REL-A-TREE Icon by:
Brenna A. McLeod

REL-A-TREE

A program for assessing patterns of relatedness without any *a priori* information

ver. 1.0

Written By:

Tim Frasier

Saint Mary's University
Department of Biology & Forensic Sciences Programme
923 Robie Street
Halifax, Nova Scotia B3H 3C3
Canada
Tel: (902) 491-6382
Fax: (902) 420-5046
E-mail: timothy.frasier@smu.ca

Table of Contents

Overview

REL-A-TREE is a program written in C designed to provide a means to cluster individuals based on their relatedness to one another. This approach should provide a means for users to assess patterns of relatedness within populations even when they don't have any other information to guide their investigations. The program works by calculating all pairwise relatedness values for a data set, and then formatting the data in a way that can be read by the PHYLIP program. In PHYLIP, a tree can be made based on these relatedness values, which then forms the means of clustering individuals based on relatedness. Bootstrap values can be obtained by bootstrapping the loci, which can aid in interpreting branching patterns. REL-A-TREE can also generate a data set containing individuals of known relatedness, based on your allele frequency file, which can act as a sort of power analyses for your data. It would be wise to run the analyses on these simulated data first, in order to identify what sort of resolution you should expect in your data set.

The bootstrap procedures, as well as the process of Mendelian inheritance within the simulated data sets, require random numbers. REL-A-TREE uses random number generators from the GNU Scientific Library (GSL, Galassi *et al.* 2006).

Legal Stuff

I tested REL-A-TREE extensively throughout its development, and subsequently tested it using several data sets upon completion. It appears to be functioning properly, and I am currently using it to analyze data from projects that I am working on. However, I cannot guarantee that it does not contain any errors, or that its results will always be reliable.

Installation

There are three ways to install REL-A-TREE: one way for Windows; and two for LINUX - one method without the source code, and the other with the source code that you can compile yourself. Details are below.

Windows Installation

Step 1: Download the zipped file from the website.

Step 2: Unzip it wherever you want to keep the program
- The folder will include:
  (a) The executable file;
  (b) A folder containing the example files; and
  (c) Other files necessary for the program to run (don't move or delete them!!!)

Step 3: You should place your infiles in the same folder as the executable file prior to running the program.

LINUX Installation #1 (No Source Code)

Step 1: Download the appropriate file from the website.

Step 2: Move the folder to wherever you want to keep the program.

Step 3: Unzip the file by typing:

```
>gunzip Rel-A-Tree_LINUX_1.0.tar.gz
```

Then:

```
>tar -xvf Rel-A-Tree_LINUX_1.0.tar
```

- This will make a folder called Rel-A-Tree_LINUX_1.0

Step 4: Go into that folder

```
>cd Rel-A-Tree_LINUX_1.0
```

- There will be a folder called "Example_Files", and a file called "tree"

Step 5: Turn this file into an executable by typing:

```
>chmod a+x tree
```

Step 6: You can now start the program by typing:

```
>./tree
```

LINUX Installation #2 (From the Source Code)

If you choose to download the source code and compile it yourself, you will need to have the GNU Scientific Library (GSL) installed. There are a few ways to get this. One is from the GNU website (http://www.gnu.org/software/gsl/). If you are using Ubuntu, you can also install the GSL using Synaptic (install both the "gsl-bin" and the "libgsl0-dev" packages). Note that you will also have to have some general programming packages installed as well: "libc6" and "build-essential", which can also be installed via Synaptic.

Once that is installed, you can install REL-A-TREE.

Step 1: Download the appropriate file from the website.

Step 2: Move the folder to wherever you want to keep the program.

Step 3: Unzip the file by typing:

```
>gunzip Rel-A-Tree_LINUX_1.0_Code.tar.gz
```

Then:

```
>tar -xvf Rel-A-Tree_LINUX_1.0_Code.tar
```

- This will make a folder called Rel-A-Tree_LINUX_1.0_Code

Step 4: Go into that folder

```
>cd Rel-A-Tree_LINUX_1.0_Code
```

- There will be a folder called "Example_Files", and a file called "Rel-A-Tree.c"

Step 5: You can compile the source code by typing:

```
>gcc Rel-A-Tree.c -lgsl -lgslcblas -lm -otree
```

- This will create and executable file called "tree"

Step 6: You can now run the program by typing:

```
>./tree
```

Infiles

There are only two infiles that you will need for any of the three analyses conducted by REL-A-TREE: your allele frequency file and your genotype file (the simulation only requires the allele frequency file). You files should be in a tab-delimited text format. The names of the files should not be longer than 40 characters (including extension), and they **should not contain any spaces**. When entering the name of your files, be sure to include the extension (if they have one).

Allele Frequency File

Unfortunately, REL-A-TREE will not calculate your allele frequencies for you, and you will have to enter these data as a separate file. Your allele frequency file should have one column for each locus, with each column containing the allele frequencies for that locus. The loci MUST be ordered in the same way as in the genotypes; meaning that the first locus in the allele frequency file should also be the first locus found in the genotypes, …and so on. A zero (0) needs to be present at the top of each column, and you cannot include locus names. ALL columns must have values for the same number of rows. This means that the number of rows will equal the number of alleles in your most polymorphic locus. You MUST fill these rows with integers > 1 as placeholders for those loci with fewer alleles. An example allele frequency file is shown below for four loci. The first locus has two alleles, the second has four alleles, the third has five alleles, and the fourth has three alleles. In this case, 5 was used as the placeholder.

```
0       0       0       0
0.2     0.1     0.1     0.4
0.8     0.2     0.3     0.1
5       0.5     0.15    0.5
5       0.2     0.2     5
5       5       0.25    5
```

Genotype File

The format for your genotype file is the basic format that you will (hopefully) typically have your data in anyway; with a column of individual IDs, followed by two columns for each locus you used (one column for each allele). There are some rules though. First, individual IDs MUST be numeric (they cannot contain text!!!!). Second, IDs MUST be 10 numbers long!!! This is a requirement of PHYLIP. If your IDs are not 10 numbers long, then REL-A-TREE will run properly, but the interpretation of those data in PHYLIP will be completely incorrect (and you may not know it!!). Third, you cannot include a row of locus names in your genotype file. Lastly, all missing data must be coded as zero (0). The way you code your alleles is up to you - as long as they are numeric. An example of a proper genotype file is below: it contains five individuals typed at three loci.

```
1000000001  120 122 203 203 98  100
1000000002  120 120 0   0   100 100
1000000003  118 122 201 203 100 102
1000000004  118 118 201 201 98  100
1000000005  120 120 203 205 98  104
```

Conducting Analyses

Pairwise Relatedness (Option #1)

The first option in the REL-A-TREE menu is to conduct pairwise relatedness for all of the individuals in your genotype file. This option is here so that once you have performed your clustering analysis (option #2) you can see the actual relatedness values for the individuals that clustered together. For this analysis you will need both your allele frequency file and your genotype file, formatted as described above.

The relatedness calculation is based on the method described in Li *et al.* (1993), with each locus weighted using the method described in Lynch & Ritland (1999) and Van de Casteele *et al.* (2001). This method was chosen out of the many available approaches for calculating relatedness because it is unbiased, it is never undefined, and it consistently performs well in a variety of situations (Van de Castelle *et al.* 2001; Wang 2002; Krützen *et al.* 2003). The equation for the relatedness at each locus (ℓ) is:

$$r_{xy}(\ell) = \frac{S_{xy} - S_o}{1 - S_o}$$

where $S_{xy} = 1$ when (genotype $x = ii$, genotype $y = ii$) or ($x = ij$, $y = ij$), $S_{xy} = 0.75$ when ($x = ii$, $y = ij$) or vice versa, $S_{xy} = 0.5$ when ($x = ij$, $y = ik$), $S_{xy} = 0$ when ($x = ij$, $y = kl$), with i, j, k, and l representing the alleles at the different allelic positions.

$$S_o = \sum_{i=1}^{n} p_i^2(2 - p_i)$$

where p_i is the population allele frequencies for allele i. Multilocus relatedness values can be obtained by multiplying r_{xy} for each locus by the weight for that locus, summing this value across loci, and then dividing the sum of all weights used. The weight for each locus is calculated based on the number of alleles in each locus (ℓ):

$$w(\ell) = \frac{n_j = 1}{\Sigma n_j = 1}$$

where n_j is the number of alleles at locus j.

So…total relatedness is:

$$r_{xy} = \frac{1}{W} \sum w(\ell) r_{xy}(\ell)$$

where W is the sum of the weights for all loci used.

Lets walk through an example using the example infiles. We will use the file "frequencies", and the file "pairwise", with a list of five individuals genotyped at two loci.

"frequencies"

```
0       0
0.1     0.2
0.3     0.4
0.5     0.4
0.1     5
```

"pairwise"

```
1   1   2   2   2
2   2   2   2   3
3   3   3   1   1
4   1   2   2   3
5   1   3   2   2
```

S_o for locus 1 = 0.019 + 0.153 + 0.375 + 0.019 = 0.566
S_o for locus 2 = 0.072 + 0.256 + 0.256 = 0.584

Weight for locus 1 = 3/6 = 0.5
Weight for locus 2 = 2/6 = 0.333

Total weights (W) = 0.833

Pair	S_{xy} locus 1	S_{xy} locus 2	r_{xy} locus 1	r_{xy} locus 2	r-value
1 & 2	0.75	0.75	0.424	0.399	0.414
1 & 3	0	0	-1.30	-1.40	-1.34
1 & 4	1	0.75	1	0.399	0.760
1 & 5	0.5	1	-0.152	1	0.309
2 & 3	0	0	-1.30	-1.40	-1.34
2 & 4	0.75	1	0.424	1	0.654
2 & 5	0	0.75	-1.30	0.399	-0.621
3 & 4	0	0	-1.30	-1.40	-1.34
3 & 5	0.75	0	0.424	-1.40	-0.305
4 & 5	0.5	0.75	-0.152	0.399	0.0683

Note that you will get a result of "nan" (in LINUX) or "-1.#INDO0" (in Windows) in those cases where relatedness could not be calculated due to missing data (e.g. when two individuals are missing data at enough loci that no loci can be compared between them). As long as at least one locus is available for comparison, REL-A-TREE will calculate relatedness.

Cluster Analysis (Option #2)

This second option is where the "magic" happens, where you can perform a cluster analysis of individuals based on their relatedness values. It is a three-step process, one involving REL-A-TREE, one involving PHYLIP, and the last involving a tree-drawing program of your choice.

First Step: REL-A-TREE

For this analysis, you will need your allele frequency file and your genotype file, formatted as described above (remember that your IDs have to be 10 numbers long!!). What the program will do first is calculate all pairwise relatedness values using the method described above.

Next, a conversion must take place. Relatedness values tend to range from -1 to 1, whereas genetic distance values only range from 0 to 1. Additionally, the values have *opposite* "meanings": a high relatedness value should be a low distance value, and vice versa. Therefore, the relatedness values need to be adjusted accordingly, so that they have "distance-like" characteristics. The equation that REL-A-TREE uses is:

$$d = \lceil 1 - ((r + 1)/2) \rceil$$

where d is the new genetic distance and r is the relatedness value. To give you some idea of how this works, some example r-values and the associated d-value calculations are given below.

r-value	*d*-value
-1	1
0	0.5
1	0

r-values < -1 are given a *d*-value of 1.

After the relatedness values have been converted to distance values, the data are formatted into an "upper-triangular matrix" that can be directly used as input into PHYLIP. The output will be given to you as a file called "Rel_Distances", which can then be analyzed in PHYLIP.

Note that REL-A-TREE formats the data **specifically for PHYLIP**. Other phylogenetic programs use different formats, that won't necessarily work with your REL-A-TREE output. If you need to analyze your REL-A-TREE data with another phylogenetic program, let me know and I may be able to change the output accordingly.

You can assess the confidence in your branching patterns by bootstrapping your data. When you do this, REL-A-TREE will bootstrap your loci, re-calculate relatedness and create a new matrix for each bootstrapped data set. All of these matrices will be placed in the same outfile ("Rel_Distances"), which can be directly input into PHYLIP. You will need to tell PHYLIP how many times you bootstrapped your data.

Second Step: PHYLIP

Once REL-A-TREE is done running, you can take your outfile ("Rel_Distances"), and analyze it in PHYLIP. Essentially, your data will be formatted in a distance matrix exactly how PHYLIP expects distance matrices to be formatted for sequence data. That allows us to take advantage of the tree-drawing programs in PHYLIP for our purposes. There are currently two tree-drawing options in PHYLIP: (1) **neighbor**, which can draw trees using either the UPGMA or neighbor-joining method (Saitou & Nei 1987); and (2) **fitch**, which uses the Fitch-Margoliash method for drawing a tree based on a distance matrix (Fitch & Margoliash 1967). Personally, when doing phylogenetic analyses based on a distance matrix method, I prefer the Fitch-Margoliash algorithm. However, it is far too slow for analyzing large data sets, and therefore with my REL-A-TREE data I tend to use the neighbor-joining method. I will use the neighbor-joining method in my examples below.

First, copy your Rel_Distances file and paste it into the folder that contains the executable files for PHYLIP. Then, select the neighbor-joining option. The program will ask you for the name of your infile, where you should type `Rel_Distances`. Then, a menu like the one below will open up and you can select different options for your data.

```
Neighbor-Joining/UPGMA method version 3.5

Settings for this run:
  N       Neighbor-joining or UPGMA tree?  Neighbor-joining
  O                       Outgroup root?  No, use as outgroup species  1
  L         Lower-triangular data matrix?  No
  R         Upper-triangular data matrix?  No
  S                       Subreplicates?  No
  J     Randomize input order of species?  No. Use input order
  M           Analyze multiple data sets?  No
  0   Terminal type (IBM PC, VT52, ANSI)?  ANSI
  1    Print out the data at start of run  No
  2  Print indications of progress of run  Yes
  3                        Print out tree  Yes
  4       Write out trees onto tree file?  Yes

Are these settings correct? (type Y or the letter for one to change)
```

There are two main options that you need to change when analyzing REL-A-TREE data:

(1) You need to select "upper-triangular data matrix". Select this by using the "R" option in the menu, so that it says "Yes" for this option.

(2) If you bootstrapped your data, you want to "Analyze multiple data sets" – option "M". When you select "M", it will ask you how many data sets you want to analyze. Here, you should enter the number of times that you bootstrapped your data.

That's about it! The program will then create one tree for each bootstrapped data set. That is not what you want, however. What you need to do next is use the **consense** program in PHYLIP to make a consensus tree. There are a couple steps to this process:

(1) **neighbor** will have given you two outfiles – "outfile" and "outtree". You need to re-name these before running **consense**. You can rename them anything you like. For example, lets say we re-named them "neighout" and "neightree", respectively.

(2) When you run **consense** it will ask you for your tree file. This is your tree file, not your outfile from **neighbor**. Using the naming from above, you would enter `neightree`. You will then get the menu below, and you can change whatever options you feel are appropriate. In most cases, the default options should be fine.

```
Majority-rule and strict consensus tree program, version 3.5c
Settings for this run:
  O                       Outgroup root?  No, use as outgroup species  1
  R         Trees to be treated as Rooted?  No
  0   Terminal type (IBM PC, VT52, ANSI)?  ANSI
  1          Print out the sets of species  Yes
  2  Print indications of progress of run  Yes
  3                        Print out tree  Yes
  4       Write out trees onto tree file?  Yes

Are these settings correct? (type Y or the letter for one to change)
```

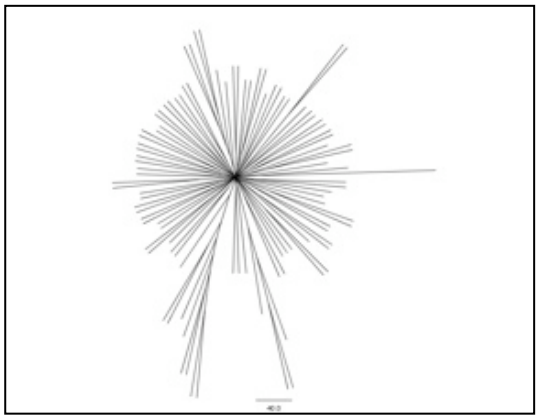
Consense will also give you two outfiles called "outfile" and "outtree". It is the "outtree" file that contains the tree information that will be used by the tree-drawing

programs.

Third Step: Tree-Drawing Program

The output file from PHYLIP contains all of the information for your consensus tree. Now you need to choose a program for viewing that tree. There are a large number of options for this. I highly recommend **FigTree** (http://tree.bio.ed.ac.uk/software/figtree/). It is extremely easy to use, and is very versatile. It is particularly good for large data sets because it allows you to zoom in and out, so that you can take a closer look at different aspects of your tree.

You can import your tree into the program, and *voila* – you have your tree based on relatedness values!!!! Hopefully you see some patterns that are interesting to you! Remember that this tree is *unrooted*, and therefore the best way to view it is as an unrooted tree (see example below).



An example of an unrooted tree.

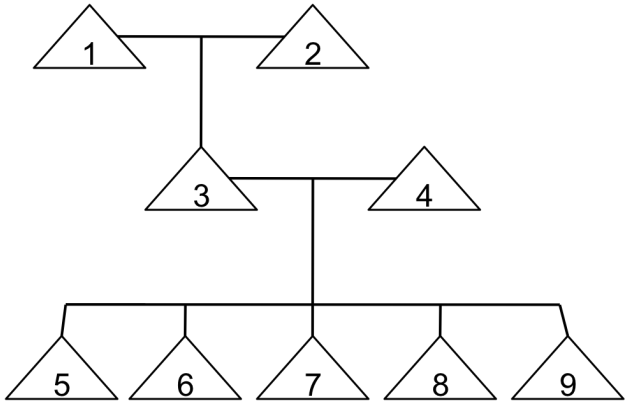
Generating Simulated Data Set (Option #3)

The third option in REL-A-TREE will generate a simulated data set, based on your allele frequencies, of 100 individuals, some of whom are of known relatedness (everyone else should be unrelated). This output file can then be run through Option #2 in order to see if the program can detect the known relatives in the data set. This provides a way for you to assess the sort of resolution that you can expect from your data.

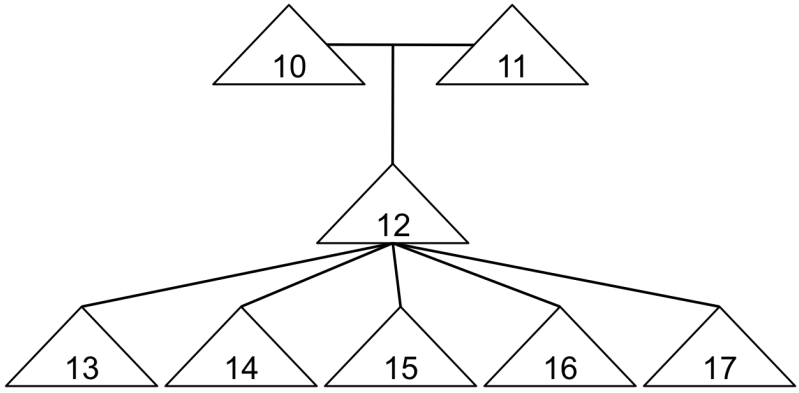
You will need to input your allele frequency file, formatted as described above. From there, the program generates 100 individuals by randomly assigning alleles to individuals in proportion to the allele frequencies. It then creates three families of varying depth and structure, and places these individuals back into the genotype file. The three family groups that are created are shown below. Note that in the output the

identifiers for the names will be 10 numbers long; but to save space I have just called them 1, 2, 3, etc… In the actual file, these same individuals would be labeled 1000000001, 1000000002, 1000000003, etc…

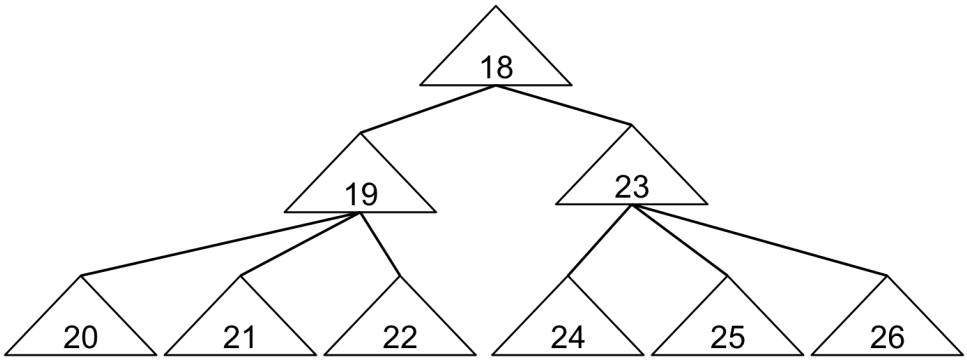
Family #1:



Family #2:
Note that individuals 13 through 17 are half-sibs. They share one parent (12), but each has a different other parent (who are not in the data set, so you won't find them).



Family #3:
Individuals 19 & 23 are half-sibs. Each of them has three offspring, each with a different mate. Therefore individuals 20 – 22 are half-sibs, and individuals 24-26 are half-sibs. Additionally, the offspring of 19 and 23 are cousins. As above, the individuals inferred, but not present, in the family tree are not in the data set.



Therefore, once you have run the simulations, and analyzed the data via Option #2, you should look to see if individuals 1 – 9 cluster together (a relatively 'tight' family), if individuals 10 – 17 cluster together (a 'looser' family), and if individuals 18 – 26 cluster together. Comparing these expected clustering patterns to what you actually get from analyzing these same data should tell you what sort of relationships you expect to pick up in your data set.

References

Fitch WM, Margoliash E (1967) Construction of phylogenetic trees. *Science* **155**: 279-284.

Galassi M, Davies J, Theiler J, Gough B, Jungman G, Booth M, Rossi F (2006) GNU Scientific Library Reference Manual - Revised Second Edition (v 18). Network Theory Ltd. Bristol UK.

Krützen M, Sherwin WB, Connor RC, Barré LM, Van de Casteele T, Mann J, Brooks R (2003) Contrasting relatedness patterns in bottlenose dolphins (*Tursiops* sp.) with different alliance strategies. *Proceedings of the Royal Society of London* Series B **270**: 497-502.

Li CC, Weeks DE, Chakravarti A (1993) Similarity of DNA fingerprints due to chance and relatedness. *Human Heredity* **43**: 45-52.

Lynch M, Ritland K (1999) Estimation of pairwise relatedness with molecular markers. *Genetics* **152**: 1753-1766.

Saitou N, Nei M (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution* **4**: 406-425.

Van de Casteele T, Galbusera P, Matthysen E (2001) A comparison of microsatellite-based pairwise relatedness estimators. *Molecular Ecology* **10**: 1539-1549.

Wang J (2002) An estimator for pairwise relatedness using molecular markers. *Genetics* **160**: 1203-1215.