De novo assembly and genotyping of variants using colored de Bruijn graphs

Zamin Iqbal[1,3+], Mario Caccamo[2+], Isaac Turner[1], Paul Flicek[3], Gil McVean[1,4*]

Supplementary Methods

[1]Wellcome Trust Centre for Human Genetics, Roosevelt Drive, Oxford OX3 7BN, UK
[2]The Genome Analysis Centre, Norwich Research Park, Norwich, NR4 7UH, UK
[3]European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton CB10 1SD, UK
[4]Department of Statistics, 1 South Parks Road, Oxford OX1 3TG, UK

*Corresponding author
mcvean@well.ox.ac.uk

Terminology and Definitions

Node: In a de Bruijn graph, a node represents a k-mer (string of length k, taken from the alphabet : A,C,G,T).
In/out-degree: Number of edges pointing in/out of a node**.**
Supernode: maximal length path through the graph with constraint that only the first/last nodes in the path may have in/out degree !=1. Given any node, there is a unique supernode containing it – extend in both directions until one reaches a node with in/out-degree <>1 (i.e. a junction). Since edge/node existence varies between colors, supernodes are color-dependent.
Bubble: A bubble is a pair of supernodes with the same start and end nodes. This generalizes straightforwardly to multiallelic sites.
Branch: Each of the supernodes constituting a bubble is called a branch.
Tip: A short path ending in a node with out-degree 0; depending on coverage, kmer, length of the path and genome complexity, one can infer that this is more likely to be created by a sequencing error than a coverage-gap due to sampling.
Confounded: A variant is called confounded if overlaps with other parts of the genome (or with itself) prevent it from forming a clean bubble.
Effective coverage, or expected k-mer coverage: The requirement that reads overlap by at least k bases before the overlap registers in the de Bruijn graph results in a reduction in coverage; this results in an "effective" coverage (of k-mers) in the graph. If R is the read-length and D is the expected per base coverage, the effective coverage is

$$D_{\text{eff}} = \frac{R-k+1}{R} D$$

Effective read-length: The requirement that reads overlap by at least k bases means that one can consider a genome of length G as a sequence of kmers (rather than bases), from which reads of length R-k+1 are sampled.

1. Theoretical predictions for variant discovery power

The following model incorporates a Poisson model of coverage on a de Bruijn graph with a model for how genome repeat content affects power to detect variants. This model was used to generate predictions for power for the Bubble-Caller (BC) algorithm as shown in Figure 2a, and Supplementary Figures 5 and 6.

1.1 Separating the effects of experimental design and and genome-complexity

The probability, P, of detecting a variant allele in a de Bruijn graph with kmer-size k, is the product of the probability that the allele is present in the graph (which depends on sequence coverage, read-length, allele-length and kmer-size and error rate), the probability, G, that the variant containing this allele is not confounded with the true genome graph, and the probability, E, that the variant containing the allele is not confounded by an error. We refer to the second probability as the "genome complexity" G, which depends only on the complexity of the genome graph, the length of the allele, and k. The genome complexity function acts as an upper bound for sensitivity (power).

1.2 Estimating genome complexity

For the human genome, the genome complexity was estimated from the human reference genome NCBI36.

1. For SNPs we estimate *G* directly, by taking the 247 Mb human chromosome 1 (from reference assembly NCBI36), and creating all 3x247 million = 741 million possible single nucleotide variants in turn. The path created by each SNP allele is then compared with the graph of the reference, and if (a) the path forms a new supernode (i.e. does not overlap with the rest of the genome) and (b) the reference allele itself forms a single supernode, then the SNP is considered *callable*. The genome complexity is calculated as the proportion of possible SNPs which are callable. This is shown in black in Supplementary Figure 4. In the context of the high coverage simulation, where a modified copy of chromosome 1 was treated as an entire genome, the above calculation was repeated replacing NCBI36 with that chromosome (shown in blue in Supplementary Figure 4).
2. The probability that a 50bp variant is callable can be approximated from the probability that a randomly chosen 50bp contig from the human reference lies entirely within one supernode. This is calculated for different values of k (k = 21, 31, 41, 55, 65 and 75). To predict power for variants of different sizes we used 50 bp as a mean for small variants (1-100bp), 550 bp for medium sized variants (101 bp – 1 kb) and 5.5kb for large variants (1 kb – 10 kb). These are shown in Supplementary Figure 5.

Estimates are expected to become progressively worse approximations as the size of variant increases as they do not take account of known families of repeats such as Alus, which are over-represented relative to their genome abundance in structural variation. We note that calculations could be performed for any particular variant type.

1.3 Poisson model incorporating read-length, coverage, kmer-size and sequencing errors

The sampling of reads from a genome may be considered as queuing problem – reads "arrive" at a queue in a Poisson process of fixed rate, λ=D/R (where D is the sequencing depth and R the read length) and queue for a fixed time R. In queuing terminology this is an $M/D/\infty$ queue. By making a linear transformation, instead of considering a Poisson process of reads of length R arriving on a genome, in a de Bruijn graph one effectively has reads of length L=R-k+1 arriving on a "genome" of length $G-k+1 \cong G$. Thus the effective depth of coverage, or expected *k*-mer coverage, in a de Bruijn graph is given by

$$D_{\text{eff}} = \frac{R-k+1}{R} D = \lambda L$$

The following illustrative examples show how choice of k, D, R affect sensitivity. If k=21 and R=100, then the effective read length in the de Bruijn graph is 80, whereas a SNP causes an allele of length 22. Thus the entire allele fits inside a single read. By contrast, if k=75 and R=100, then the effective read-length is 26, but a SNP causes an allele of length 76, and thus several reads are needed to cover the allele. Thus we expect that if, for a specified *k* and *R*, the allele of interest fits into a single effective read, the probability of covering the whole allele very rapidly approaches 1 as depth of coverage increases. The mean and variance of the distribution of contig lengths were derived by Lander and Waterman [1]; we derive a good

approximation for the whole probability distribution below; for a general treatment see Leeuwaarden et al[2].

Proposition:
If *C* is defined as the distance between the (starts of the) first and last reads in a contig, then the probability distribution function of *C* is given by

$$P(C \geq t) \cong (1-e^{-\lambda L})e^{-\lambda e^{-\lambda L}t} I_{\{t>0\}} + e^{-\lambda L} I_{\{t=0\}}$$

where I is an indicator function. The approximation is an upper bound, and becomes worse when λL is small (as k approaches R, or when D is small). The actual contig length is L+C (a contig is always as long as a read) - note that this results in a distribution which is a point mass plus an exponential distribution.

Proof:
If N is the number of reads in a contig after the initial one, then N is geometrically distributed with parameter 1-Exp(- λL). For a fixed N, approximate the waiting time until the end of the contig (i.e. the waiting time until a gap of length at least L), by the waiting time until N reads have arrived, which is given by a Gamma distribution. This approximation breaks down when the probability of a read arriving with a gap larger than L is non-negligible – i.e. when λL is small.

$$P(C=s) = \sum_{n=0}^{\infty} P(N=n)P(C=s \mid N=n) = e^{-\lambda L} I_{\{s=0\}} + \sum^{\infty} (1-e^{-\lambda L})^n e^{-\lambda L} \frac{s^{n-1}\lambda^n e^{-\lambda s}}{(n-1)!}$$

$$= e^{-\lambda L} I_{\{s=0\}} + (1-e^{-\lambda L}) e^{-\lambda(L+s)} \lambda e^{\lambda s(1-e^{-\lambda L})} I_{\{s>0\}}$$

$$= e^{-\lambda L} I_{\{s=0\}} + (1-e^{-\lambda L}) \lambda e^{-\lambda L} e^{-\lambda e^{-\lambda L} s} I_{\{s>0\}}$$

Note, this shows that *C* is a sum of a Dirac mass, and an exponentially distributed variable. The result then follows by integration.

A calculation shows that

$$\overline{C} = \frac{e^{\lambda L} - 1}{\lambda}$$

and thus for the region of parameter space of experimental interest, *C*>>*L*, except when *D* is very low. Note that if $k = 1$, the above formula reduces to the standard Lander-Waterman formula for contig length, $R(e^D - 1)/D$.

Corollary:
The probability P that an allele of length d is present in the de Bruijn graph is approximated by

$$P \cong (1 - e^{-\lambda L}) P(C \geq d)$$

Proof:
The previous proposition gives the probability distribution of the contig formed by all reads arriving after any particular read – the first read does not need to be preceded by a gap. Therefore, without loss of generality, suppose we have an allele of length *d*, starting at position *L*. Conditional on there being any coverage at all on the first node, there are some number *N* of reads covering that node, which will have start positions uniformly distributed on the interval [1,*L*].
As the depth of coverage increases, so will *N*, and the start position of the furthest left read approaches 1 – we are interested in the distribution of the length of the contig starting with that furthest-left read. In other words, conditional on having non-zero coverage at the (arbitrary) initial position *L*, as depth increases the distribution of the length of the contig of interest approaches the distribution of a contig of length *L*+*C*. This contig will contain the allele (length *d*) when *C*>=*d*, which proves the result.

1.4 Combining Poisson model and genome complexity

Recalling the discussion above, the power to detect an allele of length t>0, is given by

$$\text{Power} \cong G(t,k)(1 - e^{-\lambda L})^2 e^{-\lambda e^{-\lambda L} t}$$

where *G* depends on the species/genome in question. Note that in the de Bruijn graph, an n base-pair variant forms an allele/path of length *k*+*n*.

1.5 Integrating an error model

1.5.1 Reduction in coverage due to errors

The primary effect of sequencing errors is to reduce the arrival rate λ by a factor of $1 - k\varepsilon$, where ε is the per-base sequencing error rate (and therefore $k\varepsilon$ is the per k-mer error rate). In the case of the simulations (see below) a simple model of an Illumina er ror-profile was used: a per-base error rate 0.1% in first 80bp of read, and 5% in the final 20bp. For convenience below, these two error-rates are termed “fast” and “slow”. This resulted in a per k-mer error rate dependent on k. For example for $k = 21$, each 100bp read contained 80 21-mers, of which the first 60 21-mers consisted entirely of bases with a slow error rate, the next 21-mer contained 20 bases with a slow error rate and 1 base with a fast error rate, and so on. An average per base error rate ε was calculated for each k, based on this model, and used in the predictions for the simulations. In empirical data, the error rate can be calculated from (well-calibrated) base quality scores.

1.5.2 Effect of error-cleaning, and errors which confound bubbles despite error-cleaning

In the simulations, graphs were cleaned by clipping tips of length at most $k+1$, and then removing supernodes where every interior node had coverage $\leq a$ (typically $a=1$, in the text referred to as 'relaxed', except for 'stringent' cleaning, where $a=2$). We took these factors into account to estimate the proportion of sites uncalled due to errors confounding the bubble.

Tip clipping: if *k*>*R*/2, then a single error in a single read cannot make a full bubble, and is guaranteed to be removed by tip-clipping. Thus in this case, for an error to escape tip-clipping, it must occur twice, and form a full bubble, requiring it to occur in one read in a position with more than *k* -1 correct bases after it, and in a second read in a position with more than *k*-1 correct bases before it – the simulations used k ≥ 21, which is longer than the section at the end of the read with higher error rate. Therefore, the probability of an error repeating in such a way as to avoid tip-clipping (conditional on there being two errors, one fast and one slow) is approximately (80-*k*+1)/80. Errors which evade tip-clipping by occurring twice (once fast and once slow) therefore occur at rate $\bar{\varepsilon} = \sqrt{0.001 \times 0.05}$.

Removing low-coverage supernodes: We include a simple term to account for the probability that a site becomes uncallable because of a repeated error, using the notation $\mathrm{dpois}(n,r)$ to denote the probability that a Poisson distribution of rate *r* takes value n, and $\mathrm{binom}(n,p,i)$ to denote the probability that a Binomial distribution with parameters n,p takes value i:

$$E = 1-(2k-2)\left(\sum_{i\geq 2}\mathrm{dpois}(i,D_{\mathrm{eff}})\sum_{n=2}^{i}\mathrm{binom}(\mathrm{i},\frac{2\bar{\varepsilon}}{3},n)\left(1-\frac{1}{2^{n-1}}\right)\right)\left(\sum_{j\geq 2}\mathrm{dpois}(j,\lambda k)(1-\varepsilon)^{\mu j}\right)\frac{80-k+1}{80}$$

The central bracketed term is the probability that the same error occurs twice at a specific base and is an error that will confound the site; if the true base is (for example) an A, then there are three possible error-bases (C,G,T) of which only two will make the bubble uncallable (the other will match the alternate/other allele; for the sake of example here, we say the other allele is C), thus giving us an error rate of $2\bar{\varepsilon}/3$. The sum over *n* computes the probability that two of the *n* errors that occur are the same, conditional on both of them being one of the two confounding bases. The factor $2k-2$ in front accounts for errors at all bases which might affect a k-mer from that site (k-1 bases before and after the variant site). The final bracket ensures there are no other errors at any other base within that errored *k*-mer (i.e. within k-1 bases of the repeated sequencing error). Otherwise the errored supernode would have coverage 1, and would be removed by error-cleaning. We denote μ for the distance on either side of the error that needs to be kept error-free – given the error profile we have, errors are much more likely to occur in the final 20 bases of a read, so it usually is not necessary to keep a full $k-1$ bases clear; we approximate with $\mu = 9$ (i.e. approximately half of the length of the high error-rate region). E is the same for both homozygous and heterozygous sites; whether reads come from the reference or alternate allele, there are still only 2 bases which confound the site, and in both cases the total expected coverage is $D_{\mathrm{eff}} = \lambda L$.

The above definition of E was used to plot Figure 2a (main paper), and clearly relied on a detailed error-model. However this is only needed to predict the behaviour for *k*<*R*/2,

where sensitivity drops well below the theoretical maximum (the genome complexity G) despite there being adequate coverage. Supplementary Figure 6, which shows the results of error-free simulations (solid lines) and model predictions with no E term (dotted lines) shows that for low k-mers, the theoretical maximum is attained, and that the model matches simulation. In simulations with sequencing errors, the drop in sensitivity for low k-mers is entirely due to repeated errors evading error-cleaning. For $k>R/2$, the effect of this E term is much more limited – without it the model overestimates sensitivity slightly (for k=55, overestimate by less than 5%).

2. Variant Calling Algorithms

2.1 Bubble Caller

The Bubble Caller was implemented as a traversal of the hash table (as a block of memory) rather than of the graph. Since hash access time is constant, traversal takes time proportional to the size of the table. Pseudocode for the algorithm follows – note the entire graph is covered and each node is visited at most twice.

```
compute_supernode (n, G)
        n is a node in a de Bruijn graph, assume color is specified. G is a de Bruijn graph

        path1<-traverse forward from n until reach node with outdegree!=1 or indegree !=1
        path2<-traverse backwards from n until reach node with outdegree!=1 or indegree !=1
        return union of path1 and path2



Bubble Caller (G)
        G is a de Bruijn graph implementation
foreach node n in G

        if outdegree(n) = 2 and not_visited(n)                  ### if we are at a new bifurcation
                mark_as_visited(n)                              ### mark it, in case we return to it
                (<n1,e1>,<n2,e2>)  ←  get_outedges(n,G)         ### get the two edges pointing out
                                                                ###  and the two nodes they lead to
                path1   ←  compute_supernode (n1,G)            ### get supernode in which n1 lies
                path2   ←  compute_supernode(n2,G)             ### get supernode in which n2 lies

                length1  ←  get_length(path1)
                length2  ←  get_length(path2)
                mark_as_visited(path1[0]…path1[length1-1])  ## mark both supernodes visited
                mark_as_visited(path2[0]…path2[length2-1])
                                                        ## if the two supernodes meet at the
                                                        ##  same node (and in the same
                                                        ## orientation, as nodes store both
                                                        ## a k-mer and its reverse complement)
                if ((path1[length1] = path2[length2]
                        AND (orientation(path1[length1])=orientation(path2[length2]) )
                        bubble_found = true
```

2.2. Path Divergence Caller Algorithm

The bubble-calling algorithm relies on the detection of clean bubbles where both alleles do not touch other parts of the graph. However, for complex variants (e.g. novel sequence insertions, large deletions and inversions), the path of at least one allele is unlikely to generate a clean contig. Nevertheless, in some cases, most notably where the variant is a deletion relative to the reference, the path complexity may be restricted to the reference allele. It is possible to identify such cases by following the path of the reference through the joint graph and identifying points where the reference path breaks away from the sample graph and then returns, requiring that both breakpoints lie on a single supernode in the *sample* graph. Since by definition there are no junctions within a supernode, it must constitute a single uninterrupted (phased) alternate allele (or short haplotype), whose length is constrained by the repeat content of the genome (i.e. the length distribution of supernodes). As a result this method can generate haplotypes much longer than the read-length or the insert size – for example the longest fosmid-validated PD call on NA12878 was over 9kb long (Supplementary Table 1). The algorithm is outlined in Supplementary Figure 2 – note the asymmetry between the green flanking regions with respect to the breakpoints, reflecting the fact that we take the entire supernode from the first breakpoint as an allele. We state the algorithm more precisely below.

Given a de Bruijn graph, a trusted path (e.g. a reference genome) represented as a sequence of nodes n(i) (i=1,2..), a specified length L of flanking sequence before and after a variant, a maximum size M of variant to be searched for, and a method for obtaining the supernode containing any node in the graph, the Path Divergence Caller applies the following algorithm:

```
Path_Divergence(L,M)
For (i=1; i<= length(P); i++)                                   ### for each node in the ref. path

{
 S<- compute_supernode_in_sample_color(n(i))                  ###  get the supernode in the sample
                                                                  ### color, which may be null if this
                                                                  ### node is absent in the sample
b<-get_first_position_where_reference_and_supernode_differ()

if ( S!=NULL AND  b>0 AND S(b-L-1+j)=n(i+j) for j=1..L )
                                                                   ## supernode and reference path
                                                                  ##  agree for L nodes
                                                                  ## prior to b, where they differ.
                                                                  ## This is one anchor/flank.
        {
                                                                  ## We want to take the whole supernode
                                                                  ##  as a potential alternate allele, so
                                                                  ## take the last L nodes of the supernode
                                                                  ## and compare them with blocks of L nodes
                                                                  ##  in the reference sliding along until we
                                                                  ## have gone as far as M
        for (j=i+L+1 to i+M)
                {

                if (S(length(S)-L+k)=n(j+k) for k=1..L)     ### We have a second anchor
                        {
                        variant_found=true;
                        break out of innermost loop.
                        }
                }
        }
}
```

3. Probabilistic Classification of Graph Structures in a Population

Given a multi-colored de Bruijn graph, with each color representing sequencing data from a single diploid individual from a single population, we have developed a probabilistic framework to classify a pair of paths as either alleles of a variant, (monomorphic) repeats or error, by comparing the likelihood of the sample graphs under three corresponding models. In the simplest case (and the only case considered here) these paths would be the two branches of a bubble, but the model does not require this.

There are three key sources of information about the nature of the bubble: the total coverage of the two sides, the 'average' allele-balance (the distribution of coverage between the two branches) across samples and the 'variance' in allele balance across samples. Bubbles arising from (monomorphic) repeats will have high coverage, typically intermediate balance and no variation in balance across samples. Bubbles arising from errors will have normal coverage, but consistently low coverage of the error branch across samples. Bubbles arising from variants will also have normal coverage, but allele balance will either be 0, 1 or ½ with probability arising from the population allele frequency and Hardy-Weinberg equilibrium. Within the context of the model-based classification we can separate the contribution to the overall likelihood arising from the combined coverage (modeled as an over-dispersed Poisson distribution) and from allele balance (from a binomial model). For each bubble type we independently specify priors for genome copy number (geometric for repeats, point mass at 1 for variant and error) and allele balance (beta distribution for repeat and error with parameter shared by all samples; as above for variant). The contributions to the overall likelihood from coverage and allele balance can be computed independently.

- Under the **Variation model**, each individual has 0,1 or 2 copies of allele1 (and 2,1 or 0) copies of allele2, with population allele frequency x determined by a neutral coalescent process and genotype determined by a Binomial(2,x) distribution.
- Under the **Repeat model**, allele balance, y, is the same for all individuals and drawn from a symmetric Beta $B(2,2)$ distribution.
- Under the **Error model**, one allele has very little support compared with the other in all individuals. The allele balance, y, is the same for all individuals and drawn from a Beta $B(100, 100\,\varepsilon)$ distribution (asymmetric, sharply peaked at 0, ε *is* the sequencing error rate).

For each model we compute the likelihood by integrating over the prior distribution. Models are compared by the use of Bayes Factors. A site is classified as Variant/Repeat/Error if the log-likelihood of the data under that model is at least 10 greater than the other models ($\log_{10}$ Bayes Factor at least 10).

4. Genotyping Algorithm, for simple and complex sites

The following algorithm assumes there is a multicolored de Bruijn graph, with one color for each known allele, one color for the reference genome (with the site in question, named X, excised), and one color for the sample. For any pair of alleles (paths) γ_1, γ_2, a likelihood function was defined as follows.

1. Ignore any segment of the paths that is present in the reference-minus-X color (less informative, as repeated elsewhere in genome).

2. Decompose both paths into sections shared between them s_i and sections unique to each u_i, and count the number of reads on each section.
3. Count the number of nodes n in both the sample graph and the joint graph of all known alleles of X, but not in either allele γ_1 or γ_2 - under our model these are sequencing errors.
4. Applying the ideas explained in Supplementary Methods Section 1.3, for each of these sections, with length l_i , the probability of r_i reads arriving within the section is given by a Poisson distribution with rate $Dl_i/R = \lambda l_i =: \vartheta_i$ on the shared (effectively homozygous) segments and $\vartheta_i/2$ on the (effectively heterozygous) sections unique to one or other path.

This leads to an approximate likelihood:

$$P(\text{genotype} = \gamma_1 \bigcup \gamma_2 \mid \text{data}) = \prod_{s_i} \vartheta^{r_i} \frac{e^{-\vartheta_i}}{r_i!} \prod_{u_i} \left(\frac{\vartheta_i}{2}\right)^{r_i} \frac{e^{-\vartheta_i/2}}{r_i!} S(n)$$

where S(n) is the probability that n sequencing errors within 2 contigs of length=length(γ_1) + length(γ_2) overlap the graph of alternate alleles. Note that this probability is much smaller than the raw sequencing error rate. One needs first to multiply by the probability that (conditional on having an error) the error overlaps the rest of the genome – this is 1 minus the genome complexity at the relevant k. Secondly, conditional on an error hitting the genome, we need the probability that it hits our locus, which can be approximated as length(locus)/length(genome).

4.1 Application to bubbles

This further simplifies when the two alleles are of the same length (eg for a simple SNP). At a bi-allelic locus, a likelihood ratio test statistic can be used to compare the likelihoods of the site being homozygous non-reference, or heterozygous; if the two alleles do not intersect themselves or each other (as, for example, at a bubble), and the alleles are of the same length (so $\vartheta_1 = \vartheta_2$), then the statistic simplifies to

$$\text{LRT} = \frac{P(\text{data} \mid \text{hom})}{P(\text{data} \mid \text{het})} = \frac{2^{r_{ref}+r_{alt}} S(r_{alt})}{\vartheta^{r_{alt}}/r_{alt}!}$$

5. De Bruijn Graph Software Representation

In Cortex, the de Bruijn graph is encoded implicitly within a hash table. Kmers are identified with their reverse complements and stored in the same node. For each kmer, one binary flag (1 bit) is used for each nucleotide to specify whether or not a corresponding edge is present. This reduces memory use from 8 bytes (1 pointer on a 64 bit architecture) per kmer, to half a byte. The hash-key is a binary representation of the kmer, and the hash-value an object representing the node (containing an array of 4-byte integers for coverages, and an array of 1-byte chars for edges, one for each color in the graph). The precise memory use per node, M (in bytes) encoding c colors is given by $M = 8\left\lceil \frac{k}{32} \right\rceil + 5c + 1$. Hash-table look

up time is constant (for a fixed table size). An API separates the details of the hash table from graph operations. Thus given a k-mer, functions are provided to allow fetching the corresponding node (and hence coverage, and edges), to traverse the graph in various ways, and to apply global operations such as cleaning the graph based on coverage and/or topology. Since hash look-up is constant-time, hash traversal is linear in the size of the hash; therefore it is possible to traverse the graph in linear time (see the Bubble Caller algorithm in Supplementary Methods Section 2). The hash table itself can be replaced without requiring any modification of the graph code. The current implementation uses a publicly available hash algorithm written by Bob Jenkins (http://burtleburtle.net/bob/c/lookup3.c).

5.1 Performance

There are two performance benefits of this implementation. Firstly, Cortex is the only assembler capable of simultaneously handling multiple eukaryote genomes. Secondly, performance (speed) is improved by the associated reduction of cache-thrashing, and by the fact that there is no costly communication across a network. Furthermore, by introducing a compact binary file format for de Bruijn graphs, we allow the graph building process to be parallelised across a compute cluster, in a manner that scales well. To give an illustrative but fair comparison, some care is needed as other assemblers often have a slow paired-end step following a single-ended step equivalent to that of Cortex. We assembled a k=31 binary of 26x of human data in 16 hours (and 64Gb of RAM) on a single core, whereas comparison with published figures suggest this step would take SOAPdenovo 40 hours (140Gb RAM)[3]. AllPaths-LG[4] would presumably be slower as it took 3.5 weeks (512Gb RAM) to assemble 86.5x of human sequence data.

6. Error-cleaning

Single individual: In addition to standard "tip-clipping" and removal of nodes with coverage below a threshold, a third form of error-cleaning is implemented, intended to remove sections of the graph which are more likely to have low-coverage caused by single-base errors than by random sampling. A single-base error will create a path of length at most k; therefore given a low-coverage supernode in the graph of length $\leq k+1$, a likelihood ratio test can be carried out, comparing the likelihood of this happening due to a sequencing error, and the likelihood of it being due to sampling of reads. The current implementation removes supernodes where the coverage on all internal nodes is below a pre-specified constant ('relaxed' default=1, 'stringent'=2). For a reasonable error-profile, such as an error rate of 3% in the final 20% of a read, and 0.1% in the first 80% of a read (averaging to an overall error rate of 0.007), with 30x coverage of a genome with 100bp reads, and k=55, the effective coverage $D_{\text{eff}} = \frac{R-k+1}{R} D = 13.8$. The likelihood of a contig of length 55 having coverage everywhere 1 due to sampling is therefore $(13.8 \times e^{-13.8}) \times e^{-55 \times 0.3} = 9.6 \times 10^{-13}$. By contrast the likelihood of this being caused by an error is $0.007(1-0.007)^{55} = 0.007 * 0.993^{55} = 4.8 \times 10^{-3}$. Simulations showed that, for 30x of 100bp reads and k=55, this approach allowed the Bubble Caller to call 82% of heterozygous SNPs, whereas removing all nodes of coverage 1 only allowed access to 62% - a 32% increase in sensitivity.

Population data: Where individual samples have insufficient coverage to apply the above method, we first pool all samples into a single graph, and clean that graph using the above method (removing errors at the price of losing low frequency variation – see

Supplementary Methods Section 13 for further discussion of this). Ideally, variants can be called directly on the pool, and then uncleaned individual graphs used to genotype individuals at the discovered sites. If the memory requirements for a multicolored graph of uncleaned individuals are too high, then individual samples can be cleaned by comparison with the cleaned pool (retaining only kmers that exist in the pool).

7. Single diploid genome simulations

Three altered copies (A,B,C) of the 247Mb human chromosome 1 (from the human reference genome NCBI36), were created. One of these (A) was to act as reference, and the other two (B,C) as the two copies in an individual. Variants were created by changing, deleting or inverting sections from the original chromosome and placing those changes in either one or two of A,B,C. In this way, homozygous and heterozygous variants were created in a ratio of 1:2. The following five variant types were included : isolated SNPs, "Clean" insertions/deletions (simply involving excision or addition of sequence), complex insertion/deletions (where between 2 and 5 SNPs (number chosen uniformly at random) were placed in the 20bp preceding a clean insertion/deletion), inversions, and insertions/deletions of Alu transposons (as annotated by Ensembl). In order to roughly approximate the expected distribution of variants sizes in a genome, four different size-intervals were considered, and variant-sizes were chosen uniformly within those intervals. When counting both the isolated SNPs and those in complex variants, this totaled an average of 1 SNP per 1.5kb. Finally, 15,068 of the variants were discarded as they had been randomly placed in regions where the reference chromosome was all N's. The number of variants left after discarding those in N's are below:

Type	*1-100bp*	*101bp – 1kb*	*1kb – 10kb*	*1kb-50kb*
SNP	*112481*			
Alu		*500*		
Clean indels	*18159*	*1816*	*14*	
Complex indels	*9110*	*907*		*20*
Inversions	*9099*	*185*	*15*	

Sets of error-free and error-containing reads of length 100bp (depth 10,20,30,40,50x) were produced. The following error-profile, designed as a crude approximation of the profile seen in Illumina sequencing data, was used : in the first 80bp of a read, the per-base error rate was 0.001, and in the final 20bp of a read, the per-base error rate was 0.05. This is comparable to the 0.005 mean sequencing error rate we estimate for our NA12878 data in Section 9 by comparison with HapMap homozygous-reference sites. For each combination of (k, coverage) Cortex binaries were built both of the reads (applying the error-correction method described above) and of the A chromosome, which was treated as a reference, and a two -color graph was generated (precise command-lines to do this with Cortex can be found in the Cortex User Manual, available at cortexassembler.sourceforge.net).
The Bubble Caller and Path Divergence Caller were first run with minimal constraints (BC: any bubble branch found in the sample must have at least one node with coverage >1, PD: anchor size=2 nodes, maximum variant size searched for=50kb) to determine maximum

sensitivity across all the combinations of coverage and kmer. To analyse the effect on FDR of error-cleaning (comparing the method described above versus removing all nodes of coverage 1) and constraints (median coverage on branch=2, cannot have both branches in reference, one branch must lie in reference) combinations of these options were run for k=55 and coverage=30x. Finally, the sensitivity simulations were re-run with error-free reads.

Cortex generates variants as sequence (flank, allele, allele, flank). In order to compare with the true set of variants, we could either compare with the haplotypes/chromosomes from which the reads were generated, or compare precise sequence around the known variant sites. Since

a) some of the variants created were complex, and we could not predict in advance whether Cortex would assemble from the positive or negative strand
b) in any real use-case, if one wanted to assign position to a variant, one would have to map it to a reference

in all cases, the number of variants assembled precisely correctly (both alleles) was determined by mapping flank+allele1+flank, and flank+allele2+flank of the calls to each of the original chromosomes using Stampy[5].

8. Population simulation

Twenty haplotypes were generated based on human chromosome 22 (length 35Mb), placing 35,746 SNPs uniformly across the chromosome according to a neutral coalescent allele frequency spectrum. Sites were treated independently (no recombination or LD). Unlike the single-individual simulation, which aimed to determine sensitivity to variants of different complexities and sizes, the goal of this simulation was to study power across the frequency spectrum, and the value of our population-based filter. Thus, SNP positions within a k-mer of a pre-existing SNP were rejected. Reads were generated twice from each haplotype (each time, 5x coverage per haplotype, 100bp reads), firstly using the same error-model as for the previous simulation, and secondly with no errors. A kmer of 55 was used for the entire simulation. Binaries were then built as follows

1. A binary was built directly from each of the original haplotype fasta , and then these were merged to form a pooled binary (representing the case of "infinite coverage")
2. A binary was built for each individual from the error-free reads, and then these were merged to form a pool. Finally, an 11-color graph was built containing the pool in color0, reads from haplotypes 0 and 1 in color 1, reads from haplotypes 2 and 3 in color 2, ..etc.
3. The same process as in step 2 was applied to the errored- reads.
4. An error cleaning process was applied to the graphs built from errored reads, as follows. Firstly, the pooled graph was cleaned by removing supernodes where all interior nodes has coverage $\leq$ a threshold (values 1 and 2 were used). Secondly, all the individual graphs were cleaned by taking the intersection with the cleaned pool (any node not in the cleaned pool was removed).
5. Bubbles were called on the pooled graph, using the other colors to annotate the output with amount of coverage in each of the colors for each of the alleles/branches.
6. Calls were mapped to human chromosome 22 using Stampy to compare with the "true" variants, and then converted to VCF format.
7. The population filter described in Methods Section 3 was applied as an R script.

9. Case 1: analysis of single high coverage human

9.1. Assembly and variant calling on NA12878

Sequence data from HapMap individual NA12878 (Sequence Read Archive Accession ERP000603), consisting of 26x of 100bp reads were assembled with Cortex (k=55), with on-the-fly removal of PCR duplicates (if both reads of a mate pair each started with a kmer that had previously been a read-start, they were both discarded), and breaking reads at bases of quality<=5. This took 14 hours on one core and used 144Gb of RAM. The quality-filtering of reads reduced coverage to 24.5x, and dropped the mean read length to 90bp. Error-cleaning was then applied, taking 10 hours, and producing a Cortex binary graph file. The human reference genome (NCBI36) fasta file was then read into Cortex, and a k=55 binary graph produced. These two graphs were then loaded into a multicolor graph, and variants were called with the Bubble Caller. A minimum flanking region of one node was required (55 base-pairs); the 5-prime flanks of the calls had min/mean/max length 55/450/1055bp. The Path Divergence Caller was also run, setting maximum variant-size of 100kb. The 5-prime flanks of these calls had min/mean/max length 56/480/16273 bp. The following filters were then applied to both BC and PD callsets.

1. For each call, the 5prime flank (as called by Cortex) was taken and if longer than 1kb the excess was trimmed. These flanks were mapped to the reference (NCBI36) and the call discarded unless the mapping quality was at least 30.
2. Calls with median coverage on either branch less than 2 were discarded.
3. Calls where all nodes on the reference allele occur more than once in the reference were discarded.
4. Calls where the mapper placed the flank somewhere where the subsequent bases did not match the reference allele were discarded.

Time taken for variant calling was approximately 24 hours, the performance bottleneck being writing output to the filesystem.

The above parameters (a read-length of 90 (average length of quality filtered reads), a k-mer size of 55) and an error rate of 0.5% (estimated from coverage of reference allele of HapMap2 homozygous non-reference sites) give an effective coverage $(1-\varepsilon)\frac{R-k+1}{R}D$, of 7.1/3.6 at homozygous/heterozygous sites. Applying our model (formula in section 1.4), allowing for genome complexity at k=55, these give predicted sensitivities of 87%/62% for SNPs. By comparison, the BC callset for NA12878 contains 87%/67% of the homozygous/heterozygous sites called by the 1000 Genomes on that individual.

9.2.Validation of NA12878 variant discovery

In order to validate all the variant types called by Cortex, and to make a fair comparison with the 1000 Genomes callset, both callsets were compared with full sequence from 82 fosmid clones (Genbank id: 29893)[6] covering 3.3Mb in total.

If the fosmids were randomly sampled from the genome, then of those variant calls within a section of the genome corresponding to one of the fosmids, 50% of true heterozygotes would be expected to have the alternate allele in the fosmid, and 100% of true homozygotes. However, it is known (Jeff Kidd, personal communication) that the fosmids were in fact selected from a library for Sanger sequencing on the basis of having evidence of existence of structural variation.

The following procedure was applied to both callsets:

1. Map the fosmids in 1kb chunks to reference, creating a mask over the genome.
2. For all calls flank+reference-allele+flank, and flank-alternate_allele-flank were mapped to the fosmids. A flank of 50bp was required, and if necessary, calls were clustered to enable this. For clustered 1000genomes calls (where most SNPs were phased (by a separate process) and no indels were phased), all possible alternate haplotypes were enumerated consistent with the specified genotypes.
3. For all calls within the mask, count occurrences where the alternate allele (or any of the alternate haplotypes, in the case of clustered 1000 Genomes calls) mapped perfectly to the fosmid, but the reference-allele did not.
4. For all calls within the mask, count occurrences where the reference allele mapped perfectly to the fosmid, but the alternate-allele did not.
5. Examine all other cases.

Main results can be found in Table 1 of the main paper. We note that selection for structural-variation containing fosmids has led to correlation selection for other types of variant in that consistently 70%, rather than the expected 50%, of variant alleles at sites called as heterozygous are present in the fosmid data. We also find cases where neither the reference nor alternative alleles are observed or else both alleles are observed in the fosmid data (see below).

9.3 Breakdown of Cortex complex variants in the fosmid mask:

In the main paper, Cortex calls were broken down into SNP, indel, and Complex. We further classify Complex calls according to whether they are collections of SNPs alone, or clusters of SNPs and indels. The validation results for complex calls are broken down in Supplementary Table 1. The 1000 Genomes callset is excluded from this table as the callset did not include these types of call.

Calls where it was not true that one of the reference/alternate alleles mapped perfectly, but the other did not, were analysed as follows. Calls were broken into the following cases: putative CNV (both reference and alternate alleles seen perfectly matched, in a fosmid), "Ref observed (with errors)", meaning the reference allele mapped to the fosmid with <=2 mismatches which might be ascribed to fosmid sequence errors, but the alternate allele mapped with more mismatches, "Alt observed (with errors)", as the last case but with the roles of reference ad alternate alleles reversed , "Ref observed with breakpoint insertion", where the reference allele maps perfectly to the fosmid except for a small (<20bp) amount of inserted sequence at the breakpoint, and "Alt observed with breakpoint insertion". The results are shown in Supplementary Table 2, for both Cortex and the 1000 Genomes callsets. We saw evidence for the existence of copy number variants in both 1000 Genomes and Cortex callsets, and also that a further 40 Cortex calls (not included in the main validation figures in Table 1) are likely to be correct and do not match due to single-base errors in the fosmids.

9.4.Validation of Cortex genotyping

The genotyping likelihood described in Supplementary Methods section 9 below, and in the main paper, can be used to compare the three possible genotypes at a biallelic site. Supplementary Tables 3-6 show that at HapMap2 sites called by Cortex, Cortex called non-reference genotypes where HapMap called homozygous reference less than 0.1% of the time, dropping to 0.03% at high confidence sites. The calls for NA12878 were genotyped by comparing homozygous non-reference and heterozygous genotypes. Each call was given a genotype confidence. For a homozygous call, this was the ratio of the likelihood of the data given the site was a true homozygote, divided by the likelihood of the data given the site was a true heterozygote. For a heterozygous call, it was the reciprocal of this. Stratifying calls by this Bayes Factor allowed further control of False Discovery Rate, at the cost of sensitivity. A high confidence set of calls was made by requiring a log(Bayes Factor) greater than 4, and by removing calls which had a homopolymer of length >3 at the junction of 5prime flank and either allele. VCFTools (vcftools.sourceforge.net) was used to compare genotypes with the HapMap2 calls for the same individual. Supplementary Tables 4 and 6 show that at Hapmap2 sites also called by the Bubble Caller/Path Divergence Caller, genotype concordance can be increased to over 99%/97% by stratifying by genotype concordance.

9.5.Complex variants

The two alleles of all Cortex variant calls were aligned to each other using the Needleman-Wunsch algorithm (using the Algorithm::Needleman-Wunsch perl module from cpan.org) in order to determine if the call consisted of a collection of SNPs and indels, or was best described as one larger event. For each callset, two VCF files of variants were produced. One listed the variants as called by Cortex (.raw.vcf), which therefore had some long phased collections of SNPS and indels as single alleles. The other decomposed all calls as far as possible (.decomp.vcf), allowing better comparison with other callsets. Those which were not simple insertions or deletions, and were not phased collections of SNPs, were termed Complex.

9.6.Complex variants displayed in Figure 3b of main paper

Two complex variants were depicted in Figure 3b of the main paper. The first was called by the Path Divergence Caller, id: na12878_chr2_var_28881 which matches perfectly fosmid with clone id: ABC12-47840300E24. The second variant was called by the Bubble Caller, id: na12878_cortex_bubble_het_var_1242923, and perfectly matches fosmid id: AC205937.3. The precise alleles can be found in the .raw VCFs also released with this paper.

10. Case 2: Pooled Assembly of individuals from the 1000 Genomes Project low coverage pilot.

10.1 Assembly

All Illumina sequence data from the 1000 Genomes Low Coverage Pilot was obtained (164 individuals across 3 populations: CEU (49 individuals, of which 21 were males, 2.7x per

person,), YRI (57 individuals, of which 22 were males, 3.8x per person) and CHB/JPT (58 individuals, of which 25 were males, 2.2x per person) – this consisted of 1.9Terabases of sequence. The analysis was restricted to one sequencing technology to allow better estimates of allele frequency/copy number. A kmer-size of 29 was used. With 36bp reads, this gives an effective coverage for a singleton of $\frac{36-29+1}{36}D$, which is 0.3 for CEU, 0.4 for YRI and 0.2 for CHB/JPT. Thus this analysis has low power for low frequency variants, and we expect not to see singletons or doubletons. Although dropping the kmer would have increased this sensitivity, it would have come at a cost in graph complexity (see Supplementary Figure 4). The sequencing data from 1000 Genomes Pilot 1 is approximately 18 months older than that used elsewhere in this paper, and was of noticeably lower quality. A higher level of sequencing errors per individual led to a higher memory demand, and therefore correspondingly harder levels of filtering were required. The assembly was done in the same manner for all populations, as follows:

1. Assemble each individual using a quality filter threshold of 10 (reads were cut at a base of quality below 10, and both halves treated separately).
2. The individuals were split into two groups, and these merged into two pools. Each pool was cleaned by removing all nodes of coverage 1, and then the two pools merged to form a population graph.
3. The human reference genome NCBI36 was built as a binary.
4. A multicolor graph consisting of the reference (color 0), CEU (color 1), YRI (color 2) and CHB/JPT (color 3) was built.

These processes were run on 256Gb RAM server. Individual assemblies were run in parallel, 3 processes at a time, each process requiring 75Gb of RAM. Maximum memory use was during the merging of pools, which required 240Gb of RAM before cleaning. The final multicolored graph requires 165Gb RAM, and is stored as a binary file, 88Gb in size, and is released with this paper.

10.2 BLAST annotation

All contigs were BLAST-ed against all known sequence (using the NCBI nr database), and contaminants (non-primate sequence) removed. All remaining sequence matched either a primate BAC/fosmid/clone, or the HuRef assembly. These 24,277contigs alone were previously released in this form by the 1000 Genomes Project, as part of the pilot paper.

10.2 Further filtering and annotation

Novel contigs were annotated based on BLAST for several categories (HLA, cDNA/mRNA, olfactory, KIR). Contigs were also labeled as autosome, X or Y depending on closest BLAST match. Additional HLA annotation was done by downloading all IMGT HLA alleles (from ftp://ftp.ebi.ac.uk/pub/databases/imgt/mhc/hla/) and mapping novel contigs to them using Stampy. Any novel contigs mapped with mapping quality ≥ 30 were labeled HLA.
By measuring the length of the sequences to which these contigs matched, we calculated that those annotated as genic (HLA, KIR, mRNA, cDNA) had median distance from a gene of less than 2kb. Contigs with IGH annotations (either from BLAST or the IMGT) were filtered out, as these are likely due to somatic rearrangements (1000 Genomes sequencing was done from B-cells). After filtering, 21,281 contigs remained.

10.3. Calibration of model using HapMap

The formula described in section 1:

$$D_{\text{eff}} = (1 - k\varepsilon)\frac{R - k + 1}{R} D$$

described the expected depth of coverage at a single node (kmer) in the graph, due to coverage from a single locus in the genome with depth D; each extra copy of this sequence in the genome increases the expected coverage in the graph by D_{eff}. In the context of a pooled population, if a contig of length l base-pairs exists N times in the population, and each chromosome in the population is sequenced to depth d, then we recast the formula as:

$$D_{\text{eff}} = \frac{l - k + 1}{R}(1 - k\varepsilon)Nd$$

Since R,k,l,d are known, and D_{eff} can be estimated as the observed coverage on the contig, this allows one to estimate the copy number of a contig (N) from its coverage in the graph, provided the sequencing error rate ε is known. We therefore estimated a per-population error rate by comparison of HapMap SNPs with the graph, as follows:

1. We downloaded, for populations CEU, JPT,CHB,YRI, and chromosomes 1, X & Y: http://hapmap.ncbi.nlm.nih.gov/downloads/frequencies/latest_phaseII_ncbi_b36/fwd_strand/non-redundant/allele_freqs_chrN_POP_r24_nr.b36_fwd.txt.gz
2. Using these HapMap sites (with frequency information) and the human reference (NCBI36) we created a FASTA file for each chromosome-population pair, consisting of each SNP and 28bp (kmer size = 29) either side of it. These contigs represent the branches of the bubble created by a SNP. We aligned these to the four-colored cortex binary containing the reference and the merged populations, and printed the observed coverage on each allele for each population.
3. We excluded those HapMap SNPs where either branch included a kmer which occurred more than once in the reference, and those where the alt allele existed elsewhere in the reference, as these would bias our sequencing error estimates.
4. We then plotted (for ~269,000 SNPs on chr1) the HapMap population frequency versus the coverage for both alleles in each population in the de Bruijn graph. For each population, ε was estimated to make the best fit between the straight-line from (0,0) to (1, D_{eff}) and a rigid linear fit done in R using rlm() from Venables and Ripley's MASS library. This gives estimates of $\varepsilon = 1.3/2.1/1.6$ for CEU, YRI and CHB/JPT respectively. Having fixed the sequencing error rate, the equation above gives copy-number estimates per population (which, if divided by the number of chromosomes in the population, correspond to allele frequency estimates) for the HapMap SNPs; these estimates are noisy but nevertheless unbiased.

10.4 High differentiation contigs

The following contigs all match a 9K2-like olfactory receptor gene transcript, and are present only in YRI:
supernode_564612 (length: 829bp, frequencies CEU 0, YRI 0.21, CHB/JPT 0),
supernode_1168224 (length: 148bp, frequencies CEU 0, YRI 0.22, CHB/JPT 0),
supernode_1386395 (length: 235bp, frequencies CEU 0, YRI 0.18, CHB/JPT 0).

11. Case 3: Using population information to classify bubble structures

11.1 Assembly of 10 chimpanzees

Sequence data from ten Western chimpanzees, (identifiers PtAc, PtFr, PtGi, PtLa, PtLi, PtPe, PtRg, PtRn, PtSu, PtYo) was obtained from the PanMap project (http://panmap.uchicago.edu/). There was approximately 6x depth for each individual at the time of analysis, with 50bp Illumina reads. A kmer-size of 31 was chosen to take advantage of the fact that the graph complexity was expected to reduce significantly between 21 and 31 (see Supplementary Figure 4 to see this plotted for human, which we expect to be similar). At kmer 31, a singleton SNP would be expected to have coverage $\frac{R-k+1}{R}D = \frac{(50-31+1)\times 3}{50} = 1.2$. Pushing the kmer higher would risk completely removing singletons.

A 12-color multicolor graph was then assembled and variant calls made as follows
1. A graph was built from each sample, with no quality-filtering or PCR duplicate removal.
2. These graphs were merged into a population pool graph, which was cleaned by tip-clipping and then removing supernodes with all interior nodes of coverage 1.
3. The individual graphs were then cleaned by comparison with the pool.
4. The Chimpanzee reference genome PanTro2 was built as a graph.
5. A single multicolor graph was built from the reference (color 0), the pool (color 1) and then each of the samples.
6. Variants were called on the pooled color (ie Cortex only "looked at" the pool color, and did not use the reference in any way), and then the other colors were used simply to annotate the variant calls.

There were 3,583,205 variants called, of which 2,961,307 were SNPs and 34,667 were complex. Analysis for this paper was restricted to the SNPs.

Finally, the effects of the population filter (remove any call where likelihood of Variation model less than 10 times greater than that of the Error and Repeat models) and reference filters (remove bubbles where both branches are present in the reference) were compared.

12.Case 4 – Genotyping simple and complex variants

12.1.HLA-B data preparation for genotyping

The full set of 1429 known alleles of HLA-B were downloaded (as FASTA files) from the IMGT database[7]. These alleles consist not of full sequence for the entire gene, but just for the two most polymorphic exons, and furthermore were of different lengths. These alleles were therefore all mapped to the reference chromosome 6, and the position *P* of the furthest 5' base, and *Q* of the furthest 3' base were found. In order to create a set of paths that all started and ended at the same place, "extended alleles" were created, which all started at *P* and ended at *Q*, with padding sequence added from the reference sequence if necessary. These extended alleles were approximately 86kb long.

Sequence data for HapMap individuals NA19240 and NA12878 was used, as their HLA type had been experimentally determined already with PCR-SSOP protocols[8]. The data for NA19240 was generated as part of the 1000 Genomes Project, and that for NA12878 was produced at Oxford by the Wellcome Trust Sequencing Core. The data for NA19240 consisted of 33x coverage, split into 25x of Illumina reads and 8.1x of 454 reads, and was built into a kmer=29 binary. The data for NA12878 consisted of 20x of 100bp Illumina reads; by subsampling from the fastq for NA12878, graphs were built with 2x,4x,…20x coverage, using kmer=55.
A joint graph was built, with the reference genome (with HLA-B excised) in one color, the sample in the next color, and one color for each of the HLA-B alleles. (In the case of NA12878, the binary graph corresponding to each of the subsampling-levels was loaded into a color of its own.)
For each genotype (which consisted of two paths, one for each allele) the likelihood of seeing the sample graph was calculated as in the Methods Section 4. As described there, the probability $S(n)$ that sequencing errors could generate n nodes outside the desired genotype is modeled as a Poisson process with rate equal to sequencing error, times (1-genome complexity), times length(HLA-B)/length(genome).

For NA19240, we used kmer 31, genome complexity $G(31) = 0.75$, and so

$$\text{rate} = \varepsilon(1 - G(31))\frac{\text{length(allele)}}{\text{length(genome)}} = 0.25 \times \frac{86000}{3000000000}\varepsilon = 7 \times 10^{-6}\varepsilon$$

.
Furthermore, we used a cleaned graph for NA19240 (nodes of coverage 1 had been removed), so any error that occurred had to have occurred twice. Therefore we expect the rate to be between 7×10^{-8} and 7×10^{-10}. We repeated the following process for several ε within this range and all gave the same maximum likelihood genotype.

For NA12878, we used kmer 55, genome complexity $G(55) = 0.87$, and an uncleaned graph, so

$$\text{rate} = \varepsilon(1 - G(55))\frac{\text{length(allele)}}{\text{length(genome)}} = 0.13 \times \frac{86000}{3000000000}\varepsilon = 4 \times 10^{-6}\varepsilon$$

Thus we expected a rate of approximately 10^{-8} would be appropriate; however since we found the likelihood was extremely flat, we increased the penalty on errors, dropping this rate further to 4.3×10^{-10}.

The calculation was parallelised over 150 compute CPUs, each process requiring 3Gb/12Gb of RAM (for NA19240 alone/for 10 different subsamplings of NA12878 simultaneously), and total elapsed time for the computation being approximately 6 hours.
The genotype that maximized the likelihood for NA19240 was B*35:01:01/B*57:03:01, which agrees to 4 digits with the experimental typing. The likelihood ratio in comparison with the next most likely genotype (B*35:01:01/B*57:02) was $\sim 10^{23}$.
The genotype that maximized the likelihood for NA12878 varied with coverage; 16x of coverage was required for the maximum likelihood genotype to agree to 2-digits with the lab-based estimate of type. At 20x, B*08:03/B*56:01:01 was the maximum likelihood genotype, but B*08:13, B*08:15, B*08:36 and B*08:47 were all close candidates for the first

allele, all less than 1 unit of log likelihood below. The lab-calculated type, B*56:01/B*08:01 is marked with a dotted line in Figure 5 of the main paper.

13. Making appropriate choice of k, read-length and depth for experimental design.

The methods described in this paper allow one to predict discovery power of variants of a given length based on genome complexity, read-length, kmer-size, error-rate, and depth of coverage. Assuming that one has already estimated the genome complexity for the species in question (see Supplementary Methods Section 1.2), this determines an upper bound for sensitivity of the Bubble Caller dependent on kmer, irrespective of coverage and read-length. Given this, one can tailor an experiment to the type of variant discovery desired. We give some examples of putative experimental goals.

13.1 Aim: to find as many SNPs and small indels as possible in a single diploid genome

1. Examine the genome complexity curve for different k. For the species in question, there may be some threshold below which power drops dramatically. E.g. in human, there is a significant drop between k=31 and k=21. Having chosen a minimal acceptable k, this implies a minimal acceptable read-length (which must be longer than k). Further, power is affected by effective depth of coverage in the graph, $D(R-k+1)/R$ - thus there is incentive to increase read-length well beyond k.
2. Figure 2a of the main paper shows that if k is less than half the read-length, sequencing errors cause more problems than if k>R/2. This is simply because for k<R/2, a single error in a single read can form a full bubble in the graph, whereas for longer k, an error must form a tip, unless it occurs more than once. Thus ideally one would prefer a read-length not more than double the kmer-length.
3. Apply the power formula from Supplementary Methods Section 1.4 to determine the cost-benefit of different levels of coverage

13.2 Aim: to find large structural variants in a single diploid genome

The procedure is as for SNPs, but one sees there is a clear benefit to longer reads, both in the effect on the genome complexity (see Supplementary Figure 4 which uses the genome complexity for chromosome 1 as if it were a whole genome), and because the Bubble Caller power drops exponentially with variant length (for fixed coverage) – see the formula in Supplementary Methods section 1.4.

13.3 Aim: to explore diversity in a new species for which there is no reference genome, from sequencing of a population

The main consideration here is that the neutral frequency spectrum leads us to expect the majority of variation will be low frequency. It is therefore important to design the experiment in such a way that removal of errors does not kill all low frequency variation. This is most simply seen with two contrasted examples.

13.3.1 Example: 100 diploid individuals sequenced to 4x each.

At a given position in the genome, we expect 400x depth of coverage. With a sequencing error rate of 0.01, this means we expect 4 errors at each position in the genome; since there are 3 (2) possible errors at a monomorphic (biallelic) site, we expect at all positions in the genome, one error per position will have reinforced itself by occurring twice.
If we were to choose a read-length of 100bp and a kmer-size of 65, then we would have expected depth of coverage at a singleton of

$$\frac{R-k+1}{R}D=\frac{100-65+1}{100}\times 2=0.7.$$

Thus singletons, and in fact all variants with minor allele count <3, will have lower coverage in the graph than errors. If we clean errors by removing supernodes with coverage everywhere less than 3, would should lose the vast majority of errors, but we will also lose alleles with MAF<=5/200=2.5%.
We could ameliorate the effect by dropping k to 51 (just above half the read-length), increasing the expected coverage at a singleton to:

$$\frac{R-k+1}{R}D=\frac{100-51+1}{100}\times 2=1$$

Firstly – this means many singletons will be lost due to sampling alone. Secondly, cleaning supernodes with coverage less than 3, loses alleles with MAF<=2/100=2%.

13.3.2 Example: 50 diploid individuals sequenced to 8x each.

At a given position in the genome, we still expect 400x depth of coverage. With a sequencing error rate of 0.01, this means we expect 4 errors at each position in the genome. As above, since there are 3 (2) possible errors at a monomorphic (biallelic) site, we expect at all positions in the genome, errors will have reinforced themselves by occurring more twice. However now, the expected coverage at a singleton, given read-length 100 and kmer 51 is:

$$\frac{R-k+1}{R}D=\frac{100-51+1}{100}\times 4=2$$

So now we have parity between singletons and the worst errors, so cleaning supernodes with coverage 1 will lose variants with MAF<1/50=2%.

It is not necessary to be able to remove all sequencing errors (we believe the population filter is very effective at identifying errors) – the idea is simply to clean enough off to allow variant discovery to work. Thus ideally we would do two rounds of variant-calling: firstly after removing supernodes of coverage 1, and then after removing supernodes of coverage

2, and taking the union of these calls forward for filtering by the population filter and then genotyping.

14.Cortex details and data availability

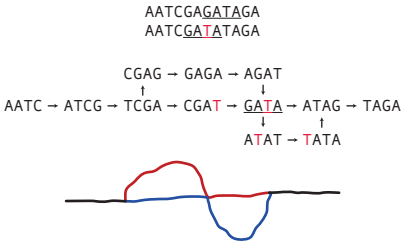
The Cortex algorithms described in this paper are implemented in an executable called cortex_var; the version used for this paper, along with a User Manual and demonstration examples can be obtained from https://sourceforge.net/projects/cortexassembler/files/cortex_var/publication_archive/2011_10/cortex_var_release_for_paper_201110.tgz.
Up-to-date versions of Cortex can be obtained from http://cortexassembler.sourceforge.net.
The variant calls made on NA12878 (in VCF format), the population graph assembled from 164 individuals in the 1000 Genomes low coverage pilot, along with novel contigs and copy number estimates, can be found at ftp.1000genomes.ebi.ac.uk:/vol1/ftp/pilot_data/paper_data_sets/companion_papers/cortex_de_novo_assembly/.

References

1. Lander, E.S. & Waterman, M.S. Genomic mapping by fingerprinting random clones: a mathematical analysis. *Genomics* **2**, 231-9 (1988).
2. Leeuwaarden, J.S.H.v., Lopker, A.H., Janssen, A.J.E.M. Connecting Renewal Age Processes with M/D/1 and M/D/∞ Queues Through Stick Breaking. *Stochastic Models* **26**, 141-163 (2010).
3. Li, R. *et al.* De novo assembly of human genomes with massively parallel short read sequencing. *Genome Res* **20**, 265-72 (2010).
4. Gnerre, S. *et al.* High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proc Natl Acad Sci U S A* **108**, 1513-8 (2011).
5. Lunter, G. & Goodson, M. Stampy: A statistical algorithm for sensitive and fast mapping of Illumina sequence reads. *Genome Res* (2010).
6. Kidd, J.M. *et al.* A human genome structural variation sequencing resource reveals insights into mutational mechanisms. *Cell* **143**, 837-47 (2010).
7. Robinson, J. *et al.* IMGT/HLA and IMGT/MHC: sequence databases for the study of the major histocompatibility complex. *Nucleic Acids Res* **31**, 311-4 (2003).
8. Leslie, S., Donnelly, P. & McVean, G. A statistical method for predicting classical HLA alleles from SNP data. *Am J Hum Genet* **82**, 48-56 (2008).

a

b

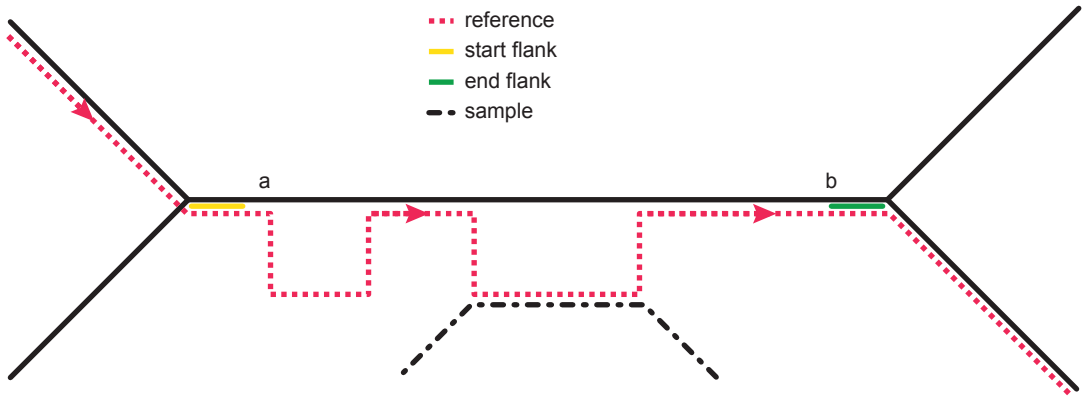


c

d

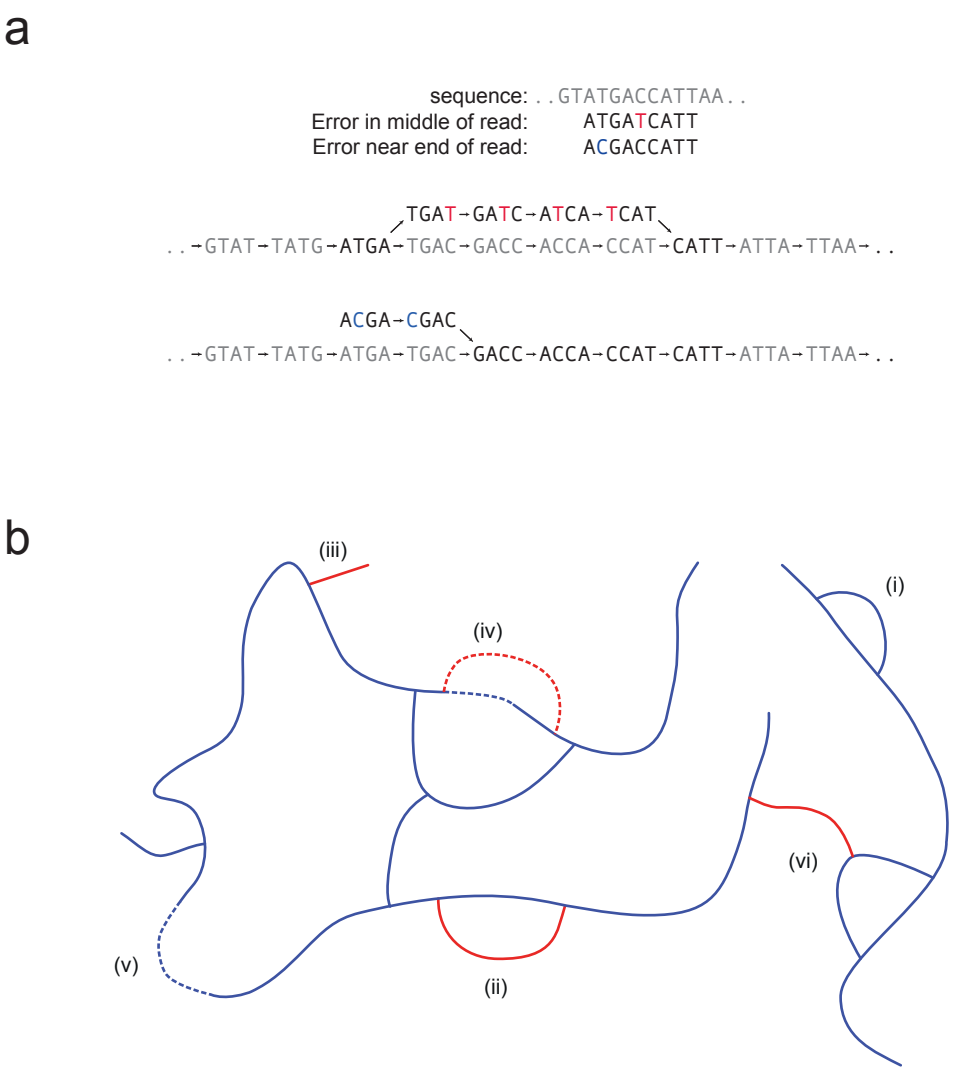
Supplementary figure 1

Examples of typical structures in de Bruijn graphs arising from genetic variation. **(a)** shows a simple SNP, first as sequence, then broken down into successive 4-mers, and finally converted into a line-representation of the graph. This example conforms to the generic case, where a SNP forms a bubble of length k. In regions of low complexity, it is possible for a SNP to form a bubble of length <k, or even more than one bubble, as shown in **(b)**. If variants are clustered closely they form a single compound bubble, as shown in **(c)**, where two SNPs and a deletion combine to form one bubble. Finally, in **(d)**, an example is given of a polymorphism that overlaps with another part of the genome (at k-mer GCTC), thus forming a confounded bubble.



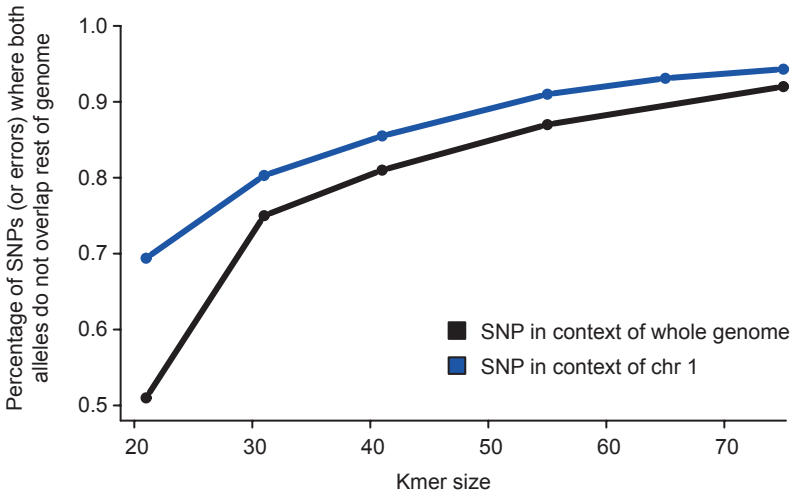
Supplementary figure 2

Key steps in the Path Divergence caller algorithm. The sample de Bruijn graph is shown in black. As the reference-path (red) is followed, we identify the point a, where the reference breaks away from the sample, and a 5-prime flank (yellow) is noted. Since the sample graph has an unbroken supernode as far as b, this sequence is all taken as the alternate allele, and therefore the desired 3prime flank is marked (in green). The reference path is then followed until it returns to hit the green flank, at which point we have determined the reference allele. The algorithm is not affected by the reference allele touching other parts of the sample or reference graphs (black dotted).



Supplementary figure 3

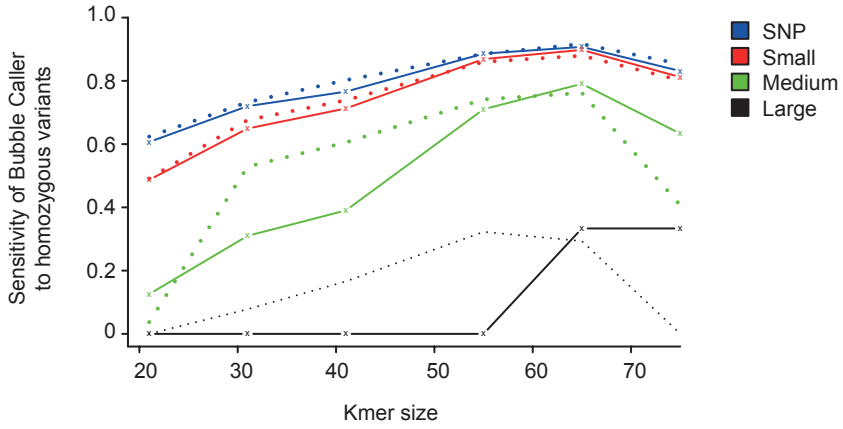
Errors and error-cleaning. **(a)** A single error that occurs once in the middle of a read is able to form a complete bubble (provided k is less than half the read-length), whereas an error at the start or end of a read must form a tip. **(b)** The effect on a graph of different error-cleaning strategies; true sequence is marked in blue, errors in red, and low coverage (=1) marked with dotted lines. In the graph as it stands, there are two callable bubbles: (i) is a true variant, whereas (ii) is due to a repeated error at a monomorphic site (coverage is higher at monomorphic sites, and repeated errors more likely). If we remove tips, very little is affected except for the tip marked (iii). If we were to remove all nodes with coverage 1, then we would remove the false branch (iv) at the price of creating a gap in the middle of the real allele at (iv), and in a real contig at (v), and the false bubble (ii) remains. However, if we remove entire supernodes if all nodes within have coverage 1 ("relaxed" cleaning), then the true allele at (iv) is preserved (for moderate coverage, it is very unlikely there would be low coverage on the whole allele) while the false one is deleted, and the low coverage segment at (v) is preserved. This still leaves the error at (ii) and the supernode at (vi) which confounds a true bubble by connecting with another part of the genome. If the maximum coverage on the error supernodes (ii) and (iv) is 2, then "stringent" cleaning (removing supernodes with coverage everywhere ≤2) eliminates these errors and makes visible the variant bubble at (vi).



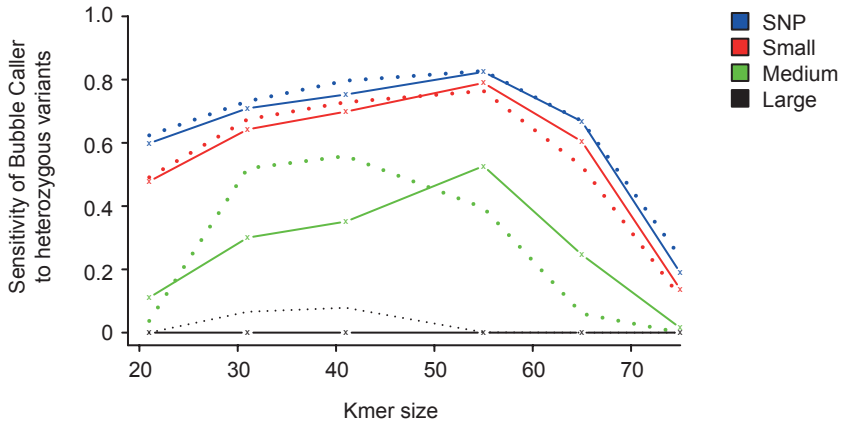
Supplementary figure 4

Genome complexity (fraction of cases where a variant generates an unconfounded bubble) for SNPs in the human genome (black) and in the context of chromosome 1 only (blue; used for simulations). At k=21, 51% of possible SNPs are callable, rising to 92% at k=75. In contrast, for chromosome 1 (treated as a genome in its own right), for k=21, 69% of possible SNPs form unconfounded bubbles. As k increases the complexities of chromosome 1 and the whole human genome approach each other.

a



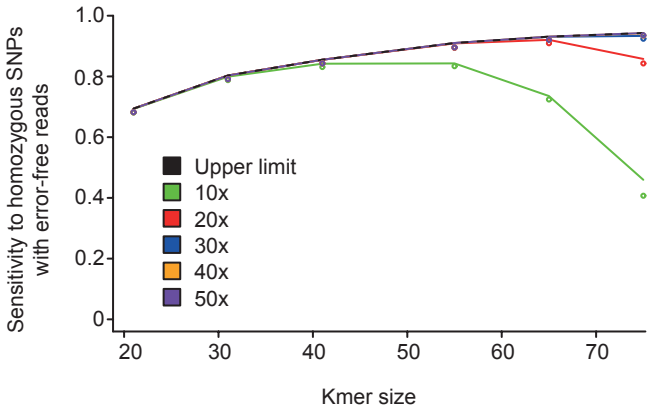
b



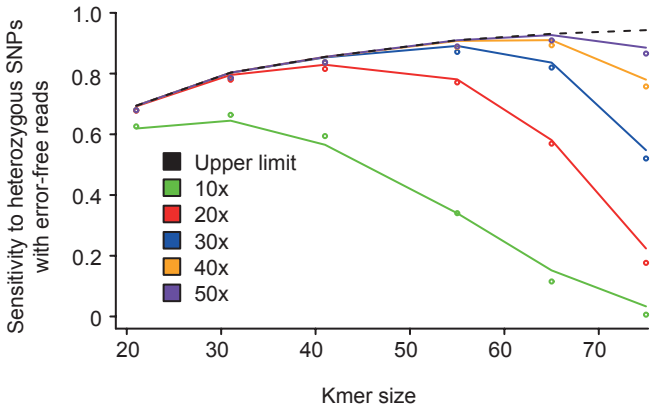
Supplementary figure 5

Comparison of predicted sensitivity and simulation results for homozygous and heterozygous variants of different sizes (Small: 1-100bp, Medium: 100bp-1kb, Large: 1001bp-50kb). Estimation of genome complexity for longer variants is more difficult than for SNPs, but nevertheless, the predictions (dotted lines) do approximate the simulation results. For SNP (blue) and small (1-100bp) indels (red) the correspondence is close, and improves with kmer, as the effect of long-range interactions between errors is reduced, and the estimate of genome complexity is better. Simulations have 30x 100 bp reads and include errors as described in the Methods.

a



b



Supplementary figure 6

Comparison of predicted and actual power to detect SNPs in simulations without errors. For low k-mer sizes, simulations and predictions both show that sensitivity attains the theoretical maximum (black), corresponding to the genome complexity function . As k increases, so does the theoretical maximum, but effective coverage drops, driving a reduction in sensitivity. Increasing sequence coverage can compensate for high k-mer values, and pushes sensitivity up to the theoretical maximum, for all k-mer values. This figure contrasts strongly with Figure 2 where for k-mer size below half the read-length, increasing coverage leads to reduced sensitivity; a result driven by sequencing errors. The curves in this figure represent a theoretical maximum achievable for error-correction algorithms.

Supplementary Tables

Supplementary Table 1. Breakdown of complex variants in Cortex calls on NA12878

Complex call type	Ref observed	Alt observed	Estimated FDR	Size of observed alt. alleles (median/max/std dev)
BC phased SNPs (hom)	3	82	3.5%	30/106/22
BC phased SNPs (het)	37	93	-	
PD phased SNPs (hom)	3	113	2.6%	287/4712/801
PD phased SNPs (het)	0	0	-	
BC SNPs+indels (hom)	0	27	0%	25/96/22
BC SNPs+indels (het)	5	28	-	
PD SNPs+indels (hom)	0	58	0%	1126/9040/1614
PD SNPs+indels (het)	0	0	-	

Supplementary Table 2. Breakdown of calls where fosmids show evidence of CNVs, or fosmid-errors, in 1000Genomes and Cortex calls

Variant Type	Putative CNV (1000g/Cortex)	Ref observed (with errors) (1000g/Cortex)	Alt observed (with errors) (1000g/Cortex)	Ref observed with breakpoint insertion (1000g/Cortex)	Alt observed with breakpoint insertion (1000g/Cortex)
SNP	35/0	0/10	0/31	0/1	0/5
Indel	25/66	0/2	0/0	0/0	0/0
Phased SNPs	0/0	0/0	0/4	0/0	0/1
Clustered SNP and indels	0/0	0/0	0/5	0/0	0/1

Supplementary Table 3: Genotype concordance of Bubble Caller main NA12878 callset with Hapmap2

	Bubble Caller R/A	Bubble Caller A/A
HM2 R/R	0.5%	0.1%
HM2 R/A	99.1%	9.9%
HM2 A/A	0.4%	90.0%
Total calls	501953	637734

Supplementary Table 4: Genotype concordance of Bubble Caller high confidence NA12878 callset with Hapmap2

	Bubble Caller (HC) R/A	Bubble Caller (HC) A/A
HM2 R/R	0.5%	0.04%
HM2 R/A	99.0%	1.2%
HM2 A/A	0.5%	98.7%
Total calls	460687	411898

Supplementary Table 5: Genotype concordance of Path Divergence caller main NA12878 callset with Hapmap2

	Path Divergence R/A	Path Divergence A/A
HM2 R/R	0.4%	0.09%
HM2 R/A	96.7%	23.1%
HM2 A/A	2.9%	76.8%
Total calls	694	151723

Supplementary Table 6: Genotype concordance of Path Divergence caller high confidence NA12878 callset with Hapmap2

	Path Divergence (HC) R/A	Path Divergence (HC) A/A
HM2 R/R	0.4%	0.02%
HM2 R/A	93.2%	2.2%
HM2 A/A	6.4%	97.8%
Total calls	249	74169

Supplementary Table 7: Proportion of HapMap2 sites with at least one alternate allele called by Bubble Caller

	BC	BC (high confidence)
HM2 (het)	69.9%	49.7%
HM2 (hom-alt)	86.5%	49.9%