GA-TA: Genetics Application - Table Adapter. User's Manual

M. M. Gamboa Lerena, G.P. Di Santo Meztler, S. del Palacio & F. G. López Armengol

September 17, 2019

Abstract

We present the GA-TA program to convert easy-to-build generic tables into more complex tables in the specific format required to use with Structure, Arlequin, and R softwares. The development of this program is motivated by the need of a simple and handy tool to ease a labor that is otherwise done by hand, which results in a highly time-consuming and inefficient task. The main characteristics of this program are that it is very simple to use, free of charge, and easily expandable to new applications. The program is written in Python 2.7 under an open source policy, allowing experienced users to download the program from a github repository and adapt it by adding new modules upon convenience. A stable version of the program can be executed online.

1 Introduction

Several informatic tools have been developed for the analysis of genetic data. Some of the most popular and widely used softwares in the field of population genetics are Arlequin [1], Structure [2], and R [3]. The problem for many scientists is that there is a non-trivial step in adapting the simple layout of the data sheet they build into the specific format in which it is required to be processed by those programs. As far as we are concerned, there is currently no freely available and straightforward-to-use tool that can bridge this gap of manufacturing the generic data table efficiently. This inconvenient leads many scientists to perform this labour manually, which is not only highly time-consuming, but also subtle to the possibility of human errors. The magnitude of this problem has escalated recently because of the current access to large databases that allows for comparison between populations of different studies. In consequence, the development of a proper tool to assess the scientific labour in this area is compelling.

The GA-TA program is capable of adapting tables from an easy-to-build format in the format required for each of the mentioned softwares (Structure, Arlequin, and R). The code was completely developed in Python and is open source, free to download and capable of being extended to adapt to other formats. Moreover, the application can be executed online without requiring to download or install any packages. The program was developed for X-chromosome but it can be applicable to autosomic and Y-chromosome.

2 GA-TA:

2.1 Functionality description

You can run GA-TA in two ways: either remotely from your web browser (generally recommended) or locally on your computer (most likely for advanced users for development purposes).

The web application[1] provides a simple and easy-to-use graphical interface for managing input and output files. Running the web application doesn't involve the installation of any program (besides a web browser, which you probably already have) nor it requires to have advanced computing skills. You simply select and upload your spreadsheet in the appropriate format (see Sect. 2.3), choose the options that you need (R, Arlequin or Structure), click on the button *convert* and download the zip file containing the requested output file(s) in an .xlsx extension. The details of the output spreadsheets produced for each program is detailed in Sect. 2.2.

[1] http://gata.fcaglp.unlp.edu.ar/

If, instead, you wish to run the application locally on your computer, you should download the GA-TA code from https://github.com/santimda/GA-TA. The code is structured as follows:

- *main.py:* The main program which takes as an input the spreadsheet and the desired output formats. This program calls the functions *readtable.py* and *<program>_structure.py*, with *<program>=Arlequin,R*, and/or *Structure*.
- *readtable.py:* This program shapes an input .xls spreadsheet into a python dataframe using *pandas*.
- *<program>_structure.py:* This function transforms the input dataframe built by *readtable.py* into the required format for <program> (with *<program>=Arlequin,R*, and/or *Structure*).

To run GA-TA on your computer you will need to install *Python* and its *numpy* and *pandas* packages.

The program runs as:

```
$ python main.py [OPTIONS] spreadsheet_name
```

[OPTIONS]: The user can specify one, two or three letters (in any order) to obtain the desired output tables. These are: 'r' for R, 'a' for Arlequin, and 's' for Structure.

The spreadsheet must be either in Excel extension ('.xlsx') or in Open Office extension ('.ods').

Examples:

```
$ python main.py ra spreadsheet.xlsx
```

will return output files for R and Arlequin.

```
$ python main.py s spreadsheet.xlsx
```

will return output files for Structure.

2.2 Applications

2.2.1 Arlequin

Given that female individuals have a genotype and male individuals have a haplotype (in the case of X-Chromosome), for statistics purposes it is possible to convert data of two male individuals into data of one 'fake woman'. If the data sample has N_f female individuals and N_m male individuals, the total number of individuals considered will be $N_\mathrm{tot} = N_\mathrm{f} + N_\mathrm{m}/2$; if N_m is odd, then one row of missing data is added to complete an additional fake woman.

The program output are three tables for each program: one containing only the male samples, one containing only the female samples, and one containing the total population.

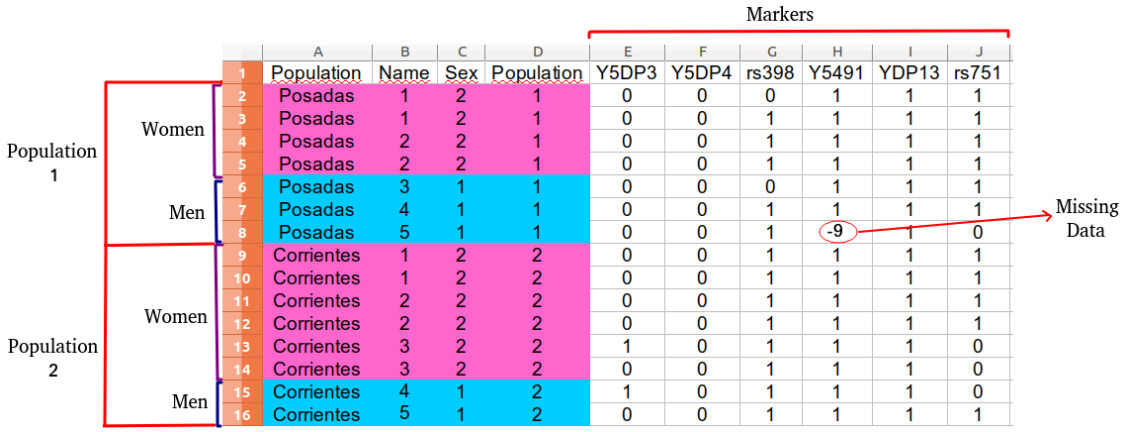
2.2.2 R

For each female individual, there are two pieces of data for each marker, which in the input table are located in two consecutive rows of the same column (Sect. 2.3). This program adds a letter 'A' or 'B' to the marker's name and places the two data values next to each other in the same row but in two separate columns. If the input table had initially a size of $n \times m$, the output table will have a dimension of $\frac{n}{2} \times (2m)$. Given that female individuals have a genotype and male individuals have a haplotype, for statistics purposes, it is possible to convert data of two male individuals into data of one fake woman. If the data sample has N_f female individuals and N_m male individuals, the total number of individuals considered will be $N_\mathrm{tot} = N_\mathrm{f} + N_\mathrm{m}/2$; if N_m is odd, then one column of missing data is added for each marker in order to complete an additional fake woman.

2.2.3 Structure

Data from female and male individuals are treated differently for adapting to Structure. For female individuals, a third row is added with values of 0.5. For male individuals, a second row with values of missing data (-9) is added and also a third column with values of 1.

The program output are three tables: one containing only males samples, one containing only the female samples, and one containing the total population.



	A	B	C	D	E	F	G	H	I	J
1	Population	Name	Sex	Population	Y5DP3	Y5DP4	rs398	Y5491	YDP13	rs751
2	Posadas	1	2	1	0	0	0	1	1	1
3	Posadas	1	2	1	0	0	1	1	1	1
4	Posadas	2	2	1	0	0	1	1	1	1
5	Posadas	2	2	1	0	0	1	1	1	1
6	Posadas	3	1	1	0	0	0	1	1	1
7	Posadas	4	1	1	0	0	1	1	1	1
8	Posadas	5	1	1	0	0	1	-9	1	0
9	Corrientes	1	2	2	0	0	1	1	1	1
10	Corrientes	1	2	2	0	0	1	1	1	1
11	Corrientes	2	2	2	0	0	1	1	1	1
12	Corrientes	2	2	2	0	0	1	1	1	1
13	Corrientes	3	2	2	1	0	1	1	1	0
14	Corrientes	3	2	2	0	0	1	1	1	0
15	Corrientes	4	1	2	1	0	1	1	1	0
16	Corrientes	5	1	2	0	0	1	1	1	1

Figure 1: Short version of the example table available at http://gata.fcaglp.unlp.edu.ar/.

2.3 Input table

In Table 1 we show an example of an input table format, whereas a complete example spreadsheet can be downloaded from http://gata.fcaglp.unlp.edu.ar/. The file must have a .xlsx or .ods extension.

The input table is built as follows. The first four columns give information about individuals and are coded such they can be easily completed using basic worksheet tools. Column 1 indicates the name of the populations (Posadas and Corrientes in the example); column 2 indicates the name of each individual (the number is repeated for women as two consecutive rows correspond to the same individual, whereas a single number is given for each men). Column 3 indicates the sex of each individual; if it is a female it is coded with number 2, whereas if it is a male it is coded with 1. Column 4 indicates the population coded by a number (in the example sheet there are two populations, Posadas coded as 1 and Corrientes coded as 2). The next columns are variables, one for each polymorphisms. These columns must have the name of the polymorphism in the first row. Alu sequences, SNP and INDEL markers should be coded as 0 or 1 while STR markers should be coded by the original number (imperfect alleles must be coded as a natural number). Missing data must be indicated as -9. **Each column must have its name in the first row** (as shown in Fig. 1): a blank cell in the first row of the worksheet will produce an error (if the column is not empty).

For Y-chromosome data, the user should define all the individuals as males while for autosomic DNA data the user should define them as females.

2.4 Extending functionalities

You can contact the GA-TA team with suggestions to extend the functionality of GA-TA to other programs and we will try to add such applications. Moreover, as the code is open source, users with basic knowledge of programming in python can also develop new applications on their own; thanks to the modularity of the GA-TA software, the programming work required is minimum. When developing new functionalities we strongly suggest to follow this steps:

1. Download (or preferably, clone) the GA-TA code from the Git reposirory https://github.com/santimda/GA-TA.
2. Establish the desired output table format.
3. Create a new application in the same folder as the others. Note that all applications have a common structure: a definition function, an output name function, and a table processing function; stick to this same structure in any new application to avoid inconveniences.
4. Add in the *main.py* program the references to the new application and the letter that should be used to call it.

References

[1] L. Excoffier and H. E. L. Lischer, *Arlequin suite ver 3.5: a new series of programs to perform population genetics analyses under linux and windows*, Molecular Ecology Resources, 10 (2010), pp. 564–567.

[2] M. J. Hubisz, D. Falush, M. Stephens, and J. K. Pritchard, *Inferring weak population structure with the assistance of sample group information*, Molecular Ecology Resources, 9 (2009), pp. 1322–1332.

[3] R. D. C. Team, *R: a language and environment for statistical computing*, 2008.