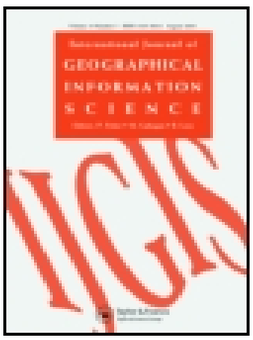

International Journal of Geographical Information Science

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tgis20

Connecting family trees to construct a population-scale and longitudinal geo-social network for the U.S.

Caglar Koylu , Diansheng Guo , Yuan Huang , Alice Kasakoff & Jack Grieve

To cite this article: Caglar Koylu , Diansheng Guo , Yuan Huang , Alice Kasakoff & Jack Grieve (2020): Connecting family trees to construct a population-scale and longitudinal geo-social network for the U.S., International Journal of Geographical Information Science, DOI: 10.1080/13658816.2020.1821885

To link to this article: https://doi.org/10.1080/13658816.2020.1821885

INTERNATIONAL JOURNAL OF GEOGRAPHICAL INFORMATION SCIENCE
https://doi.org/10.1080/13658816.2020.1821885

Taylor & Francis
Taylor & Francis Group

RESEARCH ARTICLE

Connecting family trees to construct a population-scale and longitudinal geo-social network for the U.S.

Caglar Koylu[a], Diansheng Guo[b], Yuan Huang[b], Alice Kasakoff[b] and Jack Grieve[c]

[a]Geographical and Sustainability Sciences, University of Iowa, Iowa City, IA, USA; [b]Department of Geography, University of South Carolina, Columbia, SC, USA; [c]Department of English Language and Linguistics, University of Birmingham, Birmingham, UK

ABSTRACT

We collected 92,832 user-contributed and publicly available family trees from rootsweb.com, including 250 million individuals who were born in North America and Europe between 1630 and 1930. We cleaned and connected the family trees to create a population-scale and longitudinal family tree dataset using a workflow of data collection and cleaning, geocoding, fuzzy record linkage and a relation-based iterative search for connecting trees and deduplication of records. Given the largest connected component of nearly 40 million individuals, and a total of 80 million individuals, we generated, to date, the largest population-scale and longitudinal geo-social network over centuries. We evaluated the representativeness of the family tree dataset for historical population demography and mobility by comparing the data to the 1880 Census. Our results showed that the family trees were biased towards males, the elderly, farmers, and native-born white segments of the population. Individuals were highly mobile – in our 1880 sample of parent-child pairs where both were born in the U.S., 47% were born in different states. Our findings agreed with prior studies that people migrated from East to West in horizontal bands, and the trend was reflected in the dialects and regional structure of the U.S.

ARTICLE HISTORY
Received 6 August 2019
Accepted 6 September 2020

KEYWORDS
Spatial social networks; family trees; record linkage; migration; historical GIS

1. Introduction

Data on historical population demography and mobility over a long-time span (e.g. 300 years) at country or global scale have been scarce and difficult to obtain. Traditional studies using family trees to understand historical migration have been primarily based on a small number of families (Adams and Kasakoff 1984) for whom genealogies exist in book form and are usually limited to single countries or regions within them. Nowadays, more and more historical sources are being digitized and shared through genealogy websites and applications, and ordinary citizens have become more involved in tracing their ancestors and building and sharing their family trees. For example, rootsweb.com, one of the world's largest genealogy websites, has accumulated hundreds of thousands of user-contributed family trees and over 800 million individuals and events with important spatiotemporal information such as birth and death locations

CONTACT Caglar Koylu caglar-koylu@uiowa.edu
Supplemental data for this article can be accessed here.

and dates. These crowd-sourced family trees are an untapped source of volunteered geographic information (VGI) that are created and shared publicly by millions of users. These users collectively link individuals across families, space and time, which would be a daunting task for any individual or organization to address alone. Compared with the official census data that are static snapshots of the population at a chosen year with information only on the birthplace and current residence, user-contributed family tree data offer unique opportunities to systematically analyze and understand historical population dynamics in greater detail and on a large spatial and temporal scale. However, as is true with most VGI data, user-contributed family trees require critical evaluation regarding data representativeness. In addition, user-contributed family tree data may contain missing, uncertain and duplicate information on individuals and family relationships. Trees overlap with each other, which creates multiple versions of the same families with conflicting records on events such as birth and death, and family relations such as parent-child and spouse (Guo *et al*. 2015).

In this article, we describe how we cleaned and connected user-contributed and publicly available family trees from rootsweb.com to create a population-scale family tree dataset, including 250 million individuals who were born in North America and Europe between 1630 and 1930. We cleaned and connected the family trees by using a novel workflow of data collection and cleaning, geocoding of birthplace and deathplace of individuals, fuzzy record linkage and a relation-based iterative search for connecting trees and deduplication of records. We evaluated the representativeness of the family tree data for population demography and mobility by comparing the individuals alive in 1880 in the U.S. from the family trees to the U.S. 1880 Census based on an array of summary demographic characteristics such as gender and age, and then geographical information, such as birthplace of individuals and their parents. We then compared the intergenerational migration patterns of people on the Census and those in the trees using the birth places of parents and children.

There are four major contributions of our study. First, we introduce a novel methodology for connecting family trees and removing duplicates using family relations, i.e. husband-wife and parent-child information. Many historical demographic projects are concerned with record linkage, finding the 'same' individual in many sources. We not only link individuals but also identify and remove duplicate individuals from the data. Second, our dataset is larger than others (Otterstrom and Bunker 2013, Han *et al*. 2017, Kaplanis *et al*. 2018, Charpentier and Gallic 2020) and it also goes back farther in time. Given the largest connected component of nearly 40 million individuals, and a total of 80 million individuals, we generated, to date, the largest population-scale and longitudinal geo-social network over centuries. Third, we attempt to evaluate the representativeness of the family trees by comparing them with an historical population. While others have noted the lack of African Americans in both genetic databases (Erlich *et al*. 2018) and those used by historical demographers (Goeken *et al*. 2016), there has been no attempt to explicitly compare population data from family trees with a historical census to see what systematic biases exist. Fourth, we focus on migration flows and family connections not just the locations of where people came from. This allows us to study the entire population including urban areas and frontiers where people from different origins mixed. Our flow maps provide a dynamic view of history, showing how movement unfolded over generations.

2. Related research

Family tree records carry family relationships and individuals' information, which are useful in many different domains such as medical research (Williams *et al*. 2001), local history (Hey 2010), population change and migration (Adams and Kasakoff 1984, Wrigley and Schofield 1983, Otterstrom and Bunker 2013). However, few researchers have directly used user-contributed family tree data because of the inherent uncertainties in the family trees. The major challenges for the process of name matching result from (1) names with valid spelling variations, e.g. 'Jonathan' and 'Johnathan'; (2) nicknames, e.g. 'Bill' for 'William'; (3) changed female names after marriage; and (4) the use of initials instead of given/middle names (Bloothooft and Schraagen 2015).

Kasakoff, Lawson, and van Meter (2014) linked a genealogical sample taken from family trees who were descended from nine progenitors that came to New England before 1650 to the 1860 U.S. Census to examine the effects of family on wealth. They linked 1,009 men from the nine family trees to the 1860 Census. Their results showed that the family effects have more influence on individual's wealth than spatial effects. Koylu *et al*. (2014) used these nine family trees to analyze the relationship between the density of family connections, geographic proximity of family members and their coexistence and colocation over 300 years. Although there were only nine families involved in their analyses, Koylu *et al*. (2014)'s findings align well with the geographic and demographic expansion of the U.S. between the 17th and the 20th centuries.

Recently, several genetic researchers have used big international databases to study the dispersion of related individuals working backwards from the present. Han *et al*. (2017) first clustered the genetic data of individuals born in the U.S. who had submitted material for genetic testing to Ancestry and then traced the clusters back in time to their origins using information on birth places from family trees. They mapped these clusters, and the distribution largely corresponds to what is known about the settlement of the U.S. and replicates the East to West migration patterns others (Fischer 1989) found. Kaplanis *et al*. (2018) used family tree data submitted to Heritage Quest to describe how distances between the birth places of parents and children changed over time. They started with 86 million publicly available profiles and cleaned the family trees using biological constraints. Based on the biological constraint, an individual cannot have more than two parents, and a person cannot be both the parent and child of another person (Kaplanis *et al*. 2018). Kaplanis *et al*. (2018) obtained 5.3 million disjoint family trees. The largest tree contained about 13 million individuals. However, Kaplanis *et al*. (2018) used only family relations to remove duplicates and relied on the website GINI to perform the removal of duplicates as the user inputs the data. However, in the GINI website there is no quality control for user-contributions and the user's judgement, which could result in false links. Rootsweb.com provide the largest collection of family trees with more than 250 million public records of individuals.

Within geography there have been studies of particular populations using family trees. Otterstrom and Bunker (2013) used ancestry information to trace the sources of population in different U.S. cities and to confirm the ideas of Fischer (1989) about the migration streams that created cultural differences between regions within the U.S. Kandt *et al*. (2016) used cluster analysis to map DNA from rural residents of Great Britain and compared the results with surname clusters taken from the 1881 Census. But, because

their aim was to map cultural regions, they limited their work to the stable population in rural areas. Charpentier and Gallic (2020) studied 19th century migrations using user submitted family trees from France. These studies showed the deep roots of rural cultural regions still visible today. The genetic studies validate their sample with genetic material available for a small subset of the data they have. The genealogical links are indeed validated by the genetic material. But few studies have compared their data with historical populations to identify the biases in the data. Kaplanis *et al*. (2018) acknowledge that their data is biased towards the White population of the U.S. Only Kandt *et al*. (2016) compared their samples systematically with other demographic sources such as a census, to see how the genetic populations they are studying compared with an actual population.

However, many of these studies are biased towards stable communities, rural areas and populations which did not mix. The clustering methods they use to find 'communities' (Kandt *et al*. 2016, Han *et al*. 2017, Curtis and Girshick 2017) are usually derived from individuals who did not move and who intermarried and thus formed the genetic clusters they describe. Han *et al*. (2017) note that they are unable to find certain populations in large diverse cities using their methods (p. 9). In our work, we study flows, not clusters, and are thus able to study a much more diverse set of migration experiences. We are capable of finding both those islands of stability and the places where individuals with diverse origins lived. Also, to the best of our knowledge, our data set is the largest of the data sets based upon crowd-sourced materials so far.

2.1. Record linkage

Record linkage refers to the process of identifying and linking records of the same entities in different data sources (Antonie *et al*. 2014). Record linkage is closely related to data cleaning: a process that deals with detecting and removing errors and inconsistencies from data (Rahm and Do 2000) and entity resolution: a process of extracting, matching and resolving entity mentions in various data structures (Getoor and Machanavajjhala 2012). Record linkage is challenging when there is no unique identifier for the same entity (Antonie *et al*. 2014). A typical record linkage process is performed in two steps: (1) feature construction and (2) pair classification. For an individual in a family tree, features may include surname, given name, birthplace, birth year, deathplace, death year, and family relationship(s). Pair classification is usually done using a similarity measure of the features between two records.

Phonetic encoding and pattern matching (Christen 2006) are the two most common approaches for matching text strings. Phonetic encoding attempts to convert a word or string into a code based on how it is pronounced, while pattern matching focuses on the approximate string patterns. Soundex, invented by Russell (1918), is the oldest and best known phonetic coding algorithm (Christen 2006). It keeps the first letter in a string and converts remaining letters to the phonetically similar groups of consonants (Christen 2006). Soundex is the foundation of many phonetic encoding methods, which includes Phonex, Phonix, and NYSIIS (Christen 2006). The Edit Distance (or Levenshtein Edit Distance) is a very popular pattern matching algorithm (Elmagarmid *et al*. 2007). It defines the similarity between two strings A and B based on the minimum number of edit operations (insert, delete, or replace) of single characters needed to transform string

A to string B (Levenshtein 1966). In addition, the Jaro distance metric is a revised version of the Edit Distance by incorporating string length to measure the similarity of two strings (Jaro 1978). Winkler (1990) modified this algorithm to support the idea that the differences near the start of the string are more significant than the differences near the end of the string. Another pattern matching algorithm is Q-gram, which calculates the similarity based on the number of common substrings of length q between two strings (Christen 2006). Through a comparison of different name matching techniques, Christen (2006) and Bailey *et al*. (2017) found that the pattern matching techniques result in better performance than the phonetic encoding methods. Phonetic encoding or name matching increases the false positives (matches) by 23% to 60% on average (Bailey *et al*. 2017).

Various indexing techniques are used to improve the computational efficiency in pairwise string comparison in large datasets. Antonie *et al*. (2014) used the first name and the first letter of the last name to construct blocks such that only the records within the same block are compared with each other. A major limitation of this method is that it is not able to match female names because most females change their last name after marriage. Christen (2012) evaluated the complexity, performance and scalability of the twelve variations of six indexing techniques: traditional blocking, sorted neighbourhood, Q-gram, suffix array, canopy clustering and string-map indexing. Christen (2012) found that one of the most important factors for efficient and accurate indexing for record linkage was the proper definition of blocking criterion. Also, different indexing techniques with different parameter settings result in differences in the number of matched candidate record pairs.

In addition to feature-based record linkage methods, Bhattacharya and Getoor (2004) introduced an iterative record linkage, which allows multiple passes over the data based on family relationships. They found that iterative deduplication performs better than the attribute-based deduplication. Another commonly used record linkage process relies on training data, which includes the Bayesian Inference (Newcombe *et al*. 1959), supervised machine learning (Sarawagi and Bhamidipaty 2002) and active learning models (Elmagarmid *et al*. 2007). Such models require training data to classify duplicate records. However, it is hard to collect training data that include various characteristics and family relationships, which often do not exist in most records. It is possible to match individuals from multiple genealogical datasets with individuals from family trees and use these as training data for machine learning models. However, generating such training data would require careful review of the matching pairs by human classifiers. Moreover, accuracy of machine learning models would be limited by the number, diversity and representativeness of the genealogical resources being used.

2.3. Historical census

Ancestry.com donated complete-count U.S. Census microdata under the 'Big Microdata' project, which includes the period 1790–1940 (Ruggles *et al*. 2017). However, the microdata generated by Ancestry.com consisted of variables that are not coded in a standardized way (Ruggles *et al*. 2017). The Integrated Public Use Microdata Series (IPUMS) is designed to put the historical decennial census (from 1850 through 2010) in the same format using a consistent variable coding and document the changes in the variables over time (Sobek and Ruggles 1999, Kugler and Fitch 2018). In collaboration

with genealogical organizations, by April 2018, IPUMS had produced complete census enumerations from 1790 to 1940 and released the full count data of census years 1850, 1880, 1900, 1910, 1920, 1930 and 1940 (Kugler and Fitch 2018). Also, IPUMS produced samples linking individuals from other censuses to the 1880 Census.

The full count Census data have been largely criticized for a lack of representativeness for certain population segments. Sharpless and Shortridge (1975) argued that the historical Census may have under-enumerated several important social groups, for example, black and other non-white individuals in the U.S. Steckel (1991) identified three types of errors in census data: under-enumeration, over-enumeration, and misreporting. Steckel (1991) also added that the poor, unskilled, ethnic minorities, very young, residents of large cities and residents of frontier areas were more likely to be uncounted or underrepresented in the historical Census. Goeken *et al*. (2016) compared the two enumerations of St. Louis in the 1880 Census and found that around 46% of first names were not exact matches, and the birthplaces were frequently mis-enumerated (Bailey *et al*. 2017). Although historians may question the reliability, accuracy, and the representativeness of the historical Census (Sharpless and Shortridge 1975), no other data sources can compete with the Census in regard to the population coverage and reliability (Ruggles and Menard 1995).

3. Methodology

The objective of this research is twofold: (1) to create a population-scale and longitudinal geo-social network from user-contributed trees, and (2) to assess the representativeness of the family tree data for analyzing historical population dynamics. Our methodology consisted of two steps. We first implemented an analytical workflow of geocoding, fuzzy record linkage and relation-based iterative search to clean and connect the user-contributed family trees. Second, we evaluated the representativeness of the family tree data by comparing the extracted 1880 data from the family trees with the 1880 Census based on an array of summary demographics such as race, gender, age and birthplaces of individuals and their parents. Also, we extracted and mapped state-to-state migration patterns using parent-child relations for both the 1880 Census and the tree sample to demonstrate the utility of the family tree data for capturing historical population mobility.

3.1. Family tree data processing

Figure 1 illustrates the analytical workflow for collecting, cleaning and connecting the family tree data. The details of procedures and algorithms for all steps are provided in Appendices 1 and 2. The workflow consists of five major steps: (1) Downloading of the GEDCOM files; (2) Initial cleaning of records, duplicate trees, and extraction of family trees; (3) Geocoding of individual locations; (4) Connecting (merging) family trees into tree clusters with fuzzy matching of spouse pairs; and (5) Removing duplicate records using a set of rules and iterative tree searches after merging the trees.

Step 1: Downloading GEDCOM files

The GEDCOM format is the most popular format to store and exchange genealogical data with a hierarchical lineage-linked structure that include information on an individual or a family (Gellatly 2015). GEDCOM files on rootsweb.com are publicly available for

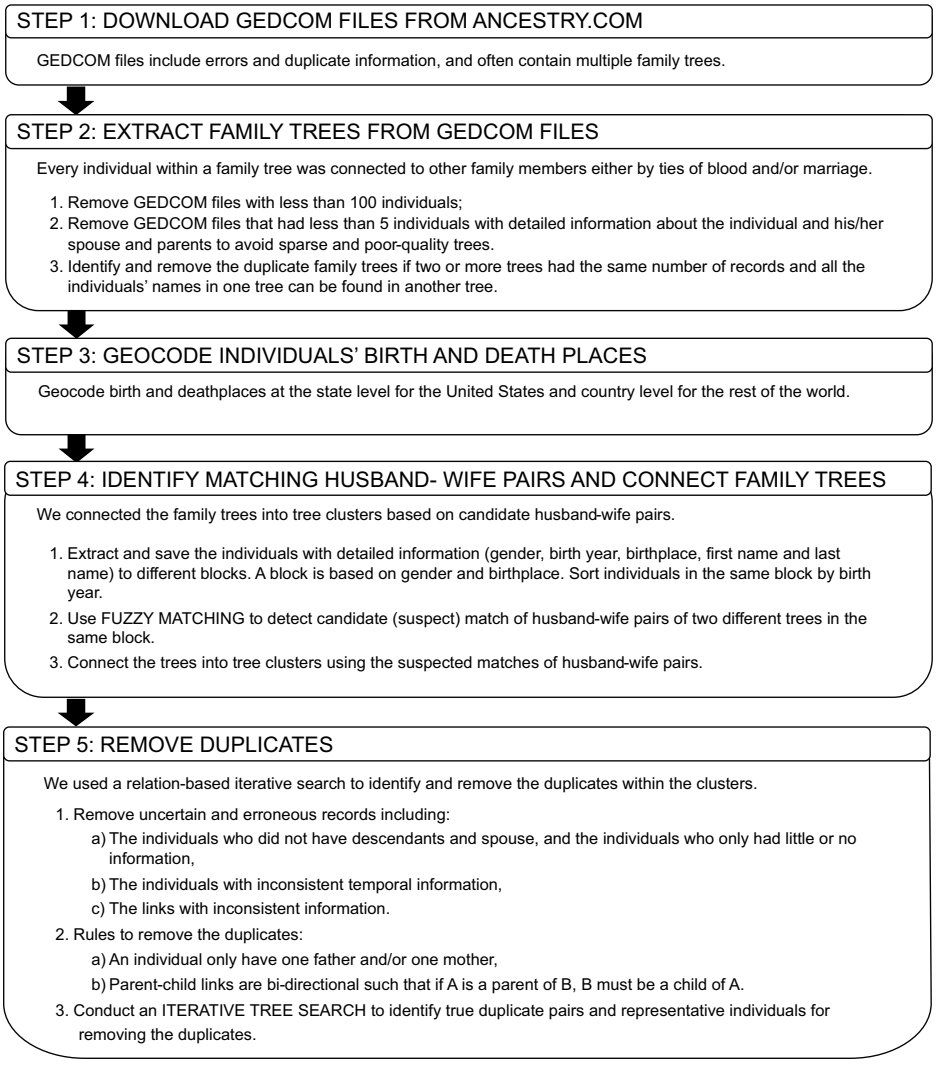

Figure 1. Analytical workflow for cleaning and connecting the family trees.

download and sorted by file name. We downloaded GEDCOM files from rootsweb.com between February and August 2015, and, thus, the most recent update date for each tree cannot be later than August 2015. User-contributed GEDCOM files include errors and duplicate information and some contain multiple family trees in the same file. The family trees are constantly being created and updated by users.

Step 2: Initial cleaning and extraction of family trees

We employed a file-level cleaning in step 2. We first removed GEDCOM files with less than 100 individuals. Second, we removed GEDCOM files that had less than 5 individuals with detailed information about the individual and his/her spouse and parents to avoid sparse and poor-quality trees. We then identified and removed the duplicate files if two or more trees had the same number of records and all the individuals' names in one tree can be found in another tree. Individual records contain detailed information on a variety of

attributes such as gender, birth year, birthplace, first name, last name, mother's, father's and spouse's information and a unique family id. However, not all records contain information on all attributes. Some attributes (features) of individuals were missing or contained values that were inconsistent or inaccurate. For example, some birth or death years contained characters instead of integers. Some records had missing names that were replaced by words such as 'null', 'unknown', 'living', 'husband' and 'wife'. Some names included a series of special characters such as backslash and semi-colon. We first filtered and cleaned these individual features so that we can extract family trees which spanned across multiple generations with standardized information. Every individual within a family tree was connected to other family members either by ties of blood and/or marriage.

Step 3: Geocoding of birth and death places

We geocoded the location of events, i.e. birth and death places, using the references: U.S. Gazetteer (https://www.census.gov), the U.S. Board of Geographic Names (http://geonames.usgs.gov), National Historical Geographic Information System (https://www.nhgis.org), Great Britain Historical Geographical Information System (http://www.gbhgis.org) and Wikipedia (https://www.wikipedia.org). We used a multi-step filtering and standardization of place names in both the reference and the family tree data. Our reference data included common misspelling, spelling variations of place names and abbreviations.

Cheshire and Longley (2012) analyzed the geographic origin, spatial extent and distribution of surnames as well as their relationship between with other surnames and place names in Great Britain. They revealed valuable insights for population genetics, historical geography and genealogy by studying changes in population structure at different scales such as local, regional and national. We geocoded all place names at state level for the U.S. and country level for the rest of the world. We first prioritized geocoding with the U.S. States reference data set, and then checked the U.K. reference and other country reference data sets. For place names that did not match with U.S. reference set, we checked whether the record matched with the other country reference sets, and the place name had to have the state and country name to be matched with those reference sets. If country name did not exist, we classified the record in 'confusing' or in other words, 'uncertain' category. This is rather a conservative approach that leaves many records unmatched. However, this was a choice we made to keep the uncertainty minimal because most of our analysis rely on location information. We reviewed these uncertain location names and adjusted our reference data and our matching criteria to resolve some of the uncertainties and reduce the number of unmatched records.

Geocoding accuracy and uncertainty in our study are affected by the large temporal scale of several centuries. Countries appear, disappear and change borders over even shorter periods of time. Some places become extinct and others are subdivided or renamed. Moreover, place names were often entered by users who relied on their personal memories, records, genealogies and census, or an educated guess about where a person in their tree was born or died. These issues pose substantial challenges for mapping the population structure and its migration, especially for a population that was expanding westward. We provide details of the geocoding process in Appendices sections: (1.1) how we handled uncertainty in location names, (1.2) the reference data sets, (1.3) our geocoding workflow, (1.4) evaluation, and (1.5) the limitations and future work for geocoding.

Step 4: Connecting family trees

For each GEDCOM file that was cleaned and geocoded through steps 1–3, we first extracted and saved the individuals with detailed information including gender, birth year, birthplace, first name, last name, individual's family tree id, mother's, father's and spouse's information into blocks and a tree data structure (Appendix 2.1). We used the blocks to search matching pairs efficiently, while we used the tree data structure in tree cleaning and deduplication process and connecting the family trees. We built each block (index) based on gender and birthplace of individuals to improve the efficiency of the search queries that match identical individuals in multiple trees. We then sorted the individuals in the same block by birth year. Second, we compared individuals to identify the records that belong to the same individual in different trees using a fuzzy matching algorithm. The fuzzy matching algorithm consists of two steps: (1) Assigning weights, and (2) spousal pairs matching and tree clustering. Using the block index, we first compared individuals whose birth years are within 5 years of each other. We then assigned weights based on the matching information on gender, birth year, death year, birthplace, deathplace, first name and last name of individuals and their fathers, mothers and spouses. The details of the weighting algorithm are provided in Appendix 2.2.1. Based on the weighting threshold of greater than or equal to 67, about two thirds of the total maximum score of 100, we identified similar (matching) pairs of individuals in GEDCOM files. Using the similar individuals, we identified whether the spouses of those individuals also have a score greater than or equal to 67. If so, the spousal pairs were classified as candidate (suspected) husband-wife pairs between different trees. These candidate husband-wife pairs were then used to connect the family trees into tree clusters. The spousal pairs matching algorithm is described in Appendix 2.2.2.

Step 5: Tree cleaning and deduplication

The output of the fuzzy match algorithm generated a list of candidate husband-wife pairs that are used to connect the trees and form the tree clusters. In step 5, starting from the largest tree cluster to the small clusters and the stand-alone trees, we first cleaned the trees (Appendix 2.3.1). Using the list of candidate husband-wife pairs we went through records within each cleaned and geocoded tree to identify the true matching husband-wife pairs (Appendix 2.3.2). To determine the 'true' matching husband-wife pairs we performed a relation-based iterative search that goes through the list of candidate (suspected) husband-wife pairs, identifies their parents' and children's information within trees. Specifically, we searched the trees to determine whether the matching husband-wife pairs' (1) parents (i.e. husband's mother and father, and wife's mother and father) were already in the candidate husband-wife pair list; (2) child (ren) had spouses, and whether at least one of the child-spouse pairs were already in the candidate husband-wife pair list. We classified the candidate husband-wife pair as true (matching) husband-wife pair if one of the father-mother and child-spouse relations were already in the list. If none of the father-mother or child-spouse relations were in the list of candidate husband-wife pairs list, we compared both mothers' and fathers' information by calculating a score of conflicting information for each candidate husband-wife pairs. If the conflicting score was less than the threshold, we classified the initial husband-wife pairs as true matching pairs and added the father-mother and the child-spouse pairs into the true matching husband-wife pair list.

During the process of clustering trees, some trees became redundant because all the information in a tree can already exist in a newly formed tree cluster. Going through the list of duplicate individuals from the matching husband-wife pairs, we deduplicated the records by identifying representative individuals for each duplicate pair based on the amount of information (Appendix 2.3.3). The person record with more known information (e.g. gender, first name, last name, birthplace, deathplace, birth year, death year, father, mother, spouse and children) was selected as the representative person. We removed each duplicate husband-wife pair until there were no more duplicate pairs.

3.2. Comparison between the family trees and the 1880 Census

We used the 1880 U.S. full count Census dataset, which is a complete count of population, to assess the representativeness of the family trees for population demography. Despite the under-enumeration and over-enumeration problems, the 1880 Census is the best reference data set that contains the whole population. Country of birth is available for all individuals in both data sets, which made it possible to compare the family tree data with the 1880 Census. We first extracted individuals alive in the U.S. in 1880. We used the following criteria to extract individuals alive in 1880 from the cleaned family trees: 1) the individual should be born before or in 1880; 2) if the individual has a death record (year), the death year should be after 1880 or in 1880; 3) if the individual did not have a death record, we checked whether the individual had an event such as a child born after or in 1880. We then used the event with the date closest to 1880 to estimate each individual's location in the U.S. in 1880 (Guo *et al*. 2015). In addition, if the death year was unknown and the birth year was known, we assumed 80 years as an age threshold to filter out individuals who were less likely to be alive in 1880. Figure 2 illustrates five events of an individual around 1880: own birth, births of three children and death. In this scenario, the closest event to 1880 is the birth of the third child (born in 1884). Therefore, we chose the birth location of the third child as the residence location of the individual in 1880. In addition, for those individuals who were in the U.S. in 1880, we extracted available information on their relatives (father, mother, spouse(s) and child(ren)) for the comparison with the Census.

We compared the family trees and 1880 Census based on locations in the contiguous U.S. Using the birth year, the estimated 1880 location and the birthplace of individuals and their parents, we first generated population pyramids for the family trees and the 1880 Census. Second, we performed linear regression and mapped the demographic and geographic differences between the two sources to identify the mismatches between the

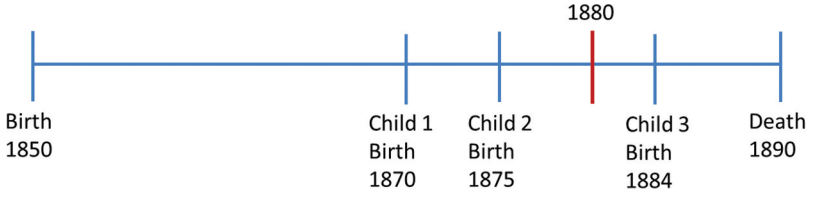

Figure 2. Estimation of an individual's location in 1880 modeled after .Guo *et al*. (2015).

two sources. Finally, we compared the family tree data to the Census in terms of historical population mobility, which we extracted through changes in parent and child birthplaces.

4. Results

4.1. Family tree data cleaning and linking

In step 1, we downloaded 92,832 public GEDCOM files from Rootsweb.com between February and August 2015. These GEDCOM files consist of 247,765,060 individuals and 92,920,152 families. In step 2, we removed 4,966 duplicate and 588 low quality GEDCOM files. After the initial cleaning, we kept 87,308 GEDCOM files that had more than 100 individuals. In step 3, we geocoded all birth and death places at the state level in the U.S. and country level for the rest of the world. We manually checked a stratified random 1% sample of 752,074 unique geocoded place names. Our evaluation included manual observation of the place names and the matches, and googling of place names to identify any other place that may have not been included in our reference sets. Overall, 98.05% of the selected records can be geocoded to the state and country level using the exact match criterion (Table 1). We provide the details of our geocoding results and evaluation and discuss the limitations and future work for geocoding in Appendices 1.4 and 1.5.

In step 4, we extracted and saved the individuals with detailed information into 490 blocks. We created each block based on gender and birthplace, and sorted each record based on birth year. For example, all males born in the same state were put in the same block. We then grouped the trees into tree clusters based on the candidate husband-wife pairs between trees (Table 2). We clustered 87,308 trees into the following clusters: 1) 9,033 trees were stand-alone trees, which were not connected to any other tree; 2) 2,866 trees formed 1,077 tree clusters; 3) There were 75,409 trees in the largest connected cluster.

In step 5, we removed 69,234,001 duplicates and 90,267,147 records with little or inconsistent information. There were 80,172,652 records left after the data cleaning, record linkage and deduplication process. The total number of trees, tree clusters, records, duplicates, records with inconsistent information, and the final records after

Table 1. Geocoding results at the country level and the state level (U.S.).

Reference Dataset	# geocoded	# evaluated	# accurate
US	533,316	5,334	5,257 (98.56%)
UK	54,595	546	514 (94.14%)
Canada/Australia/Germany/Netherlands	76,999	770	766 (99.48%)
Other European countries	21,044	211	196 (92.89%)
All other countries	6,653	67	60 (89.55%)
Historical countries	968	10	10 (100%)

Table 2. Deduplication and cleaning of the family tree clusters.

Family Tree Clusters	# trees/# clusters	# records	# duplicates	# inconsistent information	# cleaned records
Stand-alone trees	9,033	5,928,853	549,087	1,576,016	3,803,750
Small clusters	1,077	2,044,372	765,937	607,199	671,236
The largest connected cluster	75,409	231,700,575	67,918,977	88,083,932	75,697,666

deduplication are represented in Table 2 for each of the tree clusters. In addition, we removed 192 trees from the largest cluster since these trees were contained in other trees.

4.2. Comparative analyses of the family trees and the 1880 Census

There were 50,169,451 individuals in the U.S. 1880 Census. Using our methods there were 8,729,569 individuals alive in 1880 and living in the U.S. on the family trees, 17.4% of the censused individuals. Because information about their parents' birthplaces were missing for about half of these individuals, we based our comparison upon the 4,102,038 individuals with known parents' birthplaces from the family trees.

4.2.1. Age and sex distribution in the population

Figure 3 illustrates the population pyramids derived from the 1880 Census and the family trees using those individuals who were alive in 1880 and lived in the U.S. The two population pyramids both follow the classic expansive model with a broad base reflecting a greater proportion of younger people. Male percentages match for younger males (5–25) in both data sets. Both data sets highlight a historical circumstance which affected the US population: the abrupt change for both males and females in age groups 15–19 and 20–24, which was caused by a deficit of births during the Civil War. The family tree data were more biased towards males than the 1880 Census. In the larger dataset of people from trees known to be alive in 1880, the sex ratio was 115, i.e. 115 males per 100 females, but it was 134 for individuals with known parents' birthplaces in the trees. Thus, females had less information about the birthplaces of their parents, as one would expect since they changed their names at marriage and thus harder to trace. The sex ratio for the 1880 Census was 104, within the normal range for populations. In both, the ratio increased with age. Even so, it starts with 124 for children under 15 in the trees while the comparable figure was 102 for the Census. The tree were even more biased towards

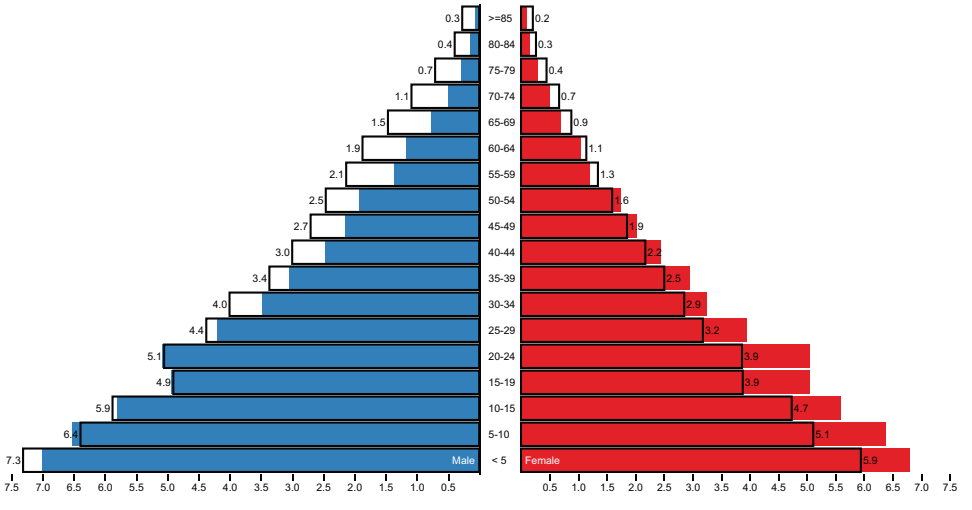

Figure 3. The population pyramids of the family trees (black outlines) and the 1880 Census (blue and red bars).

males at the older ages: the sex ratio for people ages 55 and above was 163, for the Census it was 110. Also, the family tree data included a higher proportion of elderly (people more than 55 years old), both males and females, than the 1880 Census. Individuals were not required to have descendants to be part of our data. However, because we required two events (i.e. death date or birth date of children) that bracket the census year, people without children may not have been included in our comparison because they would lack the second event if they did not have death dates. Also, we kept people alive until age 80 if there was no death date so this may have added to the surplus of older individuals on the trees.

4.2.2. Birthplaces, census residences and parents' birthplaces

Most individuals were born in the U.S. in both the 1880 Census and the family trees (Table 3). However, the family trees had fewer immigrants (people both of whose parents were born outside the U.S. and people who, themselves, were born outside the U.S.). For example, 86.49% individuals were born in the U.S. in the 1880 Census, but there were more, 96.9%, born in the U.S. in the family trees. There is little difference between the proportions of mother's birth countries and father's birth countries within each data set but again immigrants were less common in the trees than they were in the Census. The 1880 Census showed that most immigrants came from Germany or Ireland and that the proportions from each were quite similar. However, most immigrants were from Germany and the UK in the family trees and there were fewer from Ireland. Canada provided the fewest immigrants among the major countries sending people to the US in 1880, according to the Census, but there were more people born there than in Ireland in the family trees. The lack of people from Ireland in the family trees could result from the commonness of their surnames and given names, making it difficult to construct family trees due to a higher proportion of duplicates. That Irish settled in cities makes the matching process even more difficult because so many people in each city had the same names.

Figure 4 illustrates the 1880 locations, birth states of individuals and their parents as a proportion of the U.S. population for both data sets. The State of New York was home to a far greater proportion of the population in the Census (10%) than in the trees (4%). The trees had fewer in New York state because they did not have as many immigrants, especially the Irish. In 1880 the Irish accounted for about 10% of the population of New York state and were concentrated in New York City (16% of the population of New York City). Pennsylvania had the second largest number of Irish. A smaller proportion of the population lived in the deep south in the trees compared with the Census: especially South Carolina, Louisiana and Mississippi. There is no information on race in

Table 3. Country of birth by percentage in 1880 U.S. Census and family trees.

	1880 Census			Family trees		
Country of Birth	Self	Mother	Father	Self	Mother	Father
United States	86.49	71.70	69.81	96.90	89.51	88.17
Germany	3.86	8.99	9.79	0.82	3.36	3.99
Ireland	3.69	9.29	9.58	0.21	1.22	1.45
United Kingdom	1.83	2.63	4.29	0.77	2.68	3.01
Canada	1.43	1.59	1.59	0.59	1.18	1.19

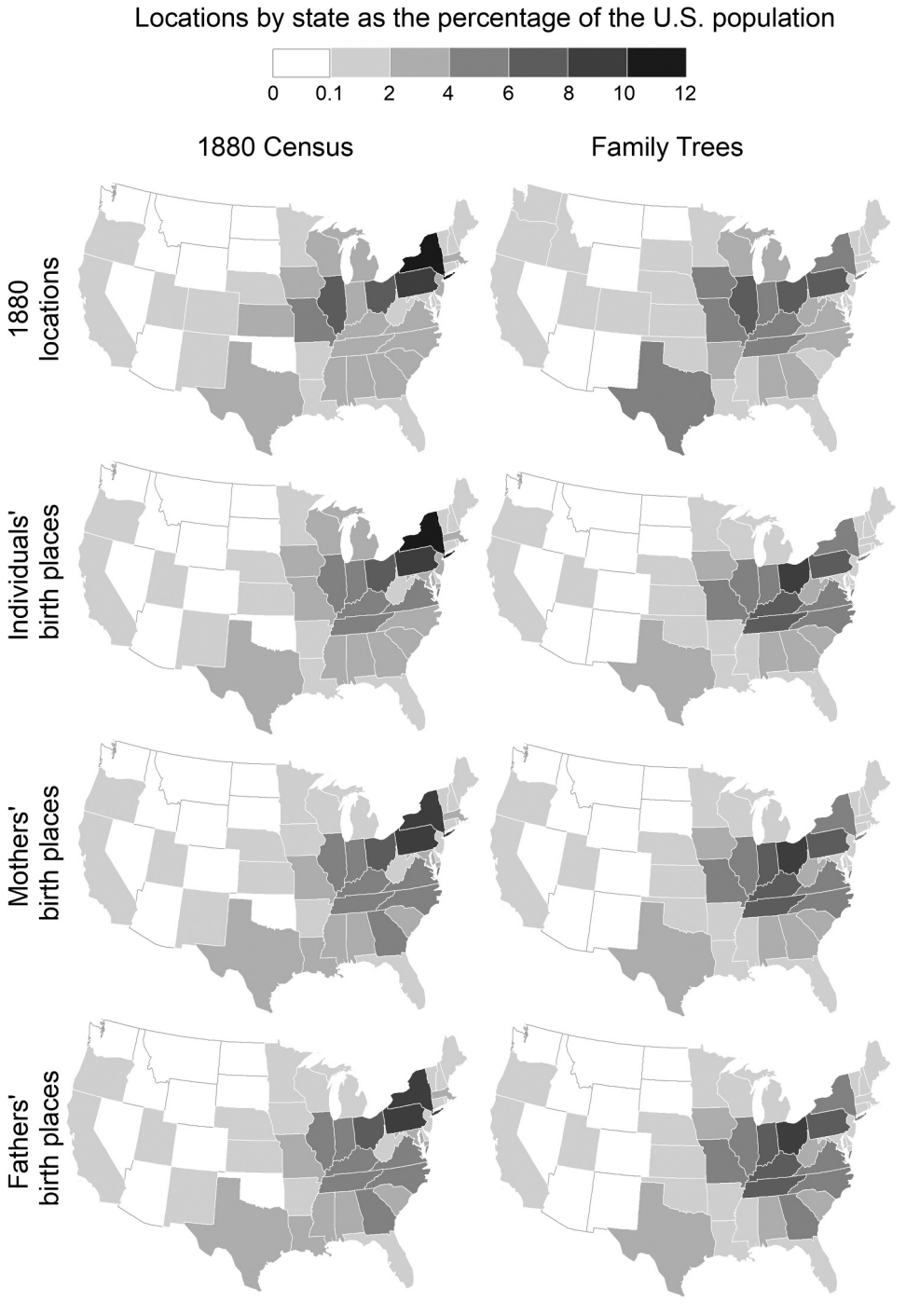

Figure 4. Residences of individuals and the birth states of individuals and their parents as the percentage of the U.S. population in the 1880 Census and in the family trees.

the family trees but there are probably far fewer Black individuals in the trees than in the actual population. Thus, it also makes sense that Kentucky and Tennessee, the states with

predominant White populations in the South in 1880, were overrepresented in the trees. The western states were also overrepresented reflecting both the lack of immigrants and Black individuals there. These effects fade as the data goes back in time to individuals' birth places and then to their parents' birth places, but they never completely disappear. The two major waves of immigration, the German and the Irish, occurred in the 1840s and 1850s. The children of those immigrants born in the U.S. would be in their 20s and 30s in 1880, and they still lived in the states where their immigrant parents had settled. Fathers' birth states on the trees were closer to the Census than mothers' birth states.

To eliminate the effect of the lack of foreign-born people in the family trees, we compared the two sources using only U.S.-born people with two U.S.-born parents (Figure 5). 68% of the total U.S. population met these criteria in 1880. In both data sets, the population was again concentrated in the middle of the country. The two data sources were more similar when the entire population was used (Figure 4). There were still more people living in New York and Pennsylvania in the Census than on the trees after the two U.S.-born parents criteria. As we go back in time through the generations, the similarity between the trees and the Census increased but the excess in the state of New York remained. The effect of where immigrants settled lasted over generations.

Figure 6 shows how well the trees corresponded with the Census by taking the ratio of the maps in each row in Figure 4 (the left panel in Figure 6) and in Figure 5 (the right panel in Figure 6). There were more individuals in the middle of the country in tree data and more in the Census in the south and northeast. Numbers were too small in one or both sources to compute a robust ratio for the far West. In the map showing residence in 1880, only Maryland, Delaware and DC were greatly overrepresented in the Census when only US-born individuals with two U.S.-born parents were used (the right panel in Figure 6). When the entire population was used regardless of the US-born condition of two parents and individuals, New York, Massachusetts and New Jersey were also among the states that were overrepresented in the Census (the left panel in Figure 6). On the other hand, West Virginia was greatly underrepresented in the Census, when the entire population was used. As we go back to the parents' generation only one state diverges greatly from the Census, West Virginia, which was again overrepresented in the trees.

4.2.3. The representativeness of the family trees for population demography

We performed a regression analysis to explore the potential bias in the family tree data (see Appendix 3). We chose the state level population proportion (the number of individuals alive in the U.S. in 1880 in the family trees divided by the number of individuals in the 1880 Census) as the dependent variable. Our candidate independent variables were state level percentage of the following population segments: white, farmer, individuals greater than 54 years old, male, individual's birthplace in the same state as the 1880 location, and individual's birthplace in foreign countries (Figure 3.1 in Appendix 3). The regression result revealed that the family trees were biased towards white people, farmers, as well as native-born (born in the U.S.) Americans compared to the 1880 Census. But the biases towards men and older people, while large, were similar over all the states. Thus, these biases were not significant in explaining the distribution of people across the different states. The difficulty of constructing Irish trees probably accounts for much of the foreign-born effect. The proportion that is white would be expected to affect the correspondence between the tree and census populations because there were probably very

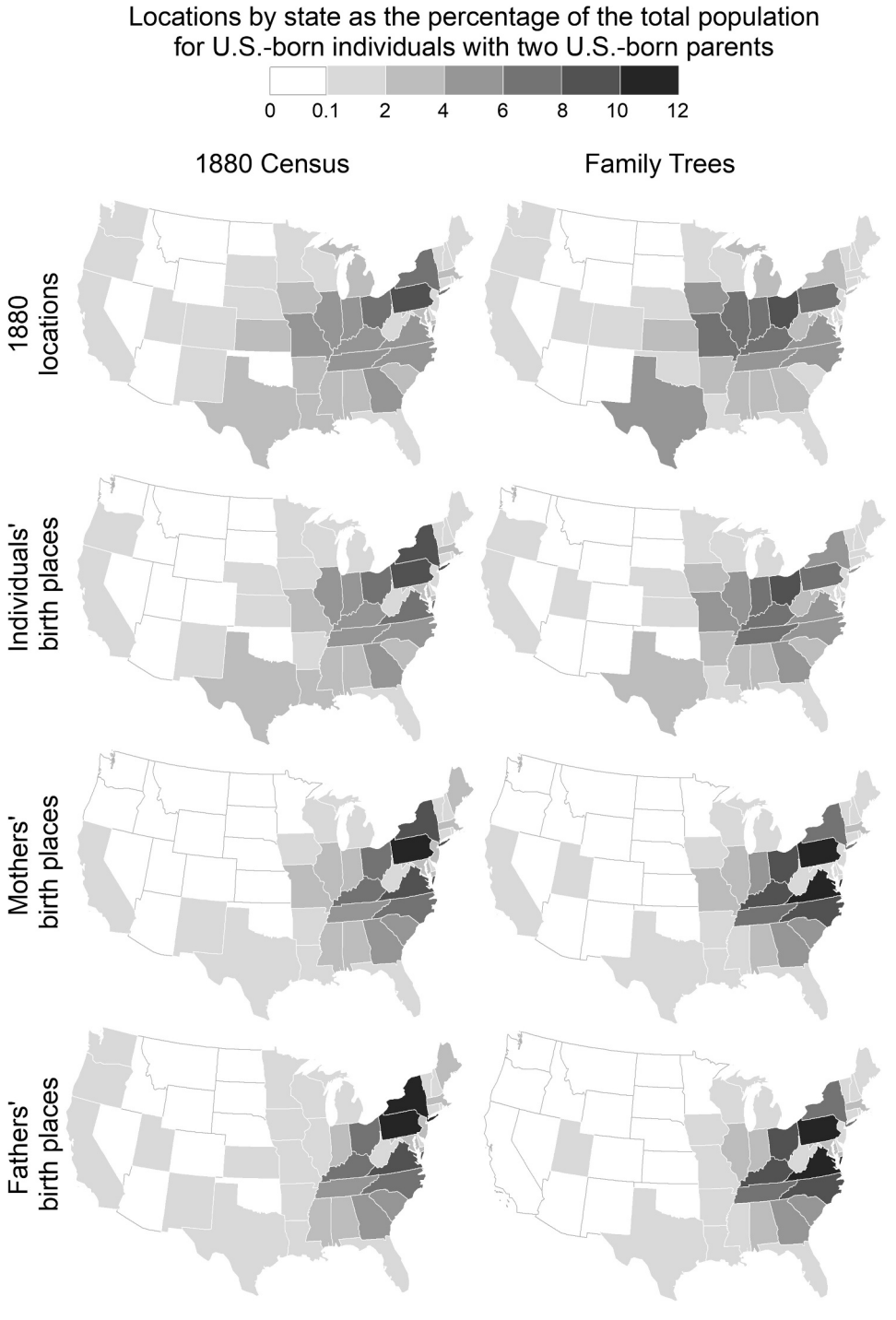

Figure 5. Residences of individuals and the birth states of U.S.-born individuals with two U.S.-born parents as the percentage of the U.S. Population in the 1880 Census and in the family trees.

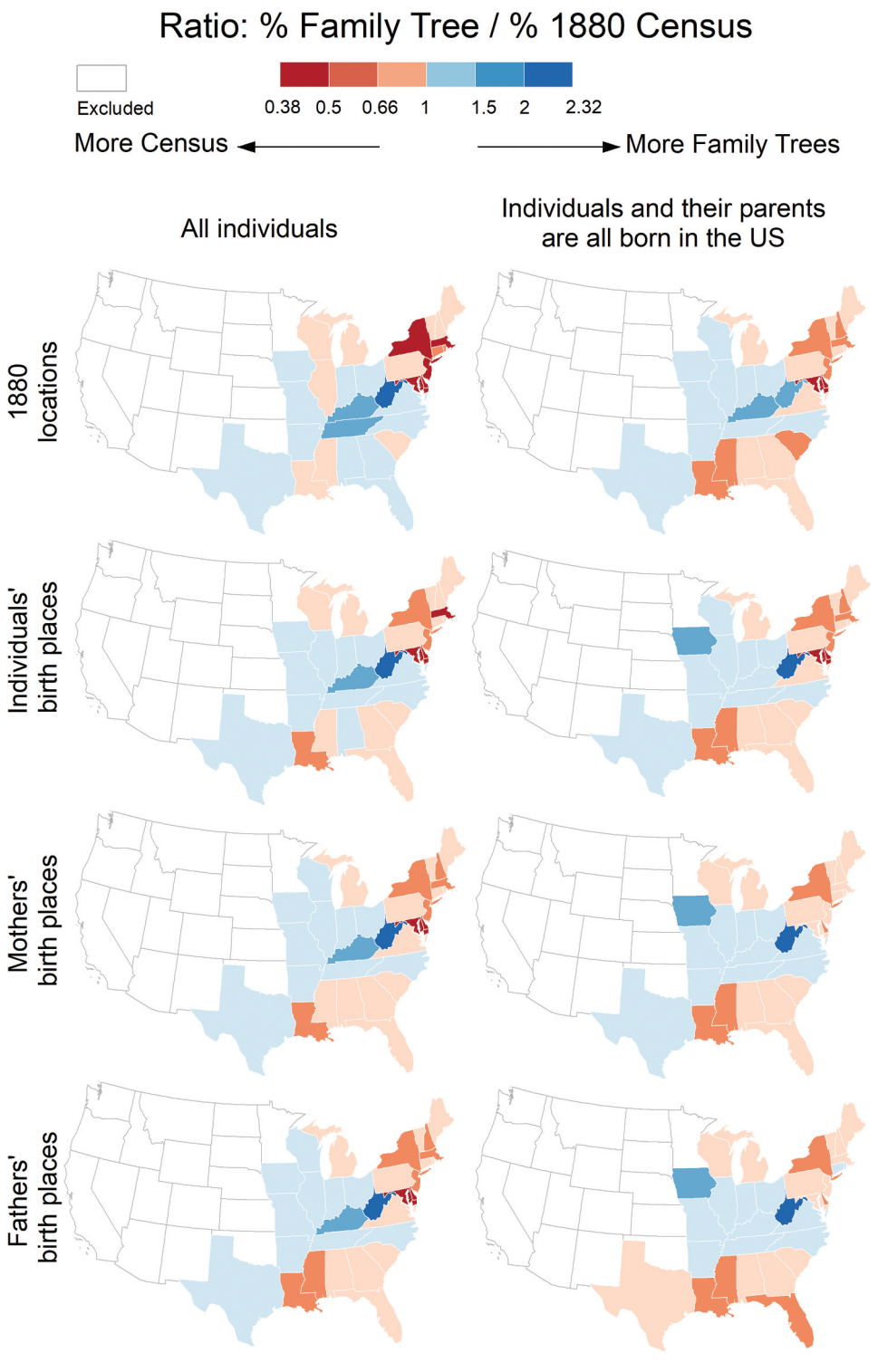

Figure 6. The ratio of the family trees to the 1880 Census for all individuals including those with US and foreign-born parents (Left) and only US-born individuals with two US-born parents (Right).

few non-white individuals in the trees. One would expect that the proportion of the 1880 population represented in tree data would be lower for the South where there was a considerable Black population. Race is not noted in the tree data; one way to find out how many Black individuals are in the tree sample would be to match individuals between the Census and the trees which we have not done. Nevertheless, the difficulty of constructing Black family trees is well known due to lack of surnames prior to emancipation. Other studies have found that Black individuals probably passed for white in large numbers and this might have also involved changing their names (Nix and Qian 2015) making it difficult to compile their trees. Linked census samples also have lower proportions of Black individuals and of foreign born than of native-born whites (Goeken *et al*. 2016). The fact that farming had a significant effect only when both foreign-born and white were included in the regression may have been due to the difficulty of compiling trees for urban residents where family members were not nearby, and several individuals with the same names could be mismatched.

4.2.4. Parent-child migration extracted from the family trees and the 1880 Census

We compared the family tree data to the Census in terms of historical population mobility, which we extracted through family relationships. Family relationships are complex, and include a variety of kinship types including parent-child, sibling, spouse, grandparent-grandchild, cousin etc. We used birth locations of parents and their children because different birth locations of a parent and child can reveal the intergenerational migration of a family. The 1880 Census was the first to ask for the birthplaces of the father and mother of each individual enumerated. Parent-child relations include both mother-child and father-child relations. Thus, the same child is in two relations: father-child and mother-child. We only counted the parent-child relation once based on gender and birthplace of children to reduce biasing the results toward large families. For example, if there were multiple children of the same sex, e.g. male, with the same birth state, mother-son and father-son relations were counted only once. If two children of the same sex were born in different states, then both relations were counted. If the two children with different sex were born in the same state, then both relations were counted. Using this strategy, we located the birthplaces of both the parent and the child for 36.9 million pairs in the Census as compared to 4.8 million pairs in the family trees.

To evaluate the population mobility and stability for states, we calculated a mobility ratio of parent-child relations in each state for the Census and the family tress. The ratio is the number of parent-child pairs in which a parent of a child was born in a different state divided by the number of parent-child pairs both born in the same state. In these statistics, we compensated for family size – there is evidence that large families were more apt to move – by counting each state of birth for only one child of each gender within each sibling set. We can do this for all individuals from the trees but in the Census data, only for individuals who lived with their parents. This is because we used information on the 'family' in the Census to correct for family size. For the others, we use the information on the birth places on parents collected for every individual in the Census. As a result, after a person had left their parental household, every person was a part of the sample of parent and child birthplaces.

$$Between\ State\ Relations\ (BSR)_i = ParentToChild_i + ChildToParent_i$$

$$Mobility\ Ratio_i = \frac{Between\ State\ Relations_i}{Within\ State\ Relations_i + BSR_i}$$

Since children were born when parents were 30 years old on average, the maps in Figure 7 display migration that occurred over a long period of the United States (people in their 70's in 1880 would have been born in 1810 but their parents in 1780, on average). The population was very mobile. The population in the trees was more mobile than in the Census. While 47% of parent-child pairs were born in different states in the trees, the percentage was 40% for the Census. This is likely due to the way immigration affected migration over generations: children and grandchildren of immigrants were probably less likely to move than descendants of people born in the U.S. Both sources showed the same spatial pattern: An East to West gradient of 'stability to mobility'. Since the West began to be settled from the East and Europe quite late, most people living there in 1880 would not have had parents who had been born there. The same islands of stability – Maine and Arizona – existed in both sources. The patterns were similar despite the overcounting of siblings in the Census compared with the trees.

Figure 8 compares the origins and destinations of people in the family trees with those in the Census. Figure 8 only shows the flows for children born in the U.S. with two U.S.-born parents. We chose a minimum flow threshold of 0.01% which illustrates 97% of all flows. In Figure 8, flows of parent-child relations in the Census and trees are displayed on top of a choropleth basemap of net flow ratio. The base map shows whether a state was gaining (red) or losing (blue) population from migration. The maps from the two sources are nearly identical. As expected, Eastern states lost while the Western states gained. Both maps have a zone down the center of the country, between the Mississippi and the Atlantic coast where ratios were low because people were both coming in and leaving. The differences between the maps from the Census and the trees were – Arizona and North Dakota who lost according to the Census but gained in the trees – are due to small numbers.

Even though we have far fewer relations on the trees (4.8 million) than we do from the Census (36.9 million), our tree sample is robust enough to replicate the patterns in the Census for this segment of the population. The flows shown by arrows are also nearly the same in the two sources, mostly from East to West, the horizontal bands Fischer (1989) found. These bands remain important today and can be seen in dialects even in contemporary Twitter data (Huang *et al*. 2016). New York State received people from New England and sent them to Michigan, Wisconsin, and also to Iowa, Ohio, Illinois and Indiana, which also received migrants from Pennsylvania. Virginia and North Carolina sent to the same states but also Tennessee and Kentucky. The southernmost band was from South Carolina and Georgia to the other southern states who also received people from Tennessee. If we could look at smaller spatial units (e.g. counties), we would most likely see that these streams went to different parts of these states. The Census had more flows than the trees, coming out of New York State. This is due to the larger number of relations for the Census making it easier to reach the percent threshold which was identical in both maps.

The western flows were also quite similar in the two sources. In general, the East to West bands broke down West of the Mississippi and people went there from almost all

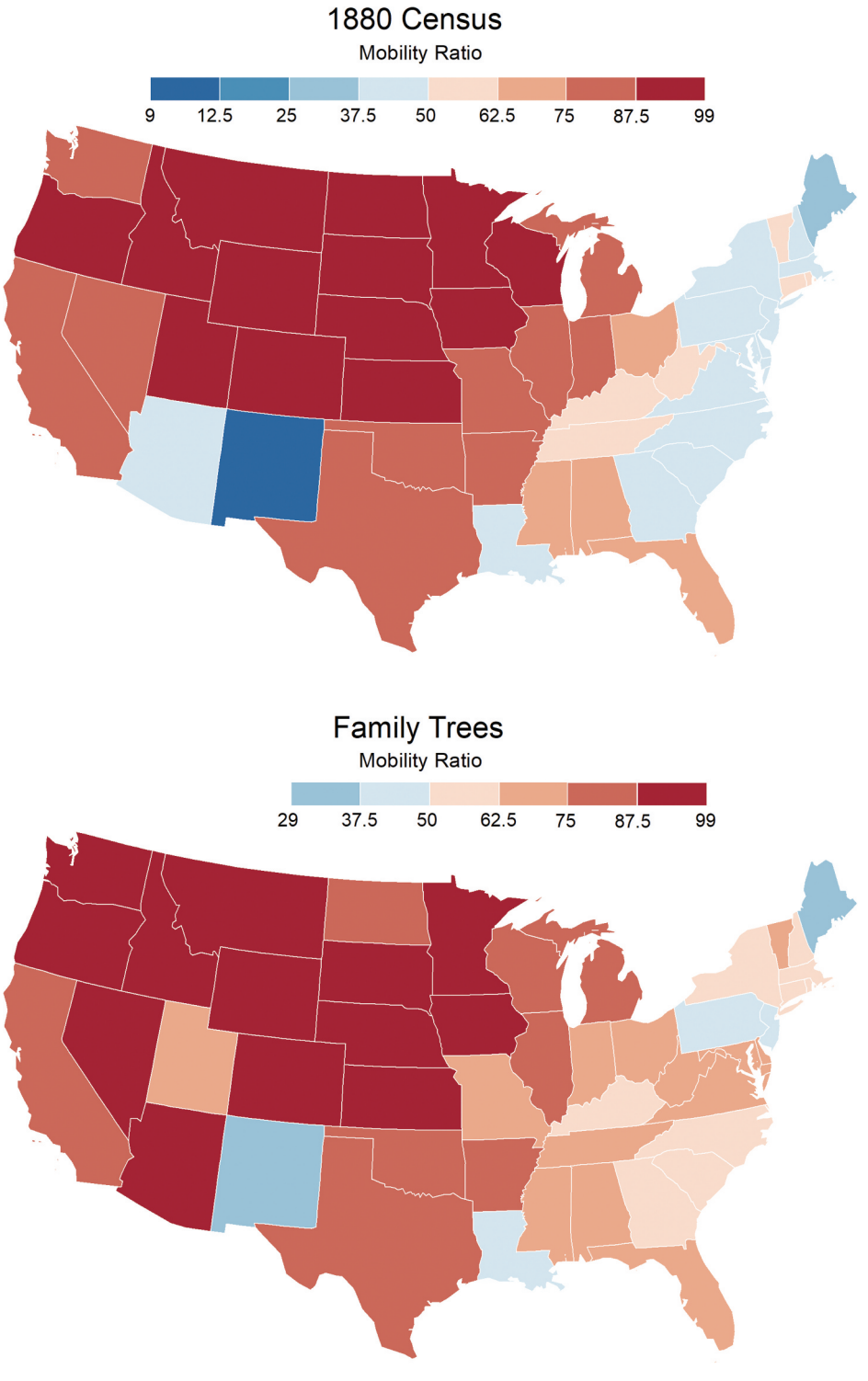

Figure 7. Mobility ratio of parent-child relations that is calculated by dividing the between-states parent-child relations by the total number of parent-child relations including within state relations.

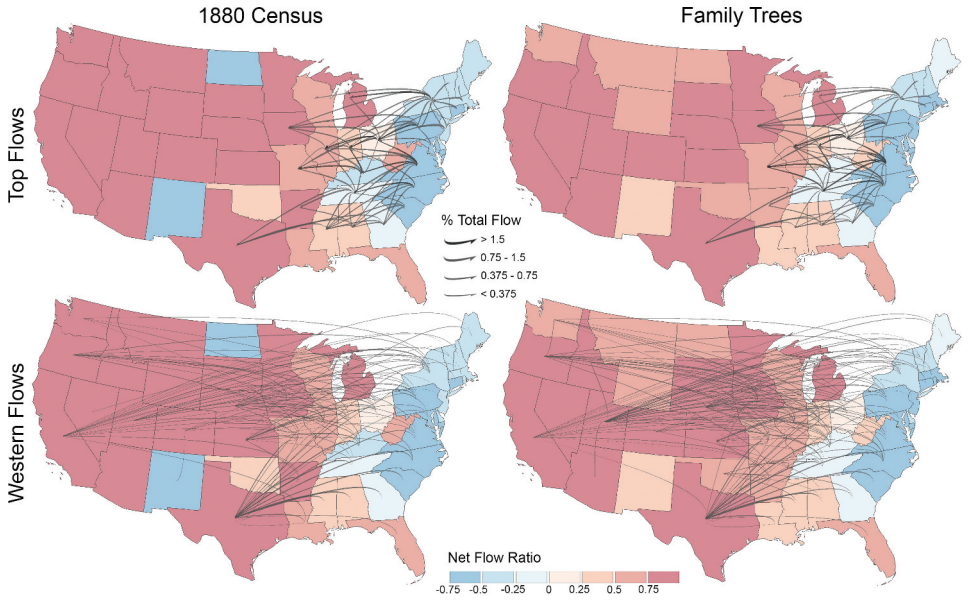

Figure 8. Comparison of the family trees and 1880 Census on parent-child flows as the percentage of total flows for individuals who were alive in 1880, whom and whose parents were both born in the U.S. The choropleth maps illustrate the net flow ratio of parent-child flows (net flow/gross flow) for each state.

states outside of the deep south though more came from the states just West of the Mississippi than from farther East. People from the South went to Texas instead.

5. Limitations and discussion

Crowd-sourced family trees have a number of limitations for studying historical populations. First, the family tree and the 1880 Census datasets are not independent of each other because users often use information from the 1880 Census to build their family trees. However, since birth year was missing for so many people in the trees but is available on the Census, we think that Census information was not used for at least half the individuals in the family trees. Therefore, family trees are only partially dependent on the Census. Second, the family trees are inevitably biased towards individuals with families. Third, the process of linking individuals in the family trees has resulted in a greater number of family trees for certain segments of the population than for others. Foreign born and their children, females, and younger people are less likely to be represented in trees than they are in the general population. There is a need to assess whether the small number of immigrants and their descendants are representative of the general population by comparing them to immigrants in the Census. Fourth, certain segments of the population are largely missing in the family tree data, for example, the Black population, the Mexican population in the southwest and the Cajun population in Louisiana. This is undoubtedly a fundamental flaw for representativeness. There are possibly more of these gaps for minorities, and it is difficult to discover these gaps without a direct matching of individuals between the trees and the Census. We geocoded data

only to the country level and the state level within the U.S. However, flows between states and countries hide substantial local variation which could be visible if we geocoded in finer spatial units such as counties or places.

6. Conclusion

User-contributed family trees are a unique source of data that can be used in social science research and are particularly useful for examining families and changes in their connections over space and time. Given the largest connected component of nearly 40 million individuals, and a total of 80 million individuals, we generated, to date, the largest population-scale and longitudinal geo-social network over centuries. This article describes the methods we used for processing, cleaning, geocoding and integrating family tree data. To the best of our knowledge, our study is among the first of its kind to formally evaluate the representativeness of user-contributed big data from family trees with ground truth data (1880 Census) for an entire nation. Our results provide valuable information for understanding biases in family trees and the segments of the population that are lacking in trees. We also showed how family trees can be used to study mobility and family connections in historical populations. Family trees provide information on family connections as they developed over a long period of time and at the scale of a continent. They provide a unique window through which we observe how kin connections were affected by migration, a process which, on a grand scale, created the cultural regions which remain important today, and, on a much smaller scale, impacted the daily lives of families by affecting the extent to which particular types of relatives were available for support nearby.

Our findings revealed that family trees are excellent for extracting and understanding the spatial flow patterns at later dates, at least for the (substantial) proportion of the population born in the U.S. whose parents were born there as well. The population was so mobile (some 40% of children born in a different state from their parents) that even large units such as states can reveal a great deal about the family and gender patterns of migration. Despite the smaller sample and biases towards males and older people in the trees, the migration flows from the two sources were quite similar for people born in the U.S. whose parents were also born in the U.S., 68% of the US population in 1880. Since immigration rates were low prior to 1840, family trees may prove to be a very good source for studying internal migration during the early history of the United States.

Roadmap for future research

We plan to conduct a series of future research that will build upon our study. The major outcome of our study is the cleaned and connected population-scale family trees. Despite our rigorous methodology for cleaning and connecting the trees, more work must be done to further compare and evaluate different strategies for data cleaning, geocoding and the effects of parameter selections on the outcome. In this article, we presented a preliminary analysis of migration through parent-child relations for individuals who were alive in 1880. We plan to extract and study migration patterns and their evolution across several centuries and countries and effects of gender on migration using both parent-child relations as well as changes in birthplaces of siblings, and individual birth-death locations. Despite the issue of representativeness, our data and analysis of population mobility is one

of the first studies to reveal dynamic migration patterns on a large spatial and temporal scale. There are few sets of data that cover as long a span of time as our family tree data. Kaplanis *et al*. (2018) identified 1850 as a turning point after which parent-child distances began to increase on all continents. We want to see how these distances changed in our data and study which segments of the population travelled farthest. We are also especially interested in how to meaningfully divide the data into temporal periods. Another study we plan to conduct is the evaluation of the family connectedness across space and time. In the connected and cleaned family trees, the individuals are linked to each other with family ties, which allow us to construct a population-scale and geographically embedded social network. We plan to analyze the social network characteristics, and how those characteristics change over time and space. In addition, we plan to analyze the relationship between kin proximity and geographic proximity to reveal characteristics of different places in terms of their family connectedness, and reasons for geographic and temporal variation in the kinship patterns. This will allow us to understand how the East to West migration and geographic expansion patterns affected proximity of parents and their children.

Acknowledgements

The authors would like to thank Xi Zhu and Chao Chen for their contribution in the early stages of the data processing of the family trees. In addition, the authors greatly appreciate the comments and suggestions from the editors and the reviewers.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by the Digging Into Data Award [LG-00-14-0030-14] by the Institute of Museum and Library Services (IMLS).

Notes on contributors

Caglar Koylu is a Professor of Geographic Information Science in the Department of Geographical and Sustainability Sciences at the University of Iowa. His research focuses on spatial interaction analysis and visualization, flow mapping, geovisual analytics and human computer interaction, and big data analytics for social media and sensor networks. He is especially interested in the development of new theory, methodologies and applications to analyze and understand large geospatial data and spatial interaction networks, i.e., networks embedded in geographic space and time such as migration, commodity flows, human mobility and communication, family trees and other geo-social networks.

Diansheng Guo is currently the director of Big Data Lab at Tencent Map. Prior to his current position he was a Professor in the Department of Geography at the University of South Carolina between 2004 and 2020. His research interests include spatial data mining, visual analytics, spatio-temporal and high-dimensional visualization and big data analytics.

Yuan Huang is currently a software development engineer at Amazon Lab 126. Yuan got a bachelor's degree in Urban Planning (2011) at Sun Yat-Sen University. She received master's degrees

in GIS (2013) from the Central Michigan University, and in Statistics (2018) and Computer Science (2019) from the University of South Carolina. Her research interests include spatial and temporal data analysis, regional linguistic variations, and spatial models.

Alice Kasakoff is Emeritus Professor of Anthropology and Research Associate in the Department of Geography at the University of South Carolina. Her research is on how social connectedness in families has changed over time, focusing on how families dispersed in space and how this affects relationships. Her presidential address "The Changing Space of Families" was published in Social Science History (2019,Volume 43:1-29). Most recently she has been studying migration over generations in Big Data which comes from family trees posted by users to Rootsweb.

Jack Grieve is a Professor of Corpus Linguistics in the Department of English Language and Linguistics at the University of Birmingham, UK. His research focuses on corpus linguistics, dialectology and forensic linguistics. He is especially interested in the analysis of regional linguistic variation in large collections of natural language.

ORCID

Caglar Koylu http://orcid.org/0000-0001-6619-6366

Data and Codes Availability Statement

The family tree data used in this study were derived from the following resource available in the public domain: https://home.rootsweb.com between the dates of February and August 2015. Based on the effective date of the Revised Terms and Conditions of Ancestry.com by 25 July 2019, we are not able to share the original GEDCOM files published on rootsweb.com. However, we provide mocked data and the entire Java code of our workflow and the instructions for reproducing the statistics and figures are available with a DOI at: https://doi.org/10.6084/m9.figshare.12349868.v1 Our generic workflow and code can be used to analyze the standard GEDCOM files from any other ancestry application such as www.familysearch.org and www.geni.com. Furthermore, we included the pseudo-code of our workflow in the Appendix 2, to enable the reproducibility of the work in other programming languages.

References

Adams, J.W. and Kasakoff, A.B., 1984. Migration and the family in colonial New England: the view from genealogies. *Journal of Family History*, 9 (1), 24–43. doi:10.1177/036319908400900102.

Antonie, L., *et al*., 2014. Tracking people over time in 19th century Canada for longitudinal analysis. *Machine Learning*, 95 (1), 129–146. doi:10.1007/s10994-013-5421-0.

Bailey, M., *et al*., 2017. *How well do automated methods perform in historical samples? Evidence from new ground truth*. Cambridge, USA: National Bureau of Economic Research.

Bhattacharya, I. and Getoor, L., 2004. Iterative record linkage for cleaning and integration. *In*: *Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, Paris, France, 11–18.

Bloothooft, G. and Schraagen, M., 2015. Learning name variants from inexact high-confidence matches. *In*: G. Bloothooft, P. Christen, K. Mandemakers and M. Schraagen, eds. *Population reconstruction*. Cham: Springer, 61–83. doi:10.1007/978-3-319-19884-2_4

Charpentier, A. and Gallic, E., 2020. Using collaborative genealogy data to study migration: a research note. *The History of the Family*, 25 (1), 1–21. doi:10.1080/1081602X.2019.1641130.

Cheshire, J.A. and Longley, P.A., 2012. Identifying spatial concentrations of surnames. *International Journal of Geographical Information Science*, 26 (2), 309–325. doi:10.1080/13658816.2011.591291.

Christen, P., 2006. A comparison of personal name matching: techniques and practical issues. *In*: *Sixth IEEE International Conference on Data Mining-Workshops (ICDMW'06)*, Hong Kong, China, 290–294.

Christen, P., 2012. *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Germany: Springer-Verlag Berlin Heidelberg.

Curtis, R.E. and Girshick, A.R., 2017. Estimation of recent ancestral origins of individuals on a large scale. *In*: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Halifax, NS, Canada, 1417–1425.

Elmagarmid, A.K., Ipeirotis, P.G., and Verykios, V.S., 2007. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19 (1), 1–16. doi:10.1109/TKDE.2007.250581.

Erlich, Y., *et al*., 2018. Identity inference of genomic data using long-range familial searches. *Science*, 362 (6415), 690–694. doi:10.1126/science.aau4832.

Fischer, D.H., 1989. *Albion's seed: four British folkways in America*. America: A Cultural History.

Gellatly, C., 2015. Reconstructing historical populations from genealogical data files. *In*: G. Bloothooft, P. Christen, K. Mandemakers and M. Schraagen, eds. *Population reconstruction*. Cham: Springer, 111–128. doi:10.1007/978-3-319-19884-2_6

Getoor, L. and Machanavajjhala, A., 2012. Entity resolution: theory, practice & open challenges. *Proceedings of the VLDB Endowment*, 5 (12), 2018–2019.

Goeken, R., *et al*., 2016. *Evaluating the accuracy of linked US census data: a household approach*. Retrieved Available at Request of the Authors.

Guo, D., *et al*., 2015. Historical Population Informatics: Comparing Big Data of Family Trees and the US 1880 Census for Migration Analysis.

Han, E., *et al*., 2017. Clustering of 770,000 genomes reveals post-colonial population structure of North America. *Nature Communications*, 8, 14238. doi:10.1038/ncomms14238.

Hey, D., 2010. *The Oxford companion to family and local history*. Oxford, England: OUP Oxford.

Huang, Y., *et al*., 2016. Understanding U.S. regional linguistic variation with Twitter data analysis. *Computers, Environment and Urban Systems*, 59, 244–255. doi:10.1016/j.compenvurbsys.2015.12.003.

Jaro, M.A., 1978. *Unimatch: A record linkage system: users manual*. Washington, D.C., USA: Bureau of the Census.

Kandt, J., Cheshire, J.A., and Longley, P.A., 2016. Regional surnames and genetic structure in Great Britain. *Transactions of the Institute of British Geographers*, 41 (4), 554–569. doi:10.1111/tran.12131.

Kaplanis, J., *et al*., 2018. Quantitative analysis of population-scale family trees with millions of relatives. *Science*, 360 (6385), 171–175. doi:10.1126/science.aam9309.

Kasakoff, A.B., Lawson, A.B., and van Meter, E.M., 2014. A Bayesian analysis of the spatial concentration of individual wealth in the US North during the nineteenth century. *Demographic Research*, 30, 1035. doi:10.4054/DemRes.2014.30.36

Koylu, C., *et al*., 2014. Mapping family connectedness across space and time. *Cartography and Geographic Information Science*, 41 (1), 14–26. doi:10.1080/15230406.2013.865303.

Kugler, T.A. and Fitch, C.A., 2018. Interoperable and accessible census and survey data from IPUMS. *Scientific Data*, 5, 180007. doi:10.1038/sdata.2018.7

Levenshtein, V.I., 1966. Binary codes capable of correcting deletions, insertions and reversals. *Dokl. Akad. Nauk SSSR,* 163 (4), 845–848.

Newcombe, H.B., *et al*., 1959. Automatic linkage of vital records. *Science*, 130 (3381), 954–959. doi:10.1126/science.130.3381.954.

Nix, E. and Qian, N., 2015. *The fluidity of race:"Passing" in the United States, 1880-1940*. Cambridge, MA, USA: National Bureau of Economic Research.

Otterstrom, S.M. and Bunker, B.E., 2013. Genealogy, migration, and the intertwined geographies of personal pasts. *Annals of the Association of American Geographers*, 103 (3), 544–569. doi:10.1080/00045608.2012.700607.

Rahm, E. and Do, H.H., 2000. Data cleaning: problems and current approaches. *IEEE Data Engineering Bulletin*, 23 (4), 3–13.

Ruggles, S., Fitch, C., and Sobek, M., 2017. *Building a national longitudinal research infrastructure*. Minneapolis, MN, USA: University of Minnesota.

Ruggles, S. and Menard, R.R., 1995. The Minnesota historical census projects. *Historical Methods: A Journal of Quantitative and Interdisciplinary History*, 28 (1), 6–10. doi:10.1080/01615440.1995.9955308.

Russell, R.C., 1918. *Index*. Google Patents.

Sarawagi, S. and Bhamidipaty, A., 2002. Interactive deduplication using active learning. *In*: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, Edmonton, Canada, 269–278.
Sharpless, J.B. and Shortridge, R.M., 1975. Biased underenumeration in census manuscripts: methodological implications. *Journal of Urban History*, 1 (4), 409–439. doi:10.1177/009614427500100403.
Sobek, M. and Ruggles, S., 1999. The IPUMS project: an update. *Historical Methods: A Journal of Quantitative and Interdisciplinary History*, 32 (3), 102–110. doi:10.1080/01615449909598930.
Steckel, R.H., 1991. The quality of census data for historical inquiry: A research agenda. *Social Science History*, 15 (4), 579–599. doi:10.1017/S0145553200021313.
Williams, R.R., *et al*., 2001. Usefulness of cardiovascular family history data for population-based preventive medicine and medical research (the Health Family Tree Study and the NHLBI Family Heart Study). *The American Journal of Cardiology*, 87 (2), 129–135. doi:10.1016/S0002-9149(00)01303-5.
Winkler, W.E., 1990. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage.
Wrigley, E.A. and Schofield, R.S., 1983. English population history from family reconstitution: summary results 1600–1799. *Population Studies*, 37 (2), 157–184.

Appendices

1. Geocoding

1.1. Uncertainty of location names

Most users of rootsweb.com who generated the family trees in our study are in the U.S. and Canada. We mainly focus on these family trees in the U.S. and their origins in other countries. Non-US and Canada locations were established much earlier, and there are more extinct places, more misspellings and spelling variations of the place names due to the change in language over time. For the U.S., changes also occurred through time with the territories acquired in the North West and the West and subdivided places in the North and Northeast. These changes pose substantial challenges in mapping the population demography and migration, and thus, studying the demographic and geographic expansion of the country. Once states were established, there were some locations that were put into a neighboring state, but these were few and involved only small parcels of land: for example some towns in Rhode Island became part of Massachusetts in 1861 and others moved from Massachusetts to Rhode Island; Alexandria moved back and forth between Virginia and the District of Columbia. More dramatically, the Honey Warin 1939 between Iowa (then a Territory) and Missouri involved a strip along the length of the boundary between them.

Place names were entered by users who relied on their personal memories, records, genealogies and census, or an educated guess of where a person in their tree was born or died. Location names entered by users may be geocoded with no problems, however, the entry could still be inaccurate. The only way to check the entries would be to identify the same individual in a census, or a genealogical record to see whether the locations match. In this study, we consider locations that we can match with a place name in our reference dataset as accurate. Future studies are needed to identify individuals in censuses and other sources to compare locations.

There are several challenges for geocoding due to the uncertainty of location names included in birth and death records. We classify uncertain location names into a category named “confusing”. We review these uncertain location names and adjust our reference datasets and our matching criteria to resolve some of the uncertainties and reduce the number of unmatched records.

1. Although the process of geocoding matches the place name with one location, sometimes two alternative locations were entered by users because the likelihood of location of birth or death were in two or more places. For example, a birthplace entry includes “North Carolina or Russia”.

Sometimes the conflict arises from the use of two different events for the location such as the location of baptism versus birthplace, or death location versus burial place.

2. Some place names include "prob. MA", which states the probable location is MA – likely to be Massachusetts, U.S. However, the location is not known with certainty.
3. Because we are using different reference datasets, there are conflicting matches for place names, especially when abbreviations are used. For example, CA matches Canada when using the country reference data, while it matches California using the US state reference data. Another example is "Beverley; WA", which may refer to "Beverley; West Australia" in the country reference set or "Beverley, Washington, USA" in the U.S. reference set. In similar cases in which location name and potential state or country name were used in lastfield1 and lastfield2, we picked the U.S. location over other locations in the world. This is because we "WA" and the location name "Beverley" both match with the U.S. reference dataset, which is prioritized and checked before the third reference dataset that Australia reference location names are in.
4. The US and states reference sets include full names, two letter, three letter and other common abbreviations, and common alternative spellings, and misspellings that we identified with Levenshtein similarity method. Some common spellings and misspellings are: ["American", "Amerika"] for the US; ["Ark", "Arkansa", "Arkanas"] for Arkansas; ["C A", "Californien", "Callifornia"] for California; ["Illinoise", "Illionois"] for Illinois; and ["Tenn", "Tenneessee"] for Tennessee.
5. There are errors in the original data. For example, a place name is entered as "Texas Co.; Montana". The abbreviation "Co." is used to refer to "County"; however, we cannot find Texas County in Montana.

1.2. Reference data

Our reference data consist of five datasets:

1. US and US states (full names, abbreviations that include two letter, three letter, and other common abbreviations, and common misspelling of names.
2. UK and historical UK place names
3. Canada, Netherland, Germany, Australia names and their first administrative division names
4. Other Europe country names
5. Other country names

1.3. Geocoding workflow

Place names require standardization since some records include special characters, numbers and multiple spaces. We used a multi-step filtering and standardization of place names in both the reference and the family tree data. We split the place name into three fields: "Name", "lastField2" and "lastField1", to categorize different levels of location hierarchy such as place names, state names and country names in the U.S. While we use "lastField1" and "lastField2" for a strict criterion of an "exact match" with the reference data, we use the "Name" field to match the place name based on "contains" criteria, which includes partial match of a place name in the reference dataset. Because the geocoding criterion are different, the reference datasets are also different for each of these three fields. In each of the reference sets we include values for match and values for exclusion to remove any unwanted matches. For example, the value "NO" matches with Norway in the country reference set when it appears in lastField1 or lastField2, while "NO" is used for exclusion if it appears in the "Name" field because "NO" is flagged with an exclusion rule in place names reference set since it is used to denote "number". Our workflow consists of three steps. In step 1, we clean the place names and extract the fields:

1. We replace all numbers (0-9), special symbols (like !, @, #…) except ";" and space with space. The output is called: "Name".

2. We extract the last field, i.e., the last meaningful segment, which is not equal to space or null, and not separated by semicolon. The output is called: "lastField1"
3. We extract the last field, i.e., the last meaningful segment separated by space. The output is called: "lastField2"

In step 2, we match the records with references based on lastField1 and lastField2, both of which are checked with our reference data using the "exact match" criterion. We compare place names with the rules and the reference datasets in the following order:

1. We check the record with the "confusing" reference data when the record may refer to different places. For example, CA may mean Canada or California. We add place names associated with Canada and California in the confusing reference data to resolve the uncertainty. This is done when we can parse the name into "lastField1" or "lastField2", and "Name".
2. We check whether the record match with lastField1, which is separated by semicolon. For example, NO (Norway), AS (Australia) are in the country reference data. However, if these abbreviations appear in middle of names, they may have other meanings (e.g., NO: number) rather than place names.
3. We check the record with the confusing reference data which includes names such as West Virginia, New Mexico, Austria-Hungary, and their variants. We generate the confusing reference set iteratively when we try to geocode unmatched place names in the confusing category. For example, a place name "Virginia" may be geocoded to West Virginia or Virginia because lastField1 and lastField2 do not exist in this record and "contains" criterion is used. To resolve this uncertainty, we update the confusing reference set and add a rule to match the record with "Virginia" when the place name cannot be parsed into those three fields.
4. We check the record with the reference datasets in the following order: U.S. reference data (4^{th}), U.K. reference data (5^{th}), Canada/German/Australia/Netherland (6^{th}), Europe countries (7^{th}), Other countries (8^{th}), historical countries (9^{th}). Both the lastField1 and lastField2 are used to match these reference data.

In step 3, we match the records with the references based on the Name field. Name field should contain the corresponding name in the reference data.

1. We check the record with "confusing" reference data.
2. We check the record with "contains" reference data.
3. We check the record with the reference datasets in the following order: U.S. reference data (3^{rd}), U.K. reference data (4^{th}), Canada/German/Australia/Netherland (5^{th}), Europe countries (6^{th}), Other countries (7^{th}), historical countries (8^{th}). The reference data used here is different from the reference data in step 1. For example, DE is deleted from U.S. reference data in step 2, since DE may have other meanings when it appears in the "Name" field.

All geocoded records are saved into eight tables based on their locations with appropriate attributes attached: confusingTable, usTable, ukTable, cagnTable (Canada / Australia / Germany / Netherlands), europeTable, othercountryTable, historicalcountryTable and unmatchedTable.

1.4. Evaluation

We checked the geocoding accuracy by checking the alternative matches in our reference sets and searching online sources (e.g., Wikipedia, Google, etc.) for the matched and unmatched place names. We geocoded all place names at state level for the U.S. and country level for the rest of the world. We first prioritized geocoding with the U.S. States reference dataset, and then checked the U.K. reference set and other country reference datasets. For place names that did not match with U.S. reference dataset, we checked whether the record match with the other country reference sets,

and the place name had to have the state and country name to be matched with those reference sets. If country name did not exist, we classified the record in "confusing" or in other words, "uncertain" category. This is rather a conservative approach that leaves many records unmatched. However, this was a choice we made to keep the uncertainty minimal because most of our analysis rely on location information. We applied the "winner takes all" approach to match uncertain place names with country information based on their frequency of occurrence to reduce the number of place names in the confusing category. Then, we matched the place names without a country or state name with the most common location occurred in those place names with the country or state name.

We selected 1% stratified sample from the geocoded data. We selected one geocoded place from every hundred place names 1st, 101st, 201st, 301st, and so on, to evaluate the geocoding accuracy. Overall, 98.05% of the selected records can be geocoded to the state and country level with exact match criterion. Among the unmatched records, we were able to match 49 out of 135 by adding the new exceptions into our reference datasets. For example, a place name that included an Australian State without the country name, "New South Wales", was wrongly geocoded to "Wales" which existed in "Wales" in the U.K. reference set. This was because records without a country name or U.S. state name are compared with the reference datasets using the "contains" criteria. "New South Wales" was matched with "Wales" in the U.K. reference set because of the prioritized order of search in the reference datasets.

There are several reasons why we did not use a geocoding software to check the accuracy. First, errors also exist in available geocoding software and packages. Second, our data is user-generated and unsuitable to be directly used for most geocoding software. Third, our reference datasets are from multiple sources and historical, and we employed different matching criterion for each of the reference data sources. These tasks are hard or even impossible to complete with the existing geocoding software and tools.

1.5. Limitations and future work

We geocode place names before eliminating the duplicates in trees. However, the process of deduplication in which we check whether locations match could be used to identify conflicting information, which we could use to evaluate the uncertainty of location names. The summary statistics of how many pairs match, and when they match what the differences could be used to enhance our geocoding methodology. Other genealogical databases have the user verify a place against an atlas with coordinates they maintain and prompt until they get a match. However, enforcing individuals to make a guess would increase the number of geocoded records as well as the uncertainty of locations.

There are several improvements to be made to the reference data. First, we included the common misspellings in our reference data and geocoded using the exact match criterion. However, in future work, we could use parts of the correct names to capture most of the misspelling names, for example, when a name contains "Pennsyl" it could be matched with "Pennsylvania" with a confidence score. Second, several reference location names are not used in step 2 because they relate to multiple locations, for example, IN, and NO. However, we may use, for example "NO", when there are no numbers following the place name. Third, we can add more reference data based on the unmatched names. We plan to geocode the unmatched data to place names in the U.S.

In addition to improving the reference data, our geocoding workflow could be improved. First, there is a need to store temporal snapshots of locations to better handle geocoding of historical place names. This may require storing country, state, and place names and their corresponding references for the most detailed temporal resolution if available. The resolution may be a decade or even a year if available. Second, we do not currently resolve some of the confusing category such as the issue of the inclusion of two different places: "Virginia or NC". In future work, we plan to use the cleaned family tree data for resolving such conflicts. For

example, we may use the nearest event such as the birth of a sibling or a child, and the death of a spouse. We could compare such events when there are two alternative locations in the confusing category to see if the nearest event matches with one of them or not. We have not used family relationships in the geocoding process. Because we already use location information to match individuals in multiple trees. Family tree relationships could be a valuable source for example, for checking the distances between spouses' birthplaces. We plan to evaluate our geocoding using family relationships in the future; however, such evaluation is challenging as the data, the trees we are studying, have high geographic mobility and diffusion over North America. Although individuals usually marry others who are from same ethnicity, origin or locations, we expect to see substantial variation of spousal relationships given the context of our study and time periods. We plan to study particularly spousal relationships in future work, that could potentially be useful to improve our geocoding process.

We have started to geocode at the county level in the US. However, the evaluation process is rather more difficult, and at this stage we report the results at the state level. In our latest experiment, we were able to geocode 99.77% of locations to the state level, while we were able to geocode 78.63% to the county level in the US.

2. Algorithms

In this section, we describe algorithms we created for steps 4 and 5 in our methodology described in section 3.1.

2.1. Saving person records into blocks

For each GEDCOM file, we first extract and save persons with detailed information including gender, birth year, birthplace, first name, last name, individual's family tree id, mother's, father's and spouse's information into blocks. Each GEDCOM file may include more than one family tree with unique id for each file. The algorithm 2.1 produces three outputs: (1) The blocks of persons indexed by birthplace and gender and sorted by birth year. We built each block (index) based on gender and birthplace of persons to improve the efficiency of the search queries to match identical persons in multiple trees. We then sort the persons in the same block by birth year. (2) A Hash map of individuals in which the key is block id, whereas the value is the person object that contain all features of an individual.

2.1. Algorithm for saving individual records into blocks for efficient processing

Input: *G:* GEDCOM file collection. Each GEDCOM file $g \in G$ include individual records.

Each individual record $i \in g$ include features: *id:* an individual's unique identification number within a tree, *bp*: birthplace, *ge*: gender, *by*: birth year, *ln*: last name, *fn*: first name, *dp*: death place, *dy*: death year, *f*: father, m: mother and *S*: spouse list and *FTID*: family tree id that is unique and equals to *g.id* (GEDCOM file id).

Output: *B:* The blocks of persons indexed by birthplace and gender and sorted by birth year.

personHashByBlock: The hash map of person hash maps by block. The key is block id and the values are the hash map of person objects that contain all attributes of a person within a block.

personID_Block: Hash map to store block id for each person.

personHashByTree: The hash map of person hash maps by each tree. The key is tree id and the values are the hash map of person objects that contain all attributes of a person within a tree. *personHashByTree* is used in Algorithm 2.3. Tree Cleaning and Deduplication.

```
initialize B as blocks, personHashByBlock, personID_Block, personHashByTree
rcnt = 0
foreach gedcom g ϵ G
    foreach person i ϵ g
        if i.ln, i.fn, i.by, i.bp and i.ge are empty or not valid
            continue
        p = createPerson(i)
        // assign unique tree id
        p.tid = i.FTID
        if i.f !=null
            p.father = createPerson(i.f)
        if i.m !=null
            p.mother = createPerson(i.m)

        foreach spouse s ϵ S && S.length < 3
            p.spouse = createPerson(s)
        blockid = i.bp + "'&" + i.ge
        if blockid is not in blocks
            blocks.add (blockid)
        personHash = personHashByBlock.get(blockid)
        if personHash == null
            initialize personHash

        personHashTree = personHashByTree.get(p.tid)
        if personHashTree == null
            initialize personHashTree

        p.id = rcnt
        personHash.put(p.id, p)
        personHashTree.put(p.id, p)
        personID_Block.put(p.id, blockid)
        personHashByBlock.add(blockid, personHash)
        personHashByTree.add(p.tid, personHashTree)
        rcnt += 1
    if rcnt > 2,000,000 or end of gedcom files
        blocks.sortByBirthYear()
```

2.2. Fuzzy matching and connecting family trees

Fuzzy matching consists of two steps: Weights assignment algorithm to match similar individuals and spousal pairs matching and tree clustering algorithms for connecting family trees.

2.2.1. Weights assignment for matching similar individuals

We use fuzzy feature weighting to add more weights to features that can be used to distinguish persons. Weights are determined based on available information about the person, his/her parents and spouse (i.e., the first spouse if a person has multiple spouses). The total weight (score) is 100 (%), and the initial match score is 25 because gender (15) and birthplace (10) are the same within each block. The weight for each feature is listed below:

Birth year: We compare the birth years of persons', their parents and spouses. If the difference is the maximum of 5 years, then the weight equals to 0.

- If birth years of persons are the same, the weight equals to 10.
- If birth years of persons' fathers are the same, then the weight equals to 2.
- If birth years of persons' mothers are the same, then the weight equals to 2.
- If birth years of persons' spouses are the same, the weight equals to 2.

2.2.1. Weight Assignment Algorithm for Fuzzy Matching

Input: *B:* Sorted blocks by birth year**.** Each person *p* include features: *id:* a person's unique identification number within a tree, *bp*: birthplace, *ge*: gender, *by*: birth year, *ln*: last name, *fn*: first name, *dp*: deathplace, *dy*: death year, *f*: father, m: mother and *S*: spouse list and *tid*: family tree id.

simJW(name1, name2): Returns the Jaro-Winkler similarity of two names
simLV(name1, name2): Returns the Levenshtein similarity of two names

maxSimLVLN(person1, person2): Returns the maximum Levenshtein similarity of two female persons' fathers', mothers' and spouses' last names.

Output: *simPairs*: The hash map of similar pairs of persons.

calculateFuzzyMatchScore(p1, p2): Returns the fuzzy match score between two persons**.**

getpairKey(p1, p2): Returns the unique comparison key for two persons p1 and p2.

```
initialize simPairs
foreach blockid b ϵ B
    for i = 0 to b.size()
        for j = i + 1 to b.size()
            p1 = b.get(i)
            p2 = b.get(j)
            if p2.by - p1.by > 5
                break
            if simJW(p1.fn, p2.fn) < 0.7 and simJW(p1.ln, p2.ln) < 0.7
                continue
            key = getPairKey(p1, p2)
            score = calculateFuzzyMatchScore(p1, p2)
            if score >= 67
                simPairs.put(key, [p1, p2])

function getPairKey(p1, p2):
    key = ""
    if p1.tid < p2.tid
        if p1.id < p2.id
            key = p1.tid + "_" + p1.id + "_" + p2.tid + "_" + p2.id
        else
            key = p1.tid + "_" + p2.id + "_" + p2.tid + "_" + p1.id
     else
        if p1.id < p2.id
            key = p2.tid + "_" + p1.id + "_" + p1.tid + "_" + p2.id
        else
            key = p2.tid + "_" + p2.id + "_" + p1.tid + "_" + p1.id
    return key

function calculateFuzzyMatchScore(p1, p2):
    // gender and birthplace known within each block:
    score = 25

    //compare birth years:
    score = score + 10 - abs(p2.by – p1.by) * 2
    score = score + 2 - abs(p2.f.by -p1.f.by) * 0.4
    score = score + 2 - abs(p2.m.by - p1.m.by) * 0.4
```

```
38      score = score + 2 - abs(p2.s.by – p1.s.by) * 0.4
39
40      //compare death years:
41      score = score + 3 - abs(p2.dy - p1.dy) * 0.6
42      score = score + 1 - abs(p2.f.dy - p1.f.dy) * 0.2
43      score = score + 1 - abs(p2.m.dy - p1.m.dy) * 0.2
44      score = score + 1 - abs(p2.s.dy - p1.s.dy) * 0.2
45
46      //compare first names:
47      score = score + (3 - simLV(p1.fn, p2.fn)) * 3.3
48      score = score + (3 - simLV(p1.m.fn, p2.m.fn)) * 2.3
49      score = score + (3 - simLV(p1.s.fn, p2.s.fn)) * 2.3
50      score = score + (3 - simLV(p1.f.fn, p2.f.fn)) * 2.3
51
52      //compare last names:
53      if p1.ge == male
54          score = score + (3 - simLV(p1.ln, p2.ln)) * 3.3
55      else
56          score = score + maxSimLVLN(p1, p2)
57
58      //compare birth places:
59      if p1.f.bp == p2.f.bp
60          score = score + 2
61      if p1.m.bp == m.f.bp
62          score = score + 2
63      if p1.s.bp == p2.s.bp
64          score = score + 2
65
66      //compare deathplaces
67      if p1.dp == p2.dp
68           score = score + 3
69      if p1.f.dp == p2.f.dp
70           score = score + 1
71      if p1.m.dp == m.f.dp
72           score = score + 1
73      if p1.s.dp == p2.s.dp
74           score = score + 1
75
76      return score
77
```

Death year: We compare the death years of persons', their parents and spouses. If the difference is the maximum of 5 years, then the weight equals to 0.

- If death years of persons are the same, then the weight equals to 3.
- If death years of persons' fathers are the same, then the weight equals to 1.
- If death years of persons' mothers are the same, then the weight equals to 1.
- If death years of persons' spouses are the same, then the weight equals to 1.

First name: Using Levenshtein similarity, we compare the first name of persons, their parents and spouses.

- If persons' first names match exactly, then the weight equals to 10.
- If persons' mothers' first names match exactly, then the weight equals to 7.
- If persons' fathers' first names match exactly, then the weight equals to 7.
- If persons' first spouses' first names match exactly, then the weight equals to 7.

Last name: We employ a gender-based comparison for last names. If persons to be compared are males, then we compare their last names directly. If persons to be compared are females, then we compare their last names, their fathers' and husbands' last names if they exist. We use the highest matching score out of the three comparisons. If persons' last names match exactly, then the weight equals to 10.

Birthplace: We compare the geocoded birthplace of persons' parents and spouses. Birth places of persons are the same since they are in the same block.

- If persons' fathers' birthplaces match, then the weight equals to 2.
- If persons' mothers' birthplaces match, then the weight equals to 2.
- If persons' spouses' birthplaces match, then the weight equals to 2.

Deathplace: We compare the geocoded deathplace of persons, their parents and spouses.

- If persons' death places match, then the weight equals to 3.
- If persons' fathers' death places match, then the weight equals to 1.
- If persons' mothers' death places match, then the weight equals to 1.
- If persons' spouses' death places match, then the weight equals to 1.

2.2.2. Spousal pairs matching algorithm

Given the suspected (candidate) pairs of persons, we applied the spousal pairs matching algorithm to identify the candidate husband-wife pairs that are similar in two trees. For each matching person pair, we first check whether they have spouses, and whether their spouses also have a match score equal to or greater than 67. We then create a hash map of candidate husband-wife pairs with their unique family tree ids. Husband-wife pairs are then used to connect and group trees into tree clusters.

2.2.2. Spousal Pairs Matching and Tree Clustering Algorithm

Input: *simPairs*: The hash map of similar pairs of persons**.** *personHashByBlock:* Hash map of person hash maps by block. *personID_Block:* Hash map to store block id for each person.

connectedComponents(Graph g): Given a graph g, this method returns all connected components into a list of subgraphs reverse-sorted by the size of nodes in each graph. Thus, the largest cluster is the first subgraph in the list. The largest connected component equals to the input graph g if all nodes are connected.

Output: *shwPairs*: The list of suspected (candidate) husband wife pairs with person ids and tree ids. There could be more than one matching husband-wife pairs for connecting the two trees.

gethwPairsKey: Returns the unique comparison key for husband-wife pairs of person1-spouse1 and person2-spouse2.

treeGraph: The graph of trees in which a node represents a family tree by id, a link represents the connection between two trees that is derived from the matching husband-wife pairs.

treeClusters: The set of subgraphs of treeGraph each of which consists of connected trees.

```
initialize shwPairs, treeGraph, treeClusters
foreach pair sp ϵ simPairs
        person1 = p.getValues[0]
        person2 = p.getValues[1]
        tree1 = person1.tid
        tree2 = person2.tid
        spouse1_block = personID_Block.get(person1.s)
        personHashSpouse1 = personHashByBlock.get(spouse1_block)
        spouse1 = personHashSpouse1.get(person1.s)
        spouse2_block = personID_Block.get(person2.s)
        personHashSpouse2 = personHashByBlock.get(spouse2_block)
        spouse2 = personHashSpouse2.get(person2.s)
        keySpouse = getPairKey(spouse1, spouse2)
        if simPairs.get(keySpouse) != null
              keyhwPair = gethwPairsKey(person1, spouse1, person2, spouse2)
              shwPairs.put(hwkey, [tree1, person1, spouse1, tree2, person2, spouse2])
              treeGraph.addLink(tree1, tree2)

treeClusters = connectedComponents(treeGraph)

function gethwPairsKey(person1, spouse1, person2, spouse2):
        keyhwPair = ""
        spousePair1 = ""
        if person1.id < spouse1.id
             spousePair1 = person1.tid + "_" + person1.id + "_" + spouse1.id
        else
             spousePair1 = person1.tid + "_" + spouse1.id + "_" + person1.id

        spousePair2 = ""
        if person2.id < spouse2.id
             spousePair2  = person2.tid + "_" + person2.id + "_" + spouse2.id
        else
             spousePair2 = person2.tid + "_" + spouse2.id + "_" + person2.id

        if person1.tid < person2.tid
              keyhwPair = spousePair1 + "_" + spousePair2
        else
              keyhwPair = spousePair2 + "_" + spousePair1

        return keyhwPair
```

2.3. Tree cleaning and deduplication

The output of the fuzzy match algorithm generates a list of candidate husband-wife pairs that are used to connect the trees and form the tree clusters. During this process, some trees become redundant because all the information in a tree can already exist in a newly formed tree cluster. We clean the trees and remove the duplicates within each tree cluster using the algorithms described in sections 2.3.1, 2.3.2. and 2.3.3. In tree cleaning and deduplication process, we use the cleaned and geocoded trees created through Steps 1-4 in our methodology. In Step 4, we use the algorithm defined in Appendix 2.1. to generate personHashByTree, the data structure that contains a hash map of person objects by each family tree.

2.3.1. Tree cleaning

First, we go through each tree cluster that consists of trees that have matching husband-wife pairs and clean trees within each cluster using a set of rules described below. We first remove persons:

- who do not have any parents, children, or a spouse.
- who only had little or no information (e.g., persons who did not have first name or birth year).

2.3.1. Tree Cleaning Algorithm

Input: *personHashByTree:* The hash map of person hash maps by each tree.

removeParent(person, parent: mother or father): Removes parent from person object.

Output: *personHashByTree:* The hash map of persons by trees.

```
1   foreach treeCluster tc ϵ treeClusters
2       foreach tree t ϵ tc
3           personHash = personHashByTree.get(t.id)
4           foreach person p ϵ personHash
5               // if person does not have descendants and spouse
6               if (p.f == null and p.m == null) or p.C == null or p.S ==null
7                   personHash.removePerson(p.id)
8                   continue
9               // if person has little or inconsistent temporal information
10              if p.by == null or p.fn == null or p.by > p.dy or abs(p.s.by – p.by) > 60
11                  personHash.removePerson(p.id)
12                  continue
13              // remove person with inconsistent links
14              if (p.by - p.f.by) < 12
15                  p.f = null
16              if (p.by - p.m.by) < 12
17                  p.m = null
18              // if person has multiple fathers or mothers
19              if p.f is Array
20                  p.f = p.f[0]
21              if p.m is Array
22                  p.m = p.m[0]
23              // if parent-child relation is bidirectional
24              personHashFather = personHashByTree.get(p.f)
25              father = personHashFather.get(p.f)
26              if p not in father.C
27                  removeParent(p, father)
28              personHashMother = personHashByTree.get(p.m)
29              mother = personHashFather.get(p.m)
30              if p not in mother.C
31                  removeParent(p, mother)
32
```

- who have inconsistent temporal information such as a record in which the birth year is greater than the death year or a person's age is greater than 120.
- who have inconsistent links. We classify a link as inconsistent if:
- The age difference between a person and his/her spouse was greater than 60.
- A person's birth year was less than 12 years of his/her father or mother's birth year.

We then examine and reconstruct the family relationship based on the following two rules:

- A person can have only one father and/or mother. We removed the parental links for persons who had multiple fathers or mothers.
- The parent-child relationship is bidirectional, which means A is listed as a child of B, then B would be one of the parents of A.

2.3.2. Iterative tree search for identifying the "true" duplicate spouse pairs

We conducted a relation-based iterative search to identify the "true" duplicate spouse pairs. For each candidate (suspected) husband-wife pair detected in the fuzzy matching process:

1. Check whether the duplicate pairs' parents (i.e., husband's mother and father, and wife's mother and father) are already in the candidate husband-wife pair list.
2. If father-mother pairs are already in the candidate husband-wife pairs list, then classify the candidate husband-wife pair as true (matching) husband-wife pair and skip step 3. If not, continue with step 4.

2.3.2. Iterative Tree Search for Identifying the "True" Matching Spouse Pairs

Input: *shwPairs*: The list of suspected (candidate) husband-wife pairs with person ids and tree ids.

shwPairsVisited: The list of visited pairs.

personHashByTree: The hash map of persons by trees.

maxSimJWLN(person1, person2): Returns the maximum Jaro-Winkler similarity of two female persons' fathers', mothers' and spouses' last names.

getpairKey(p1, p2): Returns the unique comparison key for two persons p1 and p2.

gethwPairsKey(p1, s1, p2, s2): Returns the unique comparison key for husband-wife pairs of person1-spouse1 and person2-spouse2.

treeClusters: The set of subgraphs of treeGraph each of which consists of connected trees.

Output: *hwPairs*: The list of true duplicate husband-wife pairs with person ids and tree ids.

calculateConflictScore(p1, p2): Given two persons, this function returns a score between 0 and 100, which reflects whether the two records are different persons. Conflict score is based on gender, birthplace, death place, birth year, death year, first name and last name.

conflictScoreHash: The hash map to store a pair of persons with their conflict score.

```
1     initialize hwPairs
2     while shwPairs.length > 0
3           shw = shwPairs.next()
4           if shw.key in shwPairsVisited
5                 continue
6           shwPairsVisited.add(shw.key)
7           tree1 = shw[0]
8           p1 = shw[1]
9           s1 = shw[2]
10          tree2 = shw[3]
11          p2 = shw[4]
12          s2 = shw[5]
13          p1_p2_key = getPairKey(p1, p2)
14          s1_s2_key = getPairKey(s1, s2)
15
16          // parents of persons p1 and p2
17          personHashTree1 = personHashByTree.get(tree1)
18          f1 = personHashTree1.get(p1.f)
19          m1 = personHashTree1.get(p1.m)
20          personHashTree2 = personHashByTree.get(tree2)
21          f2 = personHashTree2.get(p2.f)
22          m2 = personHashTree2.get(p2.m)
```

```
23        keyParents = gethwPairsKey(f1, m1, f2, m2)
24        // parents of spouses s1 and s2
25        fs1 = personHashTree1.get(s1.f)
26        ms1 = personHashTree1.get(s1.m)
27        fs2 = personHashTree2.get(s2.f)
28        ms2 = personHashTree2.get(s2.m)
29        keySParents = gethwPairsKey(fs1, ms1, fs2, ms2)
30
31        hwTruePair = false
32        if shwPairs.get(keyParents) != null and shwPairs.get(keySParents) != null
33              hwTruePair = true
34        else
35              // check the fathers
36              f1_f2_key =  getPairKey(f1, f2)
37              score_f1f2 = conflictScoreHash.get(f1_f2_key)
38              if score_f1f2 == null
39                    score_f1_f2 = calculateConflictScore(f1, f2)
40                    conflictScoreHash.put(f1_f2_key, score_f1_f2)
41              // check the mothers
42              m1_m2_key =  getPairKey(m1, m2)
43              score_m1m2 = conflictScoreHash.get(m1_m2_key)
44              if score_m1m2 == null
45                    score_m1_m2 = calculateConflictScore(m1, m2)
46                    conflictScoreHash.put(m1_m2_key, score_m1_m2)
47
48              // check the fathers of the spouses
49              fs1_fs2_key =  getPairKey(fs1, fs2)
50              score_fs1_fs2 = conflictScoreHash.get(fs1_fs2_key)
51              if score_fs1_fs2 == null
52                    score_fs1_fs2 = calculateConflictScore(fs1, fs2)
53                    conflictScoreHash.put(fs1_fs2_key, score_fs1_fs2)
54              // check the mothers of the spouses
55              ms1_ms2_key =  getPairKey(ms1, ms2)
56              score_ms1ms2 = conflictScoreHash.get(ms1_ms2_key)
57              if score_ms1ms2 == null
58                    score_ms1_ms2 = calculateConflictScore(ms1, ms2)
59                    conflictScoreHash.put(ms1_ms2_key, score_ms1_ms2)
60
61              // check the conflict scores to determine the true matches
62              if score_f1_f2 <= 0.3 or score_m1m2 <= 0.3 or
63              score_fs1_fs2 <= 0.3 or score_ms1_ms2 <= 0.3
64                    hwTruePair = true
65                    shwPairs.put(keyParents, [tree1, f1, m1, tree2, f2, m2])
66                    shwPairs.put(keySParents, [tree1, fs1, ms1, tree2, fs2, ms2])
67
68        if hwTruePair == true
69              hwPairs.put(shw.key(), shw)
70              hwPairs.put(keyParents, [tree1, f1, m1, tree2, f2, m2])
```

```
71                hwPairs.put(keySParents, [tree1, fs1, ms1, tree2, fs2, ms2])
72          // check whether the child-spouse pairs are already in the suspected
73          // husband-wife pairs, shwPairs. If not, add each of the child-spouse pairs to
74          // hwPairs if there is no conflict information
75          children1 = p1.C
76          children2 = p2.C
77          foreach child1 ϵ children1
78                if child1.s == null
79                      continue
80                else
81                      foreach child2 ϵ children2
82                            if child2.s !=null
83                                  c1 = personHashTree1.get(child1.id)
84                                  c2 = personHashTree2.get(child2.id)
85                                  c1_c2_key =  getPairKey(c1, c2)
86                                  score_c1c2 = conflictScoreHash.get(c1_c2_key)
87                                  if score_c1c2 == null
88                                        score_c1_c2 = calculateConflictScore(c1, c2)
89                                        conflictScoreHash.put(c1_c2_key, score_c1_c2)
90                                  if score_c1c2 > 0.3
91                                        continue
92
93                                  cs1 = personHashTree1.get(c1.s)
94                                  cs2 = personHashTree2.get(c2.s)
95                                  cs1_cs2_key =  getPairKey(cs1, cs2)
96                                  score_cs1cs2 = conflictScoreHash.get(cs1_cs2_key)
97                                  if score_cs1cs2 == null
98                                        score_cs1_cs2 = calculateConflictScore(cs1, cs2)
99                                        conflictScoreHash.put(cs1_cs2_key, score_cs1_cs2)
100                                 if score_cs1cs2 <= 0.3
101                                       keySC= gethwPairsKey(c1, c2, cs1, cs2)
102                                       if shwPairs.get(keySC) != null
103                                             shwPairs.put(keySC, [c1.tid, c1, cs1, c2.tid, c2, cs2])
104                                       hwPairs.put(keySC, [c1.tid, c1, cs1, c2.tid, c2, cs2])
105
106   function calculateConflictScore(p1, p2):
107         conflictScore = 0
108         if p1.gender != p2.gender
109               return conflictScore
110         if p1.bp != p2.bp
111               conflictScore = 0.3
112         if p1.dp != p2.dp
113               conflictScore = conflictScore + 0.2
114
115         birthyearAbsDif = abs(p1.by - p2.by)
116         if birthyearAbsDif <= 5
117               conflictScore = conflictScore + birthyearAbsDif * 0.04
118         else
```

```
            conflictScore = conflictScore + 0.2

      deathyearAbsDif = abs(p1.by - p2.by)
      if deathyearAbsDif <= 5
            conflictScore = conflictScore + birthyearAbsDif * 0.04
      else
            conflictScore = conflictScore + 0.2

      nameSimilarity = simJW(p1.fn, p2.fn)
      if nameSimilarity < 0.7
            conflictScore = conflictScore + 0.2
      else
            conflictScore = conflictScore + (1- nameSimilarity) * 0.67

      if p1.ge == male
            lastnameSimilarity = simJW(p1.ln, p2.ln)
      else
            lastnameSimilarity = maxSimJWLN(p1.ln, p2.ln)
      if lastnameSimilarity < 0.7
                  conflictScore = conflictScore + 0.2
            else
                  conflictScore = conflictScore + (1- lastnameSimilarity) * 0.67

      if p1.bp != p2.bp
            conflictScore = conflictScore + 0.2
      if p1.bp != p2.bp
            conflictScore = conflictScore + 0.2
      return conflictScore
```

3. For each candidate duplicate husband-wife pair, compare both mothers' and fathers' information by calculating a score of conflicting information. Given two persons, calculate a score of conflicting information that range between 0 and 1. The score reflects whether the two records are from different persons if the score is above the threshold of 0.3. Unlike the fuzzy matching score, conflict score considers only a few but major features of person records, that are gender, birthplace, death place, birth year, death year, first name and last name. For example, if gender does not match, then the score is 0. We check whether the conflicting scores of mothers' (husband's mother and wife's mother) and fathers' (husband's father and wife's father) exceed the predefined threshold of 0.3. If the score is below the threshold for at least one of the parents' comparisons, then classify the candidate husband-wife pair as true (matching) husband-wife pair.
4. Check whether the duplicate pairs' child (ren) has/have spouses and whether at least one of the child-spouse pairs is already in the candidate husband-wife pair list. Classify the candidate duplicate child-spouse pairs as true (matching) husband-wife pair and skip step 5. If not, continue with step 5.
5. Calculate the conflict score calculation to each child-spouse pairs if they were not already in the candidate list of husband-wife pairs. Add the child-spouse pairs into the suspect duplicate husband-wife lists if there is no conflict information.

2.3.3. Identifying representative person identification (id) numbers

We extract the representative person from the duplicates based on the amount of information. If two records were classified as duplicates, the record with more known information (e.g., gender, first name, last name, birthplace, death place, birth year, death year, father, mother, spouse, and children) was selected as representative person. If the two or more records have the same amount information, then we randomly chose one of the records as the representative person. We use the representative person to substitute other duplicate person records. Going through the true

2.3.3. Identifying representative person identification (id) numbers

Input: *hwPairs*: a list of true husband wife pairs with person ids and tree ids.

Output: *personHashByTree:* Removed duplicates from person hash maps by tree.

representativeIDHash: The hash map of each person id to a unique representative id. All person ids that point to the same representative ids are merged, and thus duplicates are removed.

duplicatesGraph: The graph in which each node is a person id, and a link is the connection between two persons, which means the two persons are the same individual.

duplicatesList: The list that contains lists of person ids that refer to the same person. *duplicatesList* is created by using *connectedComponents* function on *duplicatesGraph.*

connectedComponents(Graph g): Given a graph g, this method returns all connected components into a list of subgraphs reverse-sorted by the size of nodes in each graph. Thus, the largest cluster is the first subgraph in the list. The largest connected component equals to the input graph g if all nodes are connected.

calculateInformationScore(person): Returns a score that reflects the number of available features in a person record (i.e., gender, first name, last name, birthplace, death place, birth year, death year, father, mother, spouse and children). Each feature counts as one, and the score ranges between 0 and 11 (the total number of features).

```
1   initialize duplicatesGraph
2   foreach pair hw ϵ hwPairs
3       tree1 = hw[0]
4       person1 = hw[1]
5       spouse1 = hw[2]
6       tree2 = hw[3]
7       person2 = hw[4]
8       spouse2 = hw[5]
9       duplicatesGraph.addLink(person1, person2)
10      duplicatesGraph.addLink(spouse1, spouse2)
11  duplicatesList = connectedComponents(duplicatesGraph)
12
13  // assign representative id based on the amount of information
14  foreach duplicates d ϵ duplicatesList
15      max_score = 0
16      representativeID = -1
17      foreach person p ϵ d
18          p_score = calculateInformationScore(person)
19          if p_score > max_score
20              max_score = p_score
21              representativeID = p.id
22      foreach person p ϵ d
23          representativeIDHash.put(person.id, representativeID)
24
```

```
// remove duplicate records
foreach treeCluster tc ϵ treeClusters
    foreach tree t ϵ tc
        personHash = personHashByTree.get(t.id)
        foreach person p ϵ personHash
            representativeID = representativeIDHash.get(person.id)
            if representativeID == null
                representativeIDHash.put(person.id, person.id)
            else
                personHash.remove(person)
                continue

function calculateInformationScore(person):
    score = 0
    if person.ge !=null
        score++
    if person.ln !=null
        score++
    if person.fn !=null
        score++
    if person.by !=null
        score++
    if person.dy !=null
        score++
    if person.bp !=null
        score++
    if person.dp !=null
        score++
    if person.m !=null
        score++
    if person.f !=null
        score++
    if person.S !=null
        score++
    if person.C !=null
        score++
    return score
```

duplicate husband-wife pair list, we remove each duplicate person record until there are no more duplicate pairs.

3. Regression Results

We chose the state level population proportion (the number of individuals alive in the U.S. in 1880 in family trees divided by the number of individuals in 1880 Census) as the dependent variable. Our candidate independent variables were state level percentage of following population segments: white, farmer, individuals greater than 54 years old, male, individual's birthplace in the same state as the 1880 location, and individual's birthplace in foreign countries. The percentage of individual's birthplace in the same state as the 1880 location equals to the number of individuals born in the same state as the 1880 locations divided by the total population in 1880 in each state. The percentage of individual's birthplace in foreign countries equals to the number of individuals born in the other countries divided by the total population in 1880 in each state. Figure 3.1 illustrates the dependent variable and independent variables by states. The family trees contain a higher proportion of the population in the Middle and Eastern states of West Virginia, Indiana, Kentucky, Tennessee and Arkansas. It is interesting that in the South where there was a large Black population, the proportion of the population in trees is less than elsewhere and other states which

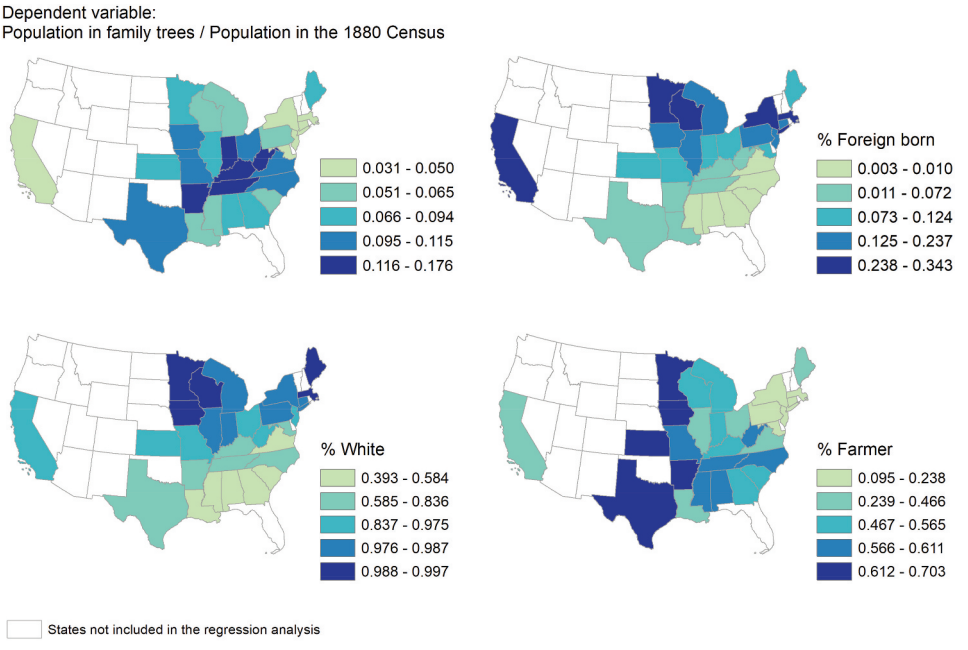

Figure 3.1. Dependent variable: 1880 family tree population / 1880 Census population and independent demographic variables.

are less well represented in trees have high proportions of foreign born. There were relatively more white and foreign born populations in the upper Eastern U.S. There were relatively more farmers in the middle U.S.

Table 3.1 shows the regression result. The coefficients of the males and individuals greater than 54 years old were not significant. When we removed these two variables from the model, the coefficient of the percentage of individuals' birthplaces in the same state became not significant. Thus, we excluded the birthplace variable from the model, the coefficient of all independent variables became significant, see Table 3.2.

We conducted further analyses to test whether the four assumptions of linear regression were satisfied (Figure 3.2). The residuals versus fitted values plot (Figure 3.2.a) showed no obvious sign of deviation in the residuals. The Shapiro normality test showed that the residuals were normally distributed at the 0.05 significance level. The studentized Breusch-Pagan test showed that the residuals had equal variance. The residuals versus leverage plot (Figure 3.2.b) indicates there were no influential cases. In addition, the multi-collinearity check showed that all variance inflation factors were less than 10. Therefore, multi-collinearity was not an issue. These test results showed that the linear regression was robust, and all model assumptions were met (Figure 3.2.c and d).

Table 3.1 Regression result 1.

Variable	Coefficients	Standard error	t value	Probability (> |t|)
Intercept	-0.28660	0.13224	-2.167	0.041310*
White percentage	0.14958	0.02720	5.500	1.58e-05***
Farm Percentage	0.10380	0.02717	3.821	0.000933***
Birthplace in the same state percentage	0.09339	0.03653	2.557	0.017983*
Birthplace in foreign country percentage	-0.19319	0.06245	-3.094	0.005304**
Age more than 54 percentage	-0.44886	0.26691	-1.682	0.106767
Male percentage	0.38135	0.22802	1.672	0.108593
Other model performance parameters				
Multiple R-squared	0.8386			
Adjusted R-squared	0.7946			
F statistics	19.05 on 6 and 22 DF, p-value: 1.097e-07			

Table 3.2. Regression result 2.

Variable	Coefficients	Standard error	t value	Probability (> |t|)
Intercept	-0.03018	0.01991	-1.516	0.14216
White percentage	0.10376	0.02359	4.399	0.000177***
Farm Percentage	0.11858	0.02214	5.357	1.48e-05***
Birthplace in foreign country percentage	-0.22897	0.04790	-4.780	6.60e-05***
Other model performance parameters				
Multiple R-squared	0.7687			
Adjusted R-squared	0.7409			
F statistics	27.69 on 3 and 25 DF, p-value: 4.111e-08			

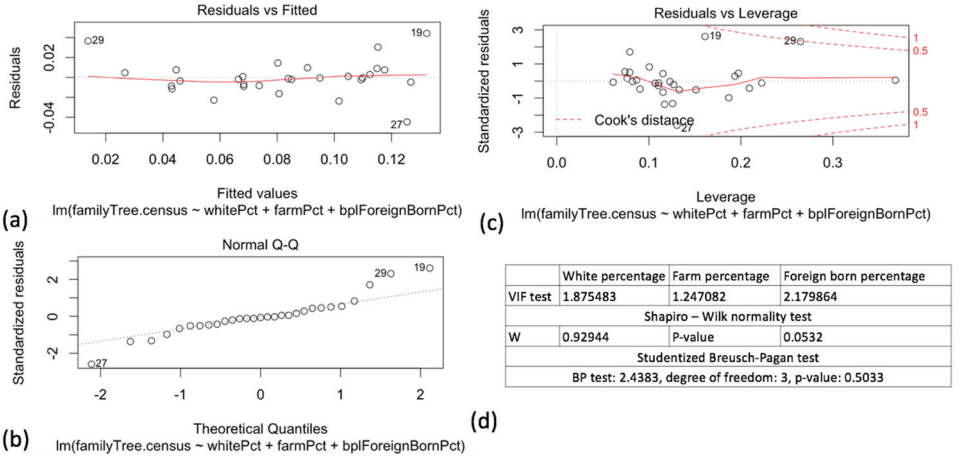

Figure 3.2. Tests for evaluating the regression model and results.