Adam Porter
Dept. Entomology & Graduate Program in Organismic & Evolutionary Biology
102 Fernald Hall
University of Massachusetts
Amherst, MA 01002-2410 USA
aporter@ent.umass.edu

This gives a short overview and reference for people using *IslandModelTest*. The test is described in detail in

> Porter, AH. A test for deviation from island-model population structure. *Molecular Ecology* (in review).

Please contact me if you wish to try these analyses before the manuscript above is published; I will be able to provide you with a copy of the manuscript.

General outline of what *IslandModelTest* does:

This program implements a parametric bootstrap to determine whether genotypic data show patterns that deviate significantly from Wright's neutral island model. Under the neutral island model, allele frequencies in different populations settle into an equilibrium shape described by a beta-distribution. The parameters of this beta-distribution are the allele frequencies of the pooled set of populations, and the standardized variance in allele frequencies, F_{ST}. Random samples of allele frequencies are repeatedly drawn from this beta-distribution, seeded using observed pooled allele frequencies and observed F_{ST}, following the same sampling pattern of the original data. The expected genotype frequencies are constructed from these allele frequencies using F_{IS}, the within-population inbreeding coefficient, because this also influences sampling patterns. In each replicate, genotypes are randomly sampled following the original data collection pattern, new values of F_{ST} are calculated for each locus and their distributions (due only to random sampling effects) are accumulated. The analysis will reveal which loci show significant deviations from the average, multilocus pattern.

This version requires genotypes from codominant markers scored in diploid individuals; future upgrades will implement haploid and dominant markers. Multiple alleles are supported.

The program uses data files formatted very similarly to Swofford's similar program BIOSYS. It also supports data in Arlequin format, which it recognizes by the suffix .arp.

The core algorithms have been checked against separate functions I have written in Mathematica, using entirely different language structure, and I have confidence in the output. Nevertheless, asserting that a program is bug-free amounts to accepting a null hypothesis, and I can't do that. So, please report any bugs you find.

What the program is not:

1) Proof that real populations follow island model assumptions. Even if you find no significant deviations from the Island Model using this test, you have not proven that the null hypothesis is true. Because of properties inherent in beta-distributions, and in testing fits to any distribution, a difference has to be pretty large to be detectable using this test. The program does, however, give feedback on the size of the deviation needed to get statistical significance.

2) Particularly user-friendly. The program is intended for specialists and is not as user-friendly as most window-based programs. It is critical to follow the grammatical conventions in setting up the data file, and even minor deviations in format can cause the program to mis-read the data. The program also does not run in the background on Macintosh machines, and large data sets may take a while to run (up to several days). If you quit in the middle of a run, the output is lost.

The program is likely to have some user-interface bugs, and these could be encountered by users who don't sweat the details while setting up their data file. I have tried to trap errors that I suspect people might make, and give hints as to what went wrong in the on-screen output. But, it is still possible to invent ways to make seemingly readable data that give complete nonsense. The program will create a new data file from its internal format after the data have been read in, and you can use this to check if the data are being read in correctly.

Setting up the data file:

> If you wish to use Excoffier's *Arlequin* format, please check his instructions for formatting (http://anthro.unige.ch/arlequin). In this case, your data file name must end with the extension '.arp', and the name cannot contain blanks.

The data file is set up very similarly to a BIOSYS file, perhaps familiar to people who have previously analyzed frequency data. Only in the first several lines are different, and there is no command structure at the end of the file. The program handles any number of individuals per population, and any number of populations, up to the memory capabilities of the user's machine.

In what follows, refer to the enclosed sample file <sampleGenotypeData>. The datafile name cannot contain blanks, and can end with or without extensions.

There are several lines at the beginning of the data file:

title line:

> This is the first line of the file. Make sure there are no lines accidentally preceding it. It can contain any characters.

```
The title line can contain any characters (spaces OK)
```

parameter line:

> This lines contains the parameter settings separated by commas (the numbers are for example only, you will set your own values)
>
> ```
> NPOP=3, NLOCI=10, MAXALL=7, NGROUPS=1
> ```
>
> These are:
>
> > `NPOP` is the number of populations you are presently analyzing. It can never exceed the number of populations in the data set. However, if `NPOP` is set to fewer populations that the data set contains, then only the first populations in the series will be analyzed. For example, if you have 10 populations in the data set, and set `NPOP=4`, then only the first 4 populations will be analyzed and the remainder will be ignored. This permits you to rearrange the populations to analyze particular subsets without accidentally deleting data.

NLOCI is the number of loci you have in the data set. Unlike NPOP, you must always specify all loci in the data set every time you run the program. If you want to drop particular loci from the analysis, it's easy, but you must use the SKIP function below to do it.

MAXALL is obsolete and you can put any value here. Originally it was used to set up arrays for data storage but this is handled automatically now. However, you cannot omit this.

NGROUPS is not used in IslandModeTest and you should just put 1 here. You cannot omit this.

The next line (or lines) contains the names of the loci.

There must be exactly as many names of loci as specified in the NLOCI= statement above. The order must be the same as in the individuals in the data below.

Each locus name should be separated by a comma.

The next line contains the SKIP= statement

SKIP tells the number of loci that you want to omit when doing a particular analysis. If you wish to analyze all loci, then you must set SKIP=0; you can't just omit this line from the file.

If you wish to omit loci from analysis, then set SKIP= the number of loci you wish to omit, put a comma after the number, and list these loci by name (in any order, but separated by commas). The names have to exactly match the list of loci, and the comparison is case-sensitive.

For example, say your 5 loci are encoded as follows on the locus name line:

```
AK,MDH-1,MDH-2,SOD,PGI
```

If you wish to analyze all loci, then set

```
SKIP=0
```

If you wish to omit, say, MDH-2, then set

```
SKIP=1,MDH-2
```

If you wish to omit both MDH-2 and AK, then set

```
SKIP=2,MDH-2,AK
```

The next block of lines contains a single population.

A population block starts with a line giving a population title. Each subsequent line describes a single individual. The final line of the block is the word NEXT, signifying the end of a population.

population title line

This can be of any length and can contain any characters. The last word of the population title can be used to identify group membership if hierarchical analyses are performed (not supported in *IslandModelTest*). I often use a taxonomic name, or a region name.

An example population title line:

```
2 mi S Great Falls MN  2.vii.92 subspecies vulgaris
```

The last word, 'vulgaris', will be used to group this population with any others marked 'vulgaris' if NGROUPS >1.

individual organism line:

The grammar of this line is quite important:

The first column is assumed to hold an individual identification code. This ID code can contain any characters except spaces, and is not otherwise used by the program. (BIOSYS requires the first 4 characters be a code for the population, but this length requirement is not enforced here). If you don't care about ID codes, you can put any character here as a placeholder. I often include the gender in the code for convenience. (If all ID codes are the same number of characters, the data will line up nicely in text-file format using Courier font.)

The remaining characters are allele codes, given in pairs and separated from other loci by a single space. The order must be exactly the same as that given in the locus names in the lines at the beginning of the file. Loci that are unscored for a particular individual must be replaced with blank spaces.

an example with 5 loci; the second individual is unscored at the 3rd locus.

```
001M AA AA BB CF DD

002F AA AA    FF CD
```

Note that EACH LOCUS MUST BE SCORED IN AT LEAST ONE INDIVIDUAL IN EACH POPULATION before the program can analyze that locus. If even one population is entirely unscored for a particular locus, then that locus (or that population) must be omitted from the analysis.

You can put additional comments or information after the genotype data, and it will be ignored. For example, morphological measurements from the same individuals can go on the same line to minimize cross-referencing. The line below is equivalent to the above for input purposes:

```
001M AA AA BB CF DD 23.76 3.04 1:16pm
```

the `NEXT` line

The word NEXT must be in uppercase characters, and should be the only 4 characters on this data line. This line tells the program to end data collection for this population, and if `NPOP` has not been exceeded, then begin reading in the data for the next population. Thus, the word `NEXT` must also be included after the last population, even though no "next" population actually exists.

Analyzing a data set:

The first step is to visually scan the data file for formatting errors. Common errors include

1) a blank line before the title line
2) populations wherein no individuals are scored for one or more loci
3) extra spaces between alleles, so that the columns are not aligned
4) forgetting that locus and allele names are case-sensitive (e.g., allele A≠a)

(To make sure spacing is correct, I like to create the files using Courier font, where each character is equal size. This way, the columns and spaces line up nicely for visual inspection. When I've got it to my satisfaction, I save the file as text.)

The next step is to save the data set as an ascii 'text' file. Files in MS Word or Excel format cannot be read.

Put the data file in the SAME FOLDER with the program on your desktop. If it's not in the same folder, the program won't be able to find it. The new output files will be placed in this same folder, too. Run it from your hard disk instead of from a floppy or zip disk.

Double-click on the program icon to start it.

Running the program:

The program will prompt you for relevant input, and do some simple tests to see if it understood you, before it reads in your data and processes it to your liking. At each prompt, type in the information and press <return> or <enter>.

? input file name =

This is the name of the file that holds your data. The file name must not have blanks in it (if yours does, rename it and start over). You have two options:

> *My format*: the file name is case sensitive and can be with or without an extension. It has to be saved in text file format.
>
> *Arlequin format:* the file name is case sensitive and has to have the extension '.arp'. It must contain codominant data.

? output file name =

This is the name of the newly created file that holds your output. The file name must not have blanks in it. This file will be created in the same folder as the program.

Do you want to use the program's default values? (Y/N)

Type Y for yes, unless you want to enter your own initialization parameters. If you type N, you get the following:

> **Enter the number of replicates used to make the null distributions. The default is: replicates = 1000.**
> **? replicates =**
>
> I don't recommend entering a number less than 1000 unless you're just playing around. The upper limit you choose depends on your patience, the size of your data set, and the speed of your machine.
>
> **Choose an integer between 1 and 32666 to seed the random number generator. Using the same seed twice gives identical results, useful if you want repeatability. If you enter 0, a unique seed will be chosen for you.**
> **? seed =**
>
> The seed you choose will appear in the output file. If you choose the same seed as a previous analysis, you will get an identical result.

<List of what you just entered>
Are these values OK? (Y/N)

Choose N to re-enter these values and Y to go on and read in the data, and start the test.

```
Are there any loci you would like to omit from the initial estimates
of Fst & Fis?
? Y/N
```

At this stage, you are estimating the parameters of the beta-distributions that determine the null hypothesis of your statistical test. A few moments later, once the null hypothesis is specified, you will draw random samples under the [statistical] assumption that the null hypothesis is true.

The first time you run this test on a data set, you should choose N.

If you choose Y, your null hypothesis will be based on a subset of the loci in your data set. For example, you may feel justified in removing a particular locus that you suspect might violate assumptions of the island model (perhaps you have independent evidence that it is under selection, for example). Here's what you get if you choose Y:

```
Type in the name of the locus exactly as it appears in the data
file. The locus name cannot have any spaces in it; e.g., "locus
1" should be changed to "locus1" in the original data file.  If
this is a problem, Quit now and make the necessary changes.
? locus name =
```

Enter names exactly as they appear in the data file, case-sensitive. You will be repeatedly prompted in case you want to remove more loci.

```
The null distributions will be calculated using Fst = xxxx and Fis =
yyyy.  However, you can estimate the null distributions of loci you
have previously omitted as well.
Are there any skipped loci you would like to restore to the data?
(Y/N)
```

This step controls which loci will be tested against the null hypothesis, using the Fst and Fis above. Often there are no additional loci to test and you would choose N. By choosing Y, you can restore loci you just removed in the step above, and even loci removed using the SKIP line in the data file.

You can walk away from the machine when you see these output lines on the screen:

```
All POPS rep: 1/1000
All POPS rep: 2/1000  time remaining 8:32:54
```

This is a gross estimate of how long your program will take in *hrs*:*mins*:*secs*, extrapolated from the time it takes to finish a single replicate. It could be significantly slower if your hard disk is short of memory, or if you haven't allocated enough operating system memory for this program. Each replicate updates its estimate of how long it will take.

At the end, tables are sent to the screen, and the exact same tables are sent to the output file. There is no reason to save the screen information when you quit — if you do, don't give it the name of your output file! The analysis will be finished when the screen reports:

```
All done.
```

There are 2 classes of output created by this program.

Screen output:

> The output on the screen is the "interactive" information of use while the program runs. It is not saved separately when the program quits, and there's no reason to, although you can save it if you want. All the results, along with key information for repeating the analysis exactly, including the data format and the random number seed, is included with the text file output.

Text file output:

> Most of the information from calculations is sent to text files with tabled information separated by tabs. They can be opened and read using word processors or spreadsheet programs; choose 'tab delimited' when opening.
>
> The main output file will have the name you assigned it, and will appear in the same folder as the program. If you choose the name of an existing file, the previous file will be overwritten; no user-friendly error checking is done to keep you from overwriting anything.

Output file contents:

Standard output includes:

- Random number seed: the integer used to initialize the random number generator. You can repeat the analysis exactly if you enter this number during the setup phase.
- A key to the populations included in the analysis.
- Three tables of estimates of F-statistics calculated using the Weir & Cockerham (1984) method:

  1) weighted averages taken over all loci and alleles, with jackknife error estimates and 95% ci around the Nm estimate (presuming that errors are normally distributed)

  2) weighted averages for each locus

  3) scores for each allele

- Frequency (null) distributions of resampled Fst scores for each locus. These can be copied into a statistical graphics program like DeltaGraph or Excel and plotted.
- Summary statistics of these resampled Fst distributions for each locus, including

  1) observed Fst scores (from the Fst table above)

  2) upper and lower 95% confidence limits obtained as percentiles of the null distribution

  3) the Lewontin-Krakauer test constant *k*, proportional to the variance of each distribution

  4) the deviation of the observed Fst value for each locus from the multilocus average Fst, measured as Nb. This is in units comparable to Nm, the gene flow rate.

  5) upper and lower 955 confidence limits of Nb obtained as percentiles of the null distribution.

6) The two-tailed significance level (*P*) of the observed Fst score compared to the null distribution.

- A large table of Fst scores of each replicate for each locus and the multilocus average. This table was crunched to generate the frequency distributions and summary statistics of the tables above.

Error messages:

It is possible that an error message will appear on the screen as the program runs. The most likely reason is that there is a formatting error in your data set, one that I didn't foresee when I wrote the data-reading algorithms. These could show up immediately as the file is being read, or potentially not until well into the analyses, depending on the details. An especially common one is a 'Bad index' message, indicating a discrepancy in an internal array. Check your data: make sure there are no loci unrepresented in populations, that the number of skipped loci is correct, that the scores for each locus line up in proper columns, etc. If you fix these and still get the error, contact me; you might have found a bug. You shouldn't trust any output that includes an error message. None of the error messages will be sent to the output file, just the screen.

It is possible that a file called 'errors.out' could be generated along with your output; this catalogs internal memory errors. You are likely to see this file if your program quits unexpectedly for some reason, or if you force an exit using the escape key, because these events bypass the proper steps for releasing internal memory. If you get this error file in any other context, I'm very interested to hear about it.

Geek stuff:

The program is written in C++ and compiled using MetroWerks CodeWarrior. Source code is available on my web site. I have compiled versions for Macintosh and Windows operating systems. There's no reason why you couldn't compile it yourself for a different OS, if you like to do this kind of thing, and don't mind porting the code over to a different compiler.

Equations that form the basis for my algorithms are developed or cited in the manuscript cited on the first page above.

Bugs:

I imagine a diversity of bugs will show up in the interface, mostly dealing with data formatting issues. Please tell me about them and I will try to help figure them out. Especially tell me if you find a bug in any of the analyses.