Merging in Genealogy Software and Genealogy Research

John A. Nairn

Oregon State University, Corvallis, Oregon, USA
Email: john@geditcom.com

A common problem for genealogists is finding two records and deciding if they are about the same person. For example, if you find a potential birth record for an ancestor, how can you be sure it is the same person you previously found in a census record with your family? The same problem arises in computer genealogy when doing collaborative research. When two researchers find results on the same new person, how can they be sure they found the same person and therefore should be merged into a single computer record? This chapter discusses matching and merging of computer genealogy records. It begins with the simpler problem of two or more people working on similar data using the same computer software. This situation can be handled reasonably well with existing computer tools known as version control systems. The more challenging problems are matching and merging data from disparate sources. Unfortunately, genealogy data are inherently fuzzy. For example, names can be spelled different ways and dates can be uncertain by indicating a range of possible dates. A potential solution is to develop a "quality" rating for degree of overlap and lack of conflict between two genealogy records. This "quality" approach is explored in general terms including suggestions for new genealogy tools, and areas for more development.

I. Introduction

Consider the problem of searching for an ancestor in a large Internet genealogy database. Before these databases were available, the main problem was finding any information at all. Now with a few text entries you can get thousands of matches. Unfortunately, many matches may be wrong, may be close to correct but lack sufficient detail to be certain, or may be repetitions of the same information. The main problem has shifted from too little information to too much. How do you search for your "needle in a haystack" of data? This problem needs attention. Genealogy databases are growing at a rapid rate. For genealogists, however, this growth seems more like accumulation of more "hay" rather than addition of new "needles." This growth needs to be accompanied by advances in searching and merging tools.

When confronted with volumes of data, the natural choice is to turn to computers. A packet of information on one individual may be called a computer

record. The computer problem is to compare two computer records and decide if they are about the same person. Computers are great at matching well defined data such as numbers or text strings, but genealogy data are fuzzy and difficult to match. For example, all genealogy research is involved with dates for events such as births, marriages, or deaths. Although computer operating systems have built-in tools for handling dates, these tools always consider a date to be a single day with a specific year, month, and date. Genealogy dates are rarely that certain. You may only know a year (born in 1858), a range of years (married before 1881), or a partial date (died in December 1927). Even these approximate dates may have errors meaning the actual date may differ by a few years from than the recorded date. Besides dates, names may be spelled different ways, appear in different forms, or may change, and place names may completely change name or switch countries since the day that a genealogy event occurred.

This chapter discusses the problem of matching computer records and dealing with such fuzzy data. A potential approach is to consider all available data and evaluate a "quality" of match based on overlap and lack of conflicts between those data. For specific types of data, such as dates, quality matching needs to be more sophisticated then whether or not two dates could be within some user-supplied tolerance. The matching needs to account for both precision and proximity of the dates. For example, matching two exact dates, such as 4 July 1776 compared to 4 July 1776, is a much better match than two partial dates, such as 1776 compared to 1776, even though unenlightened computer algorithms might consider these both to be exact matches. If enough data from two records match with sufficient quality and no data are in conflict, the records can be considered to match. All these factors and more need to be considered when comparing records.

Once matches can be found, the next step is to use that ability in computer genealogy. This chapter discusses two applications. The first is the task of merging two genealogy files. With cooperative genealogy software tools and a well-defined work flow, this problem can be solved well using existing computer tools known as version control systems. When the work brings in outside data, such merging will need access to quality matching and will normally require human input. Humans can be better than computers at the pattern-recognition skills needed to match fuzzy data. The second application is to rank search results. If search engines assigned quality scores to all matches in a genealogy search, those scores could be used to rank search results and provide the researcher with improved chances of finding useful data near the top of the list.

II. Collaborative Merging

When two or more people are researching the same family history, they usually want to coordinate their efforts. They may all be using the same software

system, but they likely will be working on different computers or even in different countries. The working group may even be a single person needing to sync research across multiple computers. The computer problem is to periodically merge each researcher's latest work into a master copy of the coordinated research. This problem is not unique to genealogists. It happens any time multiple people collaborate on projects involving computer documents. For example, large software development projects face this problem and have devised several solutions. The most popular solution is called version control, which allows a group to maintain a master copy of the computer code, for every participant to coordinate their efforts with the master copy, to track all changes made between versions, and to determine who made those changes. Three popular version control systems are Subversion (svn), Git, and Concurrent Versions System (cvs).[1] These systems are fairly easy to use and can often be directly applied to genealogy research.

The following three ingredients are crucial to successful version control of genealogy projects:

A Central Server: The master copy of version-controlled genealogy data is stored in a "repository," which needs to be on a server that is accessible to all people collaborating on the project. One good approach is to use web-based tools provided for secure and private storage of version control files, such as Beanstalk.[2] It is presented as a tool for managing collaborative, computer coding projects, but works fine, without modification, for collaborative, genealogy projects too.

Text-Based Genealogy Data: Version control systems work best when most of the important information is documented in plain-text files. These systems merge differences between two versions of a file and track the history of any changes by using text-differencing tools. One example of text-based genealogy data that works great with version control is the GEDCOM format, which stands for GENealogical Data COMmunication format.[3] Another alternative is any XML document. Unfortunately, some genealogy applications lock your data into proprietary, binary files. Such documents are unreadable by humans and not amenable to generic merging based on text methods tool. Although all version control systems can track binary documents, which is useful in genealogy projects for image files or other binary documentation, binary files have to be recopied each time a change is made. The repository will quickly grow large and it will provide little useful information on what has changed between versions. If you want to

[1] For more details, see the websites http://subversion.apache.org/ for svn, http://git-scm.com for git, and http://www.nongnu.org/cvs/ for cvs.

[2] See http://beanstalkapp.com/ - This service is free for typical genealogy projects and provides ample help for setting up and using version control by svn or git.

[3] The GEDCOM standard release 5.5 documentation can be found here: http://homepages.rootsweb.ancestry.com/~pmcbride/gedcom/55gctoc.htm

collaborate and merge efforts for genealogy projects using version control, you should always use applications that store their data in human-readable, text-based files.

Computer Skills: Although version control systems are free, they require command-line computer skills to install, to set up the repository, and in their day-to-day use. For those who prefer not to use command-line methods, various commercial applications act as version control clients that let you perform most needed tasks with a visual user interface.[4]

If the above ingredients are in place, the first step in using version control is to import an initial copy of the genealogy files to the repository on the available server. Typically a project will be a collection of text files and perhaps multimedia files located in a folder on your computer. These should be imported to a folder on the server. Once the repository is set up, each person involved in the project uses the following work flow:

1. Check out a copy of the project (or clone the entire repository when using git)
2. Make changes to any file or add and delete files
3. Commit the changes to the repository. All changes are merged into the master copy. If any conflicts arise (such as someone else changing the same data since the last saved version), you will be warned and asked to resolve them before proceeding.

A version control approach provides robust merging of genealogy records being edited by multiple members in a project. It is very generic and thus will work with any genealogy software you use, provided the key files are stored as plain-text files. You can use version control tools to track the history of changes, go back and retrieve prior versions (in case someone commits invalid data), create branches with exploratory data, tag stable versions, list changes you have made since last your last commit, and much more.

III. Merging Records for Disparate Sources

Collaborative merging, described above, is ideal when a group of people are working with the same software tools and making incremental changes to one collection of data. The more challenging problem is finding genealogy records from disparate sources and determining whether or not they are the same individual and therefore can be merged into your project. This problem arises when merging two sets of computer records into a single data collection and also occurs when searching. When searching, the problem is determining whether or not a new individual you find is the correct individual for a research question. This sec-

[4] Visual front-end applications are called “Clients” — Two examples for Mac are “Versions” (see http://versionsapp.com/) for svn and “GitX (L)” (see http://gitx.laullon.com/) for git.

tion discusses several issues related to comparing records from different sources and deciding if they are the same individual.

Unique Record Identifiers

One method used in database management to keep records straight is to assign each one a unique record identification number or ID. It is easy for computer genealogy tools to use IDs. For example, each record in a GEDCOM genealogy file has a unique ID within that file. Additional GEDCOM tags allow marking records with a permanent record number (RFN tag), with an ancestral file number from the LDS Church's Family History department (AFN tag), or an automated record ID (RIN tag). These GEDCOM tags are rarely used in GEDCOM files posted on the Internet. Some software tools mark records with universally unique IDs (UUID) or globally unique IDs (GUID).[5] UUIDs are made unique by several possible schemes. One method is to base them on a unique computer signature (known as the computer's Media Access Control, or MAC, address) along with a time stamp resolved to nanoseconds. Another approach is to randomly select a number. The enormous capacity of UUIDs (>10^{38} values) means one could randomly assign UUIDs to every human that ever lived and the chance that two would get the same ID would be so remote that it can be considered negligible.

Unfortunately, record IDs are of limited use in computer genealogy and in genealogy research. Their main use is a single user working with a single database of research on a single computer. If record IDs are kept unique within that confined system, they could function to positively identify records that refer to the same person and therefore should be merged. When a genealogy project expands to more complexity, record UUIDs becomes less useful. For example, consider two people researching one family history and each of them finding the same, new ancestor. When each researcher creates a record on their own computer, those two records would have different UUIDs. Those IDs therefore provide no help in merging those findings by two researchers. Merging in genealogy research cannot rely solely on records IDs; it also needs tools based on data in the records.

Name Matching

A good starting point for genealogy record matching is the person's name. Computer string matching can easily compare names, although it is best to break the name into parts — first, middle, and last names. The same person may be recorded with or without a middle name. Some people prefer their middle name, which might cause them to appear in records with their first and middle name switched from their birth name. Other individuals add to or replace middle names with a confirmation name from religious ceremonies. Because of these variants,

[5] For more details on UUIDs see
http://en.wikipedia.org/wiki/Universally_unique_identifier

two records with the same surname, but first and middle names having only a partial match or being switched may still refer to the same individual. An exact name match is best, but certain name mismatches do not imply a conflict.

Exact name matching by computer string comparisons is too restrictive. When it was more common for people being unable to read and write, their names were passed on orally and transcribed into records by someone else's handwriting. Such oral transmissions can cause variants in name spelling. Even when spelled correctly, later transcription of those handwritten records may misinterpret that handwriting leading to even more variants in spelling. A computer technique for dealing with modest misspellings is to convert names to a computer code based on phonetic name pronunciation rather than exact spelling. The best known system is called a Soundex code.[6] The Soundex algorithm retains the first letter (which therefore better be correct), removes vowels and a few other letters, assigns numbers to remaining letters by their sound, collapses repeated numbers into a single number, and finally keeps the first letter followed by the first three numbers (or trailing zeros if needed). For example, the name "Deveau" is converted to Soundex code D100 — "D" for the first letter, "e," "e,", "a", and "u" are removed, "1" for the "v" sound, and two zeros added to get a four-character code. If all name parts in a database are compared using Soundex codes, it may help find matching names that would be missed by strict string matching.

A Soundex code is useful, but has several limitations. First, any phonetic code depends on the language and Soundex was developed for English pronunciation. My wife's great-grandfather's name was "Felix Deveau." He was born in Cape Breton, Canada and was of French ancestry; most family records spell the name "Devot." These two names are pronounced the same by French-speaking individuals, but they are pronounced differently in English and therefore have different Soundex codes: D100 for "Deveau" and D130 for "Devot."[7] Alternative phonetic methods are available for other languages or names, such as for eastern European names,[8] but it is difficult to develop codes for all languages. Even if such codes were available, how could you be certain which language's code to use for a given name?

A second problem with phonetic codes is they do not handle spelling variations caused by misinterpretation of handwriting. For many years I could not find any history for the "Felix Deveau" mentioned above. That surname is common in

[6] Developed and patented in 1918 and 1922 by Robert C. Russell and Margaret K. Odell. The algorithm is given by Donald E. Knuth, "The Art of Computer Programming," vol. 3, pg. 391 (Addison-Wesley, Reading, MA, 1973).

[7] Starts with "D", "e" and "o" are removed, 1 for "v" sound, 3 for hard "t" sound, and one zero added to make four characters.

[8] The Daitch–Mokotoff Soundex system.

Canada and none of the records I found matched other known information. Eventually I found a Canadian archives web site[9] for census records that allowed me to search for all names beginning in "Dev." I found "Felix Devon" on Prince Edward Island. This record was transcribed from the handwritten name shown in Fig. 1. I think Felix could neither read nor write and the census recorder wrote his name as "Devow," but that handwriting was later misread and indexed in Canadian archives only as "Devon." After finding that census record, I found several church records for baptisms of his children. In those records, the name was written as "Devot," although sometimes they forgot to cross the "t" looking more like "Devoh" — see Cyril's and Oliver's names in Fig. 1. An important need for genealogy record matching is to have tools to compare names that look like each other in handwriting. Such a tool would recognize the similarity between "Devon" and "Devow" or between "Devoh" and "Devot," even though their pronunciations and Soundex codes are different.

However the quality for matching names is determined, whether it is by spelling, by pronunciation, or by appearance, name matching is never enough to confirm two records are for the same person. It is common for many people to have the same name. In fact, names are frequently reused in families when children are named after parents or other relatives. Matching genealogy records has to consider much more than names. Some options are discussed next.

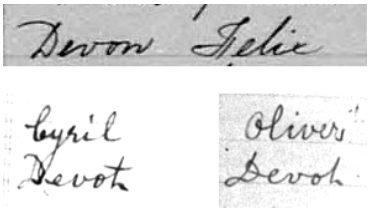

Figure 1: Felix Deveau was recorded in the 1881 Canadian census as "Felix Devow," which was later transcribed as "Devon". His son's names in church records look like "Devot" or "Devoh."

Date Matching

When comparing dates, a typical tool in genealogy database searching is to enter a date tolerance such as all individuals born 20 JAN 1895 ± 3 years. This criterion matches all individuals born between 20 JAN 1892 and 20 JAN 1898, but the number found might be enormous in a large data base. An ideal genealogy tool should rank these results by the likelihood they match the precise date of 20 JAN 1895. For example, those with birth date 20 JAN 1895 should be at the top. Those that match, but have lower precision, such as JAN 1895 or just 1895, should

[9] Canadian Census Collection: http://www.collectionscanada.gc.ca/census/index-e.html

be near the top, but lower. Finally, dates that do not match, but are still within ±3 years (*e.g.*, 1892 or 1898) should rank lowest. This section describes one computer algorithm for comparing two genealogy dates and ranking them by their match quality.

In genealogy research, many dates are "fuzzy" or imprecise dates. A date may be just a year (1895), just a month and a year (JAN 1895), or even a range of dates (between 1892 and 1898). Other common variants are open-ended ranges (before 1895) and generally uncertain dates (about 1895). A good date matching algorithm should obviously rank dates that are closer together as better matches then those that are father apart. That algorithm is trivial and can be based on time between midpoints of date ranges. A more advanced algorithm, should account for date precision as well. For example, finding an exact date match in genealogy research is a much better result than matching fuzzy dates, which means that 20 JAN 1895 and 20 JAN 1895 match much better than 1895 and 1895. To a computer, however, they both look like exact matches. Similarly, a researcher might conclude that 20 JAN 1895 matches 1895 better than it matches 15 FEB 1895. By time difference, 20 JAN 1895 is closer to 15 FEB 1895 than it is to 1895, but precise dates imply some confidence in those dates. When exact dates get farther apart, they become less likely to be referring to the same event, while the less precise date of 1895 is at least consistent with an event on 20 JAN 1895.

Genealogy dates can be mapped to a range of dates between two exact dates. These bounding exact dates can be mapped to serial day numbers or SDNs. In computer date calculations, an SDN assigns a calendar-independent integer to each day in history with day 1 being around 4500 BC and 1 JAN 2012 being 2222710. SDNs make date calculations easier for computers because they avoid calculations using months and years and avoid differences between calendars. For date calculations, genealogy dates are written as (x_1, x_2) where x_1 and x_2 are the minimum and maximum SDNs for the date. When $x_1 = x_2$, the date is an exact date. When $x_2 > x_1$, the date is a date range with imprecision or width of $\Delta x = x_2 - x_1$.

Imagine two date ranges SEP 1867 to MAR 1868 and 1868. How many days are between these two dates? The number could be -90 days (from 31 MAR 1868 to 1 JAN 1868) or it could be 486 days (from 1 SEP 1867 to 31 DEC 1868) or it could be any number between these two extremes. In other words, the difference between SEP 1867 to MAR 1868 and 1869 is between – 90 days and 486 days. But, we can say more. There is only one way the time difference can be – 90 days, while there are many ways it can be 200 days (212 ways to be exact). For any pair of dates, (x_1, x_2) and (y_1, y_2), ordered such that $\Delta y \geq \Delta x$, we can analyze their ranges and construct a probability distribution, $p(t)$, for the probability that the time difference between them is t days. The $p(t)$ for comparing SEP 1867 to MAR 1868 with 1868 is in Fig. 2. All distributions for comparing fuzzy dates have this characteristic "hat" shape. The sloped regions on the sides have length Δx, which will

become a vertical line when $\Delta x = 0$ for an exact date. The plateau's height is $1/\Delta y$ and its width is $\Delta y - \Delta x$; the plateau will collapse to a point when the two dates have the same width. For two exact dates, $p(t)$ is a vertical line at their separation in days — they have only one possible separation.

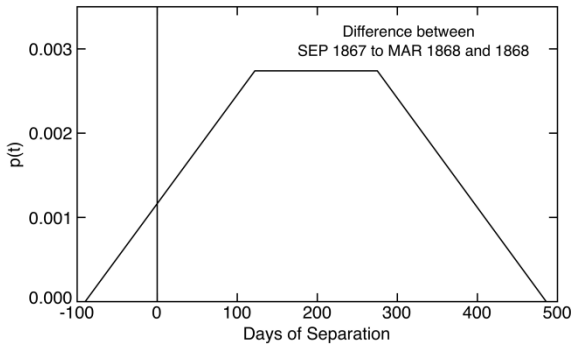

Figure 2: A probability distribution, p(t), for the probability that the difference between the date SEP 1867 to MAR 1868 and the date 1868 is t days..

In date comparisons, we are more concerned with the probability that two dates are *closer* than some number of days than of *being separated* by a number of days, as given by $p(t)$. This probability is known as a cumulative probability distribution function and is found by integrating $p(t)$:

$$\Gamma(t) = \int_{-t}^{t} p(u)du$$

In other words, $\Gamma(t)$ is the probability that the number of days between two dates is less than t, regardless of sign, and it is the area under the $p(t)$ function between $-t$ and t. Figure 3 shows $\Gamma(t)$ for various date pairs. Two matching exact dates rise immediately to 1.0 as a perfect match. Two matching, but less precise dates, rise more slowly as they become more imprecise (see curves for "Apr 1868 vs. Apr 1868" and "1868 vs 1868"). When dates are mismatched, the curve rises slower (see "1868 vs 1869") and may be offset by their minimum difference (see "Apr 1868 vs. Oct 1868"). For closer and more precise date pairs, $\Gamma(t)$ rises sooner and faster. In other words, faster rises in $\Gamma(t)$ implies higher quality date matching. This observation suggests a ranking that averages the cumulative distribution function, but gives more weight to short time differences than to longer ones. In other words, a potential date-matching quality ranking could use the following weighted average:

$$Q(\tau) = \frac{1}{\tau}\int_{0}^{\infty} e^{-t/\tau}\Gamma(t)dt = \frac{1}{\tau}\mathrm{L}(\Gamma(\mathrm{t})) = \mathrm{L}(p(t)) - \mathrm{L}(p(-t))$$

The weighting function is chosen as the exponential function, $e^{-t/\tau}$, which is 1 for $t = 0$ and decreases monotonically for higher t. The "lifetime" factor τ is a user-selectable factor that influences how the ranking treats date precision. Other weighting functions could be used, but choosing an exponential function turns the quality equation into a Laplace transform (indicated with L()). Furthermore, by using the integral relation between $\Gamma(t)$ and $p(t)$ and Laplace transforms properties, $Q(\tau)$ can be determined from transform of $p(t)$ alone, thereby eliminating the need to ever calculate $\Gamma(t)$ — evaluation speed is always beneficial to computer tools. Finally, the $1/\tau$ factor is a normalizing factor. With that factor, $Q(\tau) = 1$ for comparing two exact and identical dates (*e.g.*, 20 JAN 1895 to itself). All other comparisons will have $Q(\tau) < 1$ and the quality will depend both on the time between the dates and on their precisions.

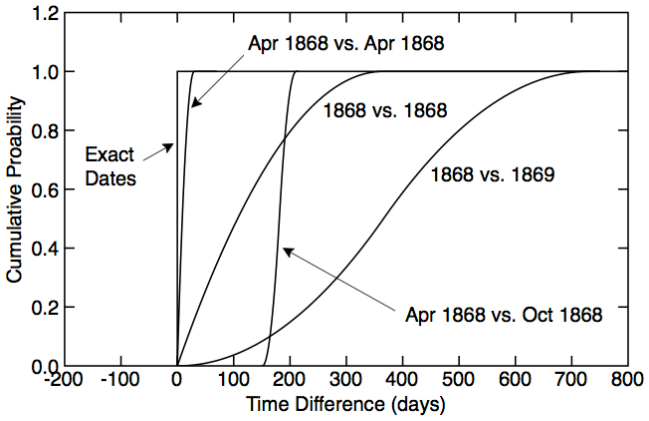

Figure 3: Cumulative probability distribution for the time difference between various dates.

Some refinements are helpful. First, $Q(\tau)$ for two dates is a positive, non-zero score, *i.e.*, it never rejects two dates as being different. A simple solution is to implement a Q_{min} below which the dates are assumed to be different. One option is to eliminate all dates with lower quality than two exact dates separated $\pm n$ days. From the evaluation of $Q(\tau)$ for exact dates, the cutoff is

$$Q_{\min} = e^{-n/\tau}$$

Second, a common date observation in genealogy research is to know an event happened "before" or "after" a date. For example, when you find a baptismal record but no birth date, that individual must have been born before their baptism. A reasonable ranking for date (y_1, y_2) to be before date (x_1, x_2) is to set x_1

= x_2, to make it an exact date at the end of its date range, and then consider only negative separations:

Table 1: Sample matching of 20 JAN 1895 to typical genealogy dates using τ = 50 days.

Date	$Q(\tau)$
20 JAN 1895	1.00000
JAN 1895	0.86394
8 JAN 1895	0.78663
6 FEB 1895	0.71177
FEB 1895	0.60782
26 FEB 1895	0.47711
1 MAR 1895	0.44933
28 NOV 1894	0.34646
MAR 1895	0.33789
24 NOV 1894	0.31982
1895	0.26818
28 MAR 1895	0.26185
6 APR 1895	0.21871
3 NOV 1894	0.21014
OCT 1894	0.14882
4 MAY 1895	0.12493
1894	0.09044
29 MAY 1895	0.07577
13 SEP 1894	0.07577

$$Q(\tau) = \begin{cases} \mathrm{L}[p(-t)] & \text{for } y_1 < x_2 \\ 0 & \text{otherwise} \end{cases}$$

For date (y_1, y_2) to be after date (x_1, x_2), set $x_2 = x_1$ and consider only positive separations:

$$Q(\tau) = \begin{cases} \mathrm{L}[p(t)] & \text{for } y_2 > x_1 \\ 0 & \text{otherwise} \end{cases}$$

The value of τ can be chosen to reflect how close you expect the actual date to be to the specified "before" or "after" date. For example, τ could be chosen small when you know an expected birth date is close to a known baptism date or could be chosen large if it is possible a person died several years after their last

appearance in a census record. Finally, a computer algorithm for finding $Q(\tau)$ including these refinements (and some more) is given in the appendix.

To see date matching in action, I searched for the date 20 JAN 1895 in my own family history file containing 18,383 typical genealogy dates using $\tau = 50$ days. Table 1 shows the top results and their quality score. The highest ranking match, with $Q(\tau) = 1$, is a match to the same exact date; all other matches have lower quality. The effect of date precision shows up in how date ranges rank among the exact dates. From example FEB 1895 matches better than 26 FEB 1895, but not as well as 6 FEB 1895. Two bigger effects have 1895 ranking higher than 28 MAR 1985 and 1894 ranking higher than 29 MAY 1895. The year dates match better even though their time difference (based on year midpoints) are farther from 20 JAN 1895 than the lower ranking exact dates. This result reflects an intuition that dates with the correct year (*e.g.*, 1895) are more consistent with 20 JAN 1895 than exact dates separated by a couple of months (*e.g.*, 28 MAR 1895).

This date quality score is just one proposal for developing genealogy searching and merging methods. It is not currently used by any genealogy systems.[10] The algorithm is more computer intensive then one based on date separation by their midpoints, but it has the advantage of accounting for date precision, which a simple separation method does not. The method could be developed further by gaining experience with large collections of genealogy dates. A key question is how to determine the optimal τ for application to specific genealogy searching and merging problems?

Place Matching

Another key datum in genealogy is a place name that documents where a birth, marriage, or death (*etc*.) occurred. Place names can be compared much like names, which means to break them into parts, do string comparisons, watch out for spelling errors, use Soundex or other codes, and be aware of transcription errors. Like first, middle, and last names as name parts, place names have hierarchical parts such as town, county, state and country in the United States and many other organizations in other countries. Genealogy records may vary considerably in the hierarchical detail they provide. For example, the birth place "Ridgewood, Bergen, New Jersey, USA" (for town, county, state, country) may appear in records as "Ridgewood", "Ridgewood, New Jersey", or just "New Jersey" rather than the full place name. Furthermore, place names can change. These changes can include moving between countries, political reorganization of jurisdictional

[10] The calculations here were done by writing a script for my software called GEDitCOM II (see http://www.geditcom.com). This application is also exploring this date method as an editing tool to find and merge duplicate genealogy records.

areas such as counties, and town name changes. My ancestor Rebecca Nurse,[11] who was executed as a witch, lived in Salem Village, Massachusetts Bay Colony, Great Britain. Her home in that "place" is preserved as a historical residence,[12] but it is now located in Danvers, Essex, Massachusetts, USA. No computer string matching could possibly conclude these are the same place. When place names do not include their full and current hierarchy, it complicates matching and merging based on their names.

Unlike people's names, place names refer to a physical location on earth. An alternative for places therefore is to supplement their names with their latitude and longitude location. If all places in a genealogy database had latitude and longitude, it would be easy to calculate the distance between any two places and accurately determine their similarity and likelihood that the two places are referring to the same location. For example, Salem Village, Massachusetts Bay Colony, Great Britain and Danvers, Essex, Massachusetts, USA, have the same latitude and longitude. Larger places, such as states or countries, should be documented with a bounding polygon of latitudes and longitudes and then any place contained within that region might be a matching place.

The need to add latitude and longitude to all recorded genealogy place names suggests the need for a genealogy tool to look up places and return their latitude and longitude. This problem is partially solved by web-based mapping tools such as Google maps.[13] You can type almost any name variation into Google maps and it will provide a map — finding a map implies it found the latitude and longitude for that place. Google maps and other web tools, have some limitations for robust genealogy work. First, many names are ambiguous. When you type "Ridgewood" into Google maps, it suggests several specific names, but if you ignore them, it returns a map for Ridgewood, New Jersey. This result is misleading because it ignores name ambiguity in favor of providing some map. A better tool would list all possible "Ridgewood"s and let you find the correct one. One genealogy-based database for places finds 67 towns in the United States containing the text "Ridgewood"[14] and any one of these might be the one you need. Another problem with mapping tools is that they are always based on today's place names. They are little help in locating historical place names that are common in genealogy. For example, searching for "Salem Village, Massachusetts Bay Colony" in

[11] Rebecca Nurse was executed as a witch in 19 JUL 1692. She was survived by eight children and later by thousands of descendants (see http://en.wikipedia.org/wiki/Rebecca_Nurse)

[12] The Rebecca Nurse Homestead: http://www.rebecca-nurse.org/

[13] http://maps.google.com

[14] This place database is available at http://www.geditcom.com/Atlas.html, although the option to search the data requires access of that web site through GEDitCOM II (http://www.geditcom.com).

Google maps suggest many businesses containing the text "Bay Colony" in "Massachusetts" and does not even mention the current place, "Danvers, Massachusetts," at all. Genealogy research would benefit from new latitude and longitude tools similar to Google maps, but with enhancements to identify name ambiguities and new capabilities to find historical place names.

Family Networking

Perhaps the most important information for identifying records for merging is family links. The purpose of genealogy is to find ancestors and descendants, which means the goal of genealogy records is to link them to their parents, to spouses, and to children with each spouse. These links can be used for merging. If you find common links between two records, that fact can provide good evidence that those two records can be merged.

Conversely, in the absence of matching family links, it can be difficult to ever have certainty about merging records. From my great, great-grandmother Elizabeth Walton's marriage record, I knew she was born in Halifax, Yorkshire, England in 1829 and her father's name was Samuel. When I looked in the 1841 UK census for Halifax I found one (and only one) Elizabeth Walton with the correct birth date and place. Unfortunately, her father's name in that census was John instead of Samuel. I had name, date, and place matching well but the only known family link was wrong. Based on the Sherlock Holmes adage that "*When you have eliminated the impossible, whatever remains, however improbable, must be the truth*,"[15] I merge Elizabeth Walton in my records, along with her parents and several siblings. Perhaps the father on her marriage record was a step father or maybe her father in the 1941 census changed his name or switched to using a middle name? But the family link information turned out to be crucial and I had the wrong Elizabeth Walton. I eventually found a more complete 1841 census[16] and it had a second Elizabeth Walton with father Samuel. I had to remove all the previous Waltons and replace them with the new ones.

The Walton example shows that family links are vital in merging genealogy records with confidence. It also reveals a weakness in Sherlock Holmes's deductive logic. The problem in genealogy is that you can never be certain if you have "eliminated the impossible," because that step assumes you have access to all possible data. In genealogy, the possible records may be incomplete, may be indexed incorrectly, or may no longer exist. Genealogy research must be based on discovery of information and never on its absence. This example also illustrates the value of census records. They provide names, dates, and places, but more importantly, they group individuals to provide family links that are vital to merging.

[15] Arthur Conan Doyle, "Sign of the Four", chapter 6 (1890).
[16] See http://www.findmypast.co.uk

Family links are powerful clues for matching records, but like names, dates, and places, they also have ambiguities. If you just have the name for a father or mother, matching that name to a known father or mother may or may not be exact. Some issues mentioned under names are switching of first and middle names and spelling. Mothers in family records may commonly appear with a married named, which is less useful than having the mother's maiden name as well. When parents are listed later in life (such as on a child's marriage record), it may not be indicated if those parents are natural, step, or adoptive parents. In other words, apparent conflicts in parent names on certain records may not be actual conflicts. Similarly, evidence for a match can be derived from spouse name matching, but obviously mismatched spouse names are not a conflict. It is always possible the person was married more than one time.

IV. Conclusion – Final Quality Matching

The previous sections described matching of records by comparing unique record identifiers, names, dates, places, and family networks. When it works (*i.e.*, in a well-contained project), matching by unique record identifiers is precise. But in most projects and in almost all genealogy research, record matching has to rely on comparison of the inherently fuzzy data. Computer algorithms work best in binary mode where something is either true or false. In genealogy merging, however, comparisons will always be "maybe" — computers do not work as well with "maybe."

A potential scheme is to develop quality scores for matching various data types. The section on date matching advances this topic farthest by calculating a quality score, $Q(\tau)$, for any two dates on a scale from 0 to 1 for mismatch to exact day match (a computer algorithm is given in the appendix). Similar quality functions should be developed for names, places, and family links with 0 indicating a conflict and values up to 1 for increasing match quality. For names, a perfect match would be full name with exact spelling match. Lower quality matches would follow if one name lacked a middle name, if first and middle names were switched, or if the names were spelled differently (although perhaps sounding similar). For places, the ideal quality score would simply be derived from the distance between them. If latitudes and longitudes are not known for the places, matching could be based on the hierarchical parts of place names, analogous to name matching. For family networking, the quality score would depend on the number and certainty of matching family links. If two records have conflicting links, they should be rejected as potential matches, unless a reasonable explanation is available (*e.g.*, step parents or multiple spouses).

If we assign N, $Q(\tau)$, P, and F, to the quality scores for matching names, dates, places, and family links, respectively, the total quality score, S, could be

$$S = w_n N + w_d Q(\tau) + w_p P + w_f F$$

where w_n, w_d, w_p, and w_f, are weighting coefficients applied to the corresponding genealogy data types. But, if any quality score is zero, indicated a mismatch or conflict, the total quality score is set to zero as well. Given weighting factors and algorithms for individual quality scores a computer algorithm could calculate S and decide if two records match. In practice, the value of S required to assure a match will be difficult to determine. A good computer algorithm might merge the records if S is sufficiently high, reject the records as matching if it is zero or too low, and revert to user advice for all values in between. Humans are often much better than computers at comparing fuzzy data and therefore will be needed in genealogy merging processes until computers have more artificial intelligence. If quality scores are applied to genealogy search results, they could be used to sort the results prior to display for the user. Such a sorting might greatly improve your chance of finding your ancestors.

In brief, computer merging of genealogy records will remain an elusive goal, but computer assisted merging and search ranking could be a powerful tool for genealogists, provided that quality scores are developed and verified on real data. The development would be to define the quality rankings - N, $Q(\tau)$, P, and F. The verification would be to adjust weighting factors (w_n, w_d, w_p, and w_f) to give good results when used with actual genealogy data and to determine key values for S to know when to accept or reject a match and when to ask the user for help. Donald Knuth wrote that programming can be an "aesthetic experience much like composing poetry or music."[17] This statement certainly holds for merging in genealogy software and in genealogy research. It will take collaborations between programmers and genealogists to devise optimal tools. The assessment of which quality rankings and weightings to use will literally be based on what "sounds" like the best matches to those experts. Hopefully such collaborations will compose algorithms for computer enhancement of genealogy research.

[17] Donald E. Knuth, "The Art of Computer Programming," vol. 1, preface (Addison-Wesley, Reading, MA, 1973)

APPENDIX

A subroutine for calculating matching quality between any two dates is given below as python code (it could easily be translated to other languages). The $Q(\tau)$ function is the Laplace transform of a piecewise linear function. These transforms are trivial and always given by a sum of exponentials. Dealing with the various pieces of $p(t)$, however, requires numerous special cases depending on the parts of the "hat" function that are present — most of the work in this algorithm is to identify each special case.

The inputs are date1 and date2, which are provided as day ranges (x_1, x_2) and (y_1, y_2). The optional order is -1, 0, or 1 to find date2 being before, near, or after date1, respectively. The optional tau and cutoff can change the τ and Q_{min} settings for the calculation. The first step is to convert date1 to an exact date if order is -1 or 1, or to switch date1 and date2 if order is 0 and the range of date1 is wider than the range of date2. Next, the signs of the dates are changed, if needed, to make sure the first date ends before the second. The switching of order or signs is simply to reduce the number of special cases needed in the algorithm. The code commented with "Refinement" is a modification not discussed here to improve calculations when the first date is entirely within the second date. Finally, for each possible case, the date quality is calculated in qual. If qual is less than cutoff, the function returns zero to indicate the dates do not match, otherwise it returns the calculated qual.

```
def DateQuality(date1,date2,order=0,tau=50.,cutoff=4.563e-7):
    # if order!=0, make date1 exact, otherwise switch dates if dx>dy
    y = [date2[0],date2[1]]
    dy = y[1] - y[0]
    if order == 1 :
        if y[1] <  date1[0] : return 0.
        x = [date1[0], date1[0]]
        dx = 0
    elif order == -1 :
        if y[0] > date1[1] : return 0.
        x = [date1[1], date1[1]]
        dx = 0
    else :
        x = [date1[0],date1[1]]
        dx = x[1] - x[0]
        if dx > dy :
            x = [date2[0],date2[1]]
            y = [date1[0],date1[1]]
            dy = dx
            dx = x[1]-x[0]
        # Refinement: if x within y, move it to midpoint
        if x[0]>=y[0] and x[1]<=y[1] :
```

```
        x[0] = y[0] + 0.5*(dy-dx)
        x[1] = x[0] + dx
# switch signs if x<y (saves number of needed cases below)
if y[1] < x[1] :
    x = [-x[1],-x[0]]
    y = [-y[1],-y[0]]
    order = -order
# Find L[(p(t)]+L[p(-t)] for each possible case
if y[0] >= x[1] :
    if x[1] > x[0] :
        e1 = -(y[0]-x[0])/tau
        e2 = -(y[0]-x[1])/tau
        e3 = -(y[1]-x[0])/tau
        e4 = -(y[1]-x[1])/tau
        qual = -math.exp(e1)+math.exp(e2)+math.exp(e3)-math.exp(e4)
        qual *= tau*tau/(dx*dy)
    else :
        if y[1] > y[0] :
            if order==-1 :
                # Refinement: if y[0]==x[1] shift y -1 day for non-zero qual
                qual = tau*(1.-math.exp(-1./tau))/dy
            else :
                e1 = -(y[0]-x[0])/tau
                e2 = -(y[1]-x[1])/tau
                qual = tau*(math.exp(e1)-math.exp(e2))/dy
        else :
            qual = math.exp(-(y[1]-x[0])/tau)
elif y[0] >= x[0] :
    e1 = -(x[1]-y[0])/tau
    e2 = -(y[0]-x[0])/tau
    e3 = -(y[1]-x[0])/tau
    e4 = -(y[1]-x[1])/tau
    qual = math.exp(e1)-math.exp(e2)+math.exp(e3)-math.exp(e4)
    qual = tau*(2*(x[1]-y[0])+tau*qual)/(dx*dy)
else :
    if x[1] > x[0] :
        e1 = -(x[0]-y[0])/tau
        e2 = -(x[1]-y[0])/tau
        e3 = -(y[1]-x[0])/tau
        e4 = -(y[1]-x[1])/tau
        qual = -math.exp(e1)+math.exp(e2)+math.exp(e3)-math.exp(e4)
        qual = tau*(2. + tau*qual/dx)/dy
    else :
        if order==1 :
            qual = tau*(1.-math.exp(-(y[1]-x[1])/tau))/dy
        else :
            if order==0 or y[1]<=x[0] :
```

```
                e1 = -(x[0]-y[0])/tau
                e2 = -(y[1]-x[1])/tau
                qual = tau*(2.-math.exp(e1)-math.exp(e2))/dy
            else :
                qual = tau*(1.-math.exp(-(x[0]-y[0])/tau))/dy
if qual < cutoff : qual = 0.
return qual
```

BIOGRAPHY

John A. Nairn is professor and Richardson Chair of Wood Science & Engineering at Oregon State University, Corvallis, Oregon, USA. He has been writing genealogy software for the Macintosh since 1994 and is the author of GEDitCOM and now GEDitCOM II. He also wrote the "Date Calculator" (in Mac App store) for calculations with fuzzy genealogy dates.