∫-LIBSHUFF

Introduction

∫-LIBSHUFF is a computer program that implements the Cramer-von Mises test statistic to answer the question, "are two libraries drawn from the same population and is one a subset of the other?". It builds upon work done by Singleton in the program LIBSHUFF. The primary differences between the two programs are summarized below:

1. ***Integral form of statistic:*** makes estimate more accurate, accelerates computational speed, and is more elegant than discrete form.
2. ***User can set number of iterations:*** allows the user to determine how precise p-values should be by increasing or decreasing value.
3. ***Multiple libraries compared in one execution:*** using one input distance matrix, all possible comparisons are made and the family-wise error rates are calculated to avoid misidentifying significant differences.
4. ***No preformatting required:*** A square or lower-triangular distance matrix in the phylip format is used as the input matrix.
5. ***Programmed in C++:*** Makes program considerably faster.

This manual is designed to achieve five goals:

1. Show how to use ∫-LIBSHUFF
2. Explain output file
3. Answer frequently asked questions

If you have any questions, complaints, or praise, please do not hesitate to contact Dr. Patrick D. Schloss at pds@plantpath.wisc.edu

How to Run ∫-LIBSHUFF

To compile s-libshuff in LINUX type the following in the folder with the makefile:

>make

∫-LIBSHUFF is run from the command line prompt. If you have a PHYLIP formatted "square" matrix then the following command will initiate the program assuming s-libshuff (or s-libshuff.exe) and mccaig.dist (the distance matrix) are in the same folder:

```
>s-libshuff mccaig.dist
```

Many people may wish to use ARB to align sequences and make phylogenetic trees. Using ARB's neighbor joining method, you can create and export a distance matrix. ARB generates a "lower triangular" matrix. If you use a distance matrix constructed in ARB the following option must be set (-l, "el"):

```
>s-libshuff -l mccaig.dist
```

The precision of the Monte Carlo procedure can be altered by increasing or decreasing the number of iterations that the sampling without replacement procedure is run. The default value is 10,000 but can be changed as follows:

```
>s-libshuff -i 10000 mccaig.dist
```

The biggest difference between ∫-LIBSHUFF and LIBSHUFF conceptually is the use of the integral statistic in ∫-LIBSHUFF. However, if you wish to use the discrete form described by Singleton et al. and used in LIBSHUFF you need to set the "-d" flag:

```
>s-libshuff -d mccaig.dist
```

The default step size for the discrete form of the statistic is 0.01 as it is implemented in LIBSHUFF. This step size can be changed with the "-z" flag:

```
>s-libshuff -d -z 0.001 mccaig.dist
```

∫-LIBSHUFF allows the user to seed the random number generator it uses to shuffle the libraries. The default seeds are 1234 and 5678. You can change these with the "-s1" and "-s2" flags:

```
>s-libshuff -s1 23431 -s2 9876 mccaig.dist
```

Finally, execution in Windows and Linux (and Mac OSX) is essentially the same. In Windows, you cannot merely double click on the icon to get the program to execute. You must use the "Command Prompt" program found by going Start -> Program Files -> Accessories -> Command Prompt. Then you must type in the path of ∫-LIBSHUFF and your distance file to execute the program:

```
C:\> "Documents and Settings\pds\Desktop\s-libshuff.exe" "Documents and
Settings\pds\Desktop\mccaig.dist"
```

Alternatively, you can change the root path to move to the desired directory and execute S-LIBSHUFF from there:

```
C:\PATH\> s-libshuff.exe mccaig.dist
```

Be forewarned that ∫-LIBSHUFF does not seem to run as quickly in Windows as it does in Linux and I would encourage everyone to align their sequences in ARB, which uses Linux or OSX, and to run ∫-LIBSHUFF in the same operating system.

Once ∫-LIBSHUFF is executed you will be prompted for the number of libraries in the distance matrix you used as the input file. For mccaig.dist, type "2".

Next, you will be prompted for the number of sequences in each library separated by a space. The number of sequences in each library must add to the number of libraries you set. For mccaig.dist, type "138 137".

Then the program will begin to churn and you will see the progress of the random iterations and some data for interpreting your results. Remember that because two pairwise comparisons are made for every comparison, it is essential that you correct for multiple comparisons.

Output Files

∫-LIBSHUFF produces one output file (*.coverage) that contains coverage data. In the default execution which uses the integral form, triplet columns containing the distance, Cx and Cxy values are presented. The numbers next to the "C" represent the library number starting at 0. If the "-d" flag is set then the first column will contain the distance, and the subsequent columns will contain the Cx followed by all possible Cxy coverages for that Cx.

Frequently Asked Questions

How do I cite ∫-LIBSHUFF?
Schloss, P.D., Larget, B.R. & Handelsman, J. 2004. Integration of microbial ecology and statistics: a test to compare gene libraries. *Applied and Environmental Microbiology*. **70**(9):5485-5492.

In windows, why does the command window open and close quickly when I double click on the ∫-LIBSHUFF icon in windows?
This is because you haven't given ∫-LIBSHUFF an input file and you will get an error message quickly followed by the screen closing. Please see the above section on how to run ∫-LIBSHUFF and remember that it must be run from a command prompt in windows.

I use XYZ program to construct distance matrices. Can I use ∫-LIBSHUFF?
We are anxious to help people use ∫-LIBSHUFF in a way that is easiest for them. Please contact me (pds@plantpath.wisc.edu) with an example distance matrix, and I will incorporate the format into ∫-LIBSHUFF.

Why doesn't ∫-LIBSHUFF do…?
If you would like to see something added to ∫-LIBSHUFF, please let me know (pds@plantpath.wisc.edu). It may take me a while to get around to implementing the feature, but I am generally reasonable. For example, providing the coverage data in an output file was implemented because people asked about it.