Interpreting DNA Matches

Michael Markowski
`mike.ab3ap@gmail.com`

May 2021

1 Introduction

DNA testing makes it possible for those interested to easily and inexpensively take a DNA test. Due to privacy concerns as well as business model, however, only partial data is made available to customers. As a result, it is difficult to piece the raw data together to build up information. Making sense of the data is a challenge.

This paper is an initial effort at drawing deeper information from raw data provided by DNA testing. Two methods are presented to model the data and how it can then be interconnected. Steps to reduce complexity of interconnection are also presented. Both methods yield a connectivity matrix, though each is reduced differently. They are then fed as input to the widely used, open source graph drawing program, Graphviz.

2 Ancestry.com Data

As an Ancestry.com subscriber, the method presented stems from my desire to make sense of the somewhat disconnected raw data from Ancestry into easier to understand information. Ancestry provides two basic categories of information:

1. Direct DNA matches. A list of people who share DNA segments with you.
2. Common DNA matches. For every Direct Match, a list of people who match both you and that direct match.

As mentioned, the challenge is that this is partial data. By limiting data access and not connecting customer trees in the fashion of WikiTree.org, customers must maintain subscriptions to continue their work. While some work will be identical performed in parallel with other customers, i.e., relatives, doing the same, this business model increases Ancestry revenue. To remain competitive, Ancestry uses some of the income for further document acquisition and digitization. So, there is a trade-off of pros and cons, both helping and hindering genealogical research.

A more practical challenge from the software developer's point of view is that there is no remote access to Ancestry's database. All information must be acquired through their web site. Web sites, by their nature, are designed for people as opposed to programmatic interaction. With these limitations in mind, the next steps are an approach to model the data and then present it in a useful manner.

3 Method 1: Complete Connectivity

This method is a useful way to view all direct matches and common matches between them and the primary person. Because it shows all data, it can also be a little overwhelming to make use of. This method in conjunction with the one presented later can make powerful allies, however, in determining where a DNA match is likely to reside in the family tree.

3.1 Manual Method

The technique is easily performed by hand, but becomes challenging with more people. The steps are simply:

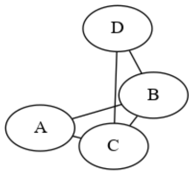
1. Write down direct matches of interest around the edges of a large sheet of paper.
2. Draw lines between all common matches.
3. Patience and perseverance!

Software does nothing more than the above, but tracks more details and of course is faster. The graphing software also has heuristics to group similar nodes together, making it easier to pick out different lines in a family tree.

3.2 Data Visualization Technique

The software version of the technique is also straightforward, and revolves around use of an adjacency matrix from graph theory. A square matrix is constructed such that rows and columns are named with each direct DNA match. Each element of the matrix is a zero if those two people do not share a common match, and a one if they do.

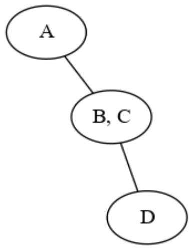
An example describes it better. Suppose we have a graph of four people whose DNA connects them like this:



The adjacency matrix looks like this:

	A	B	C	D
A	1	1	1	0
B	1	1	1	1
C	1	1	1	1
D	0	1	1	1

Notice that along the diagonal there is redundant information indicating that A is related to A, B to B, and so on. This seemingly redundant data turns out to be helpful when reducing graph complexity. Because rows B and C are identical—and they wouldn't be, if the diagonal elements weren't 1—they can be collapsed into one node:



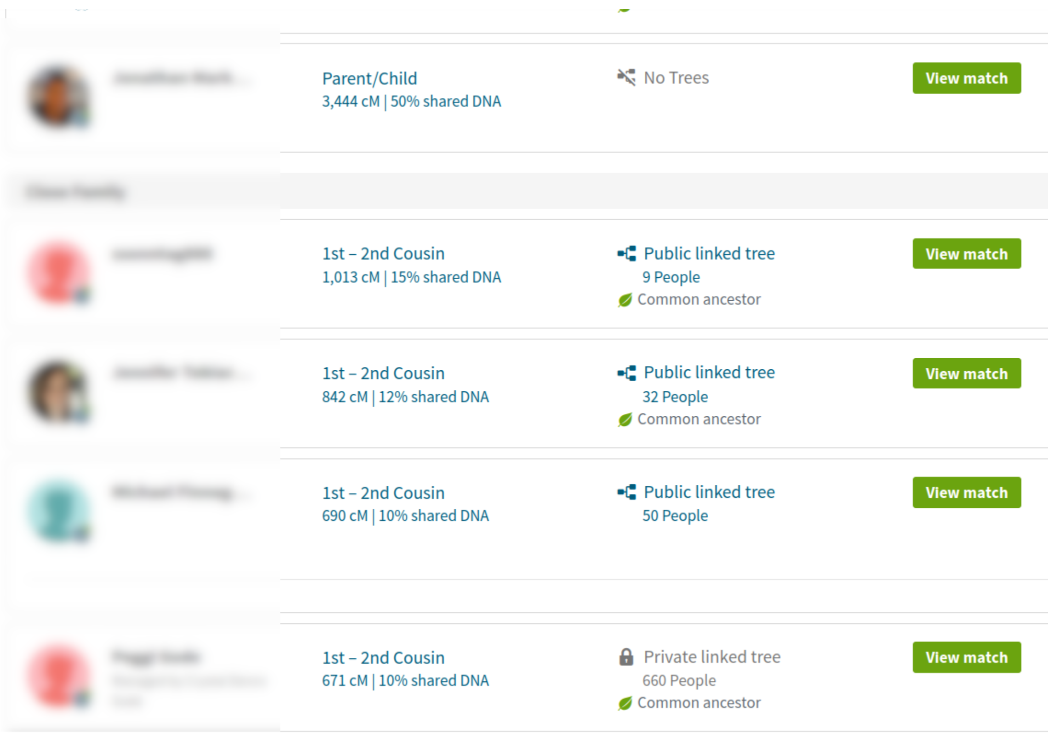
It contains the same information as the first graph, is more concise, and easier to understand. The software does nothing more than implement the steps in this example, but on a larger scale. It also tackles the problems of converting an Ancestry web page into more regularly formatted input to more easily construct the adjacency matrix.

3.3 Software Design

3.3.1 Modeling People

There are two important items in a DNA match: name and amount of shared cM's (centiMorgans). As a result, modeling a DNA match is simple. The larger challenge is modeling the relationship between people sharing DNA. It is important to note that the goal is not to build a family tree, but to create a graph showing who shares DNA segments with who. The graph is similar to a family tree, but not exactly that. By using cM levels, this technique can likely be enhanced to become a bit more like a tree. But for this first effort, simple results are the goal.

Ancestry DNA match lists are familiar to customers. Matches in 2021 look like the following.



Parent/Child
3,444 cM | 50% shared DNA
No Trees
View match

1st – 2nd Cousin
1,013 cM | 15% shared DNA
Public linked tree
9 People
Common ancestor
View match

1st – 2nd Cousin
842 cM | 12% shared DNA
Public linked tree
32 People
Common ancestor
View match

1st – 2nd Cousin
690 cM | 10% shared DNA
Public linked tree
50 People
View match

1st – 2nd Cousin
671 cM | 10% shared DNA
Private linked tree
660 People
Common ancestor
View match

Because direct database access is not available, two approaches are possible. One, is to use programmatic retrieval of web pages, parse them, and reformat name/cM data as desired. That is unquestionably the best way because it involves no human interaction. However, it works only for Ancestry pages as presented on the web today and is work intensive for the programmer. Because this is a spare time effort, I opted for the second and lazier approach. Each page of Common Matches is copy/pasted into a plain text file. Then, a program converts the data into a list like the following, where names in this paper are partially obscured with Xs:

```
XXXMurphy
XXXMarkowski|3445
XXXMarkowski|3444
XXXntag888|1013
XXXBrosnan|359
XXXFraser|210
XXXieleenj|181
XXXNolan|172
XXXBrosnan|95
XXXFitzgerald|75
XXXconway|68
XXXBrosnan|68
XXXFreeman|65
--B.|65
--C.|56
XXXAdair|55
...
```

An advantage to this is that DNA data from any source can be used as long as it is eventually presented in the form above. The program expects files in that format as input, and each file name must exactly match the name as entered. That is, suppose the file above is for direct matches with fictional customer Joe Smith. If a file exists with common matches between Joe Smith and XXXMurphy, the first name in the list, then that file of common matches must be named XXXMurphy, in this example. If such a file does not exist, XXXMurphy is ignored because he is such a distant match that he has no common matches between the Joe Smith and other matches. It isn't that he is uninteresting overall, but that this match can provide no DNA interconnection data and is unhelpful here.

3.3.2 Modeling Interconnections of People

Once all the people are read in with name and cM level stored, interconnections are modeled. This is performed just as in the small example presented earlier. Looking at intermediate output as the program runs on a small example,

```
Everyone
 0: XXXN|396
 1: XXXTobiansky|842
 2: XXXGode|361
 3: XXXFinnegan|690
 4: XXXC.|618
 5: XXXGode|671
 6: XXXBrosnan|359
 7: XXXntag888|1013
```

At this point the program has read in direct matches, then read in common matches for each direct one. In the following steps, names are replaced with numbers above for easier construction of the adjacency matrix. Next, the complete matrix is constructed. For readability, zeros are replaced with dashes.

```
Matrix Raw
 0: ['00' 111111--]
 1: ['01' 111111--]
 2: ['02' 111-11--]
 3: ['03' 11-1----]
 4: ['04' 111-11--]
 5: ['05' 111-11--]
 6: ['06' ------11]
 7: ['07' ------11]
```

The matrix is then sorted in preparation for combining identical rows, i.e., people with identical common matches, into a single row. Again, this is why it is important to set all diagonal elements to 1 in the original matrix, above.

```
Matrix Sorted
 0: ['06' ------11]
 1: ['07' ------11]
 2: ['03' 11-1----]
 3: ['02' 111-11--]
 4: ['04' 111-11--]
 5: ['05' 111-11--]
 6: ['00' 111111--]
 7: ['01' 111111--]
```

Finally, the reduced matrix is created, the final output of the program, where duplicate rows and columns have been combined.

```
Matrix Reduced
 0: ['00,01' -11-]
 1: ['02,04,05' 1---]
 2: ['03' 1---]
 3: ['06,07' ----]
```

In reality, the program does not output what is shown above. It outputs the connectivity matrix in the *dot* language, a simple formal language that the open source graph visualization package, Graphviz, uses. Currently, it uses Graphviz's `circo` program, but no problems are posed if more elaborate *dot* output is generated for any of the Graphviz programs.

The examples above illustrate a small number of relatives, from 350 to 3000 cM. A more typical example, below, ranges from 40 to 3000 cM, making apparent the appeal of software.

```
Matrix Reduced
 0: ['00' -----------------11--------1--------1----1----------1--1-1--1-------1-----]
 1: ['01' -----------------111---1---1-1-------------------1------1-----------------]
 2: ['02' ---1---1-1----1-1--------------1--11-------1--1------------------11----1--]
 3: ['03' --1--1----1-----1----------------------1----------------------1-11-----1--]
 4: ['04' -----1--1----1--------1--1------------1-1----1--1-----1--------11------1--]
 5: ['05' ---11---1-1--1--------1--1------------1-1---------1--11--------11------1--]
 6: ['06' -------------------1---11-1---1--1---1----1-1--1-------------1-----1----1-]
 7: ['07' --1-----------1-------------1--1--1-----------1------------------11----1--]
 8: ['08' ----11----1--1--------1------------1----1----1--1-1--11--------1-------1--]
 9: ['09' --1-------------------------1--1-------1------1-----------11--1--------1--]
10: ['10' ---1-1--1-------------1--1--------------1----1--1-1--11--------11------1--]
11: ['11' -----------------11----1------------1------------1------11----------1-----]
12: ['12' ---------------------1--1-1------1---1----1-1--1-------------------1--1---]
13: ['13' ----11--1-------------1--1------------111----11-1-1--11--------111-----1--]
14: ['14' --1----1--------1---1-------1--1--1----1------1------------------11----1--]
15: ['15' ----------------1---1--------------------------------------1-----1-----1--]
16: ['16' --11----------11----1------------------1------1------------1--1--1-----1--]
17: ['17' 11---------1------11---1-----1--1---1----1-------1--1--111--1-------1-----]
18: ['18,38' 11---------1-----1-1---1---1-1--1---1----1-------1-11--111--1-------1-----]
19: ['19' -1----1----------11----11-11-11-11--11----1-1-------1--111--11------1-----]
20: ['20' --------------111-----------------------------1------------1--1--1---1-1--]
21: ['21' ------------1----------11-1----------1------1--1--------------------------]
22: ['22' ----11--1-1--1-----------1------------1-1----1--1-1--11--------11------1--]
23: ['23' -1----1----1-----111-1--1-11-11-11--11---11-1--1-1--1---11---1--------1-1-]
24: ['24' ------1-----1------1-1-1--1---1--1---1----1-1--1-------------1--------1-1-]
25: ['25' ----11----1--1--------1---------------1-1------------11--------11------1--]
26: ['26' ------1-----1------1-1-11-----1--1--11----1-1--1-------------1--------1-1-]
27: ['27' 11----------------11---1-----1------1----1-------1--1--1-1--1-------1-----]
28: ['28' -------1-1----1----------------1--1-------------------1---11------1----1--]
29: ['29' -1---------------111---1---1----1---1------------1-1---11---1-------1-----]
30: ['30' ------1------------1---11-1------1---1----1-1----------------1-----1------]
31: ['31' --1----1-1----1-------------1-----1--------1--1------------------11----1--]
32: ['32' -----------------111---1-----1------1------------1------11----------1-----]
33: ['33' ------1-----1------1---11-1---1------1----1-1--1-------------1-----1--1---]
34: ['34' --1----1------1-------------1--1--------------1------------------11----1-1]
35: ['35' --1-----1-------------------------------------1---1---1---11---111---1----]
36: ['36' 1----------1-----111---1--11-1--1----------------1--1--111--1-------1-----]
37: ['37' ------1-----1------1-1-11-1---1--1--------1-1--1-------------1-----1--1---]
38: ['39' ----11-------1--------1--1-------------11-------------1--------11------1--]
39: ['40' ---1-----1---11-1---------------------1------1------------------11--1-----]
40: ['41' ----11--1-1--1--------1--1------------1------1--1----1---------11------1--]
41: ['42' 1----------------11----1---1---------------------1-----111--1-------1-----]
42: ['43' ------1-----1------1---11-1---1--1---1------1--1-------------1--------1-1-]
43: ['44' --1----------------------------1--------------1-----------1------11----1--]
44: ['45,48' ------1-----1------1-1-11-1---1--1---1----1----1-------------1-----1--1-1-]
45: ['46' ----1---1-1--1--------1----------------11-------1-----1--------11------1--]
46: ['47' --1----1-1---11-1---1----------1--11-------1---------------------11----1--]
47: ['49,72' ------1-----1--------1-11-1------1---1----1-1----------------1-----1--1-1-]
48: ['50' ----1---1-1--1--------1-----------------1----1--------1--------11------1--]
49: ['51' -1---------1-----11----1---1-1--1---1----1--------------11----------1-----]
50: ['52' -----1--1-1--1--------1------------1-----------------11--------1-------1--]
51: ['53' ------------------1----------1--------------------------1-----------1-----]
52: ['54' 1----------------111---1---1--------1------------------1-1----------1-----]
53: ['55' -----1--1-1--1--------1--1--------------1---------1---1--------1-------1--]
54: ['56' ----11--1-1--1--------1--1--1------1--1------1--1-1--1-----1---11------1--]
55: ['57' 1----------------111-------1-1------1----1----------1---11--1-------1-----]
56: ['58' -1---------1-----111---1-----1--1---1----1-------1-1---1-1--1-------1-----]
57: ['59' 1----------1-----111---1---1----1---1----1-------1--1--11---1-------1-----]
58: ['60' ---------1------------------1------1-------1---------------1-----------1--]
59: ['61' ---------1-----11---1-------1------1------------------1---1------1-----1--]
60: ['62' 1----------------111-------1-1------1----1-------------111----------1-----]
61: ['63' ------1------------1---11-1---1--1---1----1-1--1----------------------1---]
62: ['64' ---1-----1------1---1-------------------------------------------11--------]
63: ['65' ----11--1-1--1--------1--1---------1--1-1----1--1-1--11-------------------]
64: ['66' ---111----1--1--------1--1---------1--1-1----1--1-----1-------1--------1--]
65: ['67' --11---1-----1111---1----------1--11---1---1--1------------1--1---1--1-1--]
66: ['68' --1----1------1-------------1--1--1----1---1--1------------------1-----1--]
67: ['69' ------1-----1-----------------1--1---1------1--1--------------------------]
68: ['70' 1----------1-----111-------1-1--1---1----1-------1-11--111--1-------------]
69: ['71' --------------------1--------------1---1-------------------------1-----1--]
70: ['73' ------------1----------11-1------1---1----1-1--1-------------1----------1-]
71: ['74' --1111-1111--1111---1-1--1--1--1--1---1-1--1-11-1-1--11---11----111--1---1]
72: ['75' ------1----------------11-1---------------1-1--1----------------------1---]
73: ['76' ----------------------------------1------------------------------------1--]
```

3.4 Running the Software

This is hobbyist software without time or resources to knock off the rough edges! Please feel free to contribute if you are a programmer and are so inclined. Install python and Graphviz if not already on your computer. Then, to use the software:

1. Move into your work directory and then:
   ```
   unzip dna.zip
   mkdir matches; cd matches
   ```

2. Copy/paste direct DNA matches into a file. I pasted mine into a file named mike_markowski.raw. The pasted file must have extension .raw. You might have to scroll down multiple times to get all matches.

3. For every name in your direct DNA matches, bring up that person's page in your web browser, and paste the page into hisName.raw, on and on... If a name is M.M., then the file name is M.M..raw. As with direct matches, the pasted file must have extension .raw.

   That is by far the most work intensive part. With all these files in a directory, the easy part is to generate the graphs.

4. `cd ..` where the dna directory (folder) must reside.

5. At the command line, type `python dna/dna -c -d -f matches mainPerson`, where *mainPerson* is your Ancestry username and *matches* points to the directory where your match files reside that you copy/pased. For me, I'd type `python dna/dna -c -d -f ancestry mike_markowski`.

Finally, display the generated output `dnaCirco.png` and `dnaDigraph.png` can be displayed with your favorite image viewing program. The `-c` argument creates a fully interconnected, circular graph that can take many minutes with a hundred or so copy/pasted files. The `-d` option creates a directed graph, which is usually more helpful and is generated quickly.

An optional `-t name` argument will highlight a target person and all matches in common.

Note: the software is preliminary and has shortcomings. Ancestry does not require that displayed names are unique. I discovered I have DNA matches with two men who chose the identical display name. At the moment, the only way around this is to manually rename one of them and use that same new name when referenced by other matches. No doubt there are other shortcomings yet to be discovered.

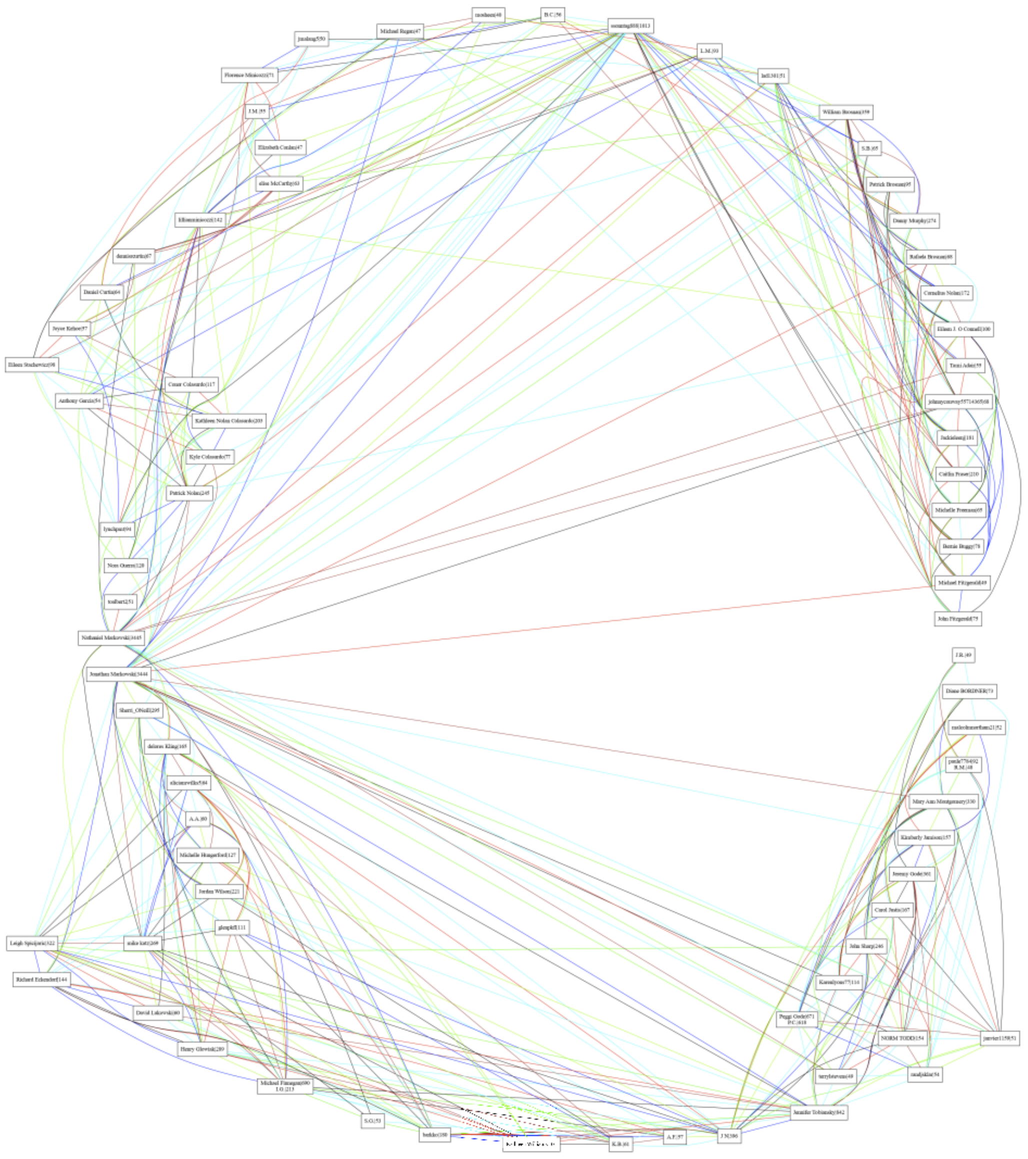
3.5 Results

The following graphs are purposely not legible, in case Ancestry customers who match me don't want their data published. That is ok, however, because it is only the shapes of the graphs that matter for this discussion.

The program, `dna`, accepts command line arguments so that you can specify low and high thresholds of shared cM values. For example, suppose I use a lower threshold of 40 cM, meaning I only want to see extended family, the command is

```
python dna -c -l 40 mike_markowski
```

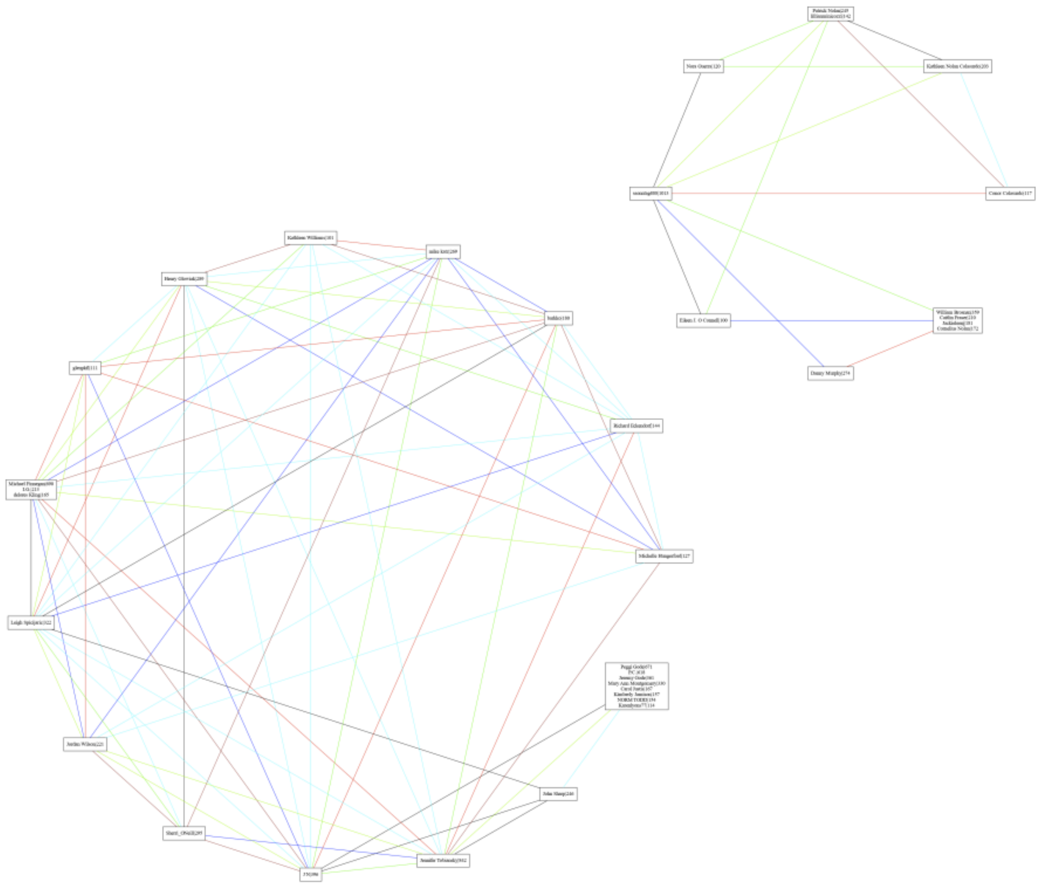
which generates a large graph of my extended family out to 40 cM shared DNA.



Suppose, however, that I filter out my sons and their many links similar to mine. Children share about 3500 cM's with parents, so I cap the cM's to 3000. Similarly, instead of going out to 40 cM, I only go out to 100 cM for this graph to keep it small:

```
python dna -c -l 100 -h 3000 mike_markowski
```

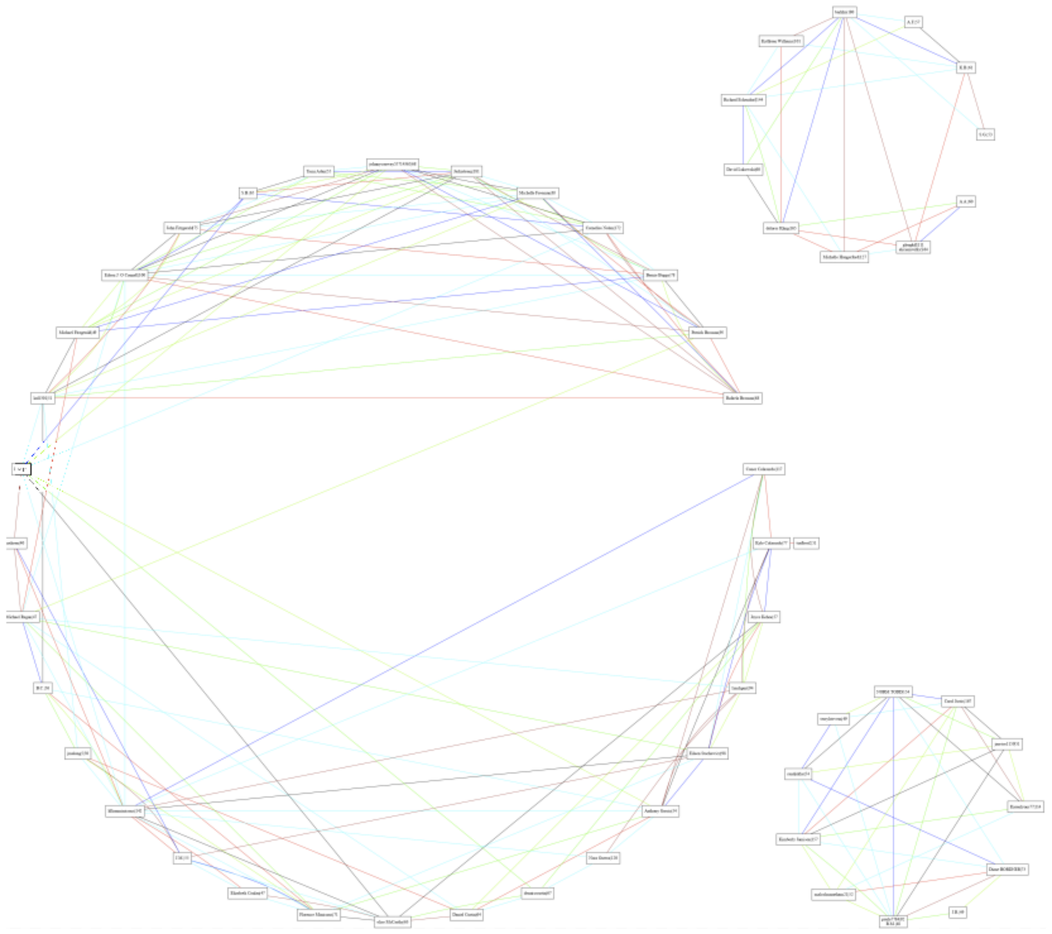
Now, my maternal and paternal trees are clearly delineated.



I can continue by narrowing the focus to my grandparents, filtering out first cousins:

```
python dna -c -l 37 -h 200 mike_markowski
```

Notice that there are three groups rather than the four expected, one for each grandparent.

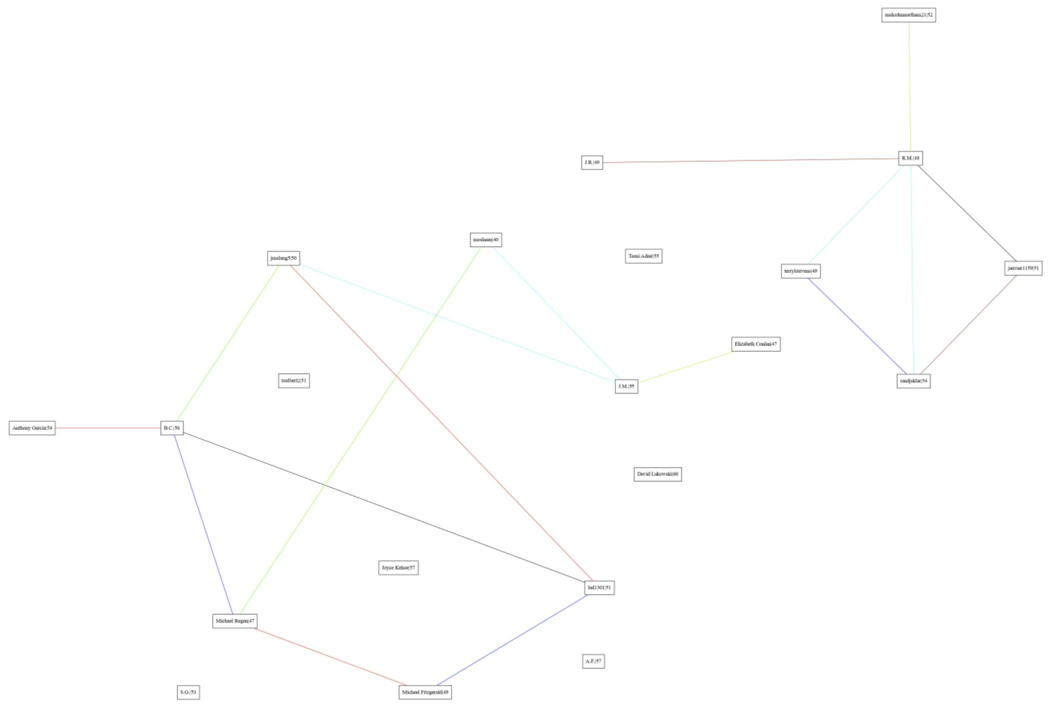


My Irish grandparents are from a small Irish farming village that even today has a population of only 700 people. Over centuries, people found spouses within walking distance. You can see on the left that the tree is almost, but not quite, split into two due to overlapping family trees.

Trying to split trees further to my great-grandparents becomes much more uncertain due to the relatively small number of people who have taken DNA tests.

```
python dna -c -l 37 -h 60 mike_markowski
```

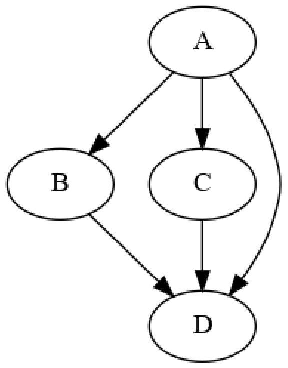
There are eight subtrees as expected.



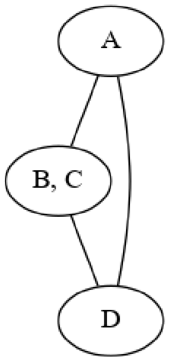
While we have eight subtrees, some have one member. Only research will show if this is correct or, more likely, simply a side effect of not having enough DNA data. When there is not enough data like in this example, it is time to resort to documents.

4 Method 2: Directed Graph

This method tries to eliminate redundant paths, making it easier to study interconnections. The graph produced is much smaller than those created by the exhaustive Method 1. As before, an example is the easiest way to describe it. Suppose we have the following graph of DNA matches:

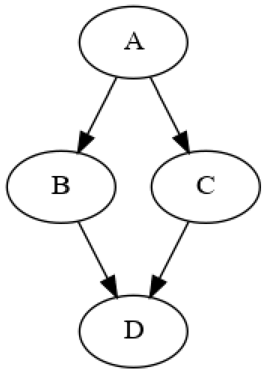


Using Method 1, the graph generated is:



For such a small graph this is ok. But imagine on a graph with a 100 nodes or more, links like the unnecessary A-D link significantly clutter the diagram.

Method 2's directed graph is reduced to this:



There is still a path from A to D, but is not explicitly shown as a direct link. In general, this is not a problem because we are looking for DNA paths. But when the detail is required, Method 1 is the preferred approach. Method 2 is clearly the preferred method when trying to view how the extended family is interconnected. It is most definitely not a family tree, but the DNA connections can give some insight into the tree underlying this data. The graphs are oriented from most shared DNA at top to least at bottom. For this method, there is no collapsing of people with similar matches into one node. This has to do with algorithm details of matrix simplification.

4.1 Algorithm

As before, an adjacency matrix is used to capture connections between common matches. The technique up to the point of creating the initial matrix in Method 1 is identical. The difference is how the matrix is simplified. In this case, paths are compared between every two nodes. In the illustration above where a link is shown between nodes A and D, the algorithm notes that the path A-B,C-D is length 2 and the path A-D is length 1. The longest path is always used because this eliminates the long, sweeping links running from top to bottom. Because this results in a sparse matrix, the method is very fast compared to Method 1.

4.2 Results

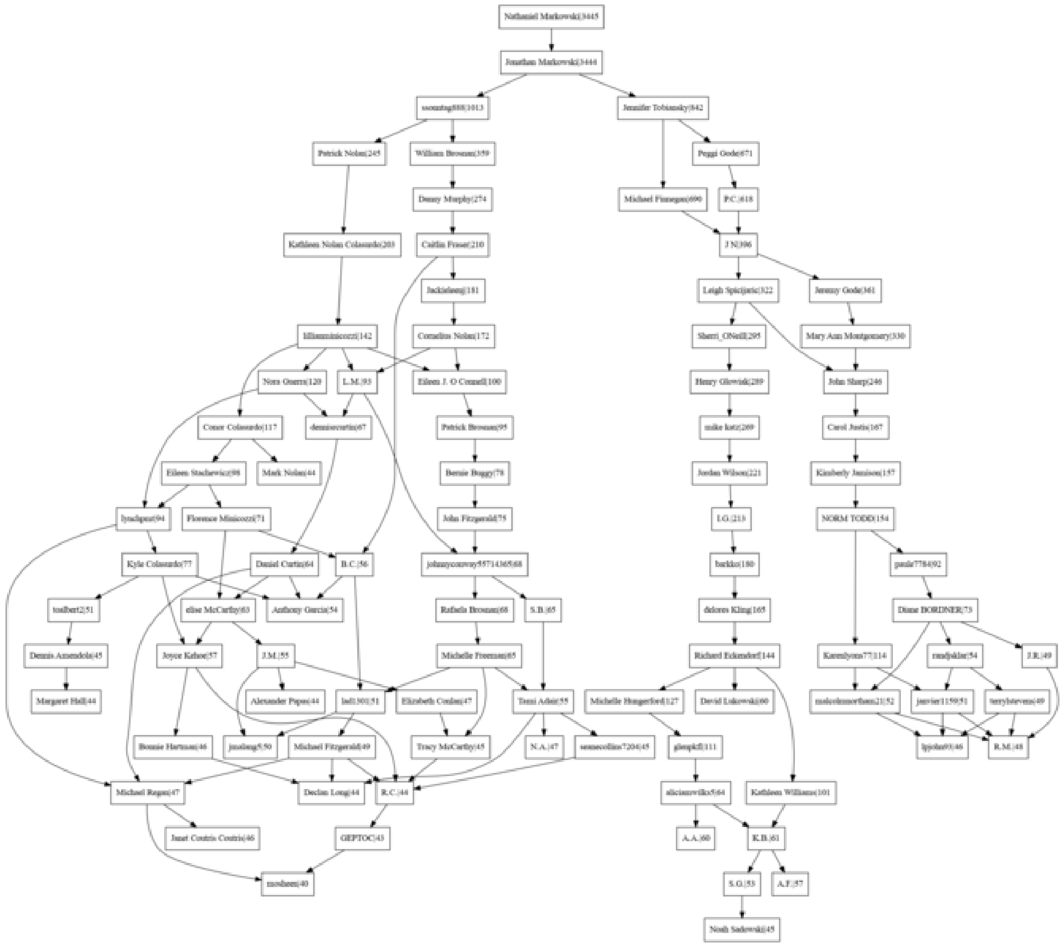
The software is run identically to Method 1 and is selected with `dna` program's `-d`, for digraph, argument. Again, everything about the program initially is, in fact, exactly what Method 1 does to create the adjacency matrix. It is only the matrix simplification and the method that Graphviz plotting that changes.

After running

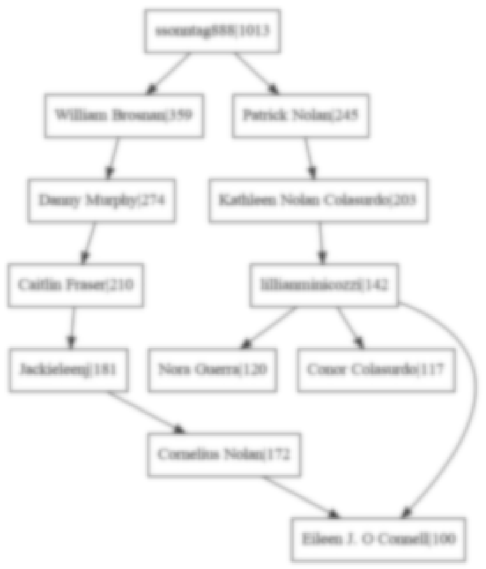
```
python dna -d -l 40 mike_markowski
```

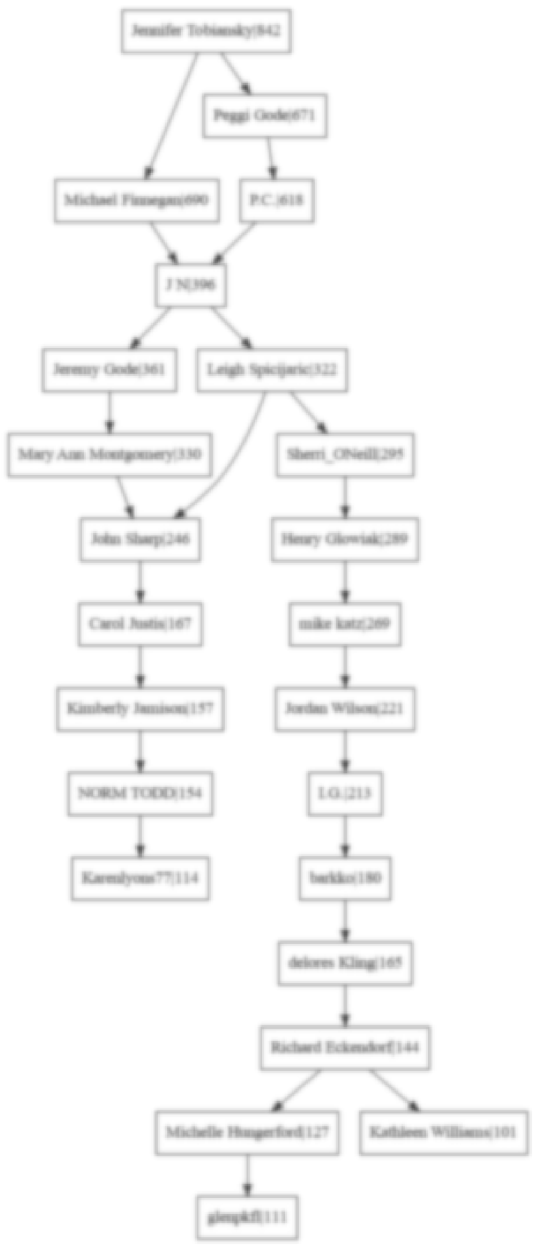
the output, again purposely blurred, shows my two sons at the very top followed by a split with paternal to left, maternal to right. Because the maternal grandparent lines represent two continents, the subtrees are neat and orderly without crossover.

My paternal side, however, represents the small Irish farming village described earlier. The farther out (less cM in common) that the graph goes, the greater crossover and tangling is seen between my grandparents' lines.

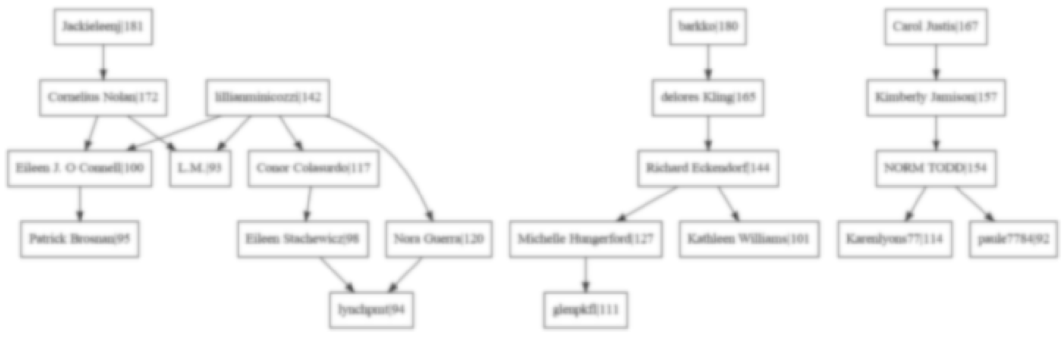


Just as the software for Method 1, the lower and higher thresholds for cM levels to consider can be adjusted. As before using thresholds of 100 cM to 3000 cM yields a graph with maternal and paternal lines cleanly split.





Threshold range of 80 cM–200 cM once results in nearly, but not quite, four graphs representing grandparent lines.



5 Utility

The goal is for the programs to useful tools, helpful in creating a more complete family tree. They present the same information as Ancestry web pages but in different ways. An advantage is that data from multiple web pages is presented in one or two pictures, coalescing more information in one place. A major use is to help locate a general area in the family where a new match might exist. Often, a DNA match tells you that you are related to someone, but the path is a mystery. While a graph can't necessarily provide all the answers, with any luck it will at least narrow down areas of search.

6 Summary

For business concerns and, to a lesser extent, privacy concerns, Ancestry limits access to its data. The simple device of an adjacency matrix allows customers to more efficiently visualize the partial DNA data provided by Ancestry. This effort can be extended, providing better graphing and representation of data. The work described here makes no effort to create a family tree because having only DNA data results in a tree with uncertainty. The uncertainty could be represented by "clouds" of unknown detail that connect known segments of the tree. Ideally, the result would be a fully correct family tree, with unknown areas represented by clouds similar to computer network diagrams.