Received: 16 February 2021 | Revised: 14 May 2021 | Accepted: 26 May 2021

DOI: 10.1111/1755-0998.13438

RESOURCE ARTICLE

MOLECULAR ECOLOGY RESOURCES WILEY

quickLD: An efficient software for linkage disequilibrium analyses

Charalampos Theodoris[1] | Tze Meng Low[2] | Pavlos Pavlidis[3] | Nikolaos Alachiotis[4]

[1]Technical University of Crete, Chania, Greece

[2]Carnegie Mellon University, Pittsburgh, USA

[3]Foundation for Research and Technology-Hellas, Heraklion, Greece

[4]University of Twente, Enschede, The Netherlands

Correspondence
Nikolaos Alachiotis, University of Twente, Enschede, The Netherlands.
Email: n.alachiotis@utwente.nl

Abstract

Software tools for linkage disequilibrium (LD) analyses are designed to calculate LD among all genetic variants in a single region. Since compute and memory requirements grow quadratically with the distance between variants, using these tools for long-range LD calculations leads to long execution times and increased allocation of memory resources. Furthermore, widely used tools do not fully utilize the computational resources of modern processors and/or graphics processing cards, limiting future large-scale analyses on thousands of samples. We present quickLD, a standalone and open-source software that computes several LD-related statistics, including the commonly used r^2. quickLD calculates pairwise LD between genetic variants in a single region or in arbitrarily distant regions with negligible memory requirements. Moreover, quickLD achieves up to 95% and 97% of the theoretical peak performance of a CPU and a GPU, respectively, enabling 21.5× faster processing than current state-of-the-art software on a multicore processor and 49.5× faster processing when the aggregate processing power of a multicore CPU and a GPU is harnessed. quickLD can also be used in studies of selection, recombination, genetic drift, inbreeding and gene flow. The software is available at https://github.com/pephco/quickLD.

KEYWORDS
computer program, high-performance software, linkage disequilibrium

1 | INTRODUCTION

Linkage disequilibrium (LD) is the nonrandom association between alleles at different genetic variants (Slatkin, 2008). It is widely employed in evolutionary biology and human genomics to provide insights into the genome structure of populations and the action of recent and strong positive selection (selective sweeps). Due to recombination, LD is mostly expected at nearby genetic variants (Koch et al., 2013). Using large-scale sequencing data, however, recent studies revealed that long-range LD is prevalent in the human genome (Park, 2019), which can provide insights into evolutionary forces at work, such as population admixture, epistatic selection and genetic drift. Yet, limited efforts have been reported on assessing LD patterns between variants separated by a large genetic distance (Koch et al., 2013; Park, 2019).

State-of-the-art software tools for linkage analyses, such as PLINK (Chang et al., 2015) and PopGenome (Pfeifer et al., 2014), compute LD-related summary statistics for all pairs of single nucleotide polymorphisms (SNPs) in one genomic region. This limitation restricts their applicability for long-range LD studies, leading to prohibitively long execution times and the need for enormous memory resources when large genomic regions are analysed. Furthermore, these tools do not fully exploit the processing power of multicore CPUs or accelerators, for example GPUs. To this end, we build upon prior work (Alachiotis et al., 2016; Binder et al., 2019; Theodoris et al., 2020) to present quickLD, a stand-alone software to efficiently compute a variety of LD statistics, such as D, D′ (Lewontin, 1964), and the commonly used r^2 (VanLiere & Rosenberg, 2008), for all SNP pairs in a single region or between arbitrarily distant genomic regions. Thus, in addition to long-range LD, quickLD can also be used to study factors that affect linkage disequilibrium in populations, such as selection, recombination, genetic drift, inbreeding and gene flow. This work extends our previous work (Theodoris et al., 2020) by combining both CPUs and GPUs into a unified high-performance framework, computing more LD-related statistics than the prior framework and providing additional capabilities such as the automated generation of heatmaps.

quickLD implements a sequence of data-transformation steps that allow the computation of LD between distant genomic regions with minimal memory requirements, irrespective of their genetic distance. Furthermore, quickLD computes LD as a matrix–matrix multiplication, allowing the exploitation of high-performance CPU/GPU implementations from dense linear algebra (DLA). To assess performance, we used various data sets with up to 100,000 sequences and up to 20,000 SNPs and compared processing times with PLINK (second-generation) (Chang et al., 2015), which, to the best of our knowledge, is the current state-of-the-art software for large-scale genome association studies. We observed between 3.2× and 12.2× faster execution than PLINK1.9 with one CPU core and between 6.1× and 21.5× faster execution with four CPU cores on a personal laptop. For the same runs, the GPU-accelerated version of quickLD run up to 49.5× faster than PLINK1.9. Both the CPU implementation and the GPU-accelerated version achieve more than 95% of the processor's theoretical peak performance thanks to leveraging lessons learnt from high-performance dense linear algebra to design efficient LD kernels for CPUs and GPUs.

2 | MATERIALS AND METHODS

2.1 | Implementation

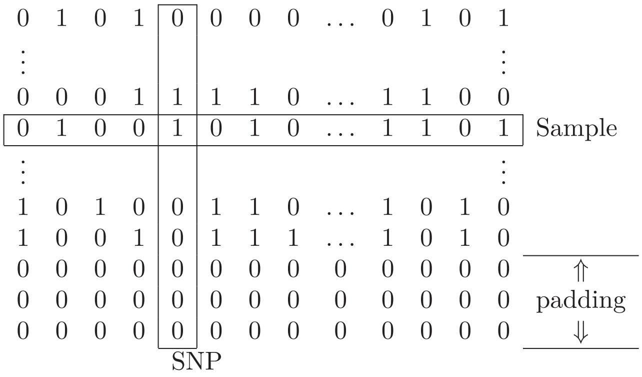
quickLD is an open-source C code for Linux environments. The CPU implementation is based on the de facto algorithm for high-performance matrix–matrix multiplication on multicore CPU systems and utilizes the BLAS-Like Instantiation Software (BLIS) (Van Zee & Van De Geijn, 2015) for high-performance DLA. The accelerated version extends the work in BLIS to computing LD on the GPU (Binder et al., 2019).

2.2 | Input/output

A quickLD command specifies a list of variant call format (VCF) input files (compressed files are also supported) and a list of genomic region pairs to be combined (regions in different VCF files can be paired as well). Optionally, a list of samples to be included in the analysis can be provided, as well as the number of CPU cores and/or GPUs to be deployed. By default, quickLD calculates r^2, applying a cut-off threshold of $r^2_{\text{th}} = 0.2$ to reduce the output report size; the threshold value can be defined by the user. quickLD handles missing data using a probabilistic genotype imputation method that operates on a per SNP basis. A random seed can be used to estimate how much scatter this method introduces; the random seed can be defined by the user. Optional parameters can be used to schedule large-scale calculations on personal computers with limited processing and memory resources. quickLD generates a report per region pair (PLINK format) that comprises all pairwise LD scores for all possible pairs of SNPs. The tool can also graphically display the results in the form of a heatmap.

2.3 | Computational workflow

quickLD separates data accessing from processing to improve performance and reduce the overhead of accessing large input files. A large VCF file is initially split into smaller VCF files that can be easily managed and processed on personal computers. Thereafter, a number of genomic region pairs are loaded from the input list, based on the amount of available memory in the system. Each pair of regions corresponds to a processing task, and all loaded tasks are processed in parallel using a scheduling algorithm that distributes tasks to the CPU cores and/or GPUs. When all the loaded tasks have been completed, the process continues by loading the next set of genomic region pairs (tasks), using the previously allocated memory. The above steps are repeated until all the region pairs have been processed, with every new set of tasks using the same memory space. This allows large-scale analyses to be conducted on off-the-shelf workstations since the memory requirements do not increase with the number of genomic regions pairs.



$$\begin{array}{cccc|c|ccccccccl}
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & \dots & 0 & 1 & 0 & 1 & \\
\vdots & & & & & & & & & & & & \vdots & \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & \dots & 1 & 1 & 0 & 0 & \\
\hline
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & \dots & 1 & 1 & 0 & 1 & \text{Sample} \\
\hline
\vdots & & & & & & & & & & & & \vdots & \\
1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & \dots & 1 & 0 & 1 & 0 & \\
1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & \dots & 1 & 0 & 1 & 0 & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Uparrow \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{padding} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Downarrow \\
 & & & & \text{SNP} & & & & & & & & &
\end{array}$$

FIGURE 1 Example of the genomic matrix *G*. Each row represents a sample while each column describes a SNP. The allocated memory per SNP is a multiple of 64 bits. This is achieved through padding by adding extra zeros to the end of each SNP. Adapted from (Alachiotis & Weisz, 2016)

TABLE 1 Performance comparison (throughput and speedup) between quickLD (C denotes CPU execution; G denotes GPU execution) and PLINK for increasing number of samples when one region and a pair of regions are processed. The region size is 5000 SNPs. Processing one region entails the calculation of 12.5 × 10^6 LD scores, while processing a region pair leads to the calculation of 25 × 10^6 LD scores

	Single region					Pair of regions	
	Throughput (LD × 10^6/s)			Speedup (×) over PLINK 1.9		Throughput (LD × 10^6/s)	
Samples (×10^3)	quickLD_C	quickLD_G	PLINK	quickLD_C	quickLD_G	quickLD_C	quickLD_G
2.5	16.53	36.55	5.14	3.21	7.11	36.76	73.10
10	11.44	27.96	1.36	8.44	20.63	14.34	55.93
20	6.60	21.48	0.69	9.61	31.25	10.35	42.96
30	4.87	17.58	0.46	10.48	37.88	5.53	35.16
40	3.79	14.76	0.35	10.75	41.88	4.23	29.52
50	3.06	13.51	0.28	10.87	48.00	3.39	27.03
60	2.60	11.19	0.23	11.11	47.79	2.86	22.38
70	2.26	8.80	0.20	11.43	44.48	2.46	17.59
80	1.98	8.08	0.17	11.56	47.22	2.17	16.16
90	1.75	7.49	0.15	11.57	49.44	1.92	14.99
100	1.61	7.01	0.14	11.69	51.06	1.74	14.03

TABLE 2 Performance comparison (throughput and speedup) between quickLD (_C denotes CPU execution, _G denotes GPU execution) and PLINK for increasingly larger regions in terms of SNPs size when a single SNP region and a pair of regions are processed. The sample size is 100,000 sequences

	Single region						Pair of regions		
		Throughput (LD × 10^6/s)			Speedup (×) over PLINK			Throughput (LD × 10^6/s)	
SNPs (×10^3)	LD scores (×10^6)	quickLD_C	quickLD_G	PLINK	quickLD_C	quickLD_G	LD scores (×10^3)	quickLD_C	quickLD_G
1	0.50	1.14	2.07	0.14	8.45	15.35	1.00	1.43	4.15
2	2.00	1.39	3.68	0.14	9.97	26.29	4.00	1.60	7.35
3	4.50	1.51	4.96	0.14	11.04	36.27	9.00	1.67	9.92
4	8.00	1.56	6.32	0.14	11.13	44.98	16.00	1.70	12.64
5	12.50	1.59	7.12	0.14	11.59	51.71	25.00	1.73	14.24
6	18.00	1.62	7.95	0.14	11.57	56.81	36.00	1.75	15.89
7	24.50	1.64	8.44	0.14	11.86	60.91	49.00	1.76	16.88
8	32.00	1.67	8.97	0.14	12.01	64.52	64.00	1.77	17.95
9	40.50	1.68	8.60	0.14	12.05	61.55	81.00	1.77	17.19
10	50.00	1.69	9.25	0.14	12.19	66.91	100.00	1.77	18.51

2.4 | Casting LD as a matrix–matrix multiplication

The key idea behind quickLD is that LD can be computed as a matrix–matrix multiplication with only ones and zeros. This formulation of the LD computation allows us to leverage decades worth of knowledge in the high-performance dense linear algebra domain to develop efficient LD implementations. This section provides a succinct summary of the underlying idea that was previously described for CPUs (Alachiotis et al., 2016) and GPUs (Binder et al., 2019).

Each SNP is represented by a vector of ones and zeros. Multiple SNPs in a genomic region can then be described as a $N_{seq} \times k$ matrix, G, where N_{seq} is the sample size and k is the number of SNPs. Each row in the matrix represents the alleles of a sample while each column represents a single SNP. Exploiting the fact that values in the genomic matrix G consist of only ones and zeros, we can reduce the storage of G by using a single bit to represent an allele. This means that each SNP is represented by a group of N_{int} unsigned integers (64-bit long each), with N_{int} defined as follows:

$$N_{int} = \left\lceil \frac{N_{seq}}{64} \right\rceil . \tag{1}$$

An example of matrix G is illustrated in Figure 1. We refer to G as the genomic matrix. In our formulation of G, we ensure that the allocated memory per SNP is a multiple of 64 bits by adding extra zeros

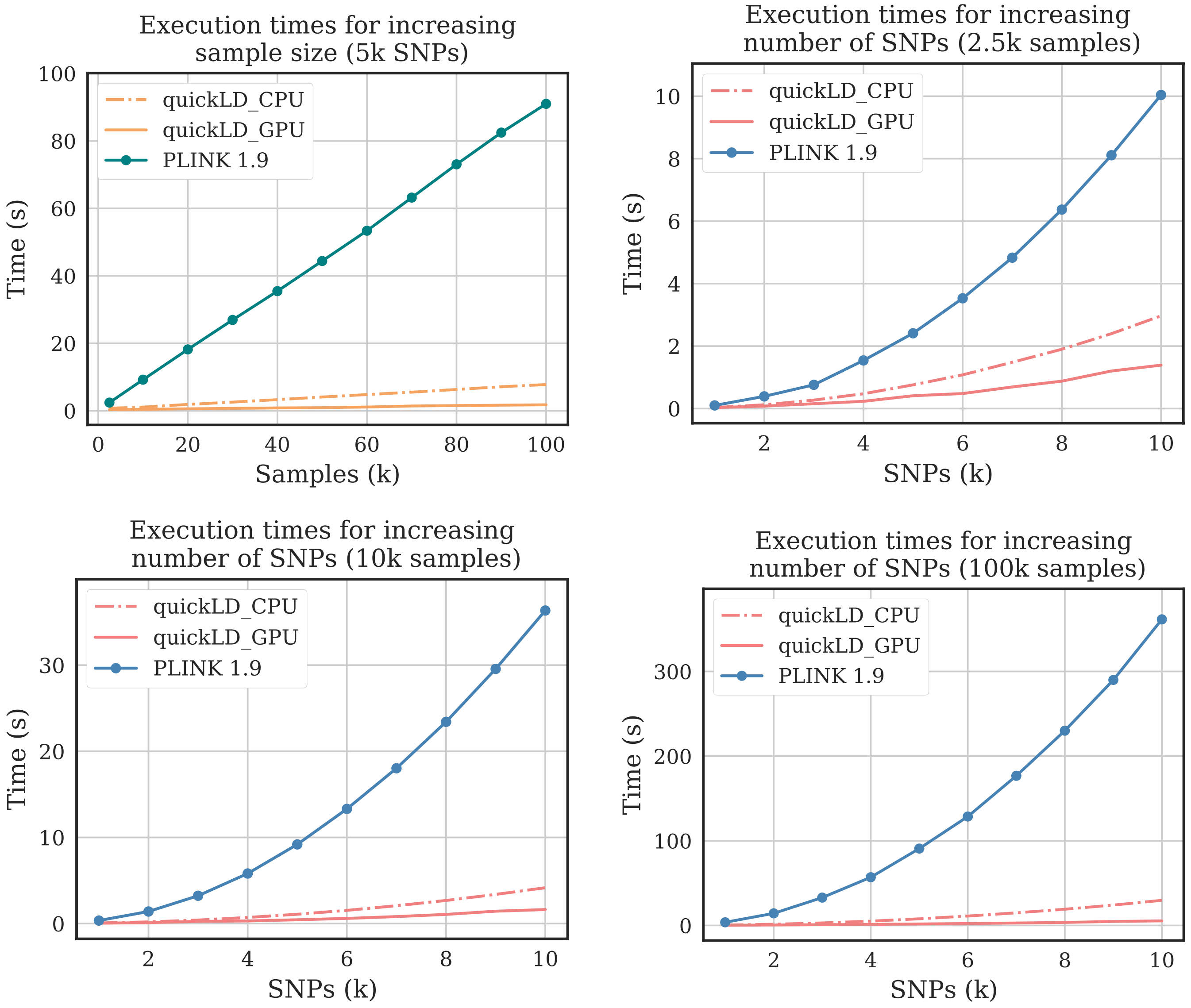


FIGURE 2 Execution times for increasing number of samples (top left) and increasing numbers of SNPs when the sample size is 2500 sequences (top right), 10,000 sequences and 100,000 sequences (bottom row)

to the end of each SNP, referred to as 'padding' in Figure 1. Different genomic regions are differentiated via subscripts (e.g. G_1 and G_2).

The desired haplotype frequency matrix, $H_{i,j}$, that is used to compute LD can be computed from the genomic matrices G_i and G_j as follows:

$$H = \frac{1}{N_{\mathrm{seq}} G_i^T G_j}. \qquad (2)$$

Equation 2 computes the haplotype frequencies and stores them in matrix H. The above formulation of H is a specific form of the general matrix multiplication (GEMM) operation ($C = \alpha AB + \beta C$) in the level 3 basic linear algebra subprograms (BLAS3), a widely used set of dense linear algebra routines. Here, $\alpha = 1/N_{\mathrm{seq}}$ and $\beta = 0$. Using our formulation of the genomic matrix, one can substitute the dot product by counting the total number of ones (set bits) remaining after a pairwise logical AND (&) operation is performed. This counting of set bits can be performed using the intrinsic POPCNT operation. Thus, the haplotype frequencies can be computed as follows:

$$P_{ij} = \frac{1}{N_{\mathrm{seq}}} \mathrm{POPCNT}(s_i \,\&\, s_j). \qquad (3)$$

Thereafter, we can compute matrix D that contains LD values by subtracting the product of the allele frequencies from H as shown in Equation 4:

$$D = H - P_i P_j^T \qquad (4)$$

where P_i and Pj are vectors of per SNP alleles frequencies that are derived from the genomic matrices G_i and G_j, respectively.

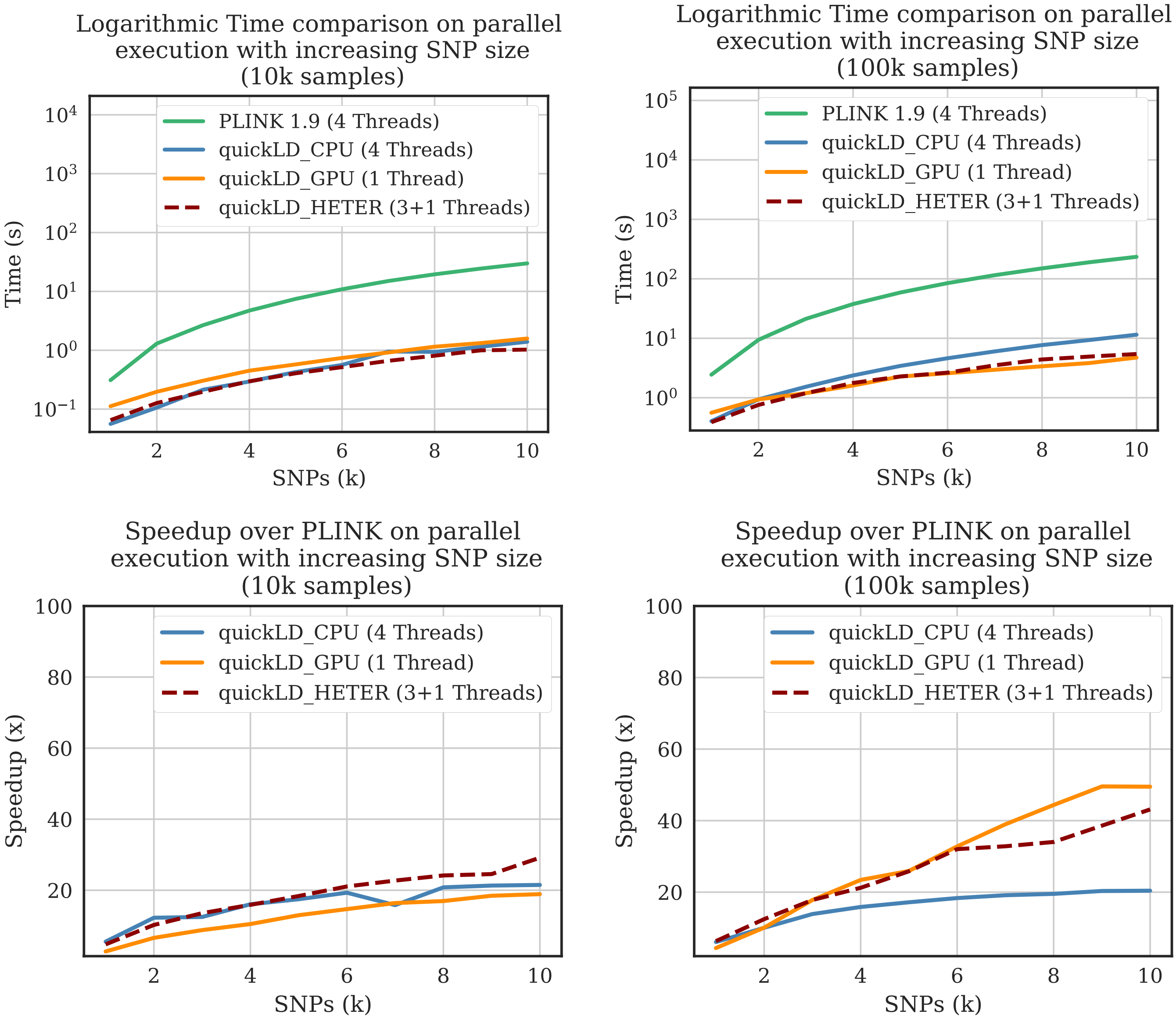


FIGURE 3 Parallel performance comparison between quickLD and PLINK1.9 in terms of execution times for 10,000 samples (top left) and 100,000 samples (top right), as well as respective speedups (bottom row). 'HETER' refers to heterogeneous execution using both the CPU and the GPU. Note the logarithmic scale for the execution times

3 | RESULTS AND DISCUSSION

3.1 | Experimental setup

We employed a personal laptop with an Intel Core i5-8300H processor with 4 cores running at 2.3 GHz (8GB of main memory) and an Nvidia GTX 1050-M GPU with 640 Cuda cores running at 1.3 Ghz (4GB of main memory). To evaluate performance, we compare quickLD with PLINK 1.9 (Chang et al., 2015), the most widely used software for linkage analyses, to the best of our knowledge. We report performance in terms of execution times and throughput. Note that we do not provide a comparison of memory consumption between quickLD and PLINK1.9 because such a comparison would not be fair since these tools handle memory in a fundamentally different way: quickLD allocates the necessary amount of memory given the data to be processed and the degree of parallelism requested by the user. PLINK1.9 follows a system-driven memory allocation approach that attempts to allocate half of the available memory of the system, and if this is not possible, then less memory is allocated, irrespective of the input data size. More information about the PLINK memory allocation scheme can be found at: https://www.cog-genomics.org/plink/2.0/other.

3.2 | Sequential execution

quickLD and PLINK differ in the way they calculate pairwise LD scores. While PLINK processes one region (one input file) and computes a diagonal matrix with pairwise LD scores, quickLD switches between different variants of matrix–matrix multiplication for

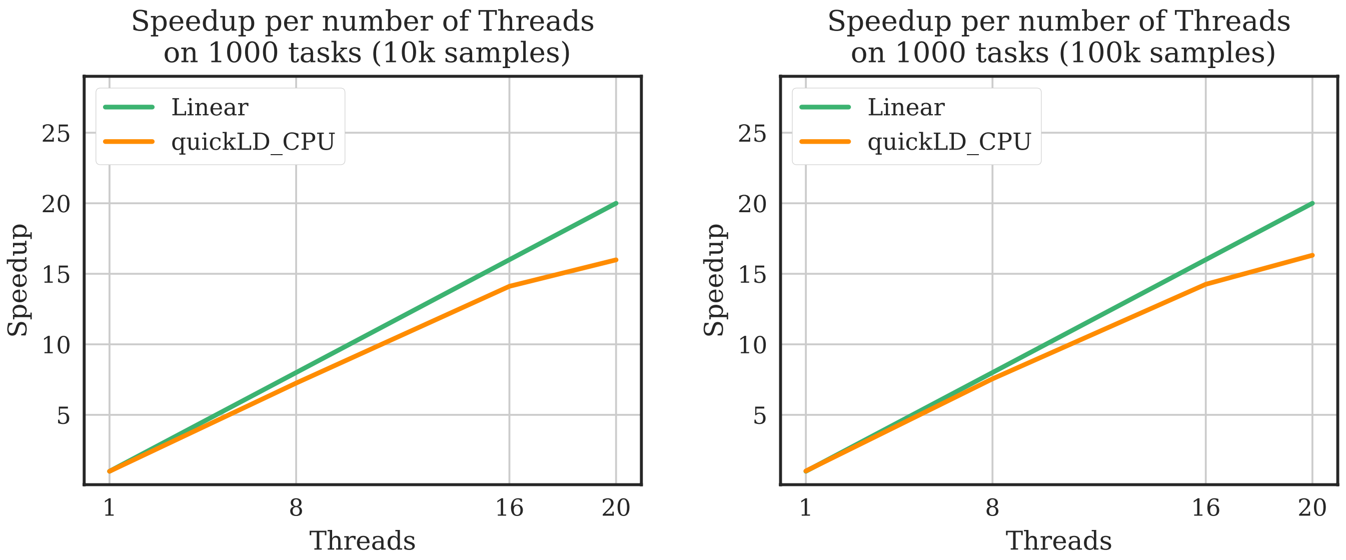


FIGURE 4 quickLD speedups for up to 20 CPU cores/threads when processing 1000 tasks (arbitrarily sized regions between 1000 and 10,000 SNPs) with 10,000 sequences (left) and 100,000 sequences (right)

efficiency purposes, depending on whether LD is computed between two different regions or within a single region. When computing with multiple regions, quickLD casts the computation as a general matrix–matrix multiplication (gemm). For single-region LD computations, quickLD switches to a symmetric rank-k update (syrk) routine where only the upper (or lower) half of the matrix is computed. The observation here is that the output matrix is symmetric when computing LD on the same region. Switching to a different variant allows quickLD to avoid redundantly computing the same LD scores.

We measured performance in terms of throughput for an increasing number of samples (Table 1) and number of SNPs (Table 2). We distinguish between processing one region and a region pair. As can be observed in the tables, quickLD is between 3.2× and 11.7× faster than PLINK1.9 when the sample sizes increase from 2500 sequences to 100,000 sequences and between 8.4× and 12.2× faster when the region size increases from 1000 SNPs to 10,000 SNPs. When quickLD deploys a GPU accelerator, it becomes between 7.1× and 51× faster than PLINK1.9 (which executes on the CPU) as the number of samples increases, and between 15.4× and 66.9× faster when the number of SNPs increases.

Figure 2 illustrates quickLD and PLINK1.9 execution times when the sample size increases up to 100,000 sequences (Figure 2A), and when the number of SNPs increases up to 10 k for fixed sample sizes of 2,5 k (Figure 2B), 10 k (Figure 2C) and 100 k (Figure 2D). Expectedly, execution times increase linearly with the number of samples and quadratically with the number of SNPs.

3.3 | Parallel execution

To assess performance of the parallel execution of quickLD over PLINK1.9, we compare execution times and report speedups when data sets with 10,000 and 100,000 sequences, with an increasing number of SNPs (up to 10,000), are processed. The results illustrated in Figure 3 show speedups up to 21.4× for the parallel guickLD CPU execution and up to 49.5× when the GPU is deployed. When the aggregate CPU/GPU performance is used by quickLD, overall throughput performance improves by up to 35%.

An important aspect of the parallel quickLD performance is the way it scales with the number of CPU cores. To assess this, we analysed 1000 arbitrarily sized, in terms of SNP size, region pairs, deploying up to 20 CPU cores/threads. For these runs, we employed a 20-core Intel Xeon E5-2660v3 processor running at 2.6 GHz (32 GB of main memory). The SNP size varied between 1000 and 10,000 SNPs. As can be observed in Figure 4, quickLD speedups are nearly linear for up to 16 cores, achieving 16.8× faster execution when deploying all 20 CPU cores.

3.4 | Application on real data

3.4.1 | Human Chromosome 22

Using a data set from the 1000 Genomes Project (1000 Genomes Project Consortium et al., 2015) (downloaded from http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/), we tested the performance of quickLD against PLINK 1.9 in an arbitrarily chosen region of chromosome 22. There are 106,730 SNPs in the chosen region (bp positions from 16050075 to 20128290). The report threshold was set to 0.8, which reduced the report file size to 30 MB and allowed for a more accurate comparison of processing times. Using 4 CPU threads, PLINK 1.9 required 429 s, whereas quickLD, also using 4 CPU threads, finished in 69.72 s. When quickLD employed a GPU in addition to the 4 CPU threads, the execution time was reduced to 51.76 s. Note that in this case,

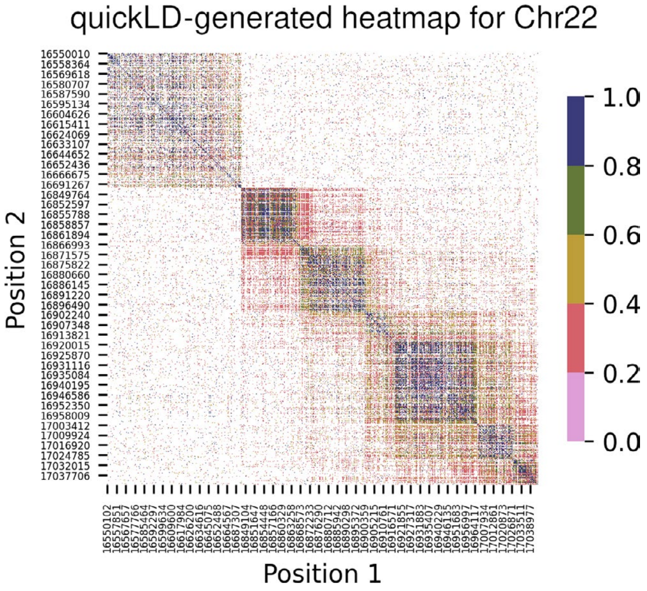


FIGURE 5 A quickLD-generated heatmap of all pairwise LD scores for 10,000 SNPs in the region 16,050,075 bp–17,050,075 bp of human chromosome 22 (2504 samples, human genome assembly GRCh37). The depicted region contains LD scores in the range of 0.2 to 1

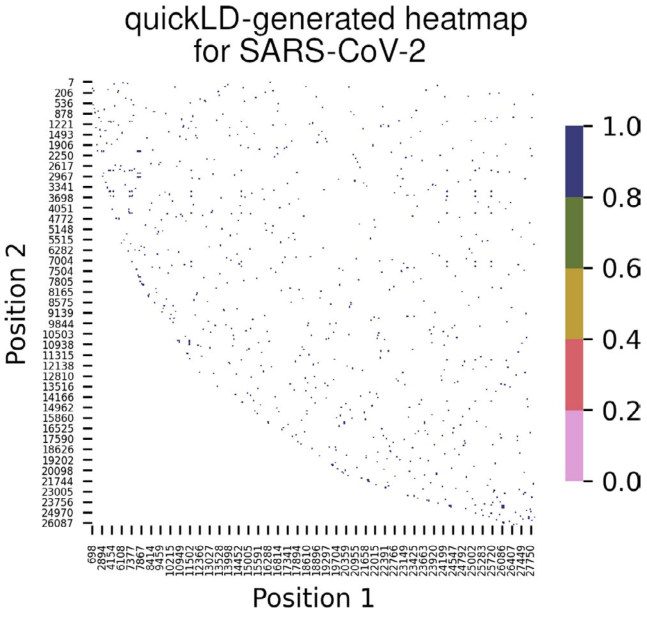


FIGURE 6 A quickLD-generated heatmap of all pairwise LD scores for 5730 variants obtained from the analysis of 22,554 SARS-CoV-2 whole genomes. The depicted region contains LD scores in the range of 0.8 to 1

1 CPU thread is exclusively used for controlling the GPU, thus leaving 3 CPU threads available for processing.

quickLD took 69.72, 91 and 51.76 s using 4 CPU threads, 1 GPU thread and the entire system (3 CPU threads, since one thread is dedicated to control the GPU, and the GPU), respectively. quickLD is effectively up to 8.3× faster than PLINK 1.9, using the entire system (heterogeneous CPU/GPU execution). Figure 5 illustrates a heatmap of a subregion of this data set, using the visualization capabilities of quickLD.

3.4.2 | SARS-CoV-2 Genomes

We employed 39,941 high-coverage SARS-CoV-2 genomes from the GISAID database (https://www.gisaid.org/). We used sequences with length equal or larger than 29,000 base pairs and trimmed ambiguous states (N) from the beginning and the end. We also excluded sequences that contained more than eight Ns and used the experimental MAFFT version (Katoh & Standley, 2013) for closely related viral genomes for multiple sequence alignment (FASTA format). The resulting data set, after applying the aforementioned filters to discard sequences, comprised 22,554 genomes. The snp-sites (Page et al., 2016) software was used for converting the FASTA file to a VCF file, and we invoked the tool's built-in option to discard columns that did not contain A, C, G, T, exclusively. The resulting data set comprised 5730 variants. PLINK 1.9 executed in 12.8 s, while quickLD ran in 0.870, 0.630 and 0.475 s using 4 CPU threads, the GPU and the entire system (3 CPU threads and the GPU), respectively. The maximum speedup of quickLD over PLINK 1.9 was 26.8×, which was observed when the aggregate system performance of the multicore CPU and the GPU accelerator was exploited. Figure 6 illustrates the quickLD-generated heatmap of the LD scores.

ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers and the editor for their comments that helped us to improve the clarity of this manuscript.

AUTHOR CONTRIBUTIONS

C.T. and N.A. designed the software; C.T. implemented the software; T.M.L. implemented the CPU/GPU kernels; P.P. tested the tool and analysed the real data; C.T., T.M.L. and N.A. wrote the manuscript.

DATA AVAILABILITY STATEMENT

The source code of quickLD, run scripts, execution instructions and user manual are available through the quickLD GitHub repository: [https://github.com/pephco/quickLD]. The first release of quickLD is archived at: [https://doi.org/10.6084/m9.figshare.14431268] (Theodoris et al., 2021).

ORCID

Pavlos Pavlidis https://orcid.org/0000-0002-8359-7257
Nikolaos Alachiotis https://orcid.org/0000-0001-8162-3792

REFERENCES

1000 Genomes Project Consortium, Auton, A., Brooks, L. D., Durbin, R. M., Garrison, E. P., Kang, H. M., Korbel, J. O., Marchini, J. L., McCarthy, S., McVean, G. A., & Abecasis, G. R. (2015). A global reference for human genetic variation. *Nature*, *526*(7571), 68–74.

Alachiotis, N., Popovici, T., & Low, T. M. (2016). Efficient computation of linkage disequilibria as dense linear algebra operations. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* (pp. 418–427). IEEE.

Alachiotis, N., & Weisz, G. (2016). High performance linkage disequilibrium: FPGAs hold the key. In *Proceedings of 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (pp. 118–127).

Binder, E., Low, T. M., & Popovici, D. T. (2019). A portable GPU framework for SNP comparisons. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* (pp. 199–208). IEEE.

Chang, C. C., Chow, C. C., Tellier, L. C. A. M., Vattikuti, S., Purcell, S. M., & Lee, J. J. (2015). Second-generation PLINK: Rising to the challenge of larger and richer datasets. *Gigascience*, *4*(1), s13742–015.

Katoh, K., & Standley, D. M. (2013). MAFFT multiple sequence alignment software version 7: Improvements in performance and usability. *Molecular Biology and Evolution*, *30*(4), 772–780.

Koch, E., Ristroph, M., & Kirkpatrick, M. (2013). Long range linkage disequilibrium across the human genome. *PLoS One*, *8*(12), e80754.–https://doi.org/10.1371/journal.pone.0080754

Lewontin, R. C. (1964). The interaction of selection and linkage. I. General considerations; heterotic models. *Genetics*, *49*(1), 49–67. https://doi.org/10.1093/genetics/49.1.49

Page, A. J., Taylor, B., Delaney, A. J., Soares, J., Seemann, T., Keane, J. A., & Harris, S. R. (2016). SNP-sites: rapid efficient extraction of SNPs from multi-FASTA alignments. *Microbial Genomics*, *2*(4), e000056. https://doi.org/10.1099/mgen.0.000056

Park, L. (2019). Population-specific long-range linkage disequilibrium in the human genome and its influence on identifying common disease variants. *Scientific Reports*, *9*(1), 1–13. https://doi.org/10.1038/s41598-019-47832-y

Pfeifer, B., Wittelsbürger, U., Ramos-Onsins, S. E., & Lercher, M. J. (2014). PopGenome: An efficient Swiss army knife for population genomic analyses in R. *Molecular Biology and Evolution*, *31*(7), 1929–1936.

Slatkin, M. (2008). Linkage disequilibrium—Understanding the evolutionary past and mapping the medical future. *Nature Reviews Genetics*, *9*(6), 477–485. https://doi.org/10.1038/nrg2361

Theodoris, C., Alachiotis, N., Low, T. M., & Pavlidis, P. (2020). qLd: High-performance computation of linkage disequilibrium on cpu and gpu. In *2020 IEEE 20th International Conference on Bioinformatics and Bioengineering (BIBE)* (pp. 65–72). IEEE.

Theodoris, C., Low, T. M., Pavlidis, P., & Alachiotis, N. (2021). *quickLD: An efficient software for linkage disequilibrium analyses*. figshare, Software. https://doi.org/10.6084/m9.figshare.14431268

Van Zee, F. G., & Van De Geijn, R. A. (2015). BLIS: A framework for rapidly instantiating BLAS functionality. *ACM Transactions on Mathematical Software (TOMS)*, *41*(3), 1–33.

VanLiere, J. M., & Rosenberg, N. A. (2008). Mathematical properties of the r2 measure of linkage disequilibrium. *Theoretical Population Biology*, *74*(1), 130–137.

How to cite this article: Theodoris, C., Low, T. M., Pavlidis, P., & Alachiotis, N. quickLD: An efficient software for linkage disequilibrium analyses. *Molecular Ecology Resources*. 2021;21:2580–2587. https://doi.org/10.1111/1755-0998.13438