Bioinformatics, 31(5), 2015, 782–784
doi: 10.1093/bioinformatics/btu704
Advance Access Publication Date: 22 October 2014
Applications Note

Genetic and population analysis

minimac2: faster genotype imputation

Christian Fuchsberger[1,*], Gonçalo R. Abecasis[1,*] and David A. Hinds[2]

[1]Department of Biostatistics, University of Michigan, Ann Arbor, MI, USA and [2]23andMe, Inc., Mountain View, CA, USA

*To whom correspondence should be addressed.

Associate Editor: Jeffrey Barrett

Received on September 16, 2014; revised on October 16, 2014; accepted on October 20, 2014

Abstract

Summary: Genotype imputation is a key step in the analysis of genome-wide association studies. Upcoming very large reference panels, such as those from The 1000 Genomes Project and the Haplotype Consortium, will improve imputation quality of rare and less common variants, but will also increase the computational burden. Here, we demonstrate how the application of software engineering techniques can help to keep imputation broadly accessible. Overall, these improvements speed up imputation by an order of magnitude compared with our previous implementation.
Availability and implementation: minimac2, including source code, documentation, and examples is available at http://genome.sph.umich.edu/wiki/Minimac2
Contact: cfuchsb@umich.edu, goncalo@umich.edu

1 Introduction

Genotype imputation is routinely used to increase the power of genome-wide association studies (GWAS; Howie *et al.*, 2012; Li *et al.*, 2009; Marchini *et al.*, 2007). The approach works by finding haplotype segments that are shared between study individuals, which are typically genotyped on a commercial array with 300 000–5 000 000 SNPs, and a reference panel of more densely typed individuals, such as those studied by The 1000 Genomes Project (The 1000 Genomes Project, 2012). In this way, the approach can accurately assign genotypes at markers that have not been directly examined, facilitating comparison of results across samples genotyped using different marker panels and easing fine-mapping efforts.

To reduce the computational burden of this procedure, we introduced an approach called pre-phasing (Howie *et al.*, 2012). In brief, this approach works in two steps. First, haplotypes are estimated for each of the GWAS individuals. Second, the estimated haplotypes are used for imputation. This two-step approach reduces the computational cost of imputation in two ways. First, the GWAS samples can be decomposed into haplotypes once, and these haplotypes can then be re-used many times. With standard methods (Browning and Browning, 2009; Li *et al.*, 2009; Marchini *et al.*, 2007) likely haplotypes for each sample are estimated every time imputation is repeated with a new or updated reference panel. Second, because we restrict searches for matching haplotypes to the most likely haplotype for each sample (or a small set of likely haplotype configurations), comparisons between study samples and reference panels proceed much faster. Previous implementations accounted for haplotype uncertainty and sought a pair of matching haplotypes (a process for which computation costs increase quadratically with sample size).

Ongoing whole-genome sequencing studies will contribute to reference panels much larger than currently available (Francioli *et al.*, 2014; Morrison *et al.*, 2013). Based on our computer simulations (Table 1), we expect substantial gains in imputation accuracy (measured as the r^2 between imputed genotypes and the true simulated genotypes) and in association information (which increases with imputation accuracy) using these panels. This is particularly pertinent for rare variants for which imputation based on current panels (which typically have <1000 samples) is relatively poor.

Since the complexity of imputation increases linearly with the number of markers and individuals in the reference panel, further improvements in computational efficiency are needed to keep imputation broadly accessible. Here, we describe and evaluate a collection of improvements that speed up imputation by 10–100-fold while maintaining the accuracy of our current method.

2 Approach

In this section, we present the software engineering techniques used to speed up pre-phasing imputation. To validate our improvements,

Table 1. Expected imputation accuracy

Reference panel sample size	Imputation accuracy (mean r^2)		
	MAF <0.1%	MAF 0.1–1%	MAF >1%
1000	0.41	0.64	0.96
10 000	0.69	0.84	0.98
20 000	0.79	0.89	0.99

Results are based on simulated haplotypes under a coalescent model using ms (Hudson, 2002) under a model consistent with European haplotype diversity (Plagnol and Wall, 2006), and imputed into GWAS data.

we compare the running times of our well-established pre-phasing method, minimac (Howie *et al.*, 2012), with its tuned version, minimac2 (Table 2). Compared with Impute2 (Howie *et al.*, 2012) and Beagle (Browning and Browning, 2009), minimac2 is 8× and >100×, respectively, faster and requires less memory (1.1 GB compared with 2.4 GB for Impute2 and 10 GB for Beagle, Table 2). To maximize efficiency of our code, we used vector and matrix operation functions from the highly optimized OpenBLAS library (http://www.openblas.net; ∼30% faster than the standard functions). We note that, due to overlapping effects, the overall speed-up is lower than the product of the speed ups measured for each single optimization.

2.1 Vectorization

The hot spots of many algorithms are loops where the same operation is applied across different array elements:

```
for (int i = 0; i <1024; i++)
  c[i] = a[i] + b[i];
```

Vectorization transforms these loops into instructions that work on multiple data items simultaneously:

```
for (int i = 0; i <1024; i+ = 4)
  c[i:i + 3] = a[i:i + 3] + b[i:i + 3];
```

Modern central processing units (CPUs) can execute multiple instructions in parallel within each core, so that the operation on the four elements takes roughly as long as the single operation, the vectorized loop runs approximately four times faster. All recent CPUs support vectorization and all major compilers apply this well-established technique by default. However, this automatic approach misses many more complicated optimization opportunities. In fact, we identified three key instances where the compiler did not automatically optimize our code and where manual loop vectorization resulted in significant speed-ups.

2.2 Data locality

The main bottleneck often is not computation time but rather memory access delay: CPUs can execute many instructions in the time needed to fetch a single word from the main memory. To guard against delays with accesses to main memory, most modern CPUs use high-speed memory to cache recently accessed elements and, crucially, their neighbors. Organizing matrices and other data structures so elements that are accessed in close succession are also located close to each other in memory can thus result in substantial speed-ups (by increasing the chance that they can be fetched from the high-speed memory cache). Two other useful techniques are to group together sections of code that use or modify the same variables and to shorten the distance between multiple uses of the same element in loops.

Table 2. Speed-up for imputing 1000 FUSION (Scott *et al.*, 2007) GWAS samples using The 1000 Genomes Project Phase 1 reference panel (1092 individuals, 37.4M SNPs) compared with minimac

Optimization	Speed-up compared to minimac	
	minimac2	minimac2 (four cores)
Data locality	4.5×	13.2×
Vectorization	3.8×	12.3×
Adaptive precision	1.5×	5.8×
Overall	18×	55×

All experiments were run on a server with four 2.4 GHz Intel Xeon, 128 GB of RAM, gcc 4.7.2, and OpenBLAS 0.2.11. minimac(2) required a maximum of 1.1 GB (2.8 GB using four cores) memory to impute the genome in 5-Mb chunks (including 0.5 Mb overlaps, total 6 Mb, up to 110 350 variants).

As with vectorization, modern compilers can automatically identify many opportunities for speed-ups and optimize memory access. However, automatic strategies fail to recognize more complex opportunities, which might require the restructuring of several nested loops, knowledge about the underlying statistics or modification of the layout of core data structures. We manually inspected our code and discovered that, for example, all the accesses to the reference haplotypes examine one marker at a time across many individuals, but that our default memory layout resulted in genotypes for different individuals at the same marker to be stored far apart in memory. Instead, our default layout placed all genotypes for each individual together. Reorganizing the haplotype representation alone led to a three-fold speed-up. Overall, improving data locality in three instances resulted in a 3.8–12.3-fold speed-up for the single and quad core minimac2, respectively (Table 2).

2.3 Adaptive precision

Floating point calculations with double precision are slower than with single precision (Press *et al.*, 2007). Based on our empirical evaluation, fitting of recombination and error rate parameters is quite sensitive to precision, but the imputation is not. By switching imputation to single precision, we obtain essentially identical imputed genotypes (maximum difference in imputed dosages for any single individual $= 10^{-3}$, SD $= 10^{-5}$) while gaining a 1.5-fold speed-up.

2.4 Parallel Processing

To take advantage of built-in parallel computing support of modern multicore CPUs, we parallelized all computationally expensive loops using OpenMP (http://openmp.org). This is especially attractive, since after pre-phasing GWAS genotypes, each haplotype can be imputed separately. We observe that in most environments, four cores provide the optimal balance between speed-up and parallelization overhead.

3 Conclusions

The computational complexity of genotype imputation together with upcoming very large reference panels could become a major bottleneck for the next generation of GWAS. The software engineering techniques described in this article helped to speed up

imputation by an order of magnitude and, therefore, will enable investigators to impute from the upcoming larger panels at no extra computational cost. As with previous eras of genetic analyses, our results illustrate the great value of combining excellent algorithms with underlying improvements in software engineering (Cottingham *et al.*, 1993; Gudbjartsson *et al.*, 2000; Purcell *et al.*, 2007).

Acknowledgements

We acknowledge M. Zawistowski for assistance with simulations. We thank the GWAS community for testing the software and providing useful feedback.

Funding

This work was funded by National Institutes of Health research grants (to G.R.A.).

Conflict of interest: none declared.

References

Browning,B.L. and Browning,S.R. (2009) A unified approach to genotype imputation and haplotype phase inference for large data sets of trios and unrelated individuals. *Am. J. Hum. Genet.*, **84**, 210–223.

Cottingham,R.W. *et al.* (1993) Faster sequential genetic linkage computations. *Am. J. Hum. Genet.*, **53**, 252–263.

Francioli,L.C. *et al.* (2014) Whole-genome sequence variation, population structure and demographic history of the Dutch population. *Nat. Genet.*, **46**, 818–825.

Gudbjartsson,D.F. *et al.* (2000) Allegro, a new computer program for multipoint linkage analysis. *Nat. Genet.*, **25**, 12–13.

Howie,B. *et al.* (2012) Fast and accurate genotype imputation in genome-wide association studies through pre-phasing. *Nat. Genet.*, **44**, 955–959.

Hudson,R.R. (2002) Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics,* **18**, 337–338.

Li,Y. *et al.* (2009) Genotype imputation. *Annu. Rev. Genom. Hum. Genet.*, **10**, 387–406.

Marchini,J. *et al.* (2007) A new multipoint method for genome-wide association studies by imputation of genotypes. *Nat. Genet.*, **39**, 906–913.

Morrison,A.C. *et al.* (2013) Whole-genome sequence-based analysis of high-density lipoprotein cholesterol. *Nat. Genet.*, **45**, 899–901.

Plagnol,V. and Wall,J.D. (2006) Possible ancestral structure in human populations. *PLoS Genet.*, **2**, e105.

Press,W. *et al.* (2007). *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, New York.

Purcell,S. *et al.* (2007) PLINK: a toolset for whole-genome association and population-based linkage analysis. *Am. J. Hum. Genet.*, **81**, 559–575.

Scott,L.J. *et al.* (2007) A genome-wide association study of type 2 diabetes in Finns detects multiple susceptibility variants. *Science*, **316**, 1341–1345.

The 1000 Genomes Project (2012) An integrated map of genetic variation from 1,092 human genomes. *Nature*, **491**, 56–65.