Implementation of the Leeds Method in R

James Sims PhD

Self-published online: Mon Sep 24 13:43:43 2018
Updated: Tue Sep 25 12:02:27 2018

Abstract

This article describes a reproducible implementation of the genetic genealogy technique known as the Leeds Method in the R statistical programming environment.

Introduction

One of the most challenging tasks genetic genealogists face is properly categorizing autosomal DNA matches as being due to DNA inherited from specific ancestors. The Leeds Method[1], developed by Dana Stewart Leeds, is a method to summarize predicted second cousin (2C) and third cousin (3C) matches at AncestryDNA. The method produces a table that seeks to answer the question, are my DNA cousins also DNA cousins? The output of the method, as originally developed, is a table that has eight or so different groups of matches representing matches that are related to the test-taker through each of the test-taker's eight great-grandparents. Whether the groups created by the grouping procedure are due to DNA inherited through each of the eight great-grandparents will vary from test-taker to test-taker for several reasons including who has tested DNA in the vendor's database.

As the technique was initially described, the names of a test-taker's predicted 2C and 3C matches at AncestryDNA are transcribed (or copied and pasted) in rank order into a table with the names of matches forming the first, left-most column of a table. A spreadsheet provides an easily modifiable way to build a table, but the method could also be implemented with paper and colored pencils or markers. If there are any matches in this list that share 400 cM of DNA or more with the test-taker, they are removed from the table. The method then applies a few simple rules to build new columns in the table by applying the rules to the entire table from the top of the table to the bottom of the table in a row-wise fashion. Each new column in the table produced by the method represents a group of shared matches.

Data sources

AncestryDNA does not provide customers with a method to download their match list or their in-common-with matches (shared matches) list for use off-line. On August 15, 2018, the commercially available DNAGedcom Client[2] software package was used to download the author's match list and the in-common-with matches in two separate .csv format files from his account at AncestryDNA. These files provide a snapshot of the author's account on that particular day.

The data in these .csv files was imported into the free version of the R statistical programming environment[3] using the free version of RStudio[4], an integrated development environment for R. The tidyverse package[5] for R was loaded first to make coding easier and more readable by humans.

[1]Dana Stewart Leeds, https://www.danaleeds.com, accessed September 25, 2018
[2]DNAGedcom LLC, DNAGedcom Client for Mac, http://dnagedcom.com/
[3]R Core Team (2018). R: A language and environment for statistical computing.R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.
[4]RStudio Team (2016). RStudio: Integrated Development for R. RStudio, Inc., Boston, MA URL http://www.rstudio.com/
[5]Hadley Wickham, Tidyverse.org, tidyverse version 1.2.1

These data sources are not included in the Github repository with this article due to privacy considerations[6].

There were 62794 rows of data, one row per match, in the matches file. There were 584404 rows of data in-common-with file.

The matches file had a total of 24 columns of information about each match. The names of those columns are given by the output of the following code chunk.

```
colnames(matches)
```

```
##  [1] "testid"             "matchid"            "name"
##  [4] "admin"              "people"             "range"
##  [7] "confidence"         "sharedCM"           "sharedSegments"
## [10] "lastlogin"          "starred"            "viewed"
## [13] "private"            "hint"               "archived"
## [16] "note"               "imageurl"           "profileurl"
## [19] "treeurl"            "scanned"            "membersince"
## [22] "ethnicregions"      "ethnictraceregions" "matchurl"
```

The in-common-with matches file had a total of 7 columns of information about each match. The names of those columns are given by the output of the following code chunk.

```
colnames(icw_matches)
```

```
## [1] "matchid"    "matchname"  "matchadmin" "icwid"      "icwname"
## [6] "icwadmin"   "Source"
```

Implementing the Leeds Method

The match list was pared down to just those matches that are predicted 2C and 3C cousins, and then a second filter was applied to remove any matches that shared 400 cM or more DNA with the author as shown in the R code chunk below.

```
# filter matches to include only predicted 2C and 3C
filtered_matches <- filter(matches, range %in% c("SECOND_COUSIN",
                                                 "THIRD_COUSIN"))

# filter out any matches that are greater than or equal to 400 cM
filtered_matches <- filter(filtered_matches,sharedCM < 400)
```

After applying these filters, there were 42 rows of data, one row per match. The filtered matches appeared to be sorted by shared cM values in descending order, and the following code explicitly does this to be sure of the order.

```
# sort filtered_matches high to low
filtered_matches <- arrange(filtered_matches, desc(sharedCM))
```

Looking at this filtered list of matches, the predicted cousin frequencies were as follows.

```
# summarize filtered_matches by predicted range so far
table(filtered_matches$range)
```

```
##
## SECOND_COUSIN  THIRD_COUSIN
##             3            39
```

The number of columns of data describing each match was reduced in this implementation because not all of the information was needed for this article.

[6] James Sims, pub_leeds repository, GitHub, accessed September 24, 2018

```
# retain all rows, but only include these columns
group_these <- select(filtered_matches,c(2,4,3,5,6,8,9))
```

The names of columns retained for each match in the grouping procedure are given in the following code output.

```
colnames(group_these)
```

```
## [1] "matchid"        "admin"          "name"           "people"
## [5] "range"          "sharedCM"       "sharedSegments"
```

A custom function was written to look-up the in-common-with matches for any given match in the table, and this is shown in the following code chunk.

```
getmatch_icw <- function(amatchid,icw_matches){
  # takes an AncestryDNA matchid value
  # takes the in-common-with match dataframe
  # returns the matchids for the shared matches
  temp <- filter(icw_matches,matchid == amatchid) %>%
    # column 4 contains the icwids
    select(4)
  temp
}
```

The Leeds Method grouping procedure was implemented as a function in R, and this is shown in the code output below.

```
get_leeds <- function(thedf,theicws){
  for(i in 1:nrow(thedf)){
  if(i==1){
    # need to add the first new column to grouped dataframe, initially empty
    grouped <- mutate(thedf,G1 = "")
    # populate column cells with the value TRUE when a match shares a match
    testing <- getmatch_icw(grouped$matchid[i],theicws)
    scoreTRUE <- which(apply(grouped,1, function(r) any(r %in% testing$icwid)))
    scoreTRUE <- c(i,scoreTRUE)
    grouped[scoreTRUE,ncol(grouped)] <- TRUE
  }
  # need to test row for any TRUE values for columns beginning with G
  # if TRUE is found, skip this row when building new columns
  # because this match has been previously grouped;
  # otherwise create a new column
  checkrow <- grouped[i,]
  if(sum(apply(checkrow,2,function(s) any(s == TRUE)))==0){
    # this is a row that needs to have a new column started for it
    varname <- paste("G",ncol(grouped) - 6,sep="")
    grouped[[varname]] <- ""
    testing <- getmatch_icw(grouped$matchid[i],theicws)
    scoreTRUE <- which(apply(grouped,1, function(r) any(r %in% testing$icwid)))
    scoreTRUE <- c(i,scoreTRUE)
    grouped[scoreTRUE,ncol(grouped)] <- TRUE
  }
  }
  # return a dataframe with output of the Leeds Method grouping
  grouped
}
```

To run the R version of the Leeds Method grouping procedure, calling the function get_leeds and passing it the table of matches to be grouped and the name of the in-common-with match table is all that is required as shown in the following code chunk.

```
# apply the Leeds Method grouping procedure
leeds_groups <- get_leeds(group_these,icw_matches)
```

In this particular case, the number of groups created by the Leeds Method is shown in the output of the code chunk below.

```
ncol(select(leeds_groups,starts_with("G")))
```

```
## [1] 8
```

The following code chunk shows the names of the columns in the table after the Leeds Method grouping procedure was applied to the table. The columns with the name starting with G are the new columns created by this implementation of the procedure. Each column name beginning with G is a group of shared matches.

```
colnames(leeds_groups)
```

```
##  [1] "matchid"        "admin"          "name"           "people"
##  [5] "range"          "sharedCM"       "sharedSegments" "G1"
##  [9] "G2"             "G3"             "G4"             "G5"
## [13] "G6"             "G7"             "G8"
```

The following code writes the grouped match table to disk as a .csv file. The file can be opened in a spreadsheet for viewing and further analysis.

```
write_csv(leeds_groups,"matches_grouped_by_leeds.csv")
```

For this article, an anonymous version of the output was saved to disk as an example that can be shared with the genetic genealogy community.

```
leeds_groups_anon <- select(leeds_groups,3:ncol(leeds_groups))
for(i in 1:nrow(leeds_groups_anon)){
  leeds_groups_anon$name[i] <- paste("match",i,sep=" ")
}
write_csv(leeds_groups_anon,"leeds_grouped_anon.csv")
```

In the case of the anonymous version of the output, the following columns (variables) of the output were written to disk as a .csv file. The *people* variable is an integer with the number of people in a tree that is linked to the match. If the match has a tree at Ancestry.com but it is not linked to the kit, the value for people will be zero. The *sharedSegments* variable is an integer with the number of shared segments each match shares with the test-taker, in this case, the author of this article.

```
colnames(leeds_groups_anon)
```

```
##  [1] "name"           "people"         "range"          "sharedCM"
##  [5] "sharedSegments" "G1"             "G2"             "G3"
##  [9] "G4"             "G5"             "G6"             "G7"
## [13] "G8"
```

In the anonymous output, the actual names were replaced with names such as match 1, match 2, etc. as shown in the output of the code chunk below.

```
leeds_groups_anon$name
```

```
##  [1] "match 1"  "match 2"  "match 3"  "match 4"  "match 5"  "match 6"
##  [7] "match 7"  "match 8"  "match 9"  "match 10" "match 11" "match 12"
## [13] "match 13" "match 14" "match 15" "match 16" "match 17" "match 18"
## [19] "match 19" "match 20" "match 21" "match 22" "match 23" "match 24"
```

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	name	people	range	sharedCM	sharedSegments	G1	G2	G3	G4	G5	G6	G7	G8
2	match 1	0	SECOND_COUSIN	295.4482	19	TRUE							
3	match 2	230	SECOND_COUSIN	286.7567	14	TRUE							TRUE
4	match 3	592	SECOND_COUSIN	206.486	10		TRUE						
5	match 4	0	THIRD_COUSIN	198.5935	11		TRUE						
6	match 5	7	THIRD_COUSIN	187.5046	9	TRUE					TRUE		TRUE
7	match 6	0	THIRD_COUSIN	185.5752	7			TRUE					
8	match 7	46785	THIRD_COUSIN	181.979	10	TRUE	TRUE						
9	match 8	0	THIRD_COUSIN	168.4967	6				TRUE				
10	match 9	0	THIRD_COUSIN	164.9278	11	TRUE							
11	match 10	592	THIRD_COUSIN	161.3144	6		TRUE						
12	match 11	0	THIRD_COUSIN	152.481	7			TRUE	TRUE				
13	match 12	0	THIRD_COUSIN	139.9216	6	TRUE							TRUE
14	match 13	4833	THIRD_COUSIN	138.714	7	TRUE					TRUE		
15	match 14	3	THIRD_COUSIN	129.6976	8	TRUE							TRUE
16	match 15	592	THIRD_COUSIN	128.502	8		TRUE						
17	match 16	0	THIRD_COUSIN	125.5835	7		TRUE						
18	match 17	17	THIRD_COUSIN	121.82	7				TRUE				
19	match 18	83	THIRD_COUSIN	121.209	6	TRUE							
20	match 19	14198	THIRD_COUSIN	117.5353	7	TRUE					TRUE		
21	match 20	0	THIRD_COUSIN	116.4197	7			TRUE					
22	match 21	0	THIRD_COUSIN	113.6529	6					TRUE			
23	match 22	0	THIRD_COUSIN	112.236	7						TRUE		
24	match 23	0	THIRD_COUSIN	109.4942	8			TRUE					
25	match 24	0	THIRD_COUSIN	108.3153	7	TRUE							
26	match 25	79	THIRD_COUSIN	108.0821	6		TRUE			TRUE			
27	match 26	0	THIRD_COUSIN	105.9008	5	TRUE					TRUE		
28	match 27	0	THIRD_COUSIN	104.569	7					TRUE			
29	match 28	55	THIRD_COUSIN	101.385	6	TRUE							TRUE
30	match 29	32	THIRD_COUSIN	98.5826	7			TRUE	TRUE				
31	match 30	272	THIRD_COUSIN	97.695	6							TRUE	
32	match 31	19	THIRD_COUSIN	97.57999	6								TRUE
33	match 32	0	THIRD_COUSIN	96.5309	5	TRUE					TRUE		TRUE
34	match 33	0	THIRD_COUSIN	96.0282	4	TRUE							
35	match 34	2139	THIRD_COUSIN	95.5647	5	TRUE							
36	match 35	881	THIRD_COUSIN	95.4718	7					TRUE			
37	match 36	29	THIRD_COUSIN	94.8102	5	TRUE							
38	match 37	0	THIRD_COUSIN	94.75136	4				TRUE				
39	match 38	0	THIRD_COUSIN	92.5517	5	TRUE							
40	match 39	23	THIRD_COUSIN	92.4669	3			TRUE					
41	match 40	0	THIRD_COUSIN	92.4521	3				TRUE				
42	match 41	0	THIRD_COUSIN	91.466	4					TRUE			
43	match 42	4833	THIRD_COUSIN	90.2593	5						TRUE		

Figure 1: Screenshot of output opened in a speadsheet.

```
## [25] "match 25" "match 26" "match 27" "match 28" "match 29" "match 30"
## [31] "match 31" "match 32" "match 33" "match 34" "match 35" "match 36"
## [37] "match 37" "match 38" "match 39" "match 40" "match 41" "match 42"
```

A screen shot of the table with anonymous output is shown in Figure 1. This was made by opening the anonymous output .csv file with a spreadsheet application and then invoking one of the screen capture tools built into the macOS. Figure 2 shows a colorized version of the table after the author applied coloring by hand.

Discussion

As implemented in this article, the R version of the Leeds Method inserts the word TRUE in a cell of a spreadsheet (that can be viewed from the .csv output of the code) rather than assigning that cell the group color when there is a shared match. This allows the genetic genealogist to use any color scheme they choose when they use the output for further analysis. In this implementation, each match group is assigned a sequential name (column name) beginning with G1, G2, through Gn rather than a color representing the "name" of the match group. This was done in anticipation of expanding the use of the Leeds Method to include hundreds or thousands of 4C matches, which may result in a inconvenient number of groups requiring different colors.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	name	people	range	sharedCM	sharedSegments	G1	G2	G3	G4	G5	G6	G7	G8
2	match 1	0	SECOND_COUSIN	295.4482	19	TRUE							
3	match 2	230	SECOND_COUSIN	286.7567	14	TRUE							TRUE
4	match 3	592	SECOND_COUSIN	206.486	10		TRUE						
5	match 4	0	THIRD_COUSIN	198.5935	11		TRUE						
6	match 5	7	THIRD_COUSIN	187.5046	9	TRUE					TRUE		TRUE
7	match 6	0	THIRD_COUSIN	185.5752	7			TRUE					
8	match 7	46785	THIRD_COUSIN	181.979	10	TRUE	TRUE						
9	match 8	0	THIRD_COUSIN	168.4967	6				TRUE				
10	match 9	0	THIRD_COUSIN	164.9278	11	TRUE							
11	match 10	592	THIRD_COUSIN	161.3144	6		TRUE						
12	match 11	0	THIRD_COUSIN	152.481	7			TRUE	TRUE				
13	match 12	0	THIRD_COUSIN	139.9216	6	TRUE							TRUE
14	match 13	4833	THIRD_COUSIN	138.714	7	TRUE					TRUE		
15	match 14	3	THIRD_COUSIN	129.6976	8	TRUE							TRUE
16	match 15	592	THIRD_COUSIN	128.502	8		TRUE						
17	match 16	0	THIRD_COUSIN	125.5835	7		TRUE						
18	match 17	17	THIRD_COUSIN	121.82	7				TRUE				
19	match 18	83	THIRD_COUSIN	121.209	6	TRUE							
20	match 19	14198	THIRD_COUSIN	117.5353	7	TRUE					TRUE		
21	match 20	0	THIRD_COUSIN	116.4197	7			TRUE					
22	match 21	0	THIRD_COUSIN	113.6529	6					TRUE			
23	match 22	0	THIRD_COUSIN	112.236	7						TRUE		
24	match 23	0	THIRD_COUSIN	109.4942	8			TRUE					
25	match 24	0	THIRD_COUSIN	108.3153	7	TRUE							
26	match 25	79	THIRD_COUSIN	108.0821	6		TRUE			TRUE			
27	match 26	0	THIRD_COUSIN	105.9008	5	TRUE					TRUE		
28	match 27	0	THIRD_COUSIN	104.569	7					TRUE			
29	match 28	55	THIRD_COUSIN	101.385	6	TRUE							TRUE
30	match 29	32	THIRD_COUSIN	98.5826	7			TRUE	TRUE				
31	match 30	272	THIRD_COUSIN	97.695	6							TRUE	
32	match 31	19	THIRD_COUSIN	97.57999	6								TRUE
33	match 32	0	THIRD_COUSIN	96.5309	5	TRUE					TRUE		TRUE
34	match 33	0	THIRD_COUSIN	96.0282	4	TRUE							
35	match 34	2139	THIRD_COUSIN	95.5647	5	TRUE							
36	match 35	881	THIRD_COUSIN	95.4718	7					TRUE			
37	match 36	29	THIRD_COUSIN	94.8102	5	TRUE							
38	match 37	0	THIRD_COUSIN	94.75136	4				TRUE				
39	match 38	0	THIRD_COUSIN	92.5517	5	TRUE							
40	match 39	23	THIRD_COUSIN	92.4669	3			TRUE					
41	match 40	0	THIRD_COUSIN	92.4521	3				TRUE				
42	match 41	0	THIRD_COUSIN	91.466	4					TRUE			
43	match 42	4833	THIRD_COUSIN	90.2593	5						TRUE		

Figure 2: Screenshot of output colorized by hand in a spreadsheet.

There are several compelling aspects of the Leeds Method for summarizing autosomal DNA matches as originally implemented. It does not require advanced knowledge of genetic genealogy to implement. It does not require advanced computer skills to implement, and it does not require any prior knowledge of the families in the match list or in the in-common-with matches. These aspects of the method make it very appealing when *beginning* to assess how matches are related to a test-taker. If there are only a couple of dozen or so matches that meet the Leeds Method criteria, the method is relatively straight forward to implement by hand on an *ad hoc* basis.

Given all of the above mentioned positive aspects of the Leeds Method, why re-implement the method in computer code? In a word, reproducibility. Reproducibility is one of the fundamental requirements of sound genealogical research. Reproducibility means given a set of genealogical and genetic data, another person can reproduce the previous analysis.

Genetic genealogists need to be aware that vendor databases containing the source data for their genetic analyzes are dynamic and not static. Match lists change over time. Not only can new matches appear over time, customers are also free to remove their data from vendor databases. Genetic genealogists need a convenient snapshot of their genetic data sources that they use in an analysis. Downloaded match lists and in-common-with match lists like those produced by the DNAGedcom Client software can be used for this purpose.

Today, if the diligent genetic genealogist chooses to go the reproducible route and perform analyses on snapshots of their genetic data, and does not reply on dynamic web site screens, they face some serious skills hurdles. For example, without the right software tools, it's not easy to deal with tens or hundreds of thousands of rows of data. Computer programming in general, and the R statistical programming environment used here, are not in the common skill set of the current generation of the most avid of genealogists. Computer programming has a very steep learning curve. That said, there are skilled programmers who can take code similar to that used here, and re-implement it in R or in some other programming language with a web page interface to reduce the computer skill set required for genetic genealogists to improve the reproducibility of their genetic genealogy research. The author of this article does not have the skills required to provide an automated Leeds Method service via the web given a pair of .csv input files. He hopes some enterprising programmer will create an easy-to-use automated tool for this purpose. Ideally, that programmer would make the implementation an open-source project, so the computer code can be inspected and tested for reproducibility by the genetic genealogy community.

If it were the case that genetic genealogists had an easy-to-use, automated Leeds Method tool that was capable of being applied to larger data sets, such as those including hundreds or thousands of 4C matches, today they still have considerable hurdles when they try to interpret the groups produced by the Leeds Method. Some of the issues are related to understanding the possibilities of an in-common-with match group and this is discussed elsewhere[7]. An additional problem is that of DNA matches with no trees and very short trees linked to their DNA kit. For example, in this article, of the 42 matches that were grouped by the Leeds Method, 21 had zero people in a tree linked to their DNA result. Hopefully, in the future people who take a DNA test for some purpose other than genealogy will become interested in genealogy and this situation will improve. For the present, inspecting matches by hand online to discover un-linked trees, and contacting matches and patiently waiting for a reply seems to be the prudent approach.

Software

DNAGedcom Client version 2.1.6 (2.18) for Mac was used to download data from the author's account at AncestryDNA. The free R version 3.5.1 (2018-07-02) was used for this analysis. The code was developed in RStudio version 1.1.456. The tidyverse package version 1.2.1 was used to make coding easier and more readable for humans. This report was produced within the RStudio integrated development environment using rmarkdown version 1.10 and the knitr package version 1.2. and its dependencies. MacTeX-2018 was

[7]Blaine T. Bettinger, Family Tree Guide to DNA Testing and Genetic Genealogy, chapter 6, Family Tree Books, 2016; available in paperback and Kindle editions

used for pdf output on a 2017 MacBook Pro running macOS version 10.13.6 (17G65), which is commonly called High Sierra.