DQ - A python package for calculating multi-locus identity and linkage disequilibria

Version 1.0

Ganeshkumar Ganapathy
gg28@duke.edu

National Evolutionary Synthesis Center, Durham, NC

January 8, 2008

1 Formal Definitions

General multilocus linkage disequilibria (LD)[1] For SNP haplotype data from m haplotypes for n bi-allelic sites in the form of a $m \times n$ 0–1 matrix H, let H_{ij} be the entry at row i, column j. Definitions:

- For site j, let p_j be the frequency of haplotypes that are in state 1 at site j. The complementary frequency is $1 - p_j$.
- For any non-empty subset of sites B (called a *partition*), let $f_{B,1}$ be the frequency of haplotypes that are in state 1 at all the sites in B. Formally, it is: $\frac{|\{i \in [0, m-1] : H_{ij} = 1 \ \forall j \in B\}|}{m}$, where $[0, m-1]$ is the set of all natural numbers from 0 through $m-1$, both included, and assuming the haplotypes are labeled 0 through $m-1$. Frequency $f_{B,0}$ is analogously defined. Note that, with this notation, $p_j = f_{\{j\},1}$ and $1 - p_j = f_{\{j\},0}$.

With the above definitions, the multilocus disequilibrium $E_{H,B}$ for the partition B of the haplotype matrix H is defined as:

$$E_{H,B} = \frac{f_{B,1} - \prod_{j \in B} p_j + f_{B,0} - \prod_{j \in B}(1 - p_j)}{1 - \prod_{j \in B} p_j - \prod_{j \in B}(1 - p_j)} \tag{1}$$

Note that there are $2^n - 1$ multilocus measures.

Level-k and pairwise LD. In Equation 1, if there are k sites in the set B, then the disequilibrium computed is the level-k LD for the partition B. If $k = 2$, then the disequilibrium computed is the pairwise LD between the two sites in B. There are $\binom{n}{k}$ level-k disequilibria.

[1] For the theory, see references: [4, 3, 1, 2]

Identity disequilibria (IDD). When phased or unphased genotype, not just haplotype, data is available, identity disequilibria can be computed. For identity disequilibria, the state at a site j for a given chromosome represents whether the chromosome is heterozygous or homozygous at the site j. By convention we will let 1 represent heterozygosity and 0 homozygosity. Formally, suppose the genotype data is available as a $(2m) \times n$ 0–1 matrix G, with rows $2i$ and $2i+1$ representing the two haplotypes of the i-th among the m chromosomes (again, we assume the chromosomes are numbered 0 through $m-1$). Define a $m \times n$ 0–1 matrix G^* such that

$$G^*_{i,j} = \begin{cases} 0 & \text{if } G_{2i,j} = G_{2i+1,j} \\ 1 & \text{if } G_{2i,j} \neq G_{2i+1,j} \end{cases}$$

Then, a multilocus IDD $I_{G,B}$ for a partition B is defined to be the multilocus LD $E_{G^*,B}$ for the same partition B of the "zygosity" data matrix G^*.

2 Installing DQ-1.0

Disclaimer, rights etc. I am distributing DQ as is, without any guarantee whatsoever. Feel free to modify it and redistribute it. I'd appreciate it if you cite the original source if you do so.

Requirements. I developed DQ using python 2.4.4 on an Apple Mac running OS X version 10.4.11 on a 1.83 GHz Intel Core 2 Duo processor, with 2 GB main memory. It uses the python scientific computing package Numpy[2] version 1.0.4. So your system must be able to run Numpy 1.0.4 as well. Currently, I have not made any attempts to test if DQ runs on any other platform, nor have I made any attempt to test if it will run with earlier versions of python or Numpy.

Binary and source distributions. The current version of DQ is 1.0. Currently, I am distributing the entire python source code. Numpy must be installed "by hand" (see instructions below). To install the source distribution follow the steps below. The instructions in `teletype` font are to be typed at the command prompt.

1. Download and install Numpy, following the instructions below.
   (a) Download the latest numpy-1.0.4.gz source distribution from http://sourceforge.net/project/showfiles.php?group_id=1369&package_id=175103 to some directory dir.
   (b) `tar -xzf numpy-1.0.4.tar.gz`
      - Execute this at the command prompt in dir
      - This will create a directory named numpy-1.0.4 under dir.
   (c) `cd numpy-1.0.4`
   (d) `python setup.py install`
   (e) Also read the README.txt file in the same directory
2. Download DQ-1.0.tar.gz, and put it in any directory, say dir

[2] http://numpy.scipy.org

3. `tar -xzf DQ-1.0.tar.gz`

   - execute the above command at the command prompt in dir.
   - This will create a directory DQ-1.0.

4. `cd DQ-1.0`

   - All the source files reside in this directory. It also contains a a file called sample_input.txt, whose purpose should be evident. The README.pdf file that you are currently reading also resides in this directory.

3 Using DQ-1.0

DQ-1.0 installation contains a script `DQ.py`. For a non-python-programming end user who just wants to calculate disequilibria, you need just the script. The other files in the source directory are python modules used by the script. Future releases of DQ will make it possible for other python applications to directly access these modules.

Before we see how to use DQ, we define some terms in the following small glossary.

A small glossary. As pointed out earlier, there are a total of $2^n - 1$ partitions for which disequilibria can be computed, for an n-site haplotype or genotype data. DQ organizes these partitions based on levels and *inter-locus distances*. Partitions and levels have already been introduced, but here they are defined again along with inter-locus distances.

- **Partition.** A partition is a non-empty set of loci. By convention, we number the loci 0 through $n-1$ if there are n loci in total. So a partition is any non-empty subset of the set $Z_{n-1} = \{0, 1, 2, \ldots, n-1\}$. But by convention, we will write the loci in a partition in ascending order, such as $(r_1, r_2, \ldots, r_k)$, with each $r_i \in Z_{n-1}$.
- **Level.** A level-k partition is one that has k loci in it. Thus, there are $\binom{n}{k}$ level-k partitions.
- **Inter-locus distances**[3]**:** Given a partition $(r_1, r_2, \ldots, r_k)$, the *vector* of inter-locus distances is $[r_2 - r_1, r_3 - r2, \ldots, r_k - r_{k-1}]$. Conversely, a given vector $[d_1, d_2, \ldots, d_{k-1}]$ of inter-locus distances corresponds to (or induces) not just one but all level-k partitions $(r_1, r_2, \ldots r_k)$ such that $r_{i+1} - r_i = d_i$. For example, for $n = 10$, an interlocus distance vector [3] induces the set of partitions $\{(0,3), (1,4), (2,5), (3,6), (4,7), (5,8), (6,9)\}$; the distance vector $[2,4]$ induces the set of partitions $\{(0,2,6), (1,3,7), (2,4,8), (3,5,9)\}$. Note that for an inter-locus distance vector to make sense, all the distances in it must sum to at most $n-1$.

Using the script DQ.py. You can use DQ to do one of the following things: (a) compute all pairwise LDs given a haplotype matrix, (b) compute the LDs of all partitions that correspond to a given distance vector, (c) compute the LD of a given partition, and (d) all the above, computing ID disequilibria instead of LD. The script should be invoked as follows:

```
python DQ.py -hapfile <input file name> [optional_parameter1 <value>] [optional_parameter2
<value>] ...
```

[3] In this document [] is used for inter-locus distance vectors and () is used for partitions. These are just conventions adopted for this document, and the notation has no python significance. In particular, both partitions and inter-locus distance vectors are represented as python lists internally.

The optional parameters are as follows:

- `-missingdata` Specify the token that's used to represent missing data in your input data.
  - DQ handles missing data as follows: while calculating the linkage/ID disequilibrium for a partition, DQ ignores any row that contains missing data in one of the loci in the partition. Such rows are **not altogether discarded from the input matrix**. They are ignored only when they contain missing data in any locus that is in the partition for which disequilibrium is currently being computed.
- `-idd_ld` Specify either `idd` or `ld` depending on which disequilibrium you want computed.
  - If you specify `idd`, the input matrix will be construed to contain genotype data. For example, DQ will expect it to contain an even number of rows.
  - Similarly, if you specify `ld`, the input matrix will be construed to contain haplotype data.
  - The default value is `ld`.
- `-allpairs` Specify `yes` or `no`. Say yes if you just want to compute all pairwise disequilibria, and `no` otherwise. The default value is `no`.
  - The output in this case is a matrix of disequilibria. The rows and columns are labeled SNP-0 through SNP-$(n-1)$.
- `-output` Specify the name of the file where you want the output written. By default, the output will be written to `sys.stdout`.
- `-level` Specify what level disequilibria you want computed.
  - There are $\binom{n}{k}$ partitions at level k, but **not all the $\binom{n}{k}$ disequilibria are computed** for reasons of running time and "graspability" of the resulting output. Instead, the following scheme is used to generate approximately $n \log_2^{k-1} n$ partitions for which the disequilibria are computed:

    Let D be the set of distances $\{(n-1)/2, (n-1)/2/2, (n-1)/2/2/2, \ldots, 1\}$. The size of D is approximately $\log_2 n$. Note that D is **not** an inter-locus distance vector as defined in the glossary, but just a subset of the possible distances between two loci. Also, note that in general $x/2/2/2$ may not be equal to $x/8$ (the same applies to other fractions in the series). Then we generate a set of valid level-k inter-locus distance vectors by (a) first computing the *ordered* Cartesian product D^{k-1} of inter-locus distance vectors, and then (b) removing all vectors whose distances sum up to more than $n-1$. What results is approximately log_2^{k-1} inter-locus distance vectors. The ordered Cartesian product is similar to the Cartesian product, except we respect order: for example, $[2, 4]$ and $[4, 2]$ are treated as two different inter-locus distance vectors.

    Each of the valid inter-locus distance vectors generated as above gives rise to approximately n partitions. The disequilibria are computed for the resulting $n \log^{k-1} n$ partitions.
  - For the output, the computed disequilibria are binned based on the inter-locus distance vector. A single vector corresponds to about n disequilibria. A line in the output looks like: $[\mathrm{d}_1, \mathrm{d}_2, \ldots]$ $[\mathrm{diseq}_1, \mathrm{diseq}_2, \ldots]$

- `-dist` Specify an inter-locus distance vector **without the opening and closing square brackets**. This will compute disequilibria for all the partitions induced by the specified vector. The output is a single line that looks like a line from the `-level` output.
- `-partition` Specify a partition **without the opening and closing brackets**. This will compute the disequilibrium for the specified partition. The output is a single disequilibrium value.
- DQ will allow you to specify **only one** of the three optional parameters `-level, -dist` and `-partition`.
- If no options are provided, DQ defaults to `-allpairs yes`.

DQ should work for any sensible values for the parameters and any sensible combination of the parameters. I haven't made much of an attempt to find what it does for degenerate input.

Acknowledgments. Marcy Uyenoyama suggested several functions in DQ. Hao Mei and Yi-Ju Li provided datasets to test the program, and Hao Mei ran the program and helped greatly in debugging.

References

[1] F. B. Christiansen. Linkage Disequilibrium in Multi-locus Genotypic Frequencies with Mixed Selfing and Random Mating. *Theoretical Population Biology*, 35:307–336, 1989.

[2] J. Hein, M. H. Schierup, and C. Wiuf. *Gene Genealogies, Variation and Evolution*. Oxford University Press, 2005.

[3] B. Weir and C. C. Cockerham. Mixed Self and Random Mating at Two Loci. *Genetics Research*, 21:247–262, 1973.

[4] S. Wright. Systems of Mating I, II, III, IV, V. *Genetics*, 6:111–178, 1921.