DIGUP – Detection of Incompatible Genealogies for Unphased Populations

September 2015

This document describes how to use DIGUP, a program for the detection of incompatible genealogies among populations using unphased data. The purpose of this program is to detect fragments coming from different histories, among populations, along the genome.

Downloading and requirements

DIGUP program and this manual are included in the file available at https://github.com/mvidalv/DIGUP.

DIGUP does not require external libraries for basic command lines, however if the output with graphical visualization of the fragments is wanted (see outputs below), Biopython and BioGraPy libraries are required.

Bipython can be downloaded at *http://biopython.org/wiki/Download*
BioGraPy can be downloaded at *http://apierleoni.github.io/BioGraPy/#download*

DIGUP usage

DIGUP is able to read both *fasta* and *ms* format and includes several arguments, ones are optional and others are required. Below, DIGUP usage and detailed explanation of each argument.

`Usage: DIGUP.py input_file -n n -i` i_1 i_2.. i_n `[-o {1,2,12}] [-ms] [-l length] [-nt` nt_1 nt_2.. nt_n`] [-G]`

-n total number of sequences (including the outgroup).

-i total number of individuals in each population (in the same order as in the input file). Last population is considered to be the outgroup.

-ms if input file is in *ms* format (default reading is for *fasta* format)

-o output type (1, 2 or both as 12) for the classification of variant positions. Default output type is 1, which includesclassification, of each population, of all variant positions. Output type 2 includes same information as type 1 but also information about the nucleotide type.

Type 2 is only valid for fasta format as *ms* format does not include nucleotides. (This argument is optional)

-nt total number of samples in each population in case of pool-seq (not including outgroup). Default value is the number of samples given by -i argument. (This argument is optional)

-l total length of sequences (only for *ms* format)

-G get a graphical visualization of the genome fragments in a .png file (This argument is optional)

The basic command line

The below line is the simplest usage of DIGUP, where the input file is a fasta format file containing aligned sequences of the populations of interest, n is the number of sequences in the input file and i is the number of individuals in each population *p* included in the input file.

```
DIGUP.py file.fas -n n -i i1 i2.. ip
```

Extended command line

Below there are three examples of more extended command lines including optional arguments.

Fasta file with sequences of three populations, with the following individuals in each population: 20, 13, 11 and 6 samples for the outgroup. It is wanted both output types for the classification of variant positions.

```
DIGUP.py file.fas –n 50 –i 20 13 11 6 -o 12
```

Fasta file with sequences of two populations, with the following individuals in each population: 32, 42 and 4 samples for the outgroup. Total individuals in each population introduced into the pool-seq were: 40 and 50. Graphical visualization of the fragments is wanted.

```
DIGUP.py file.fas –n 78 –i 32 42 4 -nt 40 50 -G
```

ms file with sequences of two populations, with the following individuals in each population: 34 54 and 10 samples for the outgrup. Sequences length is 1 Mbp. Graphical visualization of the fragments is wanted.

```
DIGUP.py file.ms –n 98 –i 34 54 10 -ms -l 1000000 -G
```

The outputs

A total of 7 outputs containing relevant information can be generated during the program run. Some of these outputs are optional and others not. All ouputs are .txt files named with an exclusive name and extended with the input file name and the date (ddmmyyyy) of the run.

The following ouput examples we present are not correlated. Only raw fragments and final incompatible fragments are correlated outputs.

Classification of the variant positions

Two available outputs for getting this information.

- Classification_1_filename_ddmmyyyy.txt:

This output includes all positions that are variant at least in one population, classification for each population the classification of positions, the number of individuals per population having a variant allele (different to the outgroup) and total number of individuals analysed (individuals with missing data for that position not included)

```
D:derived   A:ancestral   P:polymorphic   m:multiple   ancestral
M:multiple derived N:unknown ancestral U:unknown nucleotide in
population

name  position   [A/D/P]    #D    #Total
                 pop[0]           pop[1]                  pop[2]

example.txt   2  P     2     10    D     6     6     A     0     8
example.txt   3  P     1     5     A     0     6     A     0     8
example.txt   5  m     1     10    m     0     5     m     0     8
example.txt   7  P     7     9     A     0     6     P     5     8
example.txt   9  P     4     10    U     0     0     P     3     8
example.txt   9  N     1     0     N     0     0     N     0     0
```

- Classification_2_filename_ddmmyyyy.txt:

This output includes all positions that are variant at least in one population, classification for each population the classification of positions, the classification of each nucleotide regarding the outgroup (ancestral or derived) and the count of each nucleotide found in each population.

```
D:derived A:ancestral P:polymorphic m:multiple ancestral M:multiple
derived N:unknown ancestral

name    position       A:#[A/D] C:#[A/D] T:#[A/D] G:#[A/D]
                       pop[0]                                pop[1]
example.txt   257     A:10D   C:0A    T:0D    G:0D           A:5D
       C:0A    T:0D    G:0D           A:5D    C:0A    T:0D    G:0D
```

```
example.txt     271    A:10D   C:0A    T:0D    G:0D            A:5D
        C:0A    T:0D    G:0D            A:5D    C:0A    T:0D    G:0D
example.txt     305    A:0A    C:0D    T:10D   G:0D            A:0A
        C:0D    T:5D    G:0D            A:0A    C:0D    T:5D    G:0D
example.txt     306    A:0A    C:0D    T:0D    G:10D           A:0A
        C:0D    T:0D    G:5D            A:0A    C:0D    T:0D
```

Summary of the classification

This output (named summary_filename_ddmmyyyy.txt) includes the counts of the combinations (understanding combinations as the classification of all populations in one position), the total number of multiple hit positoins (M), total number of polymorphic positions in the outgroup sequence (m), number of unknown positions in the outgroup (N), total number of combinations found regarding the total number of combinations that could have been found (*i,e,*. 3^n, where n is the total number of populations - outgroup not included-) total length of the sequences analysed.

```
+------------------+
| GENERAL SUMMARY |
+------------------+
Type    #positions

AAA     38241
PAD     7854
APP     2657
AAD     267
DAD     150

M       34
m       256
N       542

Combinations found: 5
Possible combinations: 27
Sequence length: 50001

D:derived A:ancestral P:polymorphic m:multiple ancestral M:multiple
derived N:unknown ancestral
```

Incompatible positions

This output, named inc_positions_filename_ddmmyyyy.txt includes incompatible positions found by different incompatibility types. Each incompatibility type is numbered with an increasing number starting with 1 and incompatible positions within the incompatibility are tagged as *a* and *b*, being a positions incompatible with b positions, this file also includes the incompatible combinations for each incompatibility type.

```
+------------------------+
| INCOMPATIBLE POSITIONS |
+------------------------+

[1a] PPP 3054
[1b] APD 354, 358, 626, 726, 782, 1277, 1280

[2a] PPP 3054
[2b] AAD 401, 1370, 1451, 1647, 1671, 2066, 2082

[3a] DPD 2182, 2286
[3b] PDD 405, 419, 470, 892, 926, 941, 2308
```

Raw incompatible fragments

This output, called Raw_fragments_filename_ddmmyyyy.txt, includes all incompatible fragments (that is, fragments with incompatible genealogies) found. Some of these fragments are redundant. For a pair of incompatible fragments, are shown as last position from one fragment and start position of the contiguous incompatible fragment, followed by the combinations making them incompatible fragments and the weight and normalised weight of reliability for the incompatibility found.

```
+-----------------+
| RAW FRAGMENTS   |
+-----------------+

[...end]    [start...]   comb1      comb2       w     w_norm

[...981] [2165...]     PDD     DPD     0.85    0.76
[...921] [2165...]     AAD     PPP     0.63    0.54
[...2203] [2315...]    DPD     PDD     0.98    0.95
```

Final incompatible fragments

This output, called inc_fragments_filename_ddmmyyyy.txt., includes final incompatible fragment, excluding those being redundant and keeping the most informative ones (the longest ones). Format is the same as raw fragments output file.

```
+------------------------+
| INCOMPATIBLE FRAGMENTS |
+------------------------+

[...end]    [start...]   comb1      comb2       w     w_norm

[...981] [2165...]     PDD     DPD     0.85    0.76
[...2203] [2315...]    DPD     PDD     0.98    0.95
```

Visualization of the fragments

This output, called Graph_Fragments_filename_ddmmyyyy.txt, is a .png file that includes the visualization of the incompatible fragments found. Positions in

between fragments, not located into any fragment, are coloured, from white to dark blue, regarding the weight of reliability of the two fragments.

The following image belongs to a single output and it has been split into two images in order to better see the details.