Usage:

Run the LDblock binary using 1,000 Genomes binary PLINK data **split by chromosome**, as:

```
./ldblock [PLINK_PREFIX] [FLAGS]
```

The following flags are available, all are optional:

Flag	Default	Description
-frq [MAF]	0.01	MAF filtering threshold, only SNPs with MAF greater than that threshold are used to compute the (initial) break points (see also -refine below).
-win [NSNPS]	200	Correlation window in number of SNPs, correlations will be computed between SNPs that are at most this far apart in the data after MAF filtering (this is in SNP index, not genomic position; ie. by default, for a SNP it will compute the correlations with the 100 SNPs before and 100 SNPs after it)
-min-size [NSNPS]	1000	The minimum size in number of SNPs (after MAF filtering) a block must have
-min-prop [PROP]	0.1	The minimum size in number of SNPs (after MAF filtering) as a proportion of the parent block a block must have
-margin [METRIC]	0.01	Margin above the minimum metric value in a block to be split for other candidate break points to be considered equivalent; the most central of the equivalent break points is selected (see details below)
-max [METRIC]	0.25	Maximum metric value, blocks will only be split further if they contain a valid break point with metric value below this maximum
-refine [0/1]	1	If set to 1, after the break points have been computed from the MAF-filtered data, the base-pair position of the break point is further refined using any SNPs below the MAF threshold that are between the bounding SNPs (see details below)
-print-metric		If this is set, the metric values for all break points are printed to a separate output file
-out [PREFIX]	ldblock	The prefix used for output files

Details:

The program works as follows. First, the data is read in (filtering on MAF), and a thick diagonal SNP-SNP correlation matrix is computed. The thickness of the diagonal is determined by the -win parameter. This seems to work best at fairly low values like the default of 100 (at least at MAF = 5% filtering), because at larger values a lot more longer distance correlations are included. Since those are weaker, they tend to wash out the strong dependencies around the break point.

Starting from a single block encompassing all the (used) SNPs on the chromosome, the program splits that block into two, then splits the resulting blocks into two, and keeps splitting blocks until there are no blocks that can be split any further. Each pair of consecutive SNPs defines a potential break point, with a block being split in between the two consecutive SNPs (the “bounding SNPs”) for that break point.

The basic metric that is computed for each break point to evaluate where to split is defined as follows: if we split a block at a particular break point and were to analyse the two resulting subblocks (in eg. LAVA) separately, there is statistical dependency between those subblocks as a function of the LD of SNPs in one subblock with SNPs in the other subblock. Because the program only computes a thick diagonal SNP LD matrix, each break point essentially defines a triangular wedge of correlations between the two potential subblocks. The metric value is therefore computed by squaring all the correlations in that wedge and taking their mean value. This metric can therefore in principle range from 0 to 1, with higher values denoting stronger dependency between the subblocks.

When trying to split a block, the program therefore proceeds as follows. Firstly, based on the -min-size and -min-prop parameters, it determines the region within the block with potentially valid break points. If the block is smaller than twice the minimum size, no further splits will be applied to it.

Within the valid region of a block, the candidate break point with the lowest metric value is identified. If the lowest value exceeds the -max, the block is not split any further. If it is below the maximum however, if -margin is set to 0 the split is made at that break point. If -margin is not 0 however, all other candidate break points in the block with a metric value of at most 'minimum metric in block' + 'margin' are identified. The split is then made at the break point that is closest to the center (in number of SNPs) of the block. This is intended to encourage the final set of blocks to be more even in size.

The actual genomic position of the breaks is defined as follows. If the -refine option is turned off, or if there are no other SNPs between the two bounding SNPs of a selected break point, then the position of the break is computed as the average of the positions of the bounding SNPs. If there are additional SNPs between the bounding SNPs that were initially filtered out based on their MAF however, the -refine option will compute the local LD matrix (again a thick diagonal) without MAF filtering, defining a block around the two bounding SNPs. It will then evaluate all the candidate break points that lie between the two bounding SNPs to identify the one with the lowest metric value (no margin is applied here). It will then use the average of the bounding SNPs of this sub-break point as the genomic position of the original break.

Output:

The output generated by the program is a file with all the break points found, sorted by the position of the breaks. It contains the following columns. The RANK column gives the order in which the splits were made, which can be used to define a hierarchical set of blocks. Note that blocks are split in order of size, ie. when a block has been split into two subblocks, the two subblocks are returned to the collection of blocks to be split further, and then the largest block is taken from that collection to split next.

The METRIC and METRIC_MIN are the metric values at the break point. The METRIC value is the metric at the actual break point (disregarding the refining, that only affects the genomic position reported), METRIC_MIN is the metric value at the initially selected break point in the block that was being split. If the -margin parameter is set to 0, these will always be the same.

The INDEX_FILT and INDEX_ALL columns give the SNP index of the SNP right after the break. These are used to compute the sizes of blocks in number of SNPs. INDEX_FILT is the index of SNPs after MAF filtering, INDEX_ALL is the index of the SNP in the full input data file.

The POSITION column gives the genomic position of the break, computed as described above. The POS_LOWER and POS_UPPER columns give the positions of the two bounding SNPs which define the break point.

Note that the first and last entry in the file (RANK = 0) are not real breaks, but denote the lower and upper bounds of the input data. Also note that at present the program does not generate a log file, it only prints logging information to screen. It may be advisable to pipe the screen output to a log file for reference (NB: error messages are printed to stderr, not stdout).

Converting breaks to blocks

Some functions are provided in the accompanying .r file that can be used to turn the breaks into actual blocks. The load.breaks() function reads it in as a data frame, which can then be passed to the make.blocks() function to turn it into a data frame with block definitions. Note that the boundaries of these are assumed to be inclusive, ie. SNPs with base pair position equal to the start or stop position belong in the block.

The blocking solution produced can be modified by first applying the filter.breaks() function to the initial data frame, then pass the output to make.blocks() (note that applying filter.breaks() a second time on its own output is not well-defined and should be avoided).

As the name implies, doing this filters out is breaks rather than blocks, and as such the resulting blocks will always still be a full partitioning of the entire chromosome. With the max.metric and the two min.size filters, it checks if a break has a METRIC_MIN value greater than the specified maximum, or would result in any subblock with size below the specified minimum. If either is the case, the break is removed. The max.blocks setting is then applied after that, as needed reducing the number of breaks such as to result in no more than that number of blocks. This is based on the rank of the breaks, with earlier breaks (ie. rank closer to 1) being preferred.

Note that the max.metric and min.size filters here have virtually the same result as the -min-size and -max parameters in the main program; there are theoretically some small differences, but that should have little impact. Hence it is practical to run the program with more liberal thresholds for these, and then additionally filter them here to get a desired number / size of blocks etc. The min.size.all filter can only be applied here.

The max.blocks filter is present for convenience but in practice it is a bit artificial, as it just stops splitting blocks after the maximum is reached regardless of LD. The algorithm always selects the largest block to split next so the block sizes should not be too disparate unless the max.blocks value is set very low. But where the other parameters have an essentially constant interpretation across chromosomes, a given max.blocks setting will have quite different results in terms of block size and definition depending on the size of the chromosome, making it rather difficult to interpret.