OPEN ACCESS
Journal of Bioinformatics and Computational Biology
Vol. 19, No. 5 (2021) 2150026 (15 pages)

DOI: 10.1142/S0219720021500268

World Scientific
www.worldscientific.com



# Compression for population genetic data through finite-state entropy

Winfield Chen* and Lloyd T. Elliott†

*Department of Statistics and Actuarial Science*
*Simon Fraser University, Burnaby, V5A 1S6, Canada*
**winfield.chen@sfu.ca*
†*lloyd.elliott@sfu.ca*

Received 8 April 2021
Accepted 6 August 2021
Published 30 September 2021

We improve the efficiency of population genetic file formats and GWAS computation by leveraging the distribution of samples in population-level genetic data. We identify conditional exchangeability of these data, recommending finite state entropy algorithms as an arithmetic code naturally suited for compression of population genetic data. We show between $\sim 10\%$ and $\sim 40\%$ speed and size improvements over modern dictionary compression methods that are often used for population genetic data such as *Zstd* and *Zlib* in computation and decompression tasks. We provide open source prototype software for multi-phenotype GWAS with finite state entropy compression demonstrating significant space saving and speed comparable to the state-of-the-art.

*Keywords*: Statistical genetics; genome-wide association study; genotype compression; big data; multi-phenotype analysis.

## 1. Introduction

A full description of the mapping from genotype to phenotype in humans will be a centerpiece of science, and will further our understanding of the human condition, and will lead to advances in medicine. Genome-wide association study (GWAS) is currently the main tool used to create, and revise *draft* versions of this genotype-to-phenotype mapping. The pursuit of such studies is a world-wide effort conducted and considered by labs in almost every university and also at biotechnology and pharmaceutical companies.[27] The computational resources required for GWAS are large,

* Corresponding author.

and so improving the efficiency of the manipulation and storage of genetic data and of conducting GWAS will increase the pace and accessibility of GWAS research, and also allow more efficient use of funding.

To this end, researchers have made improvements to the compression of genetic data, including support for *Zlib*[10] and *Zstd*[8] libraries (replacing *gzip*, which was formerly used in the *bgen* format) in the popular *bgen* and *ped* formats.[3,5] This has allowed cheaper dissemination and storage of large-scale consortia such as UK Biobank.[3,4] The *bgen* and the new *pgen* formats have also been extended to provide variable or low bitrates for lossy compression of dosages for large genetic datasets.[3,6] In this work, we further improve the compression of population-level genetic data using recent advances in source coding[22] including finite state entropy[7] and asymmetric codes.[9] These methods are arithmetic codes[15] for compressing streams of conditionally independent symbols and show improved performance in size and speed of population genetic data compression tasks.

A demonstration of the improvements of *fse* over *Zstd* for compression size and speed is provided in Fig. 1. In this figure, a single variant is simulated for one million subjects, with varying minor allele frequency (MAF), showing improvements in compression speed for *fse* by up to 43% and in size by up to 50%. This experiment is described further in Sec. 3. The gains by *fse* over standard dictionary methods are

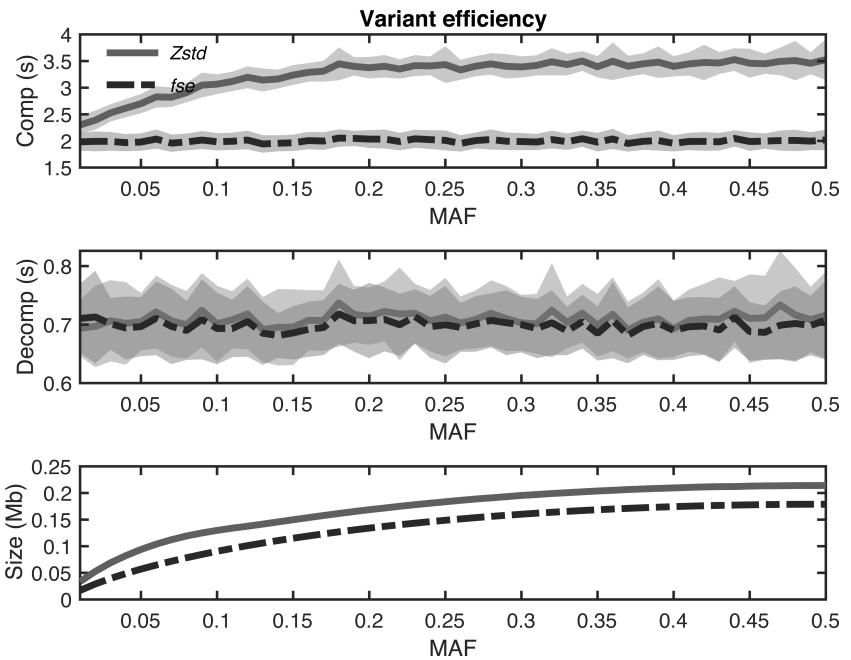


Fig. 1. Computational efficiency of encoding a single variant simulated for half a million diploid samples with a given minor allele frequency (MAF) using *Zstd* or *finite state entropy* (*fse*), with 30 replicates per condition. (Top, Middle) Runtime of compression (comp) and decompression (decomp) in seconds ($y$-axis) and one standard deviation. (Bottom) Size of encoding in megabytes ($y$-axis).

realized due to the exchangeability or conditional exchangeability of population genetic data (leading to the independence between genotypes that are required for optimal source coding[23]). We provide the *agent* software: an open source implementation (released under the BSD 2-clause license) of these ideas including features for conversion of *bgen* files to the *agent* format, and variant major GWAS for use in studies with large numbers of phenotypes.

In the remainder of this section, we describe related work and in Sec. 2, we describe exchangeability for population genetic data and introduce the *agent* software. In Secs. 3 and 4, we provide a variety of experiments on exchangeable and partially exchangeable genetic data, simulated data and real data from Thousand Genomes Project, and data with genotype dosages (as opposed to hard-calls). We provide recommendations and conclude in Secs. 5 and 6.

### 1.1. *Related work*

The variant-major organization of file formats in population-level genetics allows random access to variants, which is required for optimized GWAS and PHEWAS (phenotype-wide association study). Researchers have used linkage between adjacent variants to decrease compression size using trees, and Burrows–Wheeler transformations.[1,16] The LD-compress option in *plink2* also makes use of linkage by storing set differences.[6] We do not emphasize a comparison with linkage-based methods, as they do not allow random access. In addition, linkage-based methods are not indicated for haploid data (such as the non-pseudoautosomal Y chromosome), as such data exhibit no linkage.

We also note that recent major advances in compressing genomes (for example the Nucleotide Archival Format from Ref. 17) and alignment-free genomes[26] have led to excellent compressive bitrates for use in improved storage of *de novo* genomes and computation of distances between sequences, with application to compressing assembled genomes. Recent advances in assembled genomes such as Refs. 18 and 12 also make use of arithmetic codes and *fse*. However, these works are not relevant for the context-free and variant-major nature of compression for population-level genetic data (as they deal with whole genomes and not named variants). To our knowledge this manuscript is the first work to highlight the exchangeable or partially exchangeable aspects of population genetic data and recognize that algorithmic codes rather than dictionary methods (including *fse*) are most appropriate for compression of variant-major stores of variant blocks.

## 2. Methods

Population level genetic data are often stored in compressed variant blocks[5] in which a bitstring representation of the genotypes for all subjects are compressed separately for each variant (i.e. they are stored in a variant-major form). This organization is optimal for the purpose of fast GWAS. However, the samples collected in population-level genetic data are usually exchangeable, or partially exchangeable.[19] This

exchangeability arises in formats such as *pgen* and *bgen* because genotypes for subjects are stored in a fixed order, and this order is identical for all variants.

When a study is conducted on a homogeneous population, the fixed order does not provide information about the genotypes, leading to conditional independence among the genotypes at a fixed variant (i.e. exchangeability). The independence is conditioned on the MAF, or other variant specific measures. If a pedigree, or a genetic similarity matrix,[20] or population indicators is also considered, then the genotypes are jointly exchangeable (i.e. the joint distribution on the genotypes is still exchangeable and invariant under permutation, as long as the permutation is also applied to the pedigree, the genetic similarity matrix, or the population indicators).

For stratified or related populations, often the fixed order still provides no information about the genotypes (the subjects may be presented in an order given by sorted and random subject identification numbers). But even if a heterogeneous sample is considered in which the order indicates subpopulation identity in a block structure (leading to partial exchangeability), conditional independence still exists among all subjects in each subpopulation, and the number of blocks in the structure is often small compared to the number of samples. These considerations imply that a bitstring representation of the ordered genotypes is given marginally by a fixed law:

$$X = \ X_1 \cdots X_N, X_i \overset{\text{i.i.d}}{\sim} \mathcal{L}. \tag{1}$$

Here, $X$ is the bitstring for a marker typed for $N$ subjects and $X_i$ is the bitstring for subject $i$ and $\mathcal{L}$ is a discrete law on floating point numbers or vectors of floating point numbers (the floating points are relevant for variants that include dosage information along with discrete genotypes). For example, if $X$ involves hard-called genotypes with MAF $p$ under the Hardy–Weinberg equilibrium (HWE), then $X_i$ is a bitstring representation of the set $\{0, 1, 2\}$ and $\mathcal{L}$ is a law describing the discrete distribution that takes value 0 with probability $(1-p)^2$ and 1 with probability $2p(1-p)$ and 2 with probability $p^2$ (here the level of $X_i$ is the number of minor alleles). In other situations, such as for data in which dosages or genotype likelihoods (rather than hard-calls) or phase are recorded, $\mathcal{L}$ and the support of $X_i$ can be suitably modified to still present data in the form (1).

Population-level genetic file formats often use *Zstd*, *Zlib* or *gzip* or similar 'dictionary' style compression methods for compression of their variant blocks (these dictionary formats are used in the *pgen* and *bgen* formats). These methods work by building a dictionary of strings that occur commonly in the uncompressed bitstring and then replacing those aspects of the bitstring with keys into a dictionary.[21] However, due to the exchangeability or partial exchangeability of bitstrings of the form (1), for a dictionary value to be *viable*, many of the permutations of the dictionary value across the boundaries of the bitstring representation of a single genotype must appear in the dataset. We call this the 'exploding dictionary' problem. Here, when we say a dictionary value is *viable* we loosely mean that the value occurs often enough in the uncompressed data that its inclusion in the dictionary decreases

the size of the compressed bitstring. Roughly speaking, the independence displayed in (1) reduces the extent to which dictionary values are viable. On the other hand, source coding theory[22] is designed to describe and compress bitstrings that display independence such as (1). In particular, Ref. 11 shows a lower bound on the number of bits required to encode the output of a dictionary compression method regardless of independence. This lower bound is greater than the bound from Shannon's source coding theorem for independent data, which can be arbitrarily approached. Thus, for independent data source coding methods are advantageous. The asymmetric numerical code[9] is the most theoretically advanced source coding methods, and Ref. 7 has been benchmarked as the best *ans* implementation.

Our method, which we refer to as *agent* (*a*rithmetic codes for *gene*t*i*c data), exploits this exchangeability and independence by specifying the compression for genotype blocks with state-of-the-art *finite state entropy*[7] source coding theory. This software works by providing a variant major file format (similar to *bgen* and *pgen*), with *fse* compression. We also implement software features for GWAS with univariate linear tests. We review the *fse* algorithm in Appendix A of the supplementary material and we provide a manual and specification of the *agent* software in Appendices C and D of the supplementary material.

## 3. Experiments

In order to demonstrate the superiority of source arithmetic codes over dictionary methods for exchangeable data, in Fig. 1 we consider a simulation involving half a million diploid samples and 1 variant, with MAF varying from 0.01 to 0.5, and assuming HWE and a random subject ordering. We use reference implementations for both *fse* and *Zstd*.[7,8] The genetic data is stored in the *bgen* format, and we use a modified version of qctool (version 2.1) to compress and decompress the data, in the case of *fse* replacing qctool's compression algorithm with the *fse* implementation from *Zstd* version 1.5.0. Code to replicate this experiment is provided in the Supplementary Material. We show improved performance in both space and runtime for *fse* at all MAF levels. The largest runtime improvements are seen at high MAF (Fig. 1, Top) and the largest size improvements are seen at low MAF (Fig. 1, Bottom). The error bars for Fig. 1 (Bottom) are too small to discern: the maximum standard deviation over all MAF conditions for both *Zstd* and *fse* is 0.05 Mb. Notably, we find an apparently constant relationship between runtime and MAF in the *fse* arithmetic code. Theoretically, the runtime must be increasing with the size of the compressed variant block (even as it appears constant), as this block must be written to RAM (random access memory) or disk, and also the size of the finite state machine used by *fse* must also increase with MAF. But Fig. 1 indicates that these increases are amortized for this condition. The *fse* method improves upon *Zstd* in compression speed by 14% to 43% and in size by 16% to 49% over all MAF levels. The decompression speeds are largely unchanged between *fse* and *Zstd*, however *fse* is slightly

faster and this difference is statistically significant (a paired sample two-sided $t$-test for difference in mean decompression time yields a $p$-value of $1.3E-34$).

Motivated by this demonstration, we design four experiments (described in Sec. 3.1) exploring *fse* for population genetic data, beginning with a modification of the open source *qctool* software for the *bgen* format[3] and continuing with our *agent* software and GWAS. The results of these experiments are provided in Sec. 4. Experiments were carried out using *plink2* version 2.3 (the most recent alpha build of *plink2* available at time of writing) and *qctool* version 2.1. We used the *fse* implementation from *Zstd* version 1.5.0[8] for Fig. 1 and Experiment 1, and version 1.4.0 for Experiments 2, 3 and 4. All experiments were done on 2.10 GHz Intel Xeon E5-2683 CPUs.

### 3.1. *Experiment design*

**Experiment 1: Partially exchangeable data.** We examine the compression speed and ratio of *fse* in the partially exchangeable case in which the order of the samples in the variant blocks are provided according to stratification (in this case, stratification by ethnicity). We consider data from Phase 3: May 27th 2015 release of Thousand Genomes Project (Phase 1[a]). These data consist of 2504 subjects sampled from 26 populations (sized between 61 and 113 samples) and 81,271,745 hard-called variants. We consider two presentations of these data. In the first presentation (`EXCH`) samples are ordered according to a uniformly random permutation (with each permutation of numbers 1 to 2504 equally likely) and in the second presentation (`PART`) samples are ordered in a block structure such that the samples in each population are ordered contiguously (i.e. all samples from the first population appear first in the ordering, followed by all samples from the second population and so on). For each of these two presentations, we consider a task in which an uncompressed *bgen* file is compressed using *fse* (implemented with *qctool* + *fse*: a version of *qctool* we modified to compress variant blocks with *fse*) or *Zstd* compression algorithms (using unmodified *qctool* or *plink2*). In each case, all Thousand Genomes Project data is presented in a single *bgen* file (i.e. with all chromosomes concatenated). We record compute times and compressed file sizes for the `EXCH` and `PART` conditions.

**Experiment 2: Larger studies.** We create a simulated biologically plausible versions of human chromosome 22 with the number of samples ranging between 20,000 and 80,000. This simulated data is constructed using the *hapgen* software[25] and chromosome 22 from Thousand Genomes Project as a reference. We create one independent simulated dataset for each number of samples with 20,000, 40,000, 60,000 or 80,000 samples. After removing multi-allelic sites (a step required by *hapgen*), 1,055,452 variants remain on chromosome 22. We save each simulated dataset into a file encoded with the *bgen* format (using *Zstd*), and then consider a task in which *plink2* or *agent* is used to convert the *bgen* file into respectively the *ped*

[a] ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase1.

format (for *plink2*) or into *agent*'s file format (with *fse*). We record compute times and compressed file sizes.

**Experiment 3: Compressing dosages.** We considered a simulation study with a large homogeneous population and variants presented as dosages (formed through imputation) as well as hard calls. This situation is similar to that of large consortia such as UK Biobank.[4] We simulated 490,000 haplotypes using the *ms* software[14] under the coalescent with recombination with 100,000 segregating sites and 1000 recombination sites over a number of basepairs roughly equivalent to chromosome 22.[b] For the purposes of imputation, we use 10,000 of the haplotypes as a reference panel, and we combine the remaining 480,000 haplotypes into diploid genotypes forming a study panel of 24,000 samples. For details on study and reference panels in

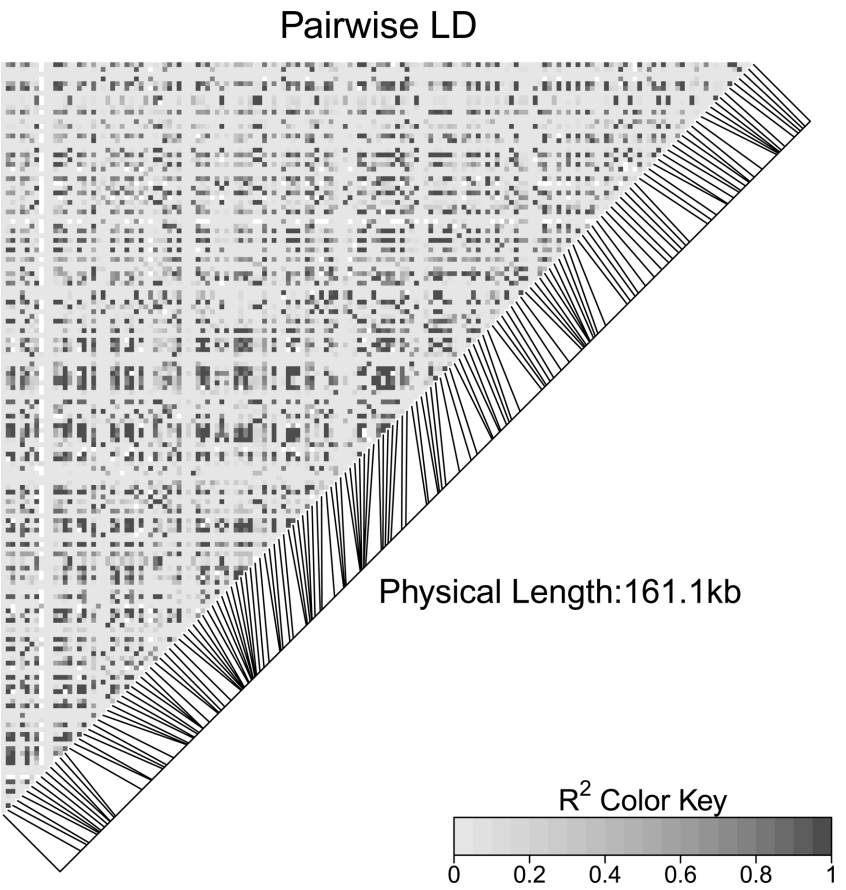


Fig. 2. Linkage disequilibrium for a chunk in Experiment 3, displayed using *LDheatmap*,[24] showing biologically plausible striated structure.

[b] Command line: `ms 500000 1 -p 15 -s 100000 -r 100 1000`.

imputation see Ref. 13. We varied the amount of imputation over the four conditions: for each condition we removed $X\%$ of the markers for $X\%$ of the samples in the study panel, and then imputed the removed genotypes using the reference panel. We stored the imputed dosages with a 16-bit representation (with 16-bit integers representing a grid over the range 0.0 to 2.0). We vary $X$ in the set $\{20\%, 40\%, 60\%$ and $80\%\}$. The same set of markers is removed for each selected sample leading to the blockwise study/reference paradigm. We conduct imputation using the *impute2* software run on 80 equally sized and partially overlapping chunks. A linkage disequilibrium plot is shown in Fig. 2 for one of the chunks from this experiment. We store the imputed datasets in the *bgen* format (with *Zstd* compression) and compare the speed and resulting file size of conversion of the stored datasets to *agent*'s file format (with *fse* compression) or *plink2*'s *pgen* format (with *Zstd* compression).

**Experiment 4: GWAS.** We perform genome-wide association studies on 30 of the chunks ($M = 19{,}922$ variants) developed in the previous experiment (compressing dosages), and we compare *agent* with *plink2*. We consider two GWAS settings. In the first GWAS setting, we consider the chunks with 20% imputation and we take all $N = 120{,}000$ samples and simulate $D = 512$ phenotypes from standard Gaussians and perform a univariate GWAS, recording the time taken. We repeat this procedure 10 times to control variance in runtimes. In the second GWAS setting, we vary the number of phenotypes (again simulated from standard Gaussians) between $D = 512$ and $D = 2048$ (in multiples of 512). For each phenotype setting, we perform the GWAS once for each imputation condition.

## 4. Results

We provide results for the experiments described in Sec. 3.1.

### 4.1. *Experiment 1: Partially exchangeable data*

In Table 1, we provide results for *qctool* + *Zstd* (i.e. unmodified *qctool*), *qctool* + *fse* and *plink2* for compressing exchangeable and partially exchangeable data from Thousand Genomes Project. We find that arithmetic coding of *bgen* files with *qctool* improves compression speed by 50% and size by 14% in the `EXCH` condition and

Table 1. Speed and size of Thousand Genome Project compression showing compression and decompression speed (in hours, with $x/y$ denoting compression speed $x$ and decompression speed $y$), and compressed size (in gigabytes) for exchangeable `EXCH` and partially exchangeable `PART` conditions.

| Method | `EXCH` speed (h) | `PART` speed (h) | `EXCH` size (Gb) | `PART` size (Gb) |
|---|---|---|---|---|
| *qctool* + *Zstd* | 27.95/12.53 | 24.65/12.42 | 13.48 | 12.99 |
| *qctool* + *fse* | 14.02/15.26 | 12.09/12.32 | 11.55 | 11.55 |
| *plink2* | 0.45/1.38 | 0.38/1.35 | 7.99 | 7.98 |

improves compression speed by 51% and size by 11% in the `PART` condition. The decompression speed is largely unchanged. Note that *Zstd* compressed size is slightly larger in the `EXCH` condition, indicating that compression ratios with the *Zstd* dictionary method depend on stratification in the *bgen* files. On the other hand, compression with *fse* does not depend on stratification.

Compression with *plink2* (using *Zstd*) outperformed both *qctool* conditions in terms of speed and size. The reduced performance in speed of *qctool* could be due to overhead in the *bgen* format (this format handles phase and polyploidy, which must be specified at each marker and decoded at each genotype). In addition, this Thousand Genome Project data is hard-called and so *plink2* uses LD-Compress (sacrificing some random access) before *Zstd*.

### 4.2. *Experiment 2: Larger studies*

In Fig. 3, we show the performance of *plink2* (using LD-*Compress* + *Zstd*) and *agent* (using *fse*) for large simulated datasets. For both pieces of software, the speed and sizes are linear in the number of samples. The *agent* software outperforms *plink2* for all conditions, but extrapolation of resulting file size indicates improved performance of *plink2* under small sample sizes (as indicated by Table 1, columns 4 and 5). Due to the large speed discrepancy between *qctool* and *plink2* in Table 1, it is unlikely that

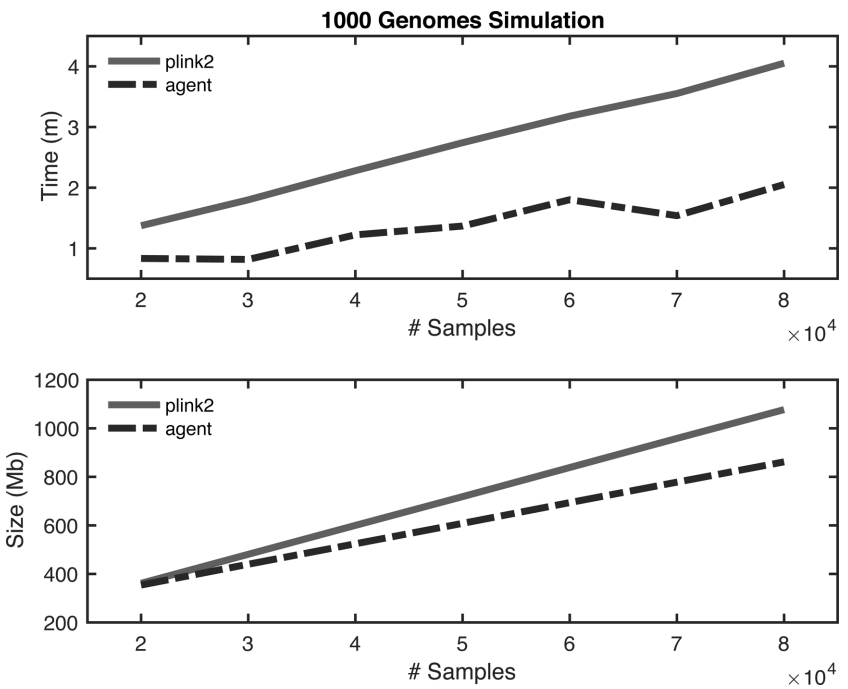


Fig. 3. Performance of *plink2* versus *agent* in a task in which between 10,000 and 80,000 samples are considered. Data is simulated based on Thousand Genomes Project's chromosome 22.

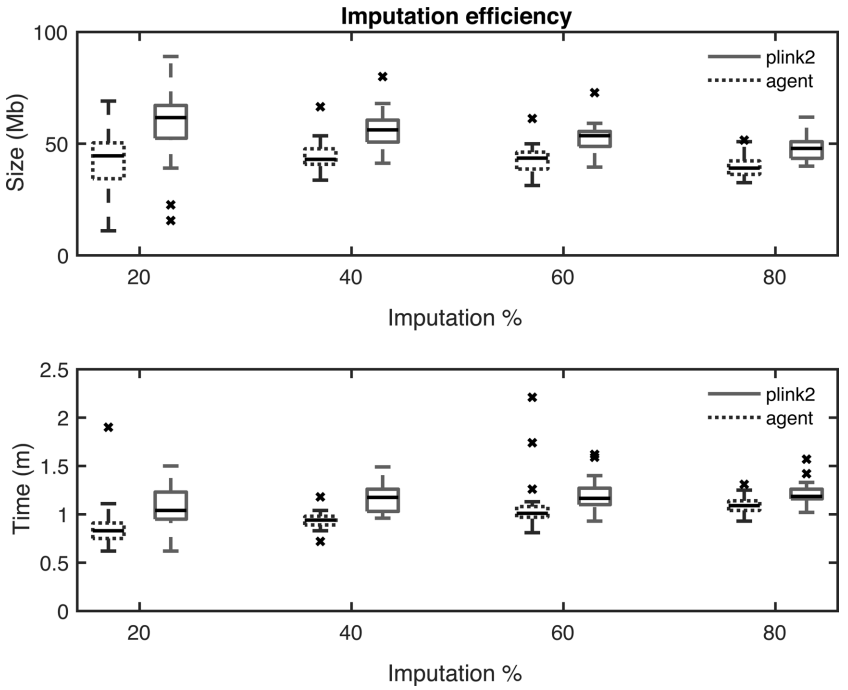


Fig. 4. Compression speeds and sizes of genetic data including imputed dosages for 240,000 samples. Box plots indicate median and quantiles for 30 replicate datasets for each imputation condition.

*qctool* (even with *fse* modification) is competitive for Experiment 2 (like *plink2*, the *agent* software strips ploidy and phase information to avoid possible overhead, optimizing for univariate human data).

### 4.3. *Experiment 3: Compressing dosages*

In Fig. 4, we show the performance of *plink2* and *agent* (using *Zstd*) for a range of imputation conditions on a large simulated biologically plausible datasets. The size of the compressed dataset and the speed of the compression are improved by *agent* under all imputation conditions, with the largest improvements seen for the 20% imputation condition.

### 4.4. *Experiment 4: GWAS*

**Experiment 5: GWAS.** We show box plots for the runtimes of the 10 repeats for the first GWAS condition in Fig. 5 (Top) and note that *plink2* is only slightly faster than *agent* in this condition. In *plink2*, no compression is done for dosage data, which may explain the improved *plink2* runtimes (*plink2* was not decompressing the files stored in the *pgen* format, as they were not compressed). Both pieces of software are optimized with Advanced Vector Extension (AVX2) instructions. Second, we

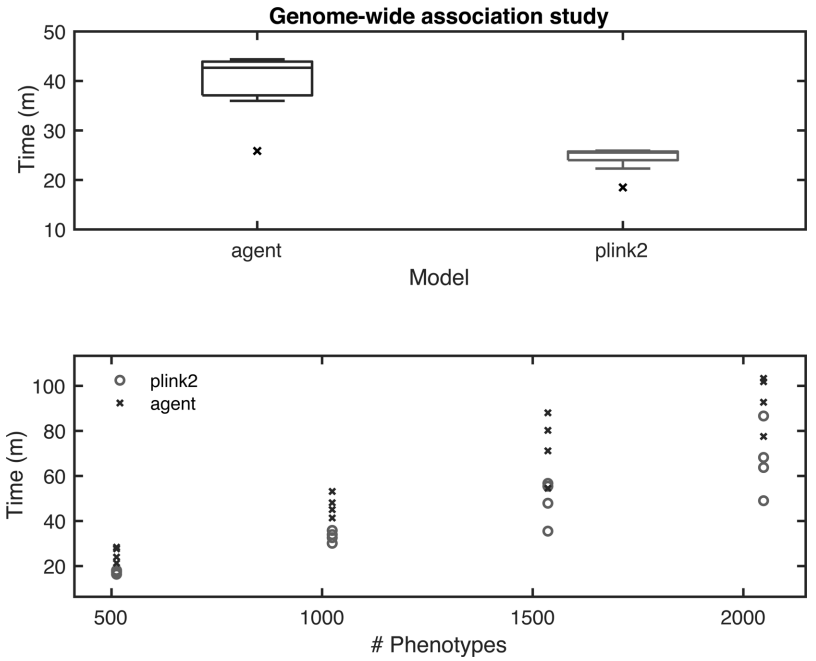


Fig. 5. Speed comparisons between *plink2* and *agent* on imputed data. (Top) Ten independent restarts are considered for a dataset with 512 phenotypes. (Bottom) The number of phenotypes is varied. In each condition, *plink2* is only slightly faster than *agent*.

consider the same genetic datasets and vary the number of phenotypes between 512 and 2048 (again, with independent standard Gaussian distributions). We perform one GWAS for each setting of the imputation proportion, yielding 5 runtimes for each method for each phenotype condition and these runtimes are plotted on Fig. 5 (Bottom). This shows that the scalings of the runtime of *agent* and *plink2* as the number of phenotypes varied are similar.

## 5. Discussion

The exchangeability of population genetic data (described in Sec. 2) recommends compression through arithmetic codes and source coding theory, rather than the commonly used dictionary methods. In our experiments, we show that gains in compression size and speed over *qctool* and *plink2* can be realized in some conditions through compression using state-of-the-art arithmetic codes (finite state entropy). This compression may be particularly recommended for dosage data stored with 16-bit fixed point numerical representations (Experiment 3). This may be due to the highly skewed distributions over genotypes that tend to arise in dosage data (examples of such skewness are illustrated in Appendix B of the Supplementary Material).

We showed that a modified version of *qctool* with *fse* support (*qctool* + *fse*) compresses faster and smaller than *qctool*'s existing *Zstd* implementation. But for small hard-called datasets, *plink2* outperformed *qctool* + *fse*. As noted in Sec. 4.1, *qctool* may incur significant overhead due to the richness of the *bgen* format. Thus, we specify and implement the *agent* software and file format for *fse* compressed genotype blocks and phenotype major GWAS, optimized for thousands of phenotypes. Our *agent* software outperforms *plink2* with both dosage and hard-calls for large sample sizes (Secs. 4.2 and 4.3). We also demonstrate speed performance similar to *plink2* on GWAS with dosage data (*plink2* was a bit faster, but we note that the *plink2* format does not compress dosage data and therefore compression and decompression were not included in the *plink2* runtimes).

## 6. Conclusion

Large consortia such as UK Biobank[4] and All of US[2] benefit from efficient storage of compressed population genetic data, as these data are replicated in thousands of labs across the world. Furthermore, fast algorithms for genome-wide association studies allow accessible research with massively multi-phenotyped data. We provide an advancement in compression and optimized GWAS code in our open source *agent* software, including features to convert population genetic data into our file format based on finite state entropy (*fse*) compression, and to perform fast GWAS.

Our experiments suggest that *fse* compression should be preferred over dictionary methods for large studies (with more than 20,000 samples). We recommend that *fse* compression be adopted by *plink2* and the *bgen* format. Several avenues of future study and improvement are suggested by our work. First, since the LD-Compress method used by *plink2* is effective (Experiment 1), further gains may be made by combining LD-Compress with *fse*. Second, compressed file sizes with *fse* may be invariant to correlation between sample stratification and sample order (Experiment 2), and so it may be possible to improve speed (without sacrificing much disk space) by reusing the same decompression *fse* decompression state machine for all variants with similar MAFs (i.e. a fixed list of 50 *fse* headers could be precomputed and keyed based on the MAF of the sample rounded to the nearest whole frequency). Finally, since the *Zstd* specification includes a step wherein the dictionary is compressed using *fse*, adoption of *fse* in file formats already supporting *Zstd* could potentially be done in a backwards-compatible way by manually crafting compressed variant blocks conforming to the *Zstd* specification but with only one (or only a few equally sized) dictionary entries that cover the entire variant block.

Based on our experiments, we recommend that *fse* compression be adopted for studies involving dosage data and many common (high MAF) variants and more than 20,000 samples. This adoption could be done by replacing dictionary methods with *fse* compression in standard tools (as *Zstd* uses *fse* for compression of its key/value dictionary, it may be possible to do this in a backwards compatible way by hand crafting a variant block in *Zstd* format such that the dictionary has only one

key/value pair), or by using *agent* in the relevant sections of the pipeline (subsetting, storing, transferring and conducting GWAS for population genetic data). Our software *agent* may also be used to further benchmark *fse* compression.

## Acknowledgments

We thank Alex Sweeten, Gavin Band and Paul Turner for helpful discussion. We thank ComputeCanada, Fred Popowich and the Big Data Hub at Simon Fraser University for help with computational resources.

## Funding

This research was supported by NSERC grant numbers RGPIN/05484-2019 and DGECR/00118-2019, and the NSERC USRA and CGS-M programs.

## References

1. Visscher PM, Wray NR, Zhang Q, Sklar P, McCarthy MI, Brown MA, Yang J, 10 years of GWAS discovery: Biology, function, and translation, *Am J Hum Genet* **101**(1):5–22, 2017.
2. Gailly J, Adler M, Zlib compression library, Available at http://www.dspace.cam.ac.uk/handle/1810/3486, 2004.
3. Collet Y, Skibiński P, Zstandard release v1.5.0. facebook/zstd, Available at https://github.com/facebook/zstd/releases/tag/v1.5.0, 2019.
4. Band G, Marchini J, BGEN: A binary file format for imputed genotype and haplotype data, bioRxiv preprint 101101/308296v2, 2018.
5. Chang CC, Chow CC, Tellier LCAM, Vattikuti S, Purcell SM, Lee JL, Second-generation PLINK: Rising to the challenge of larger and richer datasets, *GigaScience* **4**(1), 2015.
6. Bycroft C, Freeman C, Petkova D, Band G, Elliott LT, Sharp K, Motyer A, Vukcevic D, Delaneau O, O'Connell J, Cortes A, Welsh S, Young A, Effingham M, McVean G, Leslie S, Allen N, Donnelly P, Marchini J, The UK Biobank resource with deep phenotyping and genomic data, *Nature* **562**(7726):203–209, 2018.
7. Chang *et al.*, PLINK 2 File format specification draft, Available at https://github.com/chrchang/plink-ng/tree/master/pgen_spec, 2019.
8. Shannon CE, A mathematical theory of communication, *Bell Syst Tech J* **27**:3–55, 1948.
9. Collet Y, New generation entropy library, Claude Sammut and Geoffrey I. Webb (eds.). Available at https://github.com/Cyan4973/FiniteStateEntropy, 2019, pp. 81–89.
10. Duda J, Asymmetric numeral systems: Entropy coding combining speed of Huffman coding with compression rate of arithmetic coding, arXiv preprint 13112540, 2013.
11. Huffman D, A method for the construction of minimum-redundancy codes, *Proc Inst Radio Eng* **40**(9):1098–1101, 1952.
12. Shannon CE, Communication in the presence of noise, *Proc Inst Radio Eng* **37**(1):10–21, 1949.
13. Kelleher J, Wong Y, Albers P, Wohns AW, McVean G, Inferring the ancestry of everyone, bioRxiv preprint 101101/458067, 2018.
14. Adjeroh D, Zhang Y, Mukherjee A, Powell M, Bell T, DNA sequence compression using the Burrows-Wheeler transform, *Proc IEEE Computer Society Bioinformatics Conf*, pp. 303–313, 2002.

15. Kryukov K, Ueda M, Nakagawa S, Imanishi T, Nucleotide Archival Format (NAF) enables efficient lossless reference-free compression of DNA sequences, *Bioinformatics* **35**(19):3826–3828, 2019.
16. Sweeten A, Accurate alignment-free inference of microbial phylogenies, PhD Thesis, Simon Fraser University, 2019.
17. Liu Y, Peng H, Wong L, Li J, High-speed and high-ratio referential genome compression, *Bioinformatics* **33**(21):3364–3372, 2017.
18. Holmes I, Modular non-repeating codes for DNA storage, arXiv preprint 160601799v2, 2016.
19. Orbanz P, Teh YW, Bayesian nonparametric models, in *Encyclopedia of Machine Learning*, Springer, 2010.
20. Patterson N, Price AL, Reich D, Population structure and eigenanalysis, *PLOS Genet* **2**(12):2074–2093, 2006.
21. Sayood K, *Lossless Compression Handbook*, Academic Press, 2012.
22. Ganczorz M, Entropy lower bounds for dictionary compression, *Proc 30th Annual Symp Combinatorial Pattern Matching*, 2019.
23. Su Z, Marchini J, Donnelly P, HAPGEN2: Simulation of multiple disease SNPs, *Bioinformatics* **27**(16):2304–2305, 2011.
24. Hudson RR, Generating samples under a Wright-Fisher neutral model, *Bioinformatics* **18**(2), 2002.
25. Howie BN, Donnelly P, Marchini J, A flexible and accurate genotype imputation method for the next generation of genome-wide association studies, *PLOS Genet* **5**(6):1–15, 2009.
26. Shin JH, Blay S, McNeney C, Graham J, LDheatmap: An R function for graphical display of pairwise linkage disequilibria between SNPs, *J Stat Softw* **16**(3):1–10, 2006.
27. All of Us Research Program Investigators, The "All of Us" research program, *N. Engl. J. Med.* **381**(7):668–676, 2019.

**Chen Winfield** is a graduate student in the Department of Statistics and Actuarial Science at Simon Fraser University (SFU). He received his BSc in Computing Science from the School of Computing Science at SFU in 2020. He has received multiple scholarships and awards from SFU and the Natural Sciences and Engineering Research Council of Canada (NSERC) including two NSERC Undergraduate Student Research Awards and the NSERC Alexander Graham Bell Canada Graduate Scholarship. In his time with the Elliott Lab at SFU since 2019 he has contributed to research in genome-wide association studies including brain imaging genetics work with collaborators at the University of Oxford on data from the UK Biobank project which has been published in Nature Neuroscience. His research interests include bioinformatics, statistical genetics, and machine learning.

**Elliott Lloyd** is an Assistant Professor of Big Data in the Department of Statistics and Actuarial Science at Simon Fraser University, and an Honorary Academic Visitor at the Nuffield Department of Clinical Neuroscience at the University of Oxford. His research interests include machine learning, brain imaging genetics, Baysian nonparametrics, and population genetics. He currently works on SARS-CoV-2 host genetics with the HostSeq project at CGEn (Canada's national platform for genome sequencing and analysis) and the MAGPIE group (Mathematics, genomics and prediction in infection and evolution). Previously, he received his PhD from the Gatsby Unit at University College London, and did postdoctoral work in the Department of Statistics at the University of Oxford where he was a Research Member of the Common Room at.