kalis - An R Package for Quick Local Relatedness Inference & Probabilistic Haplotype Screening

Ryan Christ[1,2], Louis Aslett[2,3], David Steinsaltz[4], Xinxin Wang[5], Chris Holmes[2,4], Chris Spencer[6], Ira Hall[5]

[1]McDonnell Genome Institute, Washington University School of Medicine, USA; [2] Alan Turing Institute, UK;
[3] Department of Mathematical Sciences, Durham University, UK; 4 Department of Statistics, Oxford University, UK;
[5] Genetics, Yale University School of Medicine, USA; [6] Wellcome Trust Centre for Human Genetics, Oxford University, UK

Washington University in St. Louis School of Medicine

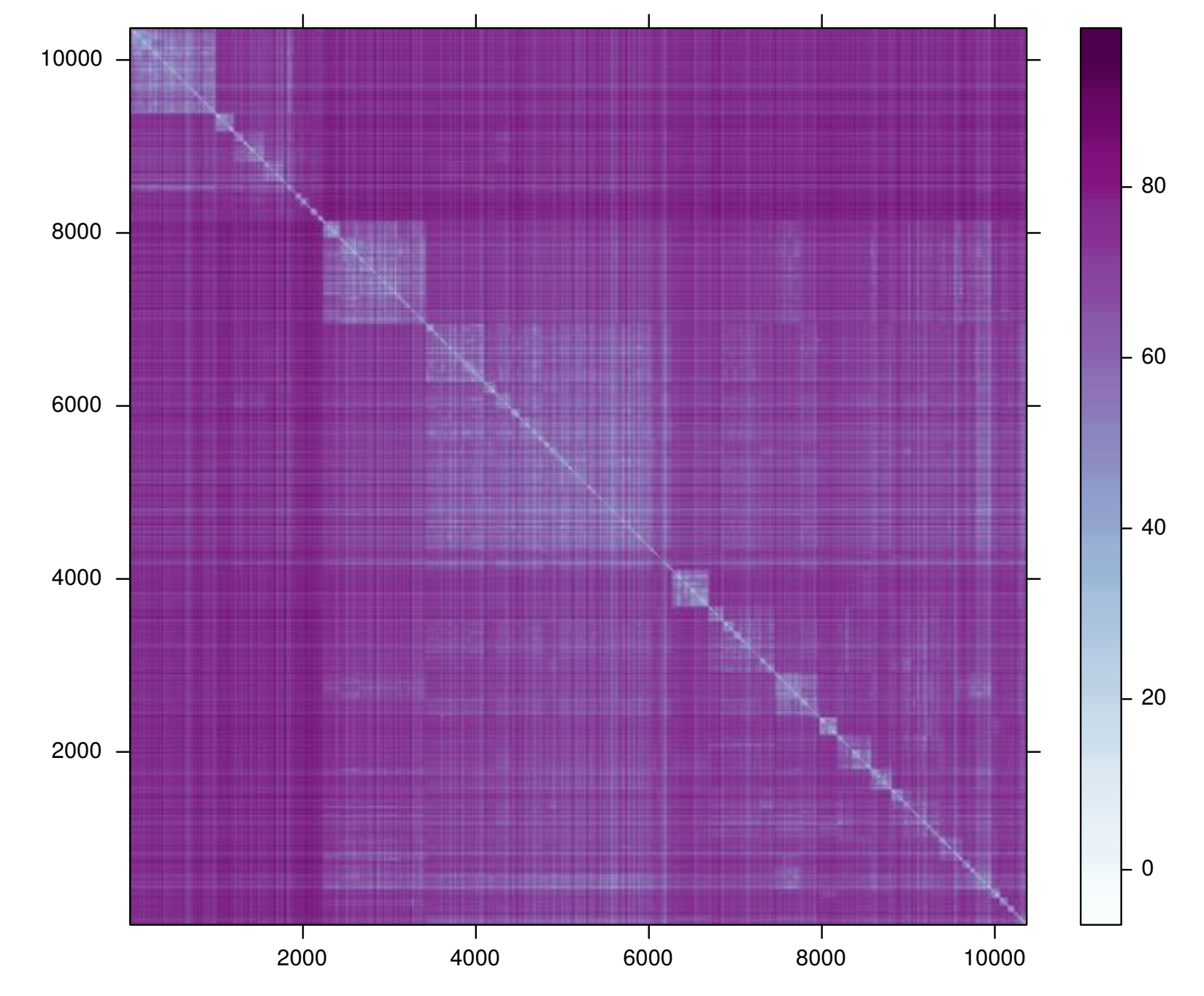
Introduction

Approximating the recent phylogeny of N phased haplotypes at sequential loci along the genome is a core problem in modern population genomics. Current leading approaches, including `tsinfer` [1] and `RELATE` [2], are rooted in the Li & Stephens (LS) copying model [3]. To facilitate further development and benchmarking in this area, we have created a fast, easy-to-use, open-source R package `kalis`: Kit for Accelerated LI and Stephens. `kalis` offers fast and memory-efficient onons of the forward and backward algorithm. They allow users to rapidly calculate the posterior copying probabilities for a given set of recipient and donor haplotypes either at a locus of interest or sequentially along a chromosome (using optimized checkpointing, see below). Helper functions enable parallelized calculation of common quantities of interest: the posterior copying probabilities at a given locus or a $N \times N$ matrix of pairwise genetic distances at a given locus (like those used in `RELATE`).

Example: Decode a Locus from a Phased VCF

Here we demonstrate how straightforward it is to use `kalis` to calculate and plot a clustered `RELATE`-style distance matrix at a given target locus starting with a VCF of phased haplotypes and a vector of the cM distances between variants, `recomb.map`.

`kalis` can run with either the classic LS mutation model or employ the asymmetric mutation kernel used in `RELATE` to utilize derived-vs-ancestral information. In this example, we use the asymmetric kernel by specifying `use.speidel = T` in the `Parameters` function. We start at the command line, using `bcftools` [4] to convert haplotypes to hap.gz format



```
bcftools convert -h my.vcf.gz
```

Then in R, we run

```
remotes::install_github("louisaslett/kalis"); library(kalis)    # Install kalis
CacheHaplotypes("my.hap.gz")                                       # Load haplotypes
pars ← Parameters(CalcRho(cM = recomb.map), use.speidel = T)  # Set HMM Parameters

fwd ← MakeForwardTable(pars)                            # Run Forward & Backward Algorithms
bck ← MakeBackwardTable(pars)
Forward(fwd, pars, target.locus.index, nthreads = 8)
Backward(bck, pars, target.locus.index, nthreads = 8)
plot(DistMat(fwd, bck, type = "minus.min")) # Plot RELATE-style Dist Matrix at target locus
```

kalis Implementation Features

- **Flexible**: among other extensions, we support
  - site-specific mutation rates
  - non-uniform prior copying probabilities
- **Scalable**: Performance features include
  - Bit-based cache for ↑ memory bandwidth (Figure 4)
  - Customized assembly-level parallelism via vector intrinsics (supporting AVX512, AVX2, and NEON architectures)
  - Novel rescaling of the LS forward and backward recursions for speed and numerical stability
- **Exact**: no approximations are used, all calculations in double precision
- **Reliable**: Ships with 100,000+ unit tests (passing on all platforms tested to date)
- **User Friendly**: delivers results directly in R for further exploration and computation with minimal data preparation

Try kalis

Installation instructions, documentation and links to the source code at

kalis.louisaslett.com

(methods for iterating sequentially over loci will be pushed to stable release soon)

Performance Benchmarks

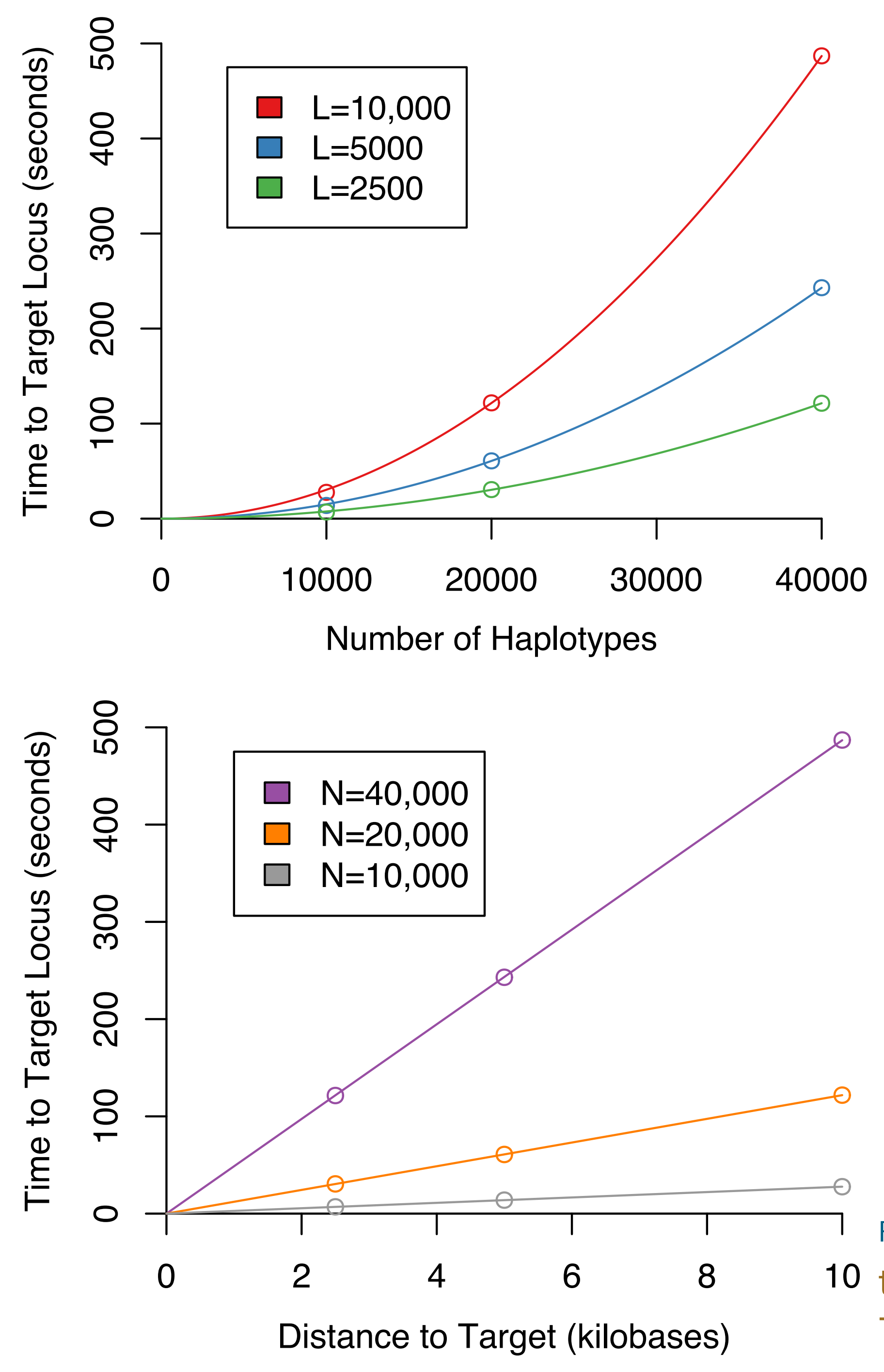


Figure 1: `kalis` shows the expected order N^2 and order L scaling of the LS model. Computed on an AWS c4.8xlarge instance (36 vCPUs, 60 GiB of RAM).

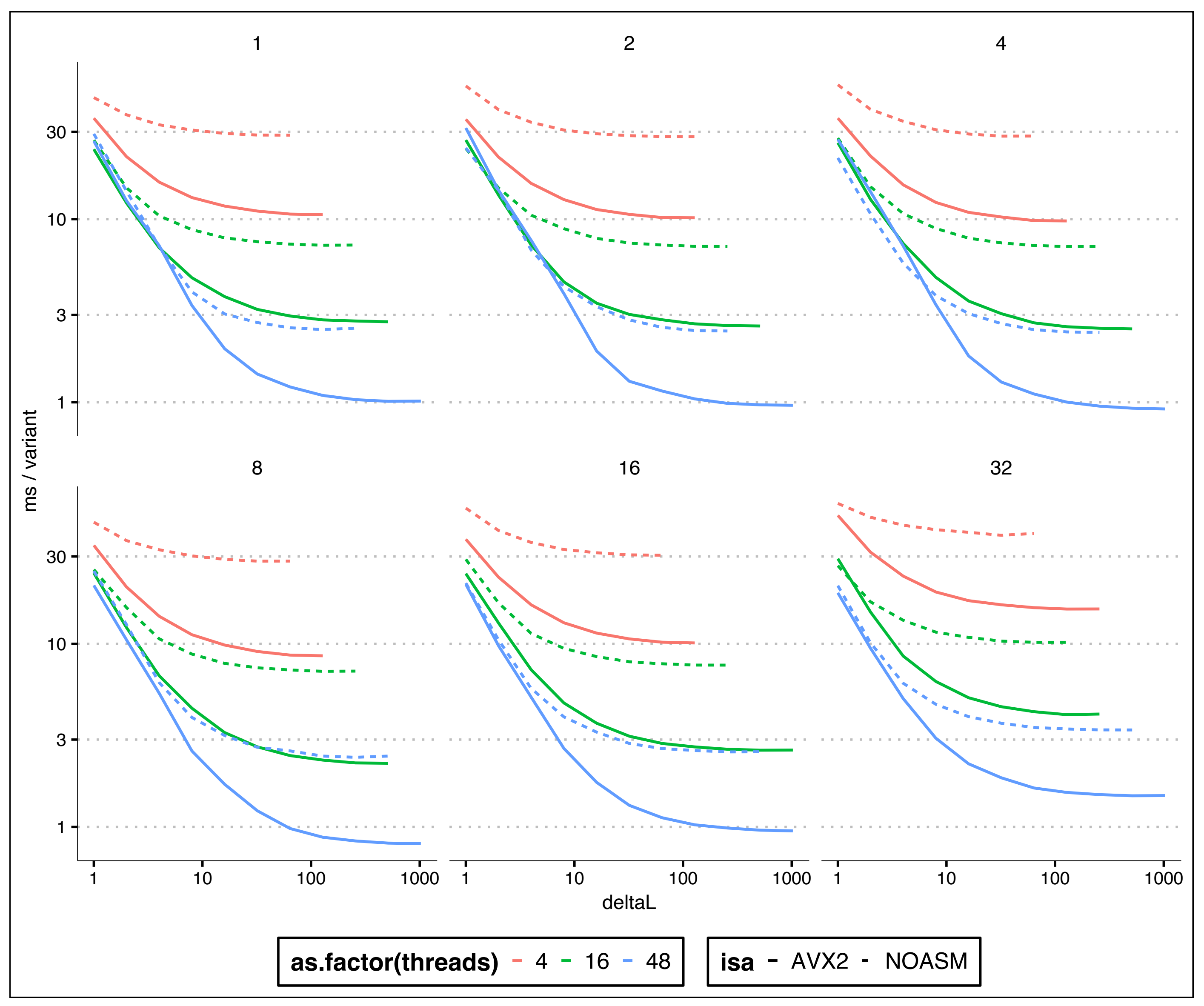


Figure 2: milliseconds / variant performance of the forward algorithm on 10,000 haplotypes across several loop unrolling levels (see title of each plot for loop unrolling level). There is clearly a cost to doing too much unrolling, but 8 loop unrolls appears to give some benefit. Using AVX2 and 48 cores, it takes less than 10 seconds to propagate a 10k × 10k forward table over 10,000 variants.

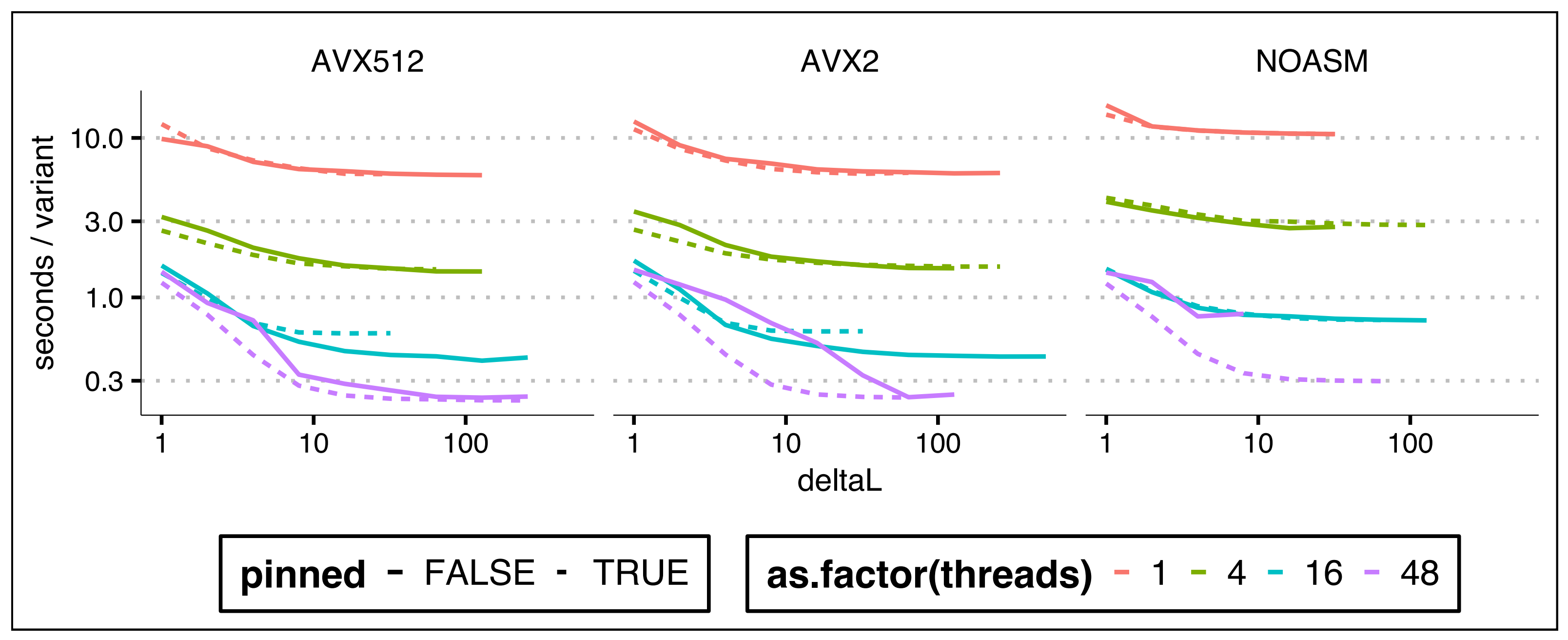


Figure 3: time / variant performance of forward algorithm on 100,000 haplotypes. We see a clear benefit from using instruction sets and pinning CPUs to avoid the cost of context switching when many cores are available. Using AVX512 and 48 cores, it takes ≈ 38 mins to propagate a 100k × 100k forward table over 10,000 variants.

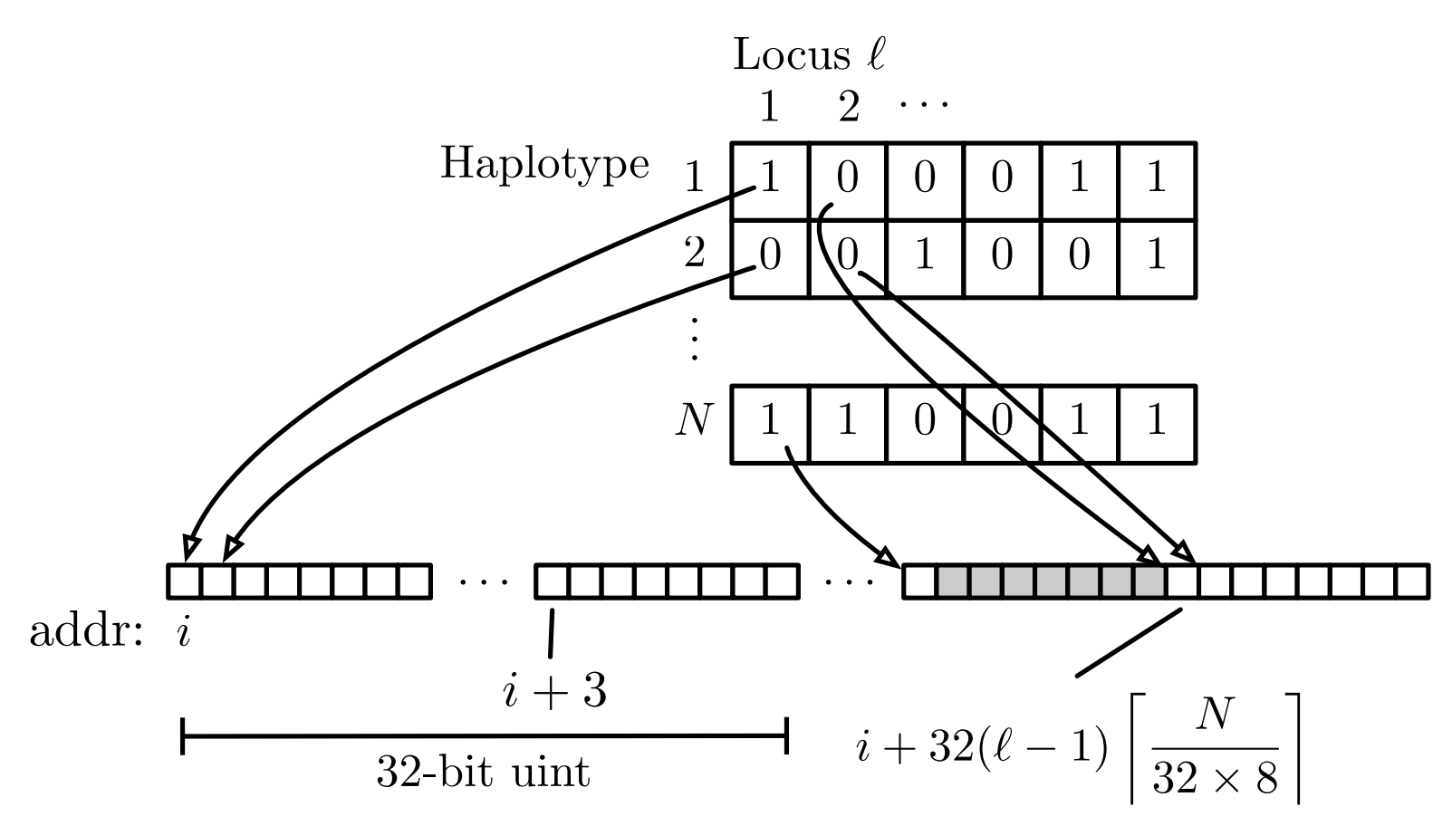


Figure 4: Haplotypes in kalis are internally stored in 32-bit chunks to maximize memory bandwidth. These bits are manually unpacked into integers for further arithmetic using BMI instructions.

References

[1] Kelleher, J., Wong, Y., Wohns, A., Fadil, C., Albers, P., and McVean, G. (2019) Inferring whole-genome histories in large population datasets. *Nat Genet*, **51**, 1330–1338.

[2] Speidel, L., Forest, M., Shi, S., and Myers, S. (2019) A method for estimating genome-wide genealogies for thousands of samples.. *Nat Genet*, **51**, 1321–1329.

[3] Li, N. and Stephens, M. (2003) Modeling Linkage Disequilibrium and Identifying Recombination Hotspots Using Single-Nucleotide Polymorphism Data. *Genetics*, **165**(4), 2213–2233.

[4] Danecek, P., Bonfield, J., Liddle, J., Marshall, J., Ohan, V., Pollard, M., Whitwham, A., Keane, T., McCarthy, S., Davies, R., and Li, H. (2021) Twelve years of SAMtools and BCFtools.. *Gigascience*, **10**.

[5] Lawson, D. J., Hellenthal, G., Myers, S., and Falush, D. (2012) Inference of population structure using dense haplotype data. *PLoS genetics*, **8**(1), e1002453.

This work is supported by the NIH/NHGRI Center for Common Disease Genomics program (UM1HG008853 and UM1HG008901)