Does the block contain the marriage of ego and partner? If not, the block is excluded from further analysis in this thesis.

2. If the block contains multiple marriages of ego and partner with the same names, the block is classified as having multiple marriages and is excluded from further analysis. In these cases it is likely that the block represents multiple couples sharing names.

3. If the block contains multiple births of ego, and the registration dates of these births do not match, the block is classified as having multiple births and is excluded from further analysis. In these cases it is likely that the block represents multiple couples sharing names.

4. If the block contains multiple deaths of ego, and the registration dates of these deaths do not match, the block is classified as having multiple births and is excluded from further analysis. In these cases it is likely that the block represents multiple couples sharing names.

5. If the block contains birth of ego, but this birth is not chronologically the first event, the block is classified as inconsistent.

6. If the block contains death of ego, and ego actively participates in events after this death, the block is classified as inconsistent. This is the case when a person marries or is born after his/her death.

After these preliminary checks, each remaining block is checked for the following:

- For each child born, the registration date of the birth has to fall within 25 years after the marriage of ego.
- For each child marrying, the estimated birth date of the child has to fall within 25 years after the marriage of ego.
- For each child dying, the estimated birth date of the child has to fall within 25 years after the marriage of ego.

If one of these checks fails for even a single record, the block is classified as inconsistent. If the checks pass the block is classified as consistent. Children being born outside of the limit are likely

children of a different couple sharing names with ego and partner. The limit of 25 years is an educated guess for when couples realistically have children. Analysis of the results of these checks will have to show if this guess is justified.

Since this method will produce a life course of an ego in combination with a single partner, events of other marriages are excluded. These can be found by inspecting blocks of the ego/parents/birth date-procedure. For each block containing multiple marriages, the life courses (from the ego/partner-procedure) containing those marriages are collected. Combining these life courses would result in the full life course of an individual.

3.6 - Implementation

The original data is stored in a MySQL-database, spread over multiple tables. While it would be possible to perform manipulation, sorting and linking all through SQL-statements, the choice has been made to export the data to a faster and more flexible format. Code was written in Python, importing relevant data into dataframes implemented by the package Pandas [16]. This structure, akin to the dataframe found in the R programming language, allows for easy extraction, manipulation and insertion of data. Python was chosen for the availability of easy-to-use scientific packages such as NumPy [17] for analysis of results. Runtime of operations is also considerably faster than comparable SQL-statements. Part of the developed code is presented in the appendix.

4 - Results and discussion

Ideally, the results produced by the different procedures of the method are evaluated against a *gold standard.* Such a standard consists of a subset of the data that has been linked and the links are verified to be correct. The method's results are then checked for the inclusion of these verified links. With such a comparison, precise scores for *recall* and *precision* can be calculated.
Recall is the amount of correct links that are found divided by the total amount of correct links, precision is the amount of correct links found divided by the total amount of links found. These scores can then be combined into an *f-measure*, a measure that considers both recall and precision to produce a single score [18]. This score could be compared with other methods. F-scores for a number of methods have for example been computed in [19].

Unfortunately, there is no gold standard for this particular dataset, and other ways to interpret the results have to be employed.
To gain insight into the results, they have been analyzed in a number of ways: in section 4.1 results between procedures are compared, in section 4.2 blocks of links are inspected for inconsistencies, in section 4.3 combined procedures are constructed and tested, in section 4.4 non-linked records are analyzed, in section 4.5 results are compared to the method used in the LINKS-project, in section 4.6 the results of the life course reconstructions are analyzed and in section 4.7 the computation time of the method is discussed.

4.1 - Comparing procedures

The first measurement is the total number of links resulting from each procedure. These numbers do not offer insight into the recall of each procedure, as the amount of links that can possibly be found is unknown. However, they can be used to evaluate procedures against other procedures within the same method.

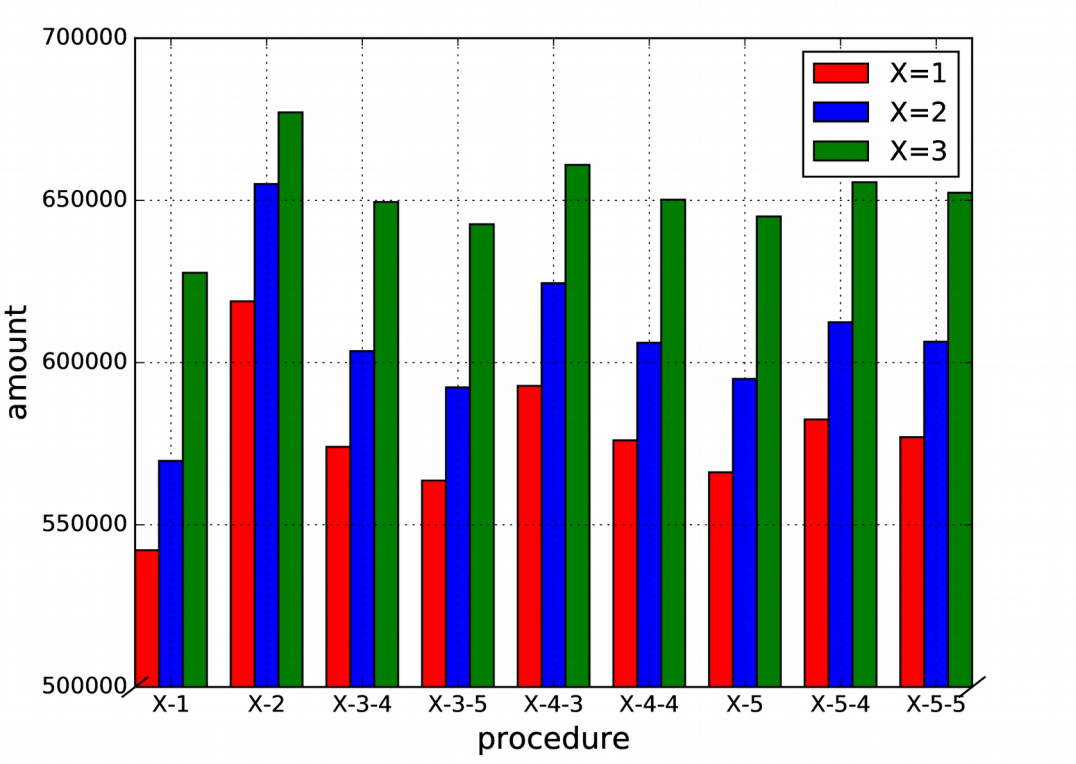

Figure 8: Number of links produced by procedures.

Figure 8 shows that, taking the strict category 1 as a baseline, procedures ignoring date of birth (category 2) produce more links for all manipulations.
Procedures of category 3, using limited birth ranges and allowing one non-matching name, are more productive then category 2 for all procedures. For procedures manipulating first names the increase over category 2 is smaller than for procedures manipulating family names. One out of three first names (that of ego) is not allowed to be non-matching. Allowing up to one non-matching field therefore has relatively less of an impact on first names.
For all procedures utilizing prefixing (X-3-N, X-4-N, X-5-N), the lower value for N produces more links.

Links of all procedures were analyzed to give insight into which non-matching fields were successfully bridged by the procedure. In order to analyze all links, blocks for each procedure have been unpacked (see section 3.3). In a large amount of cases (542.175), both records of a link match exactly. These cases are found by procedure 1-1. Since analysis of this type of links is not useful, links found by procedure 1-1 have been subtracted from the results of each procedure. This leaves the cases where non-matching fields exist in every link. For each link, the non-manipulated records were inspected for non-matching fields. For example, if two records contain a variant of the same first name for ego, and a procedure was still able to link it, the first name of ego is counted as a non-

matching field. Table 5 below shows the results of this analysis.

Procedure	Additional links	Avg. non-Matching fields	el	ef	ml	mf	ff
1-1	-	-	-	-	-	-	-
1-2	75053	1.69	-	51.29%	-	61.55%	56.40%
1-3-4	43144	1.05	40.03%	--	64.60%	-	-
1-3-5	29023	1.04	39.34%	-	64.32%	-	-
1-4-3	69438	1.13	-	34.42%	-	39.84%	38.72%
1-4-4	46160	1.09	-	33.21%	-	34.16%	41.17%
1-5	32676	1.05	42.45%	-	62.08%	-	-
1-5-4	54836	1.06	41.64%	-	64.49%	-	-
1-5-5	47378	1.06	41.75%	-	63.87%	-	-
2-1	89821	-	-	-	-	-	-
2-2	240876	1.25	-	50.85%	-	38.84%	35.19%
2-3-4	139529	1.64	43.14%	-	66.26%	-	-
2-3-5	122977	1.08	41.92%		66.25%	-	-
2-4-3	178964	1.17	-	43.79%	-	37.15%	35.93%
2-4-4	147833	1.12	-	41.32%	-	32.23%	38.59%
2-5	127556	1.09	44.64%	-	63.99%	-	-
2-5-4	153555	1.11	44.59%	-	66.23%	-	-
2-5-5	144570	1.10	44.27%	-	65.73%	-	-
3-1	120332	0.96	-	-	35.07%	33.22%	28.00%
3-2	188596	1.25	-	25.89%	30.44%	36.66%	32.51%
3-3-4	149787	1.07	14.88%	-	14.88%	30.19%	25.50%
3-3-5	140528	1.04	10.70%	-	35.81%	30.96%	26.11%
3-4-3	165883	1.16	-	19.40%	30.93%	34.96%	30.68%
3-4-4	151040	1.10	-	14.20%	31.82%	33.89%	29.97%
3-5	144015	1.05	12.92%	-	35.45%	30.76%	25.95%
3-5-4	158295	1.10	18.64%	-	36.18%	29.79%	25.10%
3-5-5	153868	1.08	16.82%	-	36.01%	30.05%	25.30%

Table 5: Analysis of links per procedure, compared to procedure 1-1. For each field, the percentage of additional links where manipulation is needed is shown. Percentages can sum to >100% in case the number of non-matching fields is larger then 1.

Each procedure involves either manipulating first names or family names. For procedures manipulating first names, no non-matching family names are bridged, and vice versa. These cases are represented by a '-' in the table. Note that for procedures of category 3, this does not hold because these allow one non-matching field after manipulation. For example, if all manipulated family names match, and the first name of the father does not, procedure 3-3-4 can still establish a link (the non-matching first name for father is ignored).

For category 2, the average number of non-matching fields is much smaller then for category 1, with most links not even containing a non-matching field. This indicates that the additional links found by this category consist mostly of records where all fields match, but the estimated birth dates differ too much (category 1 would not accept the link). Errors in age or date of event can lead to a

badly estimated birth date. These cases can be linked by procedures of category 2, and categories 1 and 3 would reject them. However, as discussed in section 3.1, birth date is necessary to distinguish multiple children of the same parents. A large amount of additional links are probably incorrect, as they link records for siblings, outweighing the gain from records with bad estimated birth dates.

Procedures of category 3 produced more links compared to their counterparts in category 1. The average number of non-matching fields for each link is only slightly higher. If two fields are non-matching in their non-manipulated form, and only one of them is equal after e.g. standardizing, procedure 1-2 would not establish a link. Procedure 3-2 would still find the link since the remaining non-matching field falls within the allowed threshold of 1. In cases like this, the amount of non-matching fields is higher for 3-2 than it is for 1-2. The fact that on average the number of non-matching fields is only slightly higher indicates that more often the threshold of category 3 will cause non-matching fields to match, where manipulation failed to do so. For example, if all fields on a record match instead of first name of father (*Johannes* and *Joehannes).* Prefixing with N=3 (*Joh, Joe)* will not allow procedure 1-4-3 to establish a link. Here, the threshold of procedure 3-4-3 allows the link to be established by ignoring this field.

While percentages of non-matching fields are lower for category 3 then for their counterparts of category 1, the amount of additional links is much larger. For example, for procedure 1-4-4 ~40% of links contained non-matching family names for ego. For procedure 3-4-4 this is ~15%. However, the amount of additional links is much smaller (~40.000 as opposed to ~150.000). The absolute amount of links containing non-matching family names for ego are therefore larger for procedure 3-4-4.

4.2 - Finding inconsistencies

As explained in the beginning of chapter 4, a precise score for precision can not be calculated because there is no golden standard for the data. Theoretically it is possible to manually inspect a number of links and decide if they are correct or incorrect. However, a large fraction of the links are between records that are very similar, and an important part of the performance of a record-linkage method is how it deals with more difficult cases. To judge these, the inspected subset of links has to be very large. Even if such a number is inspected by humans, it is not always clear when a link is correct.

Since there is no way to compare with verified links, precision will have to be predicted. A way to make this prediction is deciding in how many cases the method obviously went wrong. A block of links generated in the linking phase should contain all records concerning a person, assuming the ordering of records was correct and variations in fields have been correctly ignored or made equal by manipulations. These blocks have been analyzed. A block of links is considered inconsistent if the life course constructed by it is impossible. Each record in a block was checked for type and date of the event in question. The block must then produce a chronologically correct sequence of events: birth, one or more marriages, death. If there are multiple records for birth or death, the dates of events must match exactly. If an inconsistent block is found, this indicates that this block contains one or more records that do not belong to the person the block represents. Another possibility is that there is an error in one of the dates, e.g. a person is recorded to have died in 1949, while he actually died in 1948. The estimated birth date is based on age, so this record of death could still be sorted near other records for this person.

Table 6 on the following page shows that within a category the consistencies for each procedure are highly stable. If a procedure finds more links, a constant percentage of blocks containing new links will be consistent. Average size of the blocks increases with the amount of links found: when more links are found, blocksize will be slightly larger on average. Size of the blocks is quite small, which is explained by the fact that a large fraction of links are expected to be between records of birth and death. Child mortality was high and even if a child does not die young, it can still stay unmarried. In these cases, only two records exist for one person. The block representing this person then has size 1, as it contains only 1 link.

Links found by procedures of category 1 are expected to have very high precision, because the rules for establishing a link are very strict. Table 6 shows that this is indeed the case, with very few inconsistent blocks.

When ignoring birth date (category 2), inconsistencies rise significantly. The linking algorithm is unable to distinguish one person from another if they and their parents have identical or similar names. This effect can clearly be seen in procedure 2-2, with an inconsistency-rate of over 8%. Two children of the same parents will often be named after a parent or grandparent, with name-variations being possible. When standardizing these names, the only way to distinguish different children is through birth date. Inconsistency-rates of 6.8-8.2% are unacceptable since too many links will be false positives.

Procedures in category 3 produced similar results for blocksize and consistency-rate to those in category 1, while allowing for a significant increase in the amount of links.

Procedure	Links	Blocks	Average Blocksize	Consistent %	Inconsistent %	Consistent Blocks	Inconsistent Blocks
1-1	531226	418700	1.269	99.575%	0.425%	416922	1778
1-2	606279	469636	1.291	99.511%	0.489%	467341	2295
1-3-4	562282	440335	1.277	99.566%	0.434%	438426	1909
1-3-5	552137	433273	1.274	99.570%	0.430%	431408	1865
1-4-3	580731	452697	1.283	99.546%	0.454%	450640	2057
1-4-4	564329	441658	1.278	99.558%	0.442%	439708	1950
1-5	554679	434950	1.275	99.567%	0.433%	433066	1884
1-5-4	570561	445945	1.279	99.560%	0.440%	443983	1962
1-5-5	565204	442214	1.278	99.563%	0.437%	440283	1931
2-1	571140	430163	1.328	93.183%	6.817%	400837	29326
2-2	656739	476298	1.379	91.740%	8.260%	436958	39340
2-3-4	605086	452064	1.338	92.974%	7.026%	420302	31762
2-3-5	593858	444863	1.335	93.069%	6.931%	414028	30835
2-4-3	626086	462425	1.354	92.577%	7.423%	428099	34326
2-4-4	607685	452218	1.344	92.822%	7.178%	419758	32460
2-5	596504	446246	1.337	93.052%	6.948%	415241	31005
2-5-4	614005	457344	1.343	92.912%	7.088%	424929	32415
2-5-5	607965	453497	1.341	92.985%	7.015%	421682	31815
3-1	627677	487234	1.288	99.456%	0.544%	484584	2650
3-2	677112	520507	1.301	99.408%	0.592%	517427	3080
3-3-4	649502	502293	1.293	99.450%	0.550%	499531	2762
3-3-5	642649	497572	1.292	99.454%	0.546%	494855	2717
3-4-3	660957	509757	1.297	99.431%	0.569%	506859	2898
3-4-4	650211	502651	1.294	99.438%	0.562%	499825	2826
3-5	645032	499050	1.293	99.457%	0.543%	496339	2711
3-5-4	655595	506285	1.295	99.449%	0.551%	503494	2791
3-5-5	652363	504094	1.294	99.452%	0.548%	501334	2760

Table 6: Analysis of blocks of links for procedures.

4.3 - Combined procedures

As mentioned in section 3.4, combined procedures have been constructed. For each type of field (first name, family name) the two manipulations producing the most links have been used. Standardizing first names shows promising results: it is the manipulation producing the most links regardless of category of procedure. The second best manipulation for first names is prefixing with N=3. The most productive manipulations for family names are prefixing with N=4 and the prefixed semi-phonetic form with N=4.
Combining these manipulations led to the following combined procedures:

c1:	standardized first names,	prefixed(N=4) family names
c2:	standardized first names,	prefixed(N=4) semi-phonetic family names
c3:	prefixed(N=3) first names,	prefixed(N=4) family names
c4:	prefixed(N=3) first names,	prefixed(N=4) semi-phonetic family names

These combined procedures have been evaluated for category 3, allowing one non-matching name. This category was chosen because it increases the amount of links while maintaining high consistency, making it a strict improvement over category 1. Procedures of category 2 produce too many inconsistent blocks of links.

Procedure	Links	Blocks	Average Blocksize	Consistent %	Inconsistent %	Consistent Blocks	Inconsistent Blocks
3-c1	700010	522075	1.341	99.441%	0.559%	519159	2916
3-c2	706883	526264	1.343	99.442%	0.558%	523325	2939
3-c3	683695	511573	1.336	99.470%	0.530%	508862	2711
3-c4	690279	515622	1.339	99.471%	0.529%	512895	2727

Table 7: Analysis of blocks of links for combined procedures.

Comparing tables 6 and 7 shows that the amount of links increases when combining procedures. Average blocksize increases very slightly and consistency remains stable. This signifies that combining manipulations leads to more correct links then individual manipulations.

To analyze the results of these procedures, the blocks of links of all combined procedures were unpacked (see section 3.3). The links of all combined procedures were then aggregated and duplicates were removed. The resulting 265.012 links (with at least 1 non-matching field) have been analyzed. On average, links contained 1.3 non matching fields.

Field	Amount
family name ego	42961
first name ego	61709
family name mother	82478
first name mother	84144
first name father	73344
total	345036

Table 8: Amount of non-matching fields (before manipulation) for links of all combined procedures, split by type of field.

Non-matching fields	Amount	Percentage
1	180907	72.36%
2	47225	18.89%
3	16392	6.56%
4	4824	1.93%
5	646	0.26%

Table 9: Amount of non-matching fields (before manipulations) for all combined procedures.

Table 8 shows that names of the mother are more often non-matching. The ego's first name matches most often out of all three first names. This is unsurprising, as all combined procedures allow one non-matching field after manipulation, but the first name of the ego is excluded from this rule. Non-matching occurs about half as often for family name of ego as for family name of mother. Since the family name of ego is the first field in the sorting-order, records that do not match for this field will only be sorted together if manipulations of fields eliminate or minimize the difference between these records. The family name of the mother is placed further in the sorting-order, allowing more non-matching links.

Table 9 shows significant amounts of cases where more than one field was non-matching between records (one non-matching field is allowed, so manipulation is not necessarily responsible for bridging one non-match). This indicates that the applied manipulations were able to make these fields equal, showing the benefit of using manipulations in a record-linkage method.

Levenshtein	**el**	**ef**	**ml**	**mf**	**ff**
1	83.70%	61.16%	72.12%	57.67%	50.54%
2	13.16%	17.14%	18.47%	14.96%	16.64%
3	1.95%	10.62%	4.97%	8.63%	12.64%
4	0.50%	5.08%	1.88%	5.56%	6.28%
5	0.19%	4.61%	0.87%	4.33%	8.27%
6	0.11%	0.73%	0.53%	3.91%	2.52%
7	0.08%	0.60%	0.34%	2.34%	2.10%
8	0.07%	0.04%	0.26%	1.52%	0.75%
9	0.05%	0.01%	0.16%	0.87%	0.17%
10	0.05%	0.00%	0.13%	0.11%	0.04%
>10	0.14%	0.00%	0.25%	0.10%	0.04%

Table 10: Distribution of Levenshtein-distances for links of all combined procedures, split by field.

Table 10 shows that for first names of ego and mother, larger Levenshtein-distances between fields are found than for family names. This is true even for ego, for which the first name is not allowed to be ignored by procedures of category 3.

Combining these observations, the sorting-based method is able to bridge small differences (Levenshtein 1-2) in many cases. Additionally, a significant amount of links where larger differences are found in the non-manipulated records are also established. This shows the strength of manipulating fields.

Because the combined procedures of category 3 are shown to perform the best (finding more links while remaining consistent), they were chosen to evaluate the sorting-order that was proposed in section 2.1. Two alternative sorting-orders have been tested:
'_father' moves the first name of the father forward, positioning it after full name and sex of ego, and before the full name of mother.
'_switch' switches the names of the full names around: *first name* | *family name* as opposed to the original *family name* | *first name*.

Procedure	Links	Blocks	Average Blocksize	Consistent %	Inconsistent %	Consistent Blocks	Inconsistent Blocks
3-c1_father	702093	537529	1.306	99.398%	0.602%	534295	3234
3-c2_father	709172	542083	1.308	99.398%	0.602%	538822	3261
3-c3_father	682318	524590	1.301	99.426%	0.574%	521580	3010
3-c4_father	688937	528902	1.303	99.428%	0.572%	525876	3026
3-c1_switch	700276	536177	1.306	99.410%	0.590%	533011	3166
3-c2_switch	707044	540485	1.308	99.410%	0.590%	537295	3190
3-c3_switch	680200	523112	1.300	99.441%	0.559%	520187	2925
3-c4_switch	686689	527344	1.302	99.441%	0.559%	524395	2949

Table 11: Analysis of blocks of links for combined procedures, using alternative sorting-orders.

Comparing table 7 with 11, the alternative sorting-orders shows a very small increase in links for procedures 3-c1 and 3-c2, with a very small decrease for 3-c3 and 3-c4. Consistency is equal to the procedures using the original sorting-order. Table 6 showed that average block size correlates with the amount of links found. Average blocksize for alternative sorting-orders is overall significantly lower then for the original sorting-order. To justify spreading the links over a larger amount of blocks (and thus more persons), the procedures using the original sorting-order would have to produce more inconsistent blocks. Because this is not the case, it can be deduced that the alternative sorting-orders do not succeed in finding all links for a person and assemble them into a block. For example, if 4 records are linked into one block using the original sorting-order, using one of the alternative sorting-orders might find one block of three records and link the last record into another block. Based on these observations, the assumption that ordering fields should be based on their descending importance and ascending uncertainty is found to be correct.

4.4 - Non-linked records

Analyzing the records that have not been linked by the method can give insight into performance. If the method establishes links for all cases that are expected, the records that have not been linked should show explicable trends. 481.305 records were not linked in any of the combined procedures, on a total of 1.752.283 records. A percentage of 27.4% of all records not being matched seems high, but a link is not expected for every record. An analysis of non-linked records has been made, providing an explanation for them not being linked to any other records.

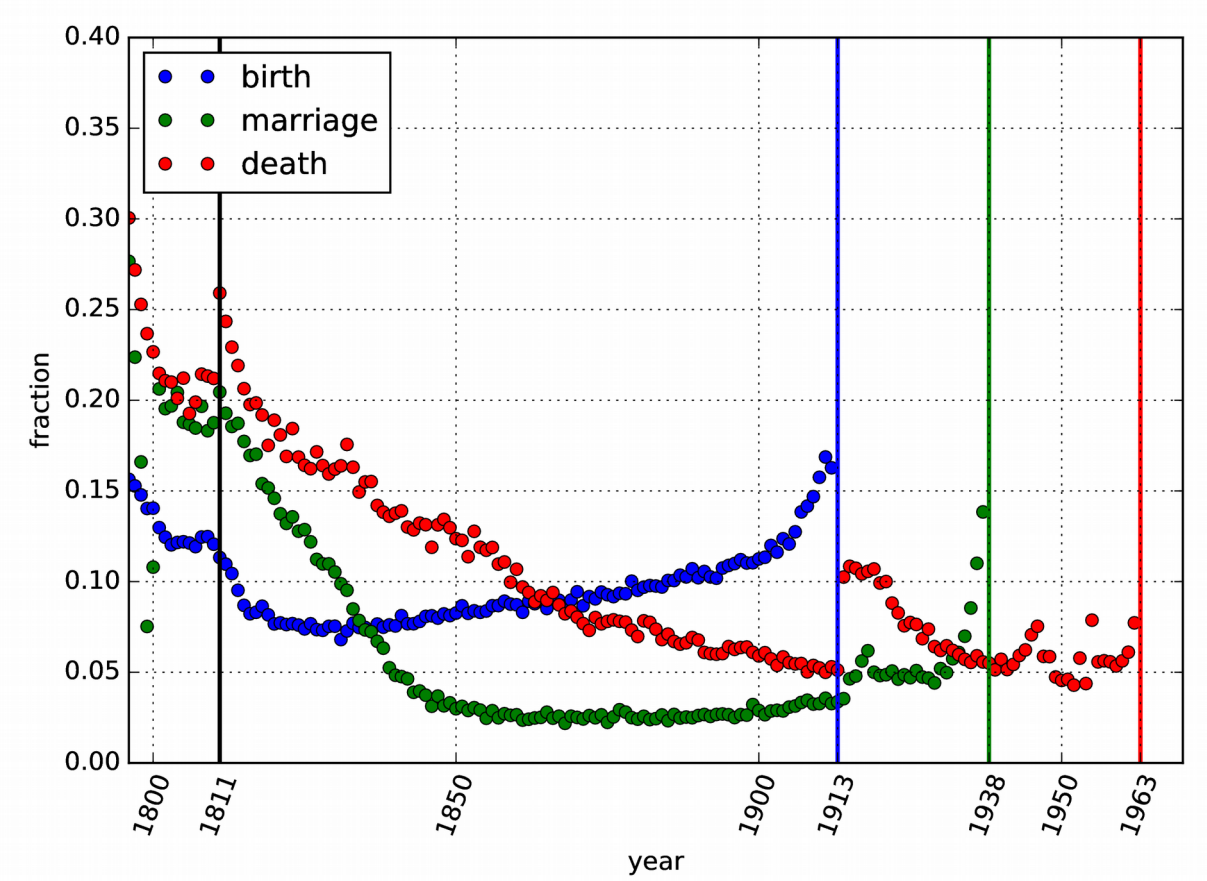

Figure 9: Non-linked records by year between 1795 and 1965. Colored vertical lines represent cut-off point for corresponding type of certificate.

Overall, not every record is able to be linked. Migration accounts for a number of non-linked records. If a person migrates into or out of the province of Zeeland, part of his records will be kept in other provinces, making them unavailable for linking. Records can also go missing for a plethora of reasons (e.g. archives burning down, records lost when moving archives) leaving records of the same person unable to be linked to them. These reasons account for a constant fraction of non-linked records, as it is suspected that these factors do not have large fluctuations over time. This leaves trends in non-linked records to be explained.

Figure 9 shows large amounts of records before 1811 have not been linked to any other records. The explanation for this fact is two-fold. Since civil registration had not started everywhere at this time, a lot of records are not present, so the number of non-linked records is expected to be higher. A person being born in a municipality where no records are being kept and then marrying or dying in

one where they are will produce a record that is unable to be linked to a birth certificate. Since records start in 1796 at the earliest and in 1811 generally, people of whom earlier records are non-existent can be found in marriage or death-certificates. This also explains the larger amount of deaths and marriages not being linked in the early years of civil registration.

The amount of births that are not linked to any other records quickly declines after 1811, stabilizing around 1820. From this point, the curve shows a slow increase. More births will not be able to be linked to deaths since an increasing number of children born will survive until the point where death certificates are no longer available (1963). This effect is increased by life expectancy increasing rapidly in the 19th century. In later years children will also not be of age of marriage at the cutoff point for marriage certificates, further increasing the amount of non-linked births. The rate of non-linked for birth certificates is consistent with what is to be expected.

Non-linked marriages show a steeply declining curve stabilizing around 1850. At this point in time, if a person was born in 1811, this person getting married would be 49 years old. Missing records from before 1811 can account for a decent chunk of missed links, but the late stabilization of non-linked marriages might be caused by flaws in the method. After stabilizing, around 2.5% of marriages are not linked. Considering each marriage consists of two records (bride and groom) this is a good result. A steep incline is seen as the cutoff-point is approached (1938). This is explained by births after 1913 not being available, so for every person older then 25 at this time no birth certificate can be found. As the cutoff-point for death certificates lies 25 years later, for everyone living for at least 25 years after marrying no death certificate can be found.

The amount of non-linked records of death show a declining curve after the expected high fractions before 1811. This curve is expected to decline, since for people dying at higher ages there will be no birth certificate present: this certificate would have to be drafted before 1811. Like with birth, higher life expectancy also contributes to a declining curve. There is a sudden and sharp increase of non-linked records of death occurring at 1914. After 1918 this amount declines again. This sudden increase can be explained by a large intake of Belgian refugees fleeing World War I. These refugees would not have any previous records in Zeeland, but records of their deaths are present. The later years of World War II also show an increase in unlinked deaths. It is possible that this is explained by a similar influx of refugees: the southern part of the Netherlands (including Zeeland) was liberated earlier then the rest of the country.

4.5 - Comparing results

Results of the sorting-based method have been compared with those of another method utilizing the same dataset. The LINKS-project aims to reconstruct all families in the Netherlands in the 19th and early 20th century. To this end the project has developed a record linking method. Comparison to this method is useful since the data used in this research was compiled and cleaned by the LINKS-project, making the dataset the same for both methods. LINKS attempts to find links for all of the Netherlands, but for this comparison only links found for the province of Zeeland have been used.

The LINKS-method utilizes a linking-algorithm that uses edit-distance techniques to bridge differences between non-matching fields.

The method of the LINKS-project is split into processes, comparable to the procedures of the sorting-based method. These processes link between certain type of records, and is not restricted to linking only egos of certificates. Only results of processes linking ego to ego have been collected: links found by linking the record of the mother of a child to a record of a bride would obviously not be present in the sorting-based method, as this method only links ego with another ego. The resulting collection of the LINKS-method contained 712.900 links.
The unpacked blocks of links for all combined procedures of the sorting-based method were again used. This aggregate contains 918.086 links.
It is possible that the large difference in amount of links found is explained by the way links are counted. Manual inspection is required to decide what causes this difference.

Links that are present in the results of one of the two methods, but not the other, have been harvested. This process was executed both ways. Of the resulting links, a number have been manually inspected and analyzed.

31.773 links were found by the LINKS-method and not by the sorting-based method. 20 of these links have randomly been selected for inspection. For each category of missed link, an example is given in Table 12**.**

1. In 8 out of 20 cases, multiple fields containing names of parents were missing in one of the records. This occurs mostly at death, in one case at marriage. The parents not being known at death can be the consequence of nobody reporting the deceased persons parents. At marriage, the bride or groom might have broken relations with the parents. If only one name would be missing, procedures of category 3 would still have established a link. Special rules for missing names of parents would have caught these missed links. For example, if both first name and family name of the mother are missing, this might be counted as 1 non-matching field, still allowing procedures for category 3 to establish a link.
2. In 4 out of 20 cases, an estimated birth date was not present, or the estimate was an obvious error (being way too low or high). The linking phase of the sorting-based method rejects these cases because it cannot be sure of a link. Since dropping the requirement for matching birth dates leads to high inconsistency-rate in the sorting-based method, special rules for these cases are necessary.
3. In 5 out of 20 cases, one name did not match exactly. While the linking phase would have ignored one field of difference, the sorting phase might not have placed them together. Incidentally, in 3 out of 5 cases the first character of the name was different. This is a downside of the sorting-based approach. Note that for the example in Table 12 for this category, all names are quite common and the family name of the mother is completely

different. It is possible that the mother has a multi-part family name (*Neeltje Adriaanse Walrave)* of which she used only one part.

4. In 2 out of 20 cases, the sex did not match. In both cases, the fields contained an obvious error: making *Jan* a woman and *Maria* a man.
5. The remaining link has all fields match with estimated birth dates very close together. These records should have been linked, and it is improbable that another record would be sorted between them. This might indicate a bug in the sorting-based method.

category	el	ef	sex	ml	mf	fl	ff	bd	role
1	goossens	johanna	f	null	null	goossens	null	666925	4
	goossens	johanna	f	kever	amelia	goossens	alphonsius	669531.5	10
2	kraker	maria	f	herrebout	janna	kraker	pieter	684857	1
	kraker	maria	f	herrebout	janna	kraker	pieter		10
3	pietersen	laurina	f	walrave	neeltje	pietersen	pieter	653488.5	4
	pietersen	laurina	f	adriaanse	neeltje	pietersen	pieter	653306.5	10
4	jonge	jan	f	reijnhout	cornelia	jonge	frans	664715	1
	jonge	jan	m	reijnhout	cornelia	jonge	frans	664719.5	10
5	luteijn	david	m	meulen	sara	luteijn	pieter	695071	1
	luteijn	david	m	meulen	sara	luteijn	pieter	695044	7

Table 12: Examples of links found by the LINKS-method and not by the sorting-based method. The codes for roles are: 1 for child, 4 for bride, 7 for groom and 10 for deceased.

These examples make clear that, while some might be incorrect links found by the LINKS-method, at least a part of the 30.000 links not found by the sorting-based method could have been found by adding more rules to deal with fringe cases.

The sorting-based method found 236.959 links that the LINKS-method did not find. Again, 20 randomly selected links have been manually inspected. These links fell into one of the following categories:

1. In 7 out of 20 cases linked records concerned children that died at a young age. LINKS reported that the results made available for this research were not complete for people under 17. This fact explains these missed links. On 4 of these cases some fields were non-matching as well, this might also have led to missing the link.
2. In 7 out of 20 cases, there were 1 to 4 fields that did not match exactly. Table 13 shows an example of a link with 4 non-matching names. Bridging these large differences shows the effectiveness of the manipulations performed: since only 1 non-matching name is allowed at least 3 have been made equal by manipulations.
3. In 5 out of 20 cases links between records of the same type were found. Two of these links concerned two records of death. Since the sorting-based method is very strict when deciding on a link between same-type records, these appear to be two records of the same event occurring involving the same person. It is possible that the LINKS-method applied cleaning steps to remove any of these 'duplicate' records, or avoided linking same-type records altogether. The other three links are between bride-bride and groom-groom. One of these appears to be two records concerning the same event, the other two are not duplicates since the age at date of event is different between records. These are in all probability two records

of the same person getting married and remarried.

4. The remaining link concerns the birth of a woman being born and her death at age 83. All fields match and estimated birth dates are consistent with dates found. The reason for the LINKS-method not finding this link is unclear.

category	el	ef	sex	ml	mf	fl	ff	bd	role	age
1	gouloze	jacobus	m	slimmen	geertruid	gouloze	david	670295	1	
	goulooze	jacobus	m	slimmen	geertruid	goulooze	david	670070.5	10	3
2	cappendijk	petronille	f	verstreepen	apollonie	cappendijk	joseph	661935	1	
	cappendijk	petronella	f	verstrepen	apolonia	cappendijk	josephus	661842.5	10	50
3	kips	honore	m	croon	maria	kips	augustinus	681157.5	7	30
	kips	honore	m	croon	maria	kips	augustinus	681048.5	7	42
4	page	pieternella	f	moerland	geertrui	page	pieter	678133	1	
	page	pieternella	f	moerland	geertrui	page	pieter	677754	10	83

Table 13: Examples of links found by the sorting-based method and not by the LINKS-method. The codes for roles are: 1 for child, 4 for bride, 7 for groom and 10 for deceased.

4.6 - Life courses

Life courses, as described in section 3.5, were constructed and evaluated.

Procedure *p-c2* has been executed on the dataset. This procedure uses the fields:

family name ego | first name ego | family name partner | first name partner

The procedure uses standardized first names and prefixed(N=4) semi-phonetic family names, and does not utilize birth dates.
This resulted in 3.490.773 links divided into 500.552 blocks. These blocks were enriched with births and deaths of ego, as described in section 3.5.2. Blocks of procedure *3-c2* were used because this procedure produces most links while consistency is equal to other combined procedures. A total of 148.451 records of birth and death were added. The resulting blocks represent a life course (here defined as the birth and death of a person plus the events of one marriage) and have been analyzed using the steps outlined in section 3.5.3.

207.840 blocks contained no marriage of ego and partner. These blocks were excluded from further analysis, since a marriage is necessary to decide on consistency.
228 blocks contained multiple births of ego (with different registration years), 457 blocks contained multiple deaths of ego (with different registration years) and 4.738 blocks contained multiple marriages of ego and partner. The dates of the marriages did not match, so they refer to different people sharing the same names. These blocks were excluded from further analysis.
The two steps checking for chronological consistency were executed simultaneously. 311 blocks were classified as inconsistent because the birth of ego was not the first event, or because the ego actively participated in events after his/her own death.

287.275 blocks remained to be analyzed. The births, marriages and deaths of the children of ego and partner were checked. The exact (where possible) or estimated birth date of these children has to fall within 25 years after the marriage of ego and partner.
For 41.277 blocks (14.48%), this was not the case, and they were classified as inconsistent. 107 of these were classified as inconsistent because a birth of a child caused an inconsistency, 196 because a marriage of a child caused the inconsistency, and in 40.974 cases the death of a child was the cause of inconsistency. The large difference between the three types of records raised suspicion: the method might not have worked correctly in these cases. Upon inspection of a number of these cases, it was concluded that the calculation of the age of stillborn children (or children dying soon after childbirth) was off in some cases. Ranges of 100 years were calculated, as if nothing was known about the age of the child. In cases where in the original date age in years was 0, and *age_literal* (a field indicating exactly what is written on the certificate) was a variation of *n.v.t (niet van toepassing*, Dutch for *does not apply*), the estimated birth year was changed to be the year of registration. It is possible that more cases have wrong estimated birth dates, but this was the only combination of fields where it was beyond doubt that the child died before reaching the age of 1.

After changing the estimated birth dates for these cases, the amount of inconsistent blocks caused by death of a child dropped from 40.974 to 1.078, netting a total of 1381 inconsistent blocks (0.59%).
The remaining 245.687 blocks (99.41%) were classified as consistent. Figure 10 on page 38 shows an example of a consistent life course.

Inconsistency is overall lower for the ego/partner procedure than for the ego/parents/birth date-procedure. This is unexpected, as the amount of fields is lower, making combinations of these fields less unique, and the absence of checks for birth dates was shown to cause inconsistency as shown in section 4.2.

Category	Amount	Blocksize	Birth Ego	Marriage Ego	Death Ego	Birth Child	Marriage Child	Death Child	Death Partner
Consistent	285583	9.90	0.28	1	0.63	3.44	1.52	2.39	0.62
Inconsistent	1692	14.86	0.30	1	0.67	5.03	2.43	4.77	0.66
Multiple births	228	11.29	2.00	1	0.55	3.53	1.63	2.02	0.56
Multiple marriages	4736	31.69	0.17	2.02	0.43	2.73	2.75	23.17	0.42
Multiple deaths	457	14.29	0.33	1	2.15	3.61	2.27	3.71	1.23
No marriage	207840	5.11	0	0	0.27	1.59	1.18	1.80	0.27

Table 14: Composition of life courses

Table 14 shows that blocksize is much higher for blocks containing multiple marriages. This indicates that these blocks concern multiple couples that share names for ego and partner after manipulation. Using different manipulations or methods to split these blocks into separate life courses may classify some of these as consistent. The average blocksize of blocks containing no marriage for ego and partner is very small. This indicates that these blocks contain records that possibly belong to another block, where a marriage is indeed present.

Blocksize is also significantly larger for inconsistent blocks than for consistent blocks. This indicates that these blocks contain some records that belong to other life courses. Another explanation is that the method to decide on consistency does not work in some cases, and blocks containing more records have a larger chance to contain these cases. 20 inconsistent life courses were manually inspected. In 6 out of 20 cases, some record was present which did not seem to belong to the life course (e.g. a child born after ego's death). In the other 14 cases, the life course appeared to be consistent, but the estimated birth date of a child did not fall within 25 years after the marriage of ego and partner. These cases all concerned children dying before reaching the age of 1. Figure 11 on page 39 shows one of these life courses. With the implementation of special rules for these kind of cases, more blocks may be classified as consistent. For example, when a child's name is *Levenloos* (Dutch for *lifeless,* indicating stillbirth), the age of the child could be considered 0 years.

For both consistent and inconsistent blocks, under half of the children that are born get married. This is expected, as child mortality is very high in large portions of the dataset, and some people remain unmarried.

For inconsistent blocks, the number of births of children roughly matches the deaths of children. For consistent blocks, one in three children will have a recorded birth, but no recorded death. This is much more then expected. Suspected is that a large number of death certificates of children was unable to be linked together with other events in a couples life course. Blocks containing multiple marriages have an average of 23.17 deaths of children, which is very high. Since the average amount of marriages is 2.02, the expected amount of deaths of children ranges from 4 to 9. A number of these additional deaths of children probably belong to another persons life course.

1865: Maria Babijn is born.

1882: Maria Babijn, age 17, marries Cornelis de Beer, age 30.

1882: child Gijsberta de Beer is born.

1883: child Geertruida de Beer is born.

1884: child Johannes de Beer is born.

1885: child Anna de Beer is born.

1886: child Pieter de Beer is born.

1887: child Maria de Beer is born.

1890: child Rene de Beer is born.

1891: child Sophia de Beer is born.

1891: child Sophia de Beer, age 0, dies.

1893: child Sophia de Beer is born.

1894: child Maria de Beer, age 6, dies.

1894: child Albert de Beer is born.

1897: child Maria de Beer is born.

1899: child Herman de Beer is born.

1899: child Hermina de Beer is born.

1899: child Herman de Beer, age 0, dies.

1899: child Hermina de Beer, age 0, dies.

1900: child Frits de Beer is born.

1900: child Frits de Beer, age 0, dies.

1900: child Gijsberta de Beer, age 17, marries Laurentius de Smet, age 22.

1902: child Betsij de Beer is born.

1903: child Johannes de Beer, age 18, marries Maria Bliek, age 19.

1910: child Pieter de Beer, age 24, marries Aaltje van Esbroek, age 22.

1917: child Sophia de Beer, age 23, marries Abraham Brevet, age 22.

1917: child Albert de Beer, age 22, marries Wilhelmina de Bert, age 22.

1920: partner Cornelis de Beer, age 67, dies.

1926: child Betsij de Beer, age 23, dies.

1943: Maria Babijn, age 78, dies.

1948: child Gijsberta de Beer, age 66, dies.

1949: child Geertruida de Beer, age 65, dies.

1950: child Sophia de Beer, age 56, dies.

Figure 10: Consistent life course detailing the life of Maria Babijn and her marriage with Cornelis de Beer

1797: Adriaan Korteville is born.

1818: child Janneke Korteville is born.

1818: child Janneke Korteville, age 0, dies.

1818: Adriaan Korteville, age 20, marries Jozina Heule, age 19.

1819: child Jannis Kortevile is born.

1820: child Levenloos Korteville, age unknown, dies.

1821: child Jacob Kortevile is born.

1821: child Jacob Korteville, age 0, dies.

1822: child Jacob Kortevile is born.

1824: child Maria Kortevile is born.

1825: child Abraham Korteville is born.

1827: child Adriaan Korteville is born.

1830: child Adriaan Korteville, age 3, dies.

1830: child Cornelis Korteville is born.

1832: child Adriaan Korteville is born.

1833: child Adriaan Korteville, age 1, dies.

1834: child Adriaan Korteville, age 0, dies.

1835: child Pieter Korteville is born.

1836: child Adriaan Korteville is born.

1837: child Pieter Korteville, age 2, dies.

1839: child Pieter Korteville is born.

1840: child Janneke Korteville is born.

1842: child Jannis Kortvile, age 22, marries Magdalena de Back, age 23.

1842: child I Korteville is born.

1842: child Isaak Korteville, age 0, dies.

1854: child Maria Kortevile, age 30, marries Johannis Deunink, age 28.

Figure 11: Inconsistent life course detailing the life of Adriaan Korteville and his marriage with Jozina Heule. The red record caused inconsistency, because the estimated birth date of the child was 1755 (the average of a 90 year range).

26.250 blocks of links produced by the ego/parents/birth date-procedure contained multiple marriages. 20.061 produced multiple consistent life courses. Table 15 below outlines the distribution of the amounts of life courses. Figure 12 shows an example of multiple life courses of one person. Combining these would give a comprehensive description of his/her major life-events.

Number of marriages	Amount
2	18503
3	1465
4	87
5	6

Table 15: Distribution of number of life courses.

1842: Catharina Vermeulen is born.
1872: child Maria Herman is born.
1872: Catharina Vermeulen, age 29, marries Izaak Herman, age 32.
1874: child Tannetje Herman is born.
1875: child Tannetje Herman, age 0, dies.
1876: child Tannetje Herman is born.
1876: child Tannetje Herman, age 0, dies.
1877: child Catharina Herman is born.
1878: child Petrus Herman is born.
1881: partner Izaak Herman, age 41, dies.
1888: child Maria Herman, age 15, dies.
1900: child Petrus Herman, age 22, marries Suzanna Beerens, age 20.
1905: child Catharina Herman, age 27, marries Marinus Moes, age 23.
1939: child Petrus Herman, age 61, dies.

1881: Catharina Vermeulen, age 38, marries Livinus Scheerens, age 52.
1882: child Abraham Scheerens is born.
1883: partner Livinus Scheerens, age 54, dies.
1903: child Abraham Scheerens, age 20, marries Adriana Simpelaar, age 19.
1928: child Abraham Scheerens, age 45, marries Catholijntje Cornelis, age 39.
1954: child Abraham Scheerens, age 71, dies.

1888: Catharina Vermeulen, age 45, marries Pieter Hubregtsen, age 29.
1911: Catharina Vermeulen, age 68, dies.

Figure 12: Multiple consistent life courses of Catharina Vermeulen, detailing her life and her marriages with Izaak Herman, Livinus Scheerens and Pieter Hubregtsen.

4.7 - Computation time

The method performs sorting of records and linking very efficiently. A measure to quantify the amount of time an algorithm takes to perform its computation is *time complexity [20]*. This measure defines time of an algorithm in terms of the size of the input. *Big O notation* is used to express this measure. For an input size n, $O(n^2)$ would for example mean that n*n comparisons would have to be performed. When input size grows, algorithms with a higher time complexity will see a relatively larger drop in time-performance.

Some manipulations have been executed as preprocessing, and are not executed with each iteration of the method. These manipulations have a complexity of O(n), since every element of the input has to be inspected and manipulated once. This process only has to be executed once, and therefore has very little influence on the performance of the method as a whole.

The sorting function for Pandas dataframes uses the *quicksort* algorithm [21]. Quicksort has a worst-case time complexity of $O(n^2)$, which is quite slow. The worst-case scenario is rarely encountered though, and on average the algorithm sorts in O(n log n), which is very efficient. In practice, the sorting of the ~1.7 million records used for this method was executed in under one minute.

The linking algorithm inspects each record in the dataset once, translating to a time complexity of O(n). This makes the algorithm efficiently scalable: the computational time increases linearly with input size. In practice, the combination of extracting from the database, sorting and linking takes between 20 and 25 minutes for $n \approx 1.7$ million. The implementation of the algorithm is not optimized for speed, so computation times can be reduced even further.

Computation time for analysis of records was severely longer than for sorting and linking. This comes from the fact that for each block, a number of searches have to be performed. For example, in the life course analysis the birth date of a child is required. To find this, the serial number of the parent has to be used to find the right certificate, and the record concerning the child has to be extracted from this certificate. Again, the program was not optimized for speed, and with a different representation of the data in the used databases, this process could be sped up.

5 - Conclusions

A sorting-based method for historical record linkage has been developed and evaluated for the population of the Zeeland province of the Netherlands, between 1796 and 1963.

5.1 - Performance

As discussed in chapter 4, it is hard to asses the quality of the sorting-based method without the existence of a gold standard. The method has therefore been evaluated in a number of other ways.

Analyzing blocks of links for inconsistent life courses has shown that for 2 out of 3 categories of procedures, very little links formed inconsistent blocks. While this is no direct measure of the quality of the links, it is a predictor of high precision.

Analyzing the records that were not linked by the method returned interpretable results, as shown in section 4.3. Trends in non-linked records can be explained, predicting high recall. In the case that the method would not have found large numbers of correct links the non-linked records would not show clear and explicable trends. While results discussed in this section give reason for optimism about the method, it might be the case that a constant fraction of links were not found. If this were the case, the non-linked results would still show these explicable trends, but the overall amount of non-linked records could have been reduced.

Comparing the sorting-based method with the LINKS-method in section 4.4 gave insights into the quality of the method. Manual inspection provided some quantification for recall and insight into the strengths and weaknesses of the method. It is important to take into account that the quality of the LINKS-method has not been assessed , and must not be confused to be a gold standard.
The comparison showed that the sorting-based method missed some correct links. These all concern fringe cases, but a possible 30.000 missed links on a total of 700.000 is a significant amount. This is not necessarily a disadvantage of the method, but indicates that additional rules might have to be developed.

Some links were not found because records were not sorted together. If they had been, the linking algorithm would have established a link between them. This is a weakness of the sorting-approach. This weakness can possibly be overcome by expanding the search space of the linking algorithm (for example, search the nearest x neighbors for links instead of only the first), which would result in higher computational time.

Analyzing links that the sorting-based method found over the LINKS-method gives insight into the advantages of the method. Combining the results of manual inspection and the results in section 4.2, the method can link records that have relatively high differences. Both large edit-distances for individual fields and numerous non-matching fields can be bridged by the method. Since manipulations are based on existing records, certain categories of variations are easier to overcome than using string similarity measures. For example, if a name is often spelled in two different ways, manipulating both names to their standard form establishes a link. These name-variants can span large edit-distances.

Using two different implementations of the sorting-based method, life courses were generated. The vast majority of life courses that have been inspected are consistent, and the small number of inconsistent life courses could be reduced further. However, nearly half of all life courses could not be analyzed, because of a lack of marriage of ego and partner. This limits the use of the method to reconstruct entire populations. If a method is developed that can classify these life courses, or

extract records from them and add them to the correct life courses, more complete reconstruction is possible.

The fact that an iteration of the method runs in a relatively small amount of time offers a significant practical advantage over more complex algorithms. Numerous values for variables (e.g. different manipulations, inclusions /exclusions of records) can be tested in the time a more complex algorithm would need to complete one iteration. This makes the method suitable for quickly measuring the influence of a change in variables.

Overall, the quality of the sorting-based method on this type of data appears to be quite high. Although no single ordering of records has been found that outperforms all others, combining a small amount of orderings (produced by the combined procedures) lead to good results. This shows that sorting the data based on manipulations of fields is a promising method. The linking algorithm developed for this research is found to be insufficiently capable of detecting cases in which the sorting went wrong. To use a sorting-based approach linking algorithms will have to be adjusted to deal with the weaknesses of structuring the data.

It is unknown how the sorting-based method performs using other datasets. Civil registration is a system that is mandatory for every citizen and the collection of records is well organized, making the data very complete. Different naming conventions might also influence it's performance. For example, the fact that a woman keeps her maiden name (the family name of her father) when getting married is a tremendous help in linking. If this name is absent linking records before and after marriage would be complicated.

5.2 - Further research

This research shows that the overall quality of the sorting-based method is high, but in some cases the linking algorithm could not cover the weaknesses of the sorted structure of data. The attractive time complexity justifies further research of this type of method. In particular, linking-algorithms utilizing a sorted dataset could to be improved.

Evaluating the sorting-based method proposed in this research (or methods based on it) on different datasets is necessary. The core approach is likely to be successful for other datasets, but parameters (such as the rules of the linking-algorithm) will have to be adjusted. With these parameters, a single method can be constructed which can, with minimal setup, operate over multiple datasets.

A project to develop a gold standard for civil records of the Netherlands is highly encouraged. Such a standard would enable a solid comparison of methods. Based on these comparisons a high quality method can be used to link all records, allowing for a very high standard of quality in systems reliant on these linked databases, e.g. *www.wiewaswie.nl*.

If the method can be refined to allow for better generation and analysis of life courses, population reconstructions are made possible. These can offer valuable statistics. It is therefore encouraged to further research the application of the sorting-based method in the reconstruction of populations.

Bibliography

[1] H. L. Dunn, “Record linkage”, *American Journal of Public Health and the Nations Health*, vol. 36, no. 12, pp. 1412–1416, 1946.

[2] W. A. Rocca, B. P. Yawn, J. L. S. Sauver, B. R. Grossardt, and L. J. Melton, “History of the Rochester Epidemiology Project: half a century of medical records linkage in a US population”, in *Mayo Clinic Proceedings*, 2012, vol. 87, pp. 1202–1213.

[3] S. Gomatam and M. D. Larsen, “Record linkage and counterterrorism”, *Chance*, vol. 17, no. 1, pp. 25–29, 2004.

[4] M. P. Schraagen, “Aspects of record linkage”, Ph.D thesis, Leiden Institute of Advanced Computer Science (LIACS), Faculty of Science, Leiden University, 2014.

[5] L. L. Roos and A. Wajda, “Record linkage strategies. Part I: Estimating information and evaluating approaches”, *Methods of Information in Medicine*, vol. 30, no. 2, pp. 117–123, Apr. 1991.

[6] I. P. Fellegi and A. B. Sunter, “A theory for record linkage”, *Journal of the American Statistical Association*, vol. 64, no. 328, pp. 1183–1210, 1969.

[7] J. L. Warren, C. N. Klabunde, D. Schrag, P. B. Bach, and G. F. Riley, “Overview of the SEER-Medicare data: content, research applications, and generalizability to the United States elderly population”, *Medical Care*, vol. 40, no. 8 Suppl, pp. IV–3–18, Aug. 2002.

[8] T. Blakely and C. Salmond, “Probabilistic record linkage and a method to calculate the positive predictive value”, *International Journal of Epidemiology*, vol. 31, no. 6, pp. 1246–1252, Dec. 2002.

[9] B. Pixton and C. Giraud-Carrier, “Using Structured Neural Networks for Record Linkage”, in *Proceedings of the Sixth Annual Workshop on Technology for Family History and Genealogical Research*, 2006.

[10] D. R. Wilson, “Beyond probabilistic record linkage: Using neural networks and complex features to improve genealogical record linkage”, in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, 2011, pp. 9–14.

[11] D. P. Huijsmans, “Op zoek naar de drie G’s”, *Gen Magazine*, vol. 20, no. 3, Sep-2014.

[12] G. Bloothooft and M. Schraagen, “Learning name variants from true person resolution”, *Proceedings of the International Workhop on Population Reconstruction*, 2014.

[13] “Levenshtein distance”, *Wikipedia, the free encyclopedia*. accessed 09-Mar-2016.

[14] “Jaro–Winkler distance”,*Wikipedia, the free encyclopedia*. accessed 11-Mar-2016.

[15] G. Bloothooft, “Corpus-based name standardization”, *History and Computing*, vol. 6, no. 3, pp. 153–167, 1994.

[16] W. McKinney, “Data Structures for Statistical Computing in Python”, presented at the Proceedings of the 9th Python in Science Conference, 2010, pp. 51–56.

[17] S. van der Walt, S. C. Colbert, and G. Varoquaux, “The NumPy Array: A Structure for Efficient Numerical Computation”, *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, Mar. 2011.

[18] “F1 score”, *Wikipedia, the free encyclopedia*. accessed 04-Feb-2016.

[19] P. Christen, “A comparison of personal name matching: Techniques and practical issues”, in *Sixth IEEE International Conference on Data Mining Workshops, ICDM Workshops*, pp. 290–294, 2006.

[20] “Time complexity”, *Wikipedia, the free encyclopedia*. accessed 16-Mar-2016.

[21] “Quicksort”, *Wikipedia, the free encyclopedia*. accessed 23-Feb-2016.

Appendix - Code

This appendix shows the most important parts of the program developed for this research. The full code is available at *https://github.com/JeltevanBoheemen/LinkingZeeland.* While the program is developed specifically for the material used in this research, it can be adjusted to work for other data.

The command for running procedure 1-1 is:

```
query1_1 = "SELECT familyname as el, firstname1 as ef, sex,
       mother_familyname as ml, mother_firstname as mf, familyname as fl,
       father_firstname as ff, birth_avg_days as bd, role, id_person as id FROM
       ps_working WHERE role IN (1,4,7,10)" #SQL-statement selecting data

proc("1-1", query1_1, ['el', 'ef', 'ml', 'mf', 'ff'], True, 0)
```

```
def proc(name, query, compare, bd_flag, t):
    """ executes an entire linking-procedure
    name is the user-defined code for the procedure
    query is the SQL statement defining which fields (and which manipulations) are used
    compare is a list containing all fields that should be considered when establishing links
    bd_flag is set to True if birth dates should be considered, False otherwise
    t is the threshhold for non-matching fields"""

    #construct a dataframe and sort on selected fields
    df = readsort(query, ['el','ef','sex','ml','mf','fl','ff','bd'])

    #generate a list of links
    matches = link(df, compare, bd_flag, t, name)

    #store the links as a python-object for further use and as a csv
    picklewrite(matches, name)
```

```
def readsort(query,sorting):
    """construct and sort a dataframe"""
    #construct a dataframe based on SQL-statement
    engine = create_engine('mysql://[redacted]@localhost/zl_work')
    df = sql.read_sql(query, engine)
    #sort the dataframe using defined sorting-order
    sorted_df = df.sort_values(sorting)
    #re-index the sorted dataframe
    sorted_df = sorted_df.reset_index(drop=True)
    #append columns for fields which are not used in sorting
    sorted_df_n = sorted_df.reindex_axis(sorting+['role','id','bdate','ddate'], axis=1)
    #return the sorted dataframe
    return sorted_df_n
```

```
def link(df, compare, bd_flag, t, name):
    """apply linking algorithm to sorted dataframe"""
    l = len(df)
    #counter for amount of links found
    m = 0
    #list containing all found links
    links = []
    #generate iterable containing all indices of dataframe
    toCheck = iter(range(0, l-1))
    #check record at index i
    for i in toCheck:
        #j is the index following i
        j = i+1
        #number of links found for record at index i
        n = 0
        if j < l:
            #check the next record for a link
            while check_match_threshold(df.ix[i], df.ix[j], compare, bd_flag, t):
                    #if a match is found:
                    n += 1
                    #record the found links as (id1, id2, block-number)
                    #the block-number is the id of the record at index i
                    links.append((df.ix[i]['id'], df.ix[j]['id'], df.ix[i]['id']))
                    m += 1
                    #increase j to inspect the next record
                    j += 1
            #any records that were linked do not need to be inspected
            #they are therefore removed from toCheck
            next(islice(toCheck, n, n), None)
    #return the list containing all links
    return links
```

```
def check_match_threshold(r1, r2, f, bd_flag, t):
    """checks if a candidate link should be accepted or rejected"""
    #a candidate link is assumed to be accepted, unless proven otherwise
    valid = True
    #if both records are of birth, birth date must match exactly
    if (r1['role'] == 1) and (r2['role'] == 1) and (r1['bdate'] != r2['bdate']):
        valid = False
    #if both records are of death, death date must match exactly
    elif (r1['role'] == 10) and (r2['role'] == 10) and (r1['ddate'] != r2['ddate']):
        valid = False
    #first name of ego must match exactly
    elif (r1['ef'] != r2['ef']):
        valid = False
    elif (r1['ef'] == r2['ef']):
        #the amount of non-matching fields encountered
        conflict = 0
        #for each field that is to be compared, check if fields of both records match exactly
        for field in f:
            if field != 'ef':
                #if this is not the case, a conflict is recorded
                if (r1[field] != r2[field]):
                    conflict += 1
        #if the amount of conflict is larger then the set treshhold t, the link is rejected
        if conflict > t:
            valid = False
    #if a procedure considers birth date
    if bd_flag:
        #if one birth date is missing, the link is rejected
        if math.isnan(r1['bd'].item()) or math.isnan(r2['bd'].item()):
            valid = False
        #a link is rejected if estimated birth dates differ more then 400 days
        elif (abs(r1['bd'].item()-r2['bd'].item()) > 400):
            valid = False
    #return wether the link is rejected (False) or accepted (True)
    return valid
```

```
def procAnalyse(proc, base, fieldlist):
    """executes analysis of links
      proc is the name of the procedure to be analysed
      base is the name of the procedure of which links should be subtracted from proc

      procedure 1-1 is often chosen as base, since this procedures only generates
      links where all original fields match exactly. This leaves only links where some
      fields do not match in the non-manipulated forms."""

    #a HDF table where different dataframes are stored
    store = HDFStore([redacted]dataframes.h5')

    #reads links from both procedures and returns only links that are found
    #in 'proc' and not in 'base'
    if base == '':
        #if not base procedure is defined, analyze all links of 'proc'
        matches = loadMatchesSingle(proc)
    else:
        matches = loadMatches(proc,base)

    #reads a previously constructed dataframe containing relevant fields for all records in the
      datasets
    df = store['qbasemar']

    #analyse the links
    matchdf = analyze(df, matches, proc, base, fieldlist)

    #generate some statistics of the analysis
    difStats(matchdf, proc, base, fieldlist)
```

```
def analyze(df, matches, proc, base, fieldlist):
    """analyzes selected links and saves information about them
    fieldlist contains all fields that are to be inspected"""

    #construct a new dataframe containing information about the links
    #in this dataframe is link is represented as a row and the columns
    #contain information about selected fields of both records of the link
    matchdf = createDf()
    matchdfc = matchdf
    #contruct a dictionary of fields in the analysis-dataframe. Each of these fields will record
      #values for each link in a list
    #the dictionary is initialzed with some standard values (0 for numbers, None for text fields)
    d = {'proc':[proc]*l, 'm1':[0]*l, 'm2':[0]*l, 'el1':[None]*l,'el2':[None]*l, 'el_d':[0]*l, 'el_lv':[0]*l,
'ef1':[None]*l,'ef2':[None]*l, 'ef_d':[0]*l, 'ef_lv':[0]*l, 'ml1':[None]*l,'ml2':[None]*l, 'ml_d':[0]*l,
'ml_lv':[0]*l, 'mf1':[None]*l,'mf2':[None]*l, 'mf_d':[0]*l, 'mf_lv':[0]*l, 'fl1':[None]*l,'fl2':[None]*l,
'fl_d':[0]*l, 'fl_lv':[0]*l, 'ff1':[None]*l,'ff2':[None]*l, 'ff_d':[0]*l, 'ff_lv':[0]*l, 'pl1':[None]*l,'pl2':
[None]*l, 'pl_d':[0]*l, 'pl_lv':[0]*l, 'pf1':[None]*l,'pf2':[None]*l, 'pf_d':[0]*l, 'pf_lv':[0]*l, 'bd_d':[0]*l,
'role1':[None]*l, 'role2':[None]*l}
```

```
    #loop over all links
    for i, (a, b) in enumerate(matches):
        #find the records based on the IDs found in the link
        m1 = df[df.id==a]
        m2 = df[df.id==b]
        #for each field, set values for both records:
        for f in fieldlist:
            #if the non-manipulated field does not match exactly:
            if m1[f].item() != m2[f].item():
                #save the value of the field for both records in the dictionary
                d[f+'1'][i] = m1[f].item()
                d[f+'2'][i] = m2[f].item()
                try:
                    #save levensthein-distance between the fields of both records
                    #if one name is not present, set levenshtein to None, otherwise it will
                    #count the distance to string 'null'
                    if m1[f].item() == 'null' or m2[f].item() == 'null':
                        d[f+'_lv'][i] = None
                    else:
                        d[f+'_lv'][i] = distance(m1[f].item(), m2[f].item())
                #if the name is not present it might be set as None or Nan when converting
                #to dataframe
                #in this case the levenshtein-distance is also set to None
                except TypeError:
                    d[f+'_lv'][i] = None
                #a value of 1 for field_d denotes the non-manipulated fields do not match
                d[f+'_d'][i] =  1
        #set IDs of both records
        d['m1'][i] = a
        d['m2'][i] = b
        #set the difference between estimated birth dates of both records
        d['bd_d'][i] = np.abs(m1['bd'].item() - m2['bd'].item())
        #set the role of both records
        d['role1'][i] = m1['role'].item()
        d['role2'][i] = m2['role'].item()
    #convert the dictionary to a dataframe and append it to the one constructed
    adddf = pd.DataFrame.from_dict(d)
    matchdf = matchdf.append(adddf,ignore_index=True)
    #return the dataframe containing information about each link
    return matchdf.reindex_axis(matchdfc.columns, axis=1)
```