Efficient compression and analysis of large genetic variation datasets

Ryan M. Layer[1*], Neil Kindlon[2], Konrad J. Karczewski[3], Exome Aggregation Consortium, and Aaron R. Quinlan[1*]

[1]Departments of Human Genetics and Biomedical Informatics, University of Utah, Salt Lake City, UT
[2]Department of Public Health Sciences, University of Virginia, Charlottesville, VA
[3]Analytical and Translational Genetics Unit, Harvard Medical School, Boston, MA

** to whom correspondence should be addressed.*

ABSTRACT

The economy of human genome sequencing has catalyzed ambitious efforts to interrogate the genomes of large cohorts in search of deeper insight into the genetic basis of disease. This manuscript introduces Genotype Query Tools (GQT) as a new indexing strategy and powerful toolset that enables interactive analyses based on genotypes, phenotypes and sample relationships. Speed improvements are achieved by operating directly on a compressed index without decompression. GQT's data compression ratios increase favorably with cohort size and therefore, by avoiding data inflation, relative analysis performance improves in kind. We demonstrate substantial query performance improvements over state-of-the-art tools using datasets from the 1000 Genomes Project (46 fold), the Exome Aggregation Consortium (443 fold), and simulated datasets of up to 100,000 genomes (218 fold). Moreover, our genotype indexing strategy complements existing formats and toolsets to provide a powerful framework for current and future analyses of massive genome datasets.

URLS

All source code for the GQT toolkit is available at https://github.com/ryanlayer/gqt. Furthermore, all commands used for the experiments conducted in this study are available at https://github.com/ryanlayer/gqt_paper.

INTRODUCTION

For the majority of common human diseases, only a small fraction of the heritability can be explained by the genetic variation we know of thus far[1,2]. One hypothesis posits that the elusive heritability is explained in part by rare, and thus largely unknown, genetic variation in the human population[3]. Multiple efforts are therefore underway to sequence hundreds of thousands of human genomes in order to catalog the full spectrum of human genetic variation from the common to the vanishingly rare. Extrapolating from current and forthcoming efforts, it is likely that more than 1 million human genomes will be sequenced in the very near term. Integrated analyses and community sharing of such population datasets will clearly be crucial for future discovery. However, in aggregate, the resulting datasets will include roughly 100 trillion distinct genotypes from the more than 100 million polymorphic loci that are likely to be discovered. Therefore, the development of creative data compression and exploration strategies will be crucial to enable discovery and to make these valuable datasets available to all researchers.

The need for computationally efficient representations of genotype datasets is not new. In 2007, Purcell et al. introduced the binary PED (BED) format for the popular PLINK variant association testing software[4]. The BED format encodes four biallelic genotypes in each byte using two bits per individual genotype. Binary encoding reduces the size of the resulting data file versus a simple text representation. The more recent Variant Call Format (VCF)[5] represents variants, sample genotypes, and flexible variant

annotations from DNA sequencing studies and its binary counterpart (BCF) provides a more efficient means of storing such datasets via compression and highly structured data. The BED and VCF/BCF formats, as well as their associated toolsets have become the standard for genome variation research. However, because these formats are intentionally structured from the perspective of individual variants, they are inherently ill suited to variant searches driven by specific genotype and phenotype combinations, inheritance patterns, or allele frequency thresholds.

RESULTS

Here we introduce a new, complementary strategy for indexing and mining individual genotypes and variants ascertained from millions of genomes. Our approach, which is made freely available in the Genotype Query Tools (GQT) software package, reorganizes and indexes genotype data such that it optimizes queries screening for variants based on the genotypes of one or more of the individuals in the study. GQT also utilizes an efficient data compression strategy to minimize the disk storage requirements of its index. Our goal is to provide a genotype compression and indexing scheme that maximizes analysis performance by allowing direct measurement and comparison of the compressed data without inflation. We demonstrate that these improvements provide extremely efficient genotype compression (as low as 0.38 bits per genotype) and efficient queries based on sample genotypes and phenotypes that are orders of magnitude faster than existing methods.

Organizing genotype data to expedite individual-centric queries.

Variant and genotype datasets in the VCF and BCF format are fundamentally structured as a matrix whose rows represent sites of genetic variation and columns represent the samples in the study (**Figure 1A**). Each cell in this matrix therefore reflects an individual's genotype at a single polymorphic locus. This "variant-centric" strategy is extremely effective for variant searches in a specific genomic region or for analyses that apply a test to every variant in the genome. The inherent consequence of a variant-centric format is that the data are poorly organized to facilitate queries in search of variants meeting specific genotype criteria for subsets of individuals (e.g., those affected with a given disease phenotype, sex, or quantitative trait), since one must iteratively extract every variant record (row) to inspect only a subset of individual genotypes (columns) (**Figure 1B**). To address queries of this type and to complement existing, variant-centric, indexing strategies, GQT first transposes the variant/genotype matrix such that rows represent individuals and columns represent variants. The resulting "individual-centric" organization minimizes the number of rows that must be extracted, and allows queries to quickly find and directly access the all of the genotypes for a given individual or set of individuals (**Figure 1C**).

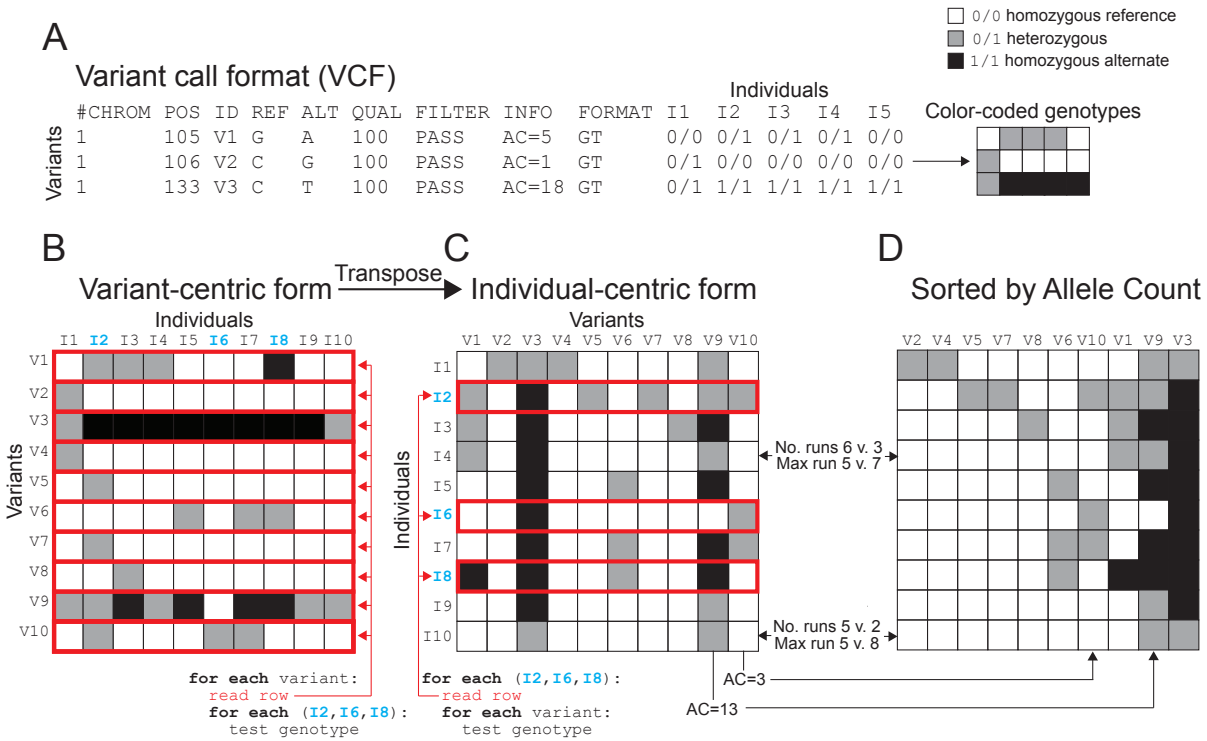


Figure 1. Individual-centric data organization. (**A**) The "variant-centric" VCF standard is essentially a genotype matrix where rows correspond to variants and columns to individuals. (**B**) The VCF standard is inefficient for queries across all genotypes and a subset of individuals since each variant row must be inspected to test the genotypes of specific individuals. (**C**) By transposing the matrix such that rows now represent the full set of genotypes for each individual, the data better aligns to individual-centric algorithms. (**D**) Sorting the columns of an individual-centric matrix by alternate allele count (AC) improves compressibility.

Optimizing data compression. Given the scale of current and future genetic variation datasets, compression must be part of any efficient indexing strategy. Genotypes are easily compressed with the variant-centric strategy (**Figure 1B**) since most genetic variation is quite rare in the human population. Therefore, the observed genotypes for the vast majority of variants will be comprised of long runs of homozygous reference genotypes that are occasionally disrupted by heterozygous or, less frequently, homozygous alternate genotypes. Long runs of identical genotypes are easily compressed with strategies such as run-length encoding or the more optimized alternative we discuss below.

We have chosen an alternative individual-centric data organization strategy that, while it facilitates queries based on individual genotypes, destroys the inherent compression of the genotype runs in the variant-centric approach. This loss of compression is the direct consequence of the fact that records in the individual-centric approach reflect the genotypes for a given individual at each site of variation in the genome. Runs of identical genotypes are far shorter, on average, than with the variant-centric approach and therefore, the individual-centric strategy will yield poor compression. The question then becomes how to leverage the query efficiency of individual-centric data organization while also retaining the opportunity for data compression? GQT solves this by sorting the variant columns of the transposed matrix in order of allele frequency (**Figure 1D**). This results in fewer, longer runs of identical genotypes in each individual's row of genotypes (**Figure 2**). For example, we compared the effect of sorting variants on genotype runs using chromosome 20 from Phase 3 of the 1000 Genomes Project. As expected, sorting variants by allele frequency caused both a dramatic increase in the mean length (10.7 versus 23.2) of identical genotype runs and a reduction in the median number of runs per individual (158,993.5 versus 70,718.5). Fewer, longer identical genotype runs allows greater compression of the each individual's (reordered) variant genotypes.

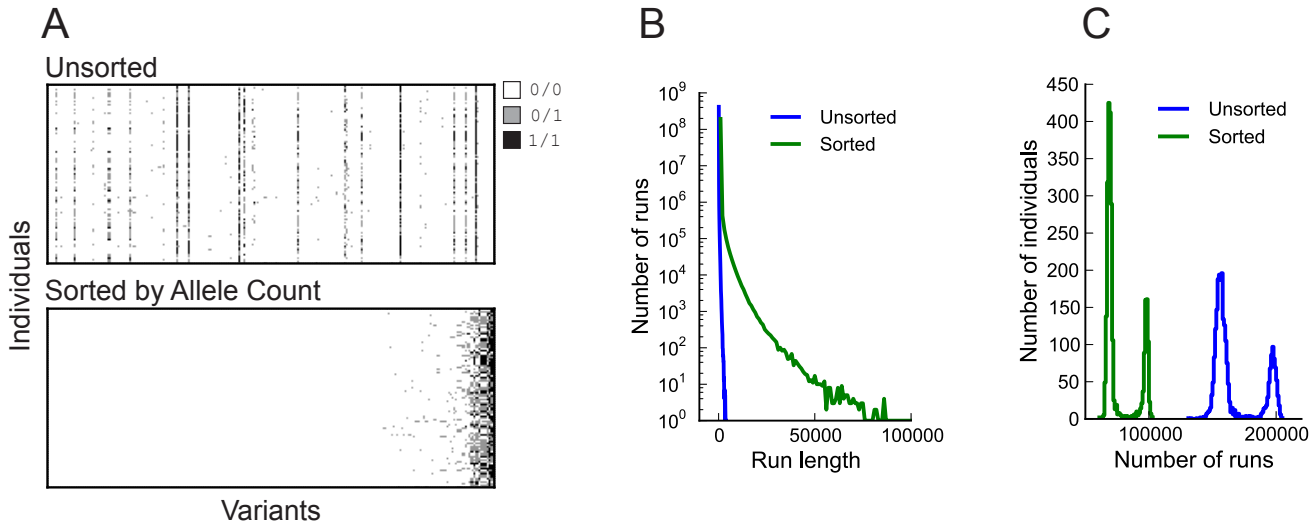


Figure 2. Sorting variants by allele frequency improves compression. (**A**) A graphical comparison of the genotype distribution of individuals (rows) and variants (columns) before and after sorting. These data represent genotypes from the 1000 Genomes Project, phase 3, for a portion chromosome 20. (**B**) The run length distribution for unsorted and sorted genotypes. (**C**) The distribution of the number of runs for sorted and unsorted data. In both cases the second peak is composed predominantly of individuals from African decent (AFR); 604/661 AFR are in the second peak in the sorted case, and 640/661 AFR in the unsorted case.

Representing sample genotypes with bitmap indices. The fundamental advantage of individual-centric data organization is the fact that all of an individual's genotypes can be accessed at once. This enables algorithms to quickly compare all variant genotypes from multiple samples in search of variants that meet specific inheritance patterns, allele frequencies, or enrichment among subsets of individuals. Despite the improved data alignment, comparing sample genotypes can still require substantial computation. For VCF, which encodes diploid genotypes as "`0/0`" for homozygotes of the reference allele, "`0/1`" for heterozygotes, "`1/1`" for homozygotes of the alternate allele, and "`./.`" for unknown genotypes (**Figure 3A, Figure S1**), comparing the genotypes of two or more individuals requires iterative tests of each genotype for each individual.

Recognizing this inefficiency, we encode each individual's vector of genotypes with a bitmap index. A Bitmap index ("bitmap") is an efficient strategy for indexing attributes with discrete values that uses a separate bit array for each possible attribute value. In this case of an individual's genotypes, a bitmap is comprised of four distinct bit arrays corresponding to each of the four possible (including "unknown") diploid genotypes. As illustrated in **Figure 3A**, the bits in each bit array are set to true (`1`) if the individual's genotype at a given variant matches the genotype the array encodes. Otherwise, the element is set to false (`0`). In turn, bitmap encoding facilitates the rapid comparison of individuals' genotypes with highly optimized bitwise logical operations. As an example, a bitmap search for variants where all individuals are heterozygous involves a series of pairwise `AND` operations on the entire heterozygote bit array from each individual. The intermediate result of each pairwise `AND` operation is subsequently compared with the next individual, until the final bit array reflects the variants where all individuals are heterozygous (**Figure 3B**). Such queries are expedited owing to the ability of modern CPUs to simultaneously test multiple bits (i.e., genotypes) with a single bitwise logical operation.

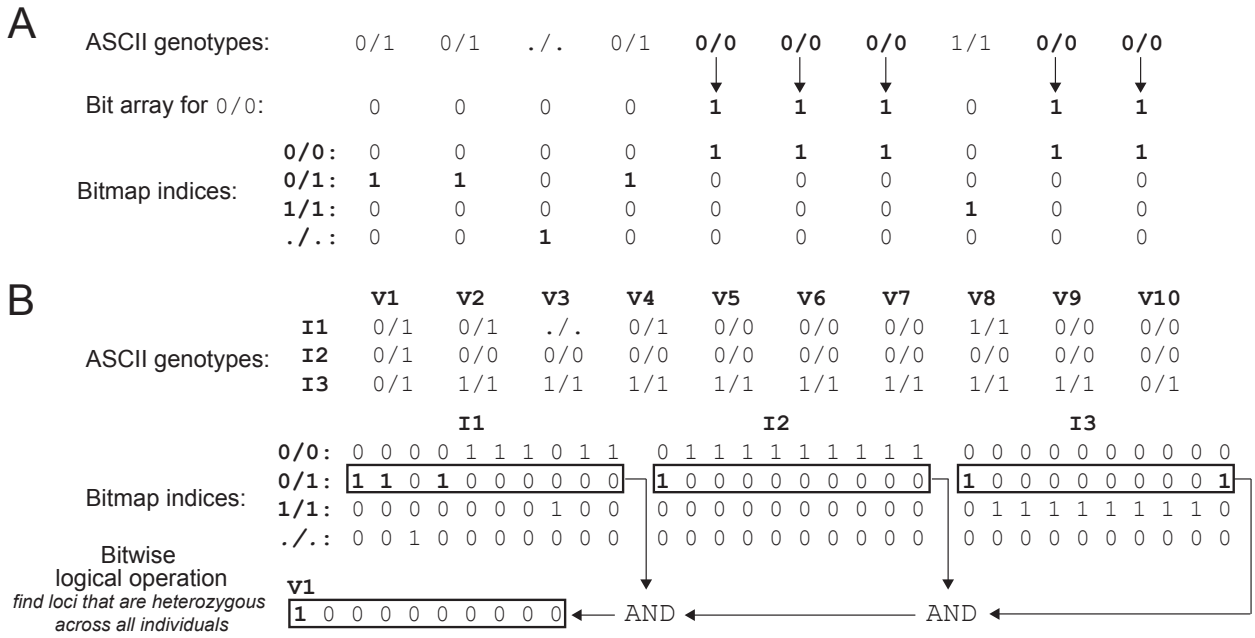


Figure 3. Bitmaps enable rapid genotype comparisons en masse. (**A**) A bit array marks the existence of one genotype state (for example, homozygous reference; "0/0") for all variants. Similarly, a bitmap index is composed of a distinct bit array for each possible genotype state. (**B**) Example genotypes in VCF format are presented for three individuals (I1, I2, I3) at 10 variant sites (V1-V10). A bitwise AND of the bit arrays corresponding to the heterozygous genotype yields the variant that is heterozygous in all individuals.

Using Word-Aligned Hybrid to directly query compressed data. Efficient methods for population-scale cohorts must address both data analysis and data compression challenges. Combining bitmap indices and bitwise logical operations to represent and compare genotypes has the potential to minimize processing time and compression reduces the size of the resulting datasets. Unfortunately, these two strategies are often at odds. The tradeoff for most compression algorithms is that they require inflation of the compressed data prior to analysis; moreover, higher compression ratios typically come at the cost of longer inflation times.

GQT strikes a balance between these two interests by compressing bitmap indices using Word-aligned Hybrid (WAH) encoding[6]. WAH is a succinct data structure that provides near-optimal compression while also enabling algorithms to directly process the compressed genotype data. Fundamentally, WAH encoding is similar to classical run-length encoding. However, WAH compresses bit arrays using an improved run encoding strategy that maintains bit alignment (see **Supplementary Note** for details). Maintaining bit alignment across words (hence the name) preserves the ability to perform logical operations between compressed values without full inflation. The cumulative advantage of the WAH compression strategy is three-fold. First, compression results in a substantial reduction in the index size. Second, the runtime of these comparisons becomes a function of the compressed input size and not the number of variants. Lastly, logical operations on compressed bit arrays can compare hundreds of (compressed) genotypes with a single operation.

Index compression performance. We assessed GQT's index compression performance by comparing its encoding of the VCF file from phase 3 of the 1000 Genomes project (2,504 individuals and 84,739,846 variants, 1.3 TB) to the compressed encodings produced by the BCF and PLINK BED formats. To understand how compression scales with the number of individuals in the dataset, we also considered subsamples of 1,000, 500, and 100 individuals with 58.2, 44.1, and 24.7 million variants, respectively.

Direct compression comparisons must account for the fact that each tool compresses different subsets of the variant and sample genotype sections of a VCF file (**Figure 4A**). By default, the BCF format encodes all of the data and metadata in both sections into binary values, and then compresses those values using blocked LZ77 encoding. Therefore, complete BCF files exhibited constant compression across population sizes with a 9.6X improvement (138.4 GB) for the 2,504 individuals dataset. PLINK ignores both variant and sample genotype metadata, does not compress the variant data, and simply encodes each genotype with two bits without compression. Since it ignores metadata, PLINK's absolute compression was higher than BCF and its compression ratio was also constant, yielding 24.1X reduction (55.1 GB) for the full data set. In contrast, GQT uses a hybrid strategy for compressing VCF files. It retains all of the variant data and only the genotype values (no metadata) in the genotype section. The variant data is compressed with LZ77 encoding. In addition, the GQT index of individual genotypes is created with the above strategy of transposition, sorting and WAH-compression.

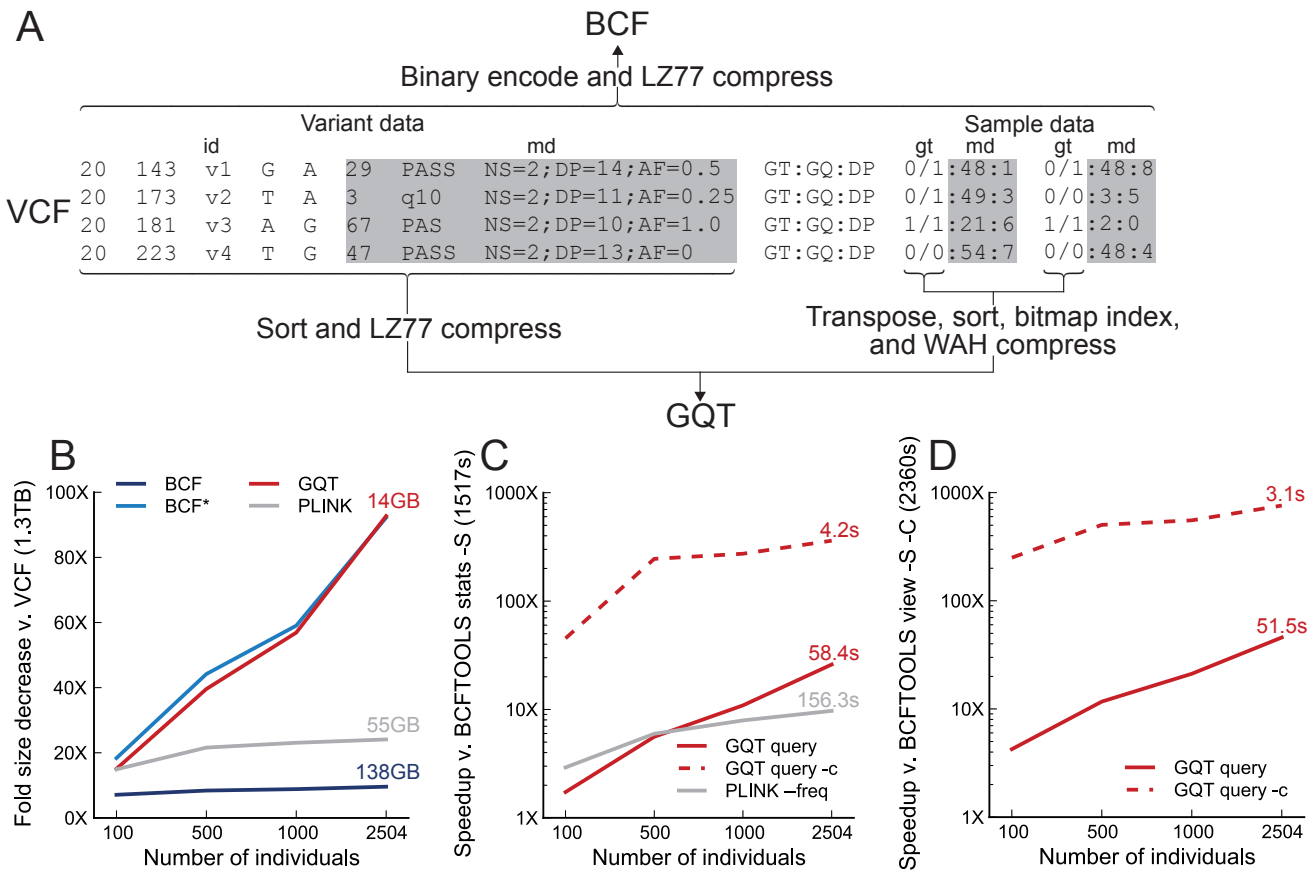


Figure 4. GQT compression and query performance for 1000 Genomes data. (**A**) VCF files are composed of a variant data section and a sample genotype section and each of these include both core information (e.g. variant position and alleles, and genotype, respectively) and extra metadata. BCF encodes both sections into a binary format, and then compresses the data using blocked LZ77 compression. GQT compresses all variant data with LZ77 and genotypes (without metadata) with WAH. PLINK (not shown) omits all metadata, uses binary encoded for genotypes, and does not compress. (**B**) File size reduction for 1000 genomes phase 3, which is comprised of 2,504 individuals and over 84 million variants. Compression ratios describe the fold reduction in file size relative to an uncompressed VCF file. BCF* omits sample metadata. (**C**) Fold speedup for computing the alternate allele frequency count for a targeted subset of 10% of the 2,504 individuals. The baseline for comparison was the BCFTOOLS "`stats`" command. (**D**) Fold speedup for finding the variants where all individuals in a target 10% are heterozygous. The baseline for comparison was BCFTOOLS "`view -C`". Note that PLINK v1.9 was excluded from this comparison because it does not directly compute this operation.

By using the WAH encoding to compress genotypes, we are, in principle, sacrificing some amount of compression so comparison can be made without inflation. In practice, we saw GQT's compression

performance steadily improving as the number of individuals and variants increased, yielding a 92.7X (14.3 GB) reduction for 2,504 individuals and requiring, on average, 0.54 bits per genotype. Interestingly, when we excluded genotype metadata (BCF* in **Figure 4B**), BCF and GQT compression are essentially identical. This means that, for genotypes from a large-scale human cohort, WAH is compressing on par with the LZ77 algorithm and GQT is not sacrificing any amount of compression.

Data query performance. The typical tradeoff for high data compression is the requirement that compressed data must be inflated prior to analysis. We chose WAH to compress genotypes precisely to avoid this tradeoff, as it enables algorithms to operate directly on compressed data without inflation. To demonstrate the analytical efficiency of WAH-encoded VCF files, we compared the query performance of GQT to both BCFTOOLS and a recent, vastly more efficient implementation of PLINK (version 1.9). When computing the alternate allele frequency among a target set of 10% of individuals from the 2504 samples in the 1000 Genomes VCF, BCFTOOLS required 1517.5 seconds. Both PLINK and GQT were substantially faster, requiring 156.3 and 58.4 seconds, respectively, yielding 9.6 fold and 26.0 fold speedup over BCFTOOLS. GQT's performance also improved more substantially than PLINK's as the number of individuals in the dataset increased (**Figure 4C**). Moreover, the majority of GQT's runtime is spent emitting the results of the query. Whereas GQT's complete runtime was 58.4 seconds for the full 2,504 individual dataset, merely 4.2 seconds were required to calculate the alternate allele frequencies from the WAH index (**Figure 4C**).

The speed with which GQT can summarize genotypes across individuals is partly owing to the use of Advanced Vector Extensions (AVX2), which exploits the Single Instruction Multiple Data (SIMD) parallelism available on modern CPUs. As the name implies, SIMD instructions perform the same operations across many elements in an array. This allows AVX2 to operate on up to eight 32-bit integers at a time and provides a nearly four-fold speedup over the default sequential calculation.

The GQT indexing strategy also enables queries that identify variants based on the genotypes of specific subsets of samples. We illustrate the performance of such queries by measuring the time required to identify rare (AAF<1%) variants among a randomly chosen subset of 10% of the individuals (**Figure 4D)**. GQT demonstrated similar performance improvements over BCFTOOLS for such queries, yielding up to 45.8-fold (51.5 seconds) over BCFTOOLS (2360.5 seconds).

Scaling to extremely large cohorts. Based on the observation that GQT's performance for both data compression and query efficiency improves as more individuals are included in the analysis, we explored its performance for datasets involving larger cohorts than that of the 1000 Genomes Project.

We first simulated variants on a 100Mb genome for cohorts ranging from 100 (588,830 variants) to 100,000 (2,061,889 variants) individuals (**Methods**). As expected, GQT's data compression and query performance improved dramatically with larger cohorts, and, since the simulation did not include genotype metadata, compression performance remained on par with that of the BCF format (**Figure 5**). Simulating variants from one million or more individuals is computationally intractable for this study; however, linear regression predicted compression ratios of at least 51.1 fold and query performance that was at least 218.1 fold faster than BCFTOOLS. We emphasize that the MACS variant simulation tool assumes a constant population size that creates far smaller proportion of very rare variants than have been observed in recent large-scale studies of human genetic variation. As such, these simulations likely serve as a lower bound; we anticipate that actual compression and query performance will far exceed these estimates when applied to forthcoming datasets.

The Exome Aggregation Consortium (ExAC)[7] has recently released a catalog of 9.36 million exonic variants identified among an aggregation of 60,706 human exomes. Given that the ExAC dataset contains 568.4 billion genotypes largely comprised of extremely rare (indeed, more than half of the variants are heterozygous in a single individual) variants, it is an informative predictor of the scale of data

compression and query performance we can expect for emerging, population-scale datasets. Indeed, a GQT index of the 14.1TB ExAC dataset used merely 0.38 bits per genotype (28GB). Moreover, GQT required only 2.1 minutes to find and report rare variants (9.98 seconds when excluding the time required to report the variants) among a target 10% of the individuals, whereas BCFTOOLS required 931.4 minutes, which represents a 443.5 fold speedup.

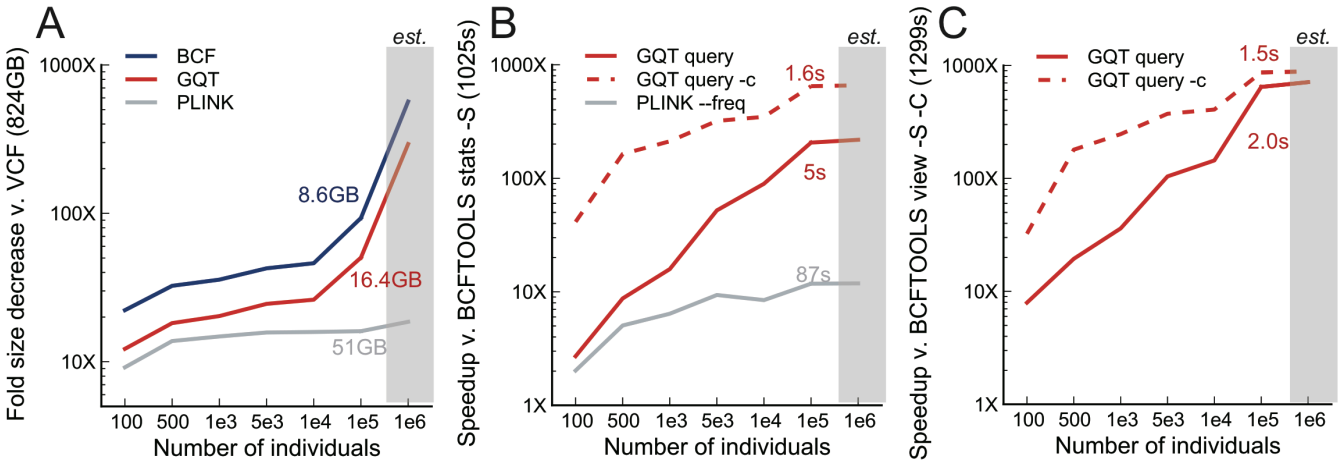


Figure 5. GQT compression and query performance for simulated genomes. A comparison of BCFTOOLS, GQT, and PLINK for simulated genotypes on a 100 Mb genome with between 100 and 100,000 individuals. (**A**) The fold reduction each tool provided relative to the original, uncompressed VCF file. (**B**) Fold speedup for computing the alternate allele frequency count for a target 10% of individuals as in Figure 4. (**C**) Fold speedup for finding the variants where all individuals in a target 10% are heterozygous as in Figure 4. Note that this simulation did not include either variant or sample metadata and that the metrics for 1 million individuals (*est.*) were estimated using a simple linear fit. As in Figure 4, PLINK was excluded from this comparison because it does not directly compute this operation.

A simple query interface for flexible data exploration. While the GQT index achieves high compression and performance for large genome datasets, greater analytical power comes from the ability to easily leverage the index to answer a broad range of questions. The GQT toolset provides a simple interface for variant queries based on sample genotypes and metadata (**Figure 6**). GQT creates an SQLite database from a PED file describing the names, relationships, phenotypes, and other attributes of the samples in an associated VCF file (**Figure 6A**). The sample database complements the GQT index of the original VCF/BCF file and allows GQT to identify the rows of sample genotypes in the GQT index that match the query. **Figure 6B** demonstrates a hypothetical GQT query in search of variants where all affected individuals ("`phenotype==2`") are heterozygous. Using these criteria, the GQT query tool issues the appropriate SQL query to the sample database to retrieve the relative position within the index of the WAH-compressed genotype bitmaps of three affected individuals. Once identified, the bit arrays for heterozygous genotypes are `AND`'ed (see **Figure 3**) to isolate the one variant where all three individuals are heterozygous.

Queries may be based on any attribute that is defined in the PED file and queries may combine multiple criteria. As examples, one may search for variants having markedly different minor allele frequencies in different world sub-populations (**Figure 6C**) or case versus control samples (**Figure 6D**). Moreover, query results are returned in VCF format, thereby allowing sophisticated analyses combining GQT queries with other, variant-centric tools such as BEDTOOLS[8], VCFTOOLS[5] and BCFTOOLS (unpublished).

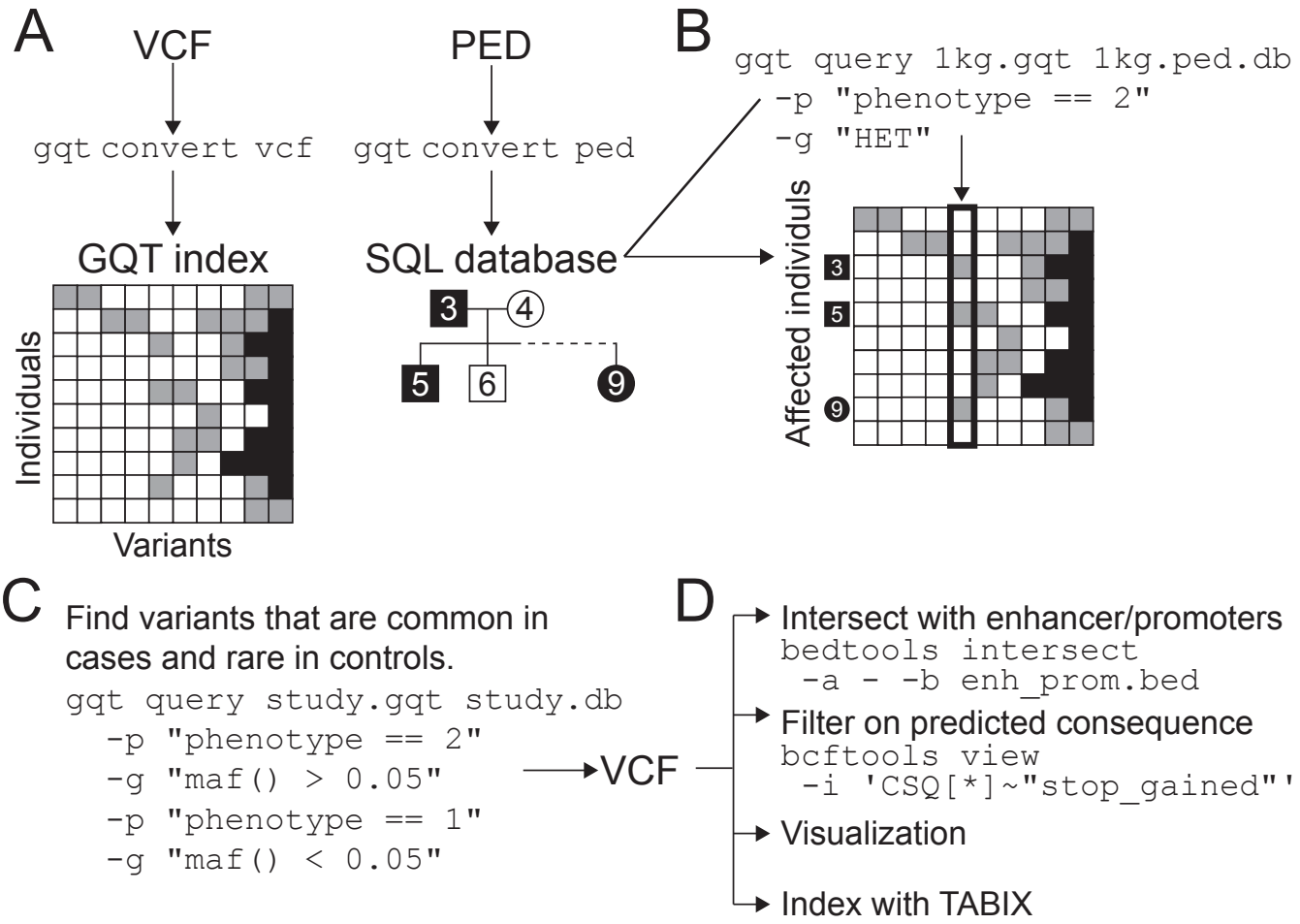


Figure 6. GQT enables queries based on sample genotypes and phenotypes. (A) GQT creates an index of the sample genotypes in a VCF or BCF file. In addition, GQT will create a SQLite database of a PED file describing the familial relationships, gender, ancestry, and various phenotypes of the samples in the VCF file. (**B**) The PED database allows GQT to quickly find the specific sample genotype rows in the genotype index that correspond to the samples that meet a user's search criteria. For example, a search for variants where all affected individuals are heterozygous begins with a query to the PED database, which identifies samples 3,5 and 9 as affected. In turn, the GQT index is used to quickly find the sole variant where samples 3,5, and 9 are all heterozygous. (**C**) Providing multiple query conditions can further refine variant searches. (**D**) Since GQT returns the identified rows in VCF format, it can be combined with other, variant-centric tools such as BEDTOOLS and BCFTOOLS to enable sophisticated analyses based on sample genotypes, variant properties, and genome annotations. This also supports visualization or variant-centric indexing with tools such as TABIX.

The flexible query framework accommodates many different experimental contexts, ranging from population genetics research to complex and rare disease studies, as well as more clinically focused datasets curated by health care systems or biotechnology companies. For example, we screened for high confidence de novo mutations in the CEPH 1473 pedigree sequenced as part of the Illumina Platinum Genomes project (**Supplementary Figure 3A**). A naïve search for candidate de novo mutations in the extensively studied NA12878 daughter involves screening for variants that are homozygous for the reference allele in NA12878's parents, yet are heterozygous in NA12878. GQT identifies 11,172 such candidates from more than 8 million total variants in 0.04 seconds (**Supplementary Figure 3B**). A more sophisticated GQT query recognizes that true de novo mutations in the germline of NA12878 should be inherited by her offspring, reducing the set of candidates to 3,002 (**Supplementary Figure 3C**). Excluding suspicious variants that lie in low-complexity regions (http://arxiv.org/abs/1404.0929) reduces the set of de novo mutation candidates by another 12% (N=2,659). While there are many more remaining candidates than would be predicted by the 1.2×10^{-8} per generation base pair mutation rate observed in the CEU pedigree[1], the 1000 Genomes Project

employed additional filters based on genotype likelihoods, proximity other variants, and other properties of the sequence alignments[1]. Moreover, the intent of this analysis is to demonstrate GQT's analytical power in the context of both large studies of unrelated individuals as well as family-based studies of disease.

DISCUSSION

Recognizing the scaling challenges posed by current and future genome datasets, our motivation was to explore new data compression and indexing strategies that minimize storage requirements while also enabling highly efficient analyses of the underlying data. We have shown that variant searches driven by sample genotypes and phenotypes are substantially faster when genotype data is transposed from a variant-centric to individual-centric organization. Competitive data compression is achieved by arranging variants in order of allele frequency, followed by word-aligned hybrid encoding of genotype bitmaps. Moreover, because WAH compression of the genotype index minimizes data inflation, GQT also achieves substantially better analysis performance for queries that interrogate the genotypes of the individuals in the dataset.

The GQT approach is well suited to very large datasets where many variants are extremely rare (and thus highly compressible), but provides less benefit for datasets involving few variants, samples, or both. This is illustrated in our comparisons to PLINK and BCFTOOLS for smaller subsets of the 1000 Genomes project dataset (**Figure 4**), and in our analyses (**Supplementary Table 1**) of variant datasets from the Mouse Genome Project (28 samples, 68.1 million variants) and the Drosophila Genome Reference Panel (205 samples, 6.1 million variants). However, we emphasize that while our analyses have been focused on human genomes, the strategies we have devised are generally applicable to large-scale datasets from any organism.

Individual-centric data organization also has the potential to expedite a wide range of population genetics statistics such as kinship coefficients and PCA-based measures of genetic similarity since these measures fundamentally depend on the comparison two or more individuals' genotypes. In contrast, haplotype-based analytical approaches[9] and statistical measures[10,11,12] are not appropriate for GQT indices since variant sites are ordered not by chromosome position but by allele frequency. However, WAH-compressed bitmaps are not inherently constrained to this organization. We plan to explore the development of variant-centric bitmap indices to enable efficient screens for selective sweeps and to quickly search for haplotypes that are identical by descent.

GQT's current indexing strategy is restricted to queries in search of variants based on sample genotypes and phenotypes. We recognize that more general queries will require indices of other genotype metadata such as allele-specific sequencings depths, genotype phase, and genotype likelihoods in order to impose stricter quality control over the variants that are returned, especially in the context of disease studies. Bitmap indices are poorly suited to attributes having continuous values (e.g., genotype likelihoods), since one bit array must be created for each distinct value. However, based on the success of previous studies[13], binning continuous values into large sets of discrete values is likely to be a straightforward approach to indexing continuous values with minimal information loss. Indeed, such binning is already employed in the VCF specification for Phred-scaled[14] integer representations of genotype likelihoods.

Since GQT optimizes individual-centric and genotype-focused analyses, it is a natural complement to the variant-centric indexing strategies employed by methods such as Tabix[15] and BCFTOOLS. Integration of these naturally complementary indexing strategies will provide the basis for a rich query interface supporting genetic data exchange efforts such as the Global Alliance for Genomics and Health. Based on the efficiency and inherent flexibility of WAH-encoded bitmap indices, we expect GQT to be a broadly useful general indexing strategy enabling the exploration of massive genomics datasets involving thousands or even millions of genomes.

METHODS

We compared GQT v0.0.1 to PLINK v1.90p and BCFTOOLS (https://github.com/samtools/bcftools) 1.1 in terms of file size and query runtime against four large-scale cohorts and simulated data sets. The cohorts included: 2,504 human genomes from the 1000 Genomes Project phase 3, 28 mouse genomes from the Mouse Genomes Project, 205 fly genomes from the *Drosophila* Genetic Reference Panel (DGRP), and 60,706 human exomes from Exome Aggregation Consortium (ExAC). Query comparisons included time to compute the alternate allele frequency count for a target 10% of the population, and time to find rare (details below) variants among a target 10% of the population. Both target sets were comprised of the last 10% of individuals. For all runtime comparisons BCFTOOLS considered a BCF file, PLINK considered a BED and BIM file, and GQT considered a GQT index and BIM file. Runtimes for GQT considered two different modes, the default mode that reports all matching variants in valid VCF and the "`count`" mode (specified by the "`-c`" option) that only reports the number of matching variants. The count mode is a useful operation in practice, and also demonstrates speed without I/O overhead.

File size. File size comparisons used an uncompressed VCF as a baseline, BCFTOOLS used a compressed binary VCF (BCF) to store both variant and sample data, PLINK used the binary plink format (BED) to store sample data and a BIM file to store variant data, and GQT used a GQT index file to store WAH-encoded sample genotype data as well as a BIM file to store LZ77-compressed variant data.

Alternate allele count. The baseline runtime for finding the alternate allele count was the BCFTOOLS "`stats`" command with the "`-S`" option to select the subset of individuals, the PLINK command was "`--freq`" with the "`--keep`" option to select individuals, and the GQT command was "`query`" (with and without the "`-c`" option) with the "`-g "count(HET HOM_ALT)"`" option to specify the allele count function and the "`-p "Ind_ID >= N"`" option to select the subset (where `N` was the ID of the range that was considered).

Identifying rare variants. The baseline runtime for selecting the variants was the BCFTOOLS "`view`" command with the "`-S`" option to select the subset of individuals and the "`-C`" option to limit the frequency of the variant, and the GQT command was "`query`" (with and without the "`-c`" option) with the "`-g "count(HET HOM_ALT)<=F"`" option to specify the allele count filter (where `F` was the maximum occurrence of the variant) and the "`-p "Ind_ID >= N"`" option to select the subset (where `N` was the ID of the range that was considered). In both cases the limit was set to either 1% of the subset size or 1, whichever was greater. PLINK was omitted from this comparison because third-party tools are required to complete this operation, and in our opinion it is not fair assign the runtime of those tools to PLINK.

Experimental Data sets.

- **1000 Genomes phase 3**: Individual chromosome VCF files were retrieved from [2] (last accessed December 10, 2014) and combined into a single file using the BCFTOOLS "`concat`" command. To understand how each tool scaled as the number of samples and variants increased, we subsampled the full data set (which included 2,504 individuals) to create new sets with 100, 500, and 1,000 individuals. To create each data set size, we randomly selected the target number of samples, then used the BCFTOOLS "`view`" command with the "`-s`" option to return just the genotypes of the target samples. We then recomputed the allele frequency of each variant with the BCFTOOLS "`fill-AN-AC`" plugin, and filtered all non variable sites with the BCFTOOLS "`view`" command and the "`-c 1`" option.
- **Mouse Genomes Project**. Data was retrieved in VCF format from (ftp://ftp-mouse.sanger.ac.uk/current_snps/mgp.v4.snps.dbSNP.vcf.gz) and was last accessed November 25, 2014.
- **Drosophila Genetic Reference Panel**. Data was retrieved in VCF format from (http://dgrp2.gnets.ncsu.edu/data/website/dgrp2.vcf) and was last accessed November 25, 2014.

- ***Exome Aggregation Consortium* (ExAC)**. Version 3 of the ExAC dataset was analyzed and run times were measure on the computing infrastructure at the Broad Institute.
- ***CEPH 1473 pedigree***. A VCF file of variants in the CEPH 1473 pedigree that was sequenced as part of the Illumina Platinum Genomes Project was downloaded from ftp://ftp-trace.ncbi.nih.gov/giab/ftp/data/NA12878/variant_calls/RTG/cohort-illumina-wgs.vcf.gz.

Simulated data sets. Genotypes were simulated using the MaCS[16] simulator version 0.5d with the mutation rate and recombination rate per site per 4N generations set to 0.001, and the region size set to 100 megabases. Since our simulation considered between 100 and 100,000 diploid samples, and MaCS only simulates haplotypes, we simulated 2X haplotypes for each case and combined 2 haplotypes to create a single diploid genome. It was computationally prohibitive to produce a data set for 1 million individuals (the 100,000 simulation ran for over four weeks), so we used a simple linear fit to estimate the file size and runtimes for 1 million individuals.

32-bit WAH word size. A fundamental choice for WAH-encoding bit arrays is the word size. Modern processors support up to 64 bits, but smaller words of 32, 16, and 8 are also possible, and the choice will affect both the compression ratio and query runtime. Since WAH uses one bit of each word to indicate the type of word (fill or literal), it would seem that larger words would be more efficient. An eight-bit word will have seven useful bits to every overhead bit, while a 64-bit word will have a 63:1 ratio. However, there is a significant amount of waste within fill words. Considering that the first two bits of a fill indicate the word type and run value, and the remaining give the length of the run in words, a 64-bit fill word can encode a run that is 1.4e20 bits long. That is enough bits to encode 46.1 billion human genomes. In fact, we only need 27 bits to cover the full genome, meaning that every 64-bit fill word would have at least 35 wasted bits. This would seem to indicate that smaller words are more efficient, but as the word size decrease the speed up of the bit-wise logical operations also declines. A single operation between two 64-bit words compares 32x more bits (and their associated genotypes) than an operation between two 8-bit words. Taken together, our test show that 32-bits gives the best balance between size and speed.

Computing environment. GQT is a tool and API written in C, which uses the htslib (https://github.com/samtools/htslib) to interact with VCF and BCF files and zlib (http://www.zlib.net/) to compress and inflate variant metadata. All experiments were run on Ubuntu Linux v3.13.0-43, with gcc v4.9.2, 4 Intel Core i7-4790K 4.00GHz CPUs with the Haswell microarchitecture, and a 550 MB/s read/write solid-state hard drive.

ACKNOWLEDGEMENTS

We are grateful to Colby Chiang for conceptual discussions, Igor Levicki for helpful advice on AVX2 operations, and both Shane McCarthy and Petr Danecek for their guidance with `htslib`. The authors would also like to thank the Exome Aggregation Consortium and the groups that provided exome variant data for comparison. A full list of contributing groups can be found at http://exac.broadinstitute.org/about
This research was supported by an NHGRI award to A.R.Q. (NIH R01HG006693).

REFERENCES

1. 1000 Genomes Project Consortium *et al.* An integrated map of genetic variation from 1,092 human genomes. *Nature* **491,** 56–65 (2012).
2. Keinan, A. & Clark, A. G. Recent Explosive Human Population Growth Has Resulted in an Excess of Rare Genetic Variants. *Science* **336,** 740–743 (2012).
3. Zuk, O. *et al.* Searching for missing heritability: Designing rare variant association studies. *Proc. Natl. Acad. Sci.* **111,** E455–E464 (2014).
4. Purcell, S. *et al.* PLINK: a tool set for whole-genome association and population-based linkage analyses. *Am. J. Hum. Genet.* **81,** 559–575 (2007).
5. Danecek, P. *et al.* The variant call format and VCFtools. *Bioinforma. Oxf. Engl.* **27,** 2156–2158 (2011).
6. Kesheng Wu, Otoo, E. J. & Shoshani, A. Compressing bitmap indexes for faster search operations. in 99–108 (IEEE Comput. Soc, 2002). doi:10.1109/SSDM.2002.1029710
7. Exome Aggregation Consortium (ExAC), Cambridge, MA. (URL: http://exac.broadinstitute.org, version 0.3) [January 2015].
8. Quinlan, A. R. & Hall, I. M. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* **26,** 841–842 (2010).
9. Durbin, R. Efficient haplotype matching and storage using the positional Burrows-Wheeler transform (PBWT). *Bioinforma. Oxf. Engl.* **30,** 1266–1272 (2014).
10. Sabeti, P. C. *et al.* Detecting recent positive selection in the human genome from haplotype structure. *Nature* **419,** 832–837 (2002).
11. Voight, B. F., Kudaravalli, S., Wen, X. & Pritchard, J. K. A Map of Recent Positive Selection in the Human Genome. *PLoS Biol.* **4,** e72 (2006).
12. Sabeti, P. C. *et al.* Genome-wide detection and characterization of positive selection in human populations. *Nature* **449,** 913–918 (2007).
13. Hsi-Yang Fritz, M., Leinonen, R., Cochrane, G. & Birney, E. Efficient storage of high throughput DNA sequencing data using reference-based compression. *Genome Res.* **21,** 734–740 (2011).
14. Ewing, B. & Green, P. Base-calling of automated sequencer traces using phred. II. Error probabilities. *Genome Res.* **8,** 186–194 (1998).

15. Li, H. Tabix: fast retrieval of sequence features from generic TAB-delimited files. *Bioinforma. Oxf. Engl.* **27,** 718–719 (2011).

16. Chen, G. K., Marjoram, P. & Wall, J. D. Fast and flexible simulation of DNA sequence data. *Genome Res.* **19,** 136–142 (2008).

SUPPLEMENTARY NOTE

Efficient comparisons using bitmaps.

By using bitwise logical operations we can compare many genotypes in a single operation, rather than comparing each individual value. At a low level, bitwise logical operations are performed on words, which are the fixed-size unit of bits used by the CPU. Modern processors typically use either 32- or 64-bit words. When a bitwise logical operation is performed between two bit arrays (each of which correspond to the genotypes of two individuals), the processor completes this operation on one pair of words at a time. For example, if the word size is 32, then computing the bitwise `AND` of two bit arrays that are each 320 bits long would require only 10 `AND`s. This optimization is equivalent to a 32-way parallel operation with zero overhead. This concept is demonstrated in **Figure S1**, where we are searching for the loci where all three individuals are heterozygous among eight variants. When genotypes are encoded in ASCII (as they are in VCF), the algorithm must loop over every individual and every variant to find the common sites. In total, this requires 24 iterations (**Figure S1A**). In contrast, encoding genotypes with a bitmap allows the same computation to be completed with only three bit-wise `AND` operations (**Figure S1B**). In effect, bit-wise logical operations compare all eight genotypes in parallel in a single step. For brevity an 8-bit word is used, and only the heterozygous bit arrays are show, but the same principles hold for the larger word sizes employed by GQT.

A ASCII Genotypes:

```
I1: 0/1 0/0 0/1 0/0 0/0 0/1 0/1 0/1
I2: 0/1 0/0 1/1 0/0 0/0 0/1 0/1 0/0
I3: 0/1 0/0 0/0 0/1 0/0 0/1 0/0 0/0
```

ASCII-based algorithm

```
IS_HET = [1,1,1,1,1,1,1,1]
for ind in [I1, I2, I3]
  for i in [1..8]
    if ind[i] != "0/1" then
      IS_HET[i] = 0
```

ASCII result

```
[0,1,0,0,0,0,1,0,0]
```

B Bit arrays for `1/0`

```
I1: 10100111
I2: 10100110
I3: 10010100
```

Bit-wise algorithm:

```
IS_HET = 0xff // hex of binary 11111111
for ind in [I1, I2, I3]
      IS_HET = IS_HET & ind
```

Binary result

```
0x84 // hex of binary 10000100
```

Figure S1. Efficiency improvements for genotype comparisons when using a bitmap index. (**A**) When considering genotypes in ASCII format (e.g., VCF), an algorithm searching for the set of variants that are heterozygous in all individuals must operate on every genotype for each for every individual separately. (**B**) In contrast, when genotypes are represented with a bitmap index, where a set of genotypes are encoded into a single CPU word (for brevity, only the bit arrays associated with the heterozygous state are shown), bitwise logical operations can be used to operate on all of the genotypes in the word with a single operation. This example assumes a word size of 8, but modern CPU support up to a 64-bit word. For the 24 genotypes given here (3 individuals, 8 genotypes each), the ASCII-base algorithm executes the "if" statement 24 times, while the bit-wise algorithm executes the logical AND ("&") only three times, with both algorithms producing equivalent results.

Compressing bit arrays.

While bitwise logical operations can drastically improve query runtime performance, bitmap indices require double the amount of space over the minimum two bits per genotype. To address this issue we can look to compressing that data. While genotype data can be compressed with standard run-length encoding, bitwise logical operations require that the bits associated with variants must be aligned, which is difficult to ensure with run-length encoding (RLE). Instead we use the Word Aligned Hybrid (WAH) encoding strategy, which represents run length in words not in bits. As shown in **Figure S2A**, RLE encodes stretches of identical values ("runs") to a new value where the first bit indicates the run value and the remaining bits give the number of bits in the run. WAH is similar to RLE, except that it uses two different types of values. The "fill" type encodes runs of identical values, and the "literal" type encodes uncompressed binary. This hybrid approach address an inefficiency in RLE where short runs map to

larger encoded values. The first bit in a WAH value indicates whether it is a fill (`1`) or literal (`0`). For a fill, the second bit gives the run value and the remaining bits give the run length in words (not bits, like in RLE). For a literal, the remaining bits directly encode the uncompressed input. Since each WAH-encoded value represents some number of words, and bitwise logical operations are performed between words, these operation can be performed directly on compressed values.

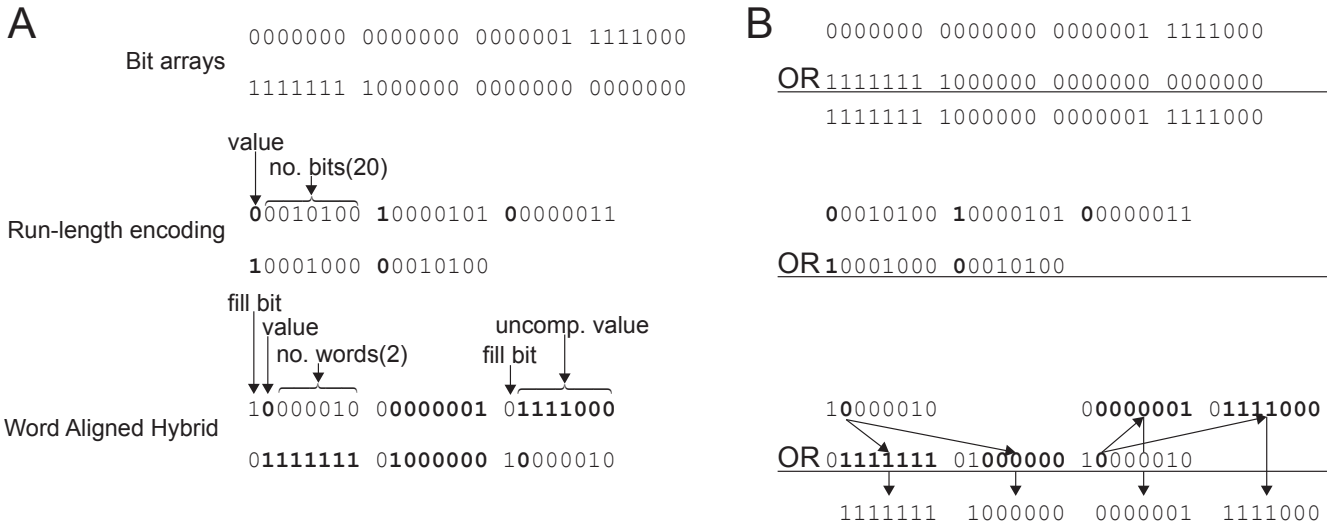


Figure S2. A comparison of binary encodings and associated bit-wise logical operations. (**A**) Two bit arrays are given in three different binary encodings: the uncompressed bit array on top, followed by the run-length encoding (RLE) and word-aligned hybrid encoding (WAH). RLE maps each set of consecutive bits to a new value that uses one bit for the run value, and the remaining bits for the number of bits in the run. WAH maps bits to one of two types of values: those that include runs and those that encode the raw binary. The first bit in a WAH value indicates if the value encodes a run (the fill bit). If that bit is set then the second bit gives the run value and the remaining bits give the run length in number of ***words*** (i.e., not number of ***bits*** as in RLE). If the fill bit is not set then the remaining bits are the uncompressed binary values. (**B**) The logical OR for the three encodings is given. For the uncompressed binary, the OR follows bit for bit across both values. For RLE, the logical OR is undefined (without inflation) because the two encoded values have different lengths. For WAH, the values are aligned based on their run length, then the logical OR is performed (in this case) between the value bit and the associated uncompressed values.

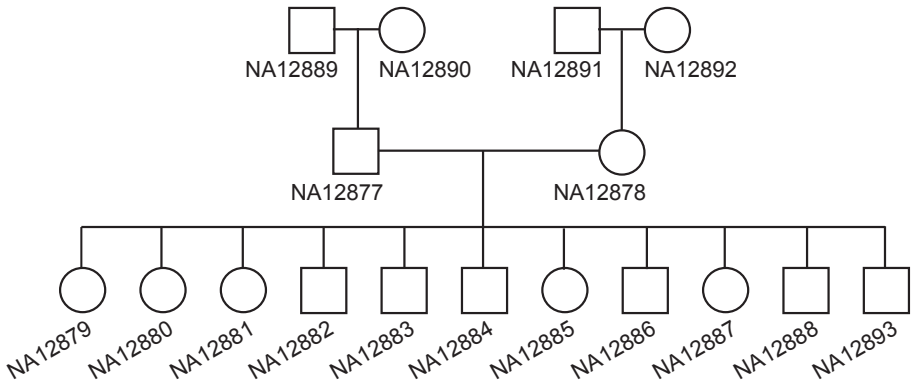
The algorithm that performs bitwise logical operations is straightforward (**Figure S2B**). To operate on two uncompressed bit arrays we simply move in unison between the two arrays from the first to last word, and find the result of each pair of words. Since each WAH value encodes one or more words, we need to move across each WAH array independently. At each step we track the number of words that have been considered in the current value and only move to the next word once the words have been exhausted. Bitwise logical operations improve the performance of most queries by computing many genotype comparisons in parallel, but some higher-level queries cannot be resolved with these operations alone.

Finding the allele frequency among a set of individuals is one such query. In this case, each bit corresponds to the genotype of a particular individual at a particular genomic position, and the allele frequency for that position is the summation of the corresponding bits across all individuals. Since 32 bits are packed into a single word, this process can be reduced to a series of bitwise sums between words, which, unfortunately, is not a standard operation. While no architecture provides single-instruction support for bitwise sum, the operation does exhibit a high-degree of parallelism. The sum of each position is independent of all other positions, which allows (in principle) the sum of all positions to be found concurrently. This classic Single Instruction Multiple Data (SIMD) scenario can be exploited through the use of the vector processor registers and instructions that are supported by the most recent Intel CPUs (Haswell and beyond). These special registers are designed to perform instructions on a list of values in parallel, and by combining several instructions (logical shift, AND, and sum) from the AVX2 instruction set we can perform the bitwise sum of 8 words in parallel. While the 8-way parallelism lags behind what is possible for other operations, it still represents a significant speed up for an operation that

is expected to be part of many queries. The index we describe above has the ability to identify variants that meet a complex set of conditions among millions of individuals and billions of genotypes in seconds.

SUPPLEMENTARY DATA

A



B

```
gqt query -i ceph1463.gqt -d ceph1463.ped.db
  -p "sample_id in ('NA12891','NA12892')" -g "HOM_REF"
  -p "sample_id = 'NA12878'" -g "HET"
```

↓

11,172 candidates (0.04 seconds)

C

```
gqt query -i ceph1463.gqt -d ceph1463.ped.db
  -p "sample_id in ('NA12891','NA12892','NA12877')" -g "HOM_REF"
  -p "sample_id = 'NA12878'" -g "HET"
  -p "maternal_id = 'NA12878'" -g "pct(HET)>=0.3"
```

↓

3,002 candidates (0.19 seconds)

D

```
gqt query -i ceph1463.gqt -d ceph1463.ped.db
  -p "sample_id in ('NA12891','NA12892','NA12877')" -g "HOM_REF"
  -p "sample_id = 'NA12878'" -g "HET"
  -p "maternal_id = 'NA12878'" -g "pct(HET)>=0.3"
| bedtools intersect -a - -b LCR-hs37d5.bed.gz -v -sorted
```

↓

2,659 candidates (0.82 seconds)

Figure S3. De novo mutation discovery in the CEPH 1463 pedigree. (**A**) The CEPH 1463 pedigree. Our analysis is focused on the discovery of de novo mutations in NA12878, the daughter of NA12891 and NA12892. (**B**) A GQT query for de novo mutations based on the expected genotypes (homozygous for the reference allele) in NA12878's parents, as well as an expected heterozygous genotype in NA12878. (**C**) True de novo mutations in NA12878's germline should be passed on to 50% of her offspring, on average. Allowing for genotyping error and binomial expectation, we filter for more confident de novo mutation candidates by requiring that the apparent mutation allele is passed on to at least 30% of NA12878's children. (**D**) A GQT query that further filters candidate mutations by excluding those lying in low complexity regions of the genome.

Experiment	Tool	**MGP** *(28 samples; 68.1 million variants)*	**DGRP** *(205 samples; 6.1 million variants)*	**1KGp3** *(2,504 samples; 84.7 million variants)*	**ExAC** *(60,706 samples; 9.3 million variants)*
File size (Gb)	BCFTOOLS	14.94	2.99	138.41	2119.7
	PLINK	**2.37**	0.47	55.10	142.29
	GQT	2.97	**0.43**	**14.33**	**27.96**
Alternate allele count time (min.)	BCFTOOLS	2.79	0.43	14.30	544.30
	PLINK	**0.41**	0.05	2.61	20.10
	GQT	1.07	**0.04**	**0.97**	**1.50**
Rare variant search time (min.)	BCFTOOLS	4.44	0.64	39.34	931.40
	PLINK	NA	NA	NA	NA
	GQT	**0.75**	**0.03**	**0.86**	**2.10**

Table S1. The performance of BCFTOOLS, PLINK, and GQT across four cohorts of different sizes and sample populations. This analysis included three whole-genome data sets from three different species: mouse genome project (MGP), *Drosophila* genome reference panel (DGRP), human from 1000 Genomes phase 3 (1KGp3), and a whole-exome for human from the Exome Aggregation Consortium (ExAC).