A natural encoding of genetic variation in a Burrows-Wheeler Transform to enable mapping and genome inference

Sorina Maciuca, Carlos delOjo Elias, Gil McVean, and Zamin Iqbal

Wellcome Trust Centre for Human Genetics,
University of Oxford, Roosevelt Drive, Oxford OX3 7BN, UK
{sorina.maciuca,gil.mcvean,zamin.iqbal}@well.ox.ac.uk
{carlos.delojoelias}@ndm.ox.ac.uk

Abstract. WABI We show how positional markers can be used to encode genetic variation within a Burrows Wheeler Transform (BWT), and use this to construct a generalisation of the traditional "reference genome", incorporating known variation within a species. Our goal is to support the inference of the closest mosaic of previously known sequence to the sample under analysis.
This encoding results in an increased alphabet size, and by using a wavelet tree encoding of the BWT we reduce the performance impact on rank operations. We give a specialised form of the backward search that allows variation-aware exact matching. We implement this, and demonstrate the cost of constructing an index of the whole human genome with 80 million genetic variants is 31Gb of RAM, below that of previous implementations. We also show that inferring a closer reference can close large kilobase-scale holes in mapping pileups in *P. falciparum*

Keywords: pan-genome, Burrows-Wheeler Transform, FM index

1 Introduction

DNA molecules can be modelled as strings drawn from an alphabet of four characters: A,C,G,T. Genome sequencing seeks to infer the string that makes up a specific biological sample.This process involves breaking the DNA into smaller fragments, identifying substrings (called "reads") and then inferring properties of the genome (ideally the original string). Thus a sequencing experiment generates a set of oversampled substrings from a single long string (4 million characters (or "bases") for bacteria, or 3 billion bases for humans), with errors dependent on the technology. Recently, sequencing has dropped in price and it has become possible to study within-species genetic variation on a large scale [6–8], where the dominant approach is to match substrings to the canonical "reference genome" which is constructed from an arbitrary individual. This problem ("mapping") has been heavily studied (see [5]) and there are now extremely efficient tools to do it. The use of the Burrows Wheeler Transform, a well known algorithm from computer science originally designed for compression [2], underlies

the two dominant mappers - bwa [3] and bowtie [4]. Mapping reads to a reference genome is a very effective means of detecting genetic variation involving single character changes (SNPs - single nucleotide polymorphisms). However this method becomes less effective the further the genome differs from the reference. Many important regions of genomes are highly diverse, and so sequence reads from one individual can very easily fail to map to the reference. Generally most species have some regions like this and in bacteria this can make use of a single reference highly problematic. Furthermore, many natural experiments, such as sequencing the bacteria infecting a patient, or sequencing cancer, fundamentally and unavoidably involve sequencing a mixture of different genomes in unknown proportions.

Although the idea of encoding many genomes together as a graph is becoming of wider interest, there is no consensus yet on how they are to be used. In offering a new data structure, we need to outline an entire workflow to explain how this could be used. We want to build a representation of the genome of a species which incorporates N genomes and supports the following inference. We take as input, sequence data from a new sample of our species, and an estimate of how many genomes the sample contains and their relative proportions - e.g. a normal human sample would contain 2 genomes in a 1:1 ratio, a bacterial isolate would contain 1 genome, and a malaria sample might contain 3 genomes in the ratio 10:3:1. We would then infer the sequence of the underlying genomes. In this paper we describe a a method for encoding genetic variation designed to enable this approach.

Our data structure is motivated by a biological observation. Genomes evolve (broadly speaking) by two processes - mutation (changing a single character, or less frequently, inserting or deleting a few) and recombination (either two chromosomes exchange a chunk of DNA, or one chromosome copies a chunk from another). Thus once we have seen many genomes of a given species, a new genome is likely to look like a mosaic of genomes we have seen before. If we can infer a close mosaic (or pair of mosaics for a diploid species), we have found a "personalised reference genome", and the mapping approach should now work much better, as reads are more likely to exact-match.

This approach was first described in [9], applied to to the human MHC region. They combined a multiple sequence alignment of high quality assemblies with catalogs of known SNP and indel variation into a directed acyclic graph that they termed a Population Reference Genome (PRG); reads were compared against the PRG and counts were collected at informative positions, after which a hidden markov model (HMM) was used to infer a pair of paths across the PRG which together best explained the data. In principle reads should then be mapped against this pair of genomes to discover genetic variation that was not in the PRG already, although that step was approximated. Furthermore, although the method worked, the implementation was quite specific to the region and would not scale to the whole genome. Valenzuela et al [1] have recently also espoused a find-the-closest reference approach, and highlighted the lack of suitable software to do this.

There are also algorithms to the problem of encoding many very similar genomes (e.g. [14]), but in our opinion the greatest biological benefit comes from supporting higher diversity species and incorporating occasional regions which are unusually divergent (but often important). Other "reference graph" approaches have been published, generally approaching just the alignment step and leaving open the problem of how this is incorporated into standard analyses: Schneeberger *et al* [10] produced an aligner which locally chose the best reference to align to. Siren *et al* developed an method (GCSA [11]) for indexing all subpaths within a directed acyclic graph representation of a multiple sequence alignment, which would enable mapping - however construction costs for a whole human genome plus a set of SNPs required more than 1 Tb of RAM. Huang et al [12] developed an FM index encoding of a reference genome-plus-variation ("bwbble") by extending the genetic alphabet to encode single-character variants with new characters (eg an A versus C mutation would be encoded as M) and then concatenating padded indel variants (padding length $R-1$, where R is read length) to the end of the reference genome. Bwbble allows reads to align smoothly across multiple SNPs, since these are encoded in situ within the reference, but for other variants the padding prevents this. We do something similar, but treat all variation in an equivalent manner. While completing this paper, the preprint for GCSA2 was published ([13]), which drops RAM usage of human genome index construction to <100Gb at the cost of >1Tb of disk I/O.

We show below how to encode a set of genomes, or a reference plus genetic variation, in an FM index which naturally distinguishes alternative sequences which come from the same place (known as "alleles"). We extend the well known BWT backward search, and show how read-mapping can be performed in a way that allows reads to cross multiple variants, allowing recombination to occur naturally. Our data structure supports bidirectional search, in principle allowing detection of Maximal Exact Matches (MEMs) and Super Maximal Exact Matches (SMEMs) that underly the algorithm of bwa-mem [3], but currently we have only implemented exact matching. We show that index construction is relatively cheap, encoding the human genome and 80 million genetic variants from 2535 individuals from the 1000 genomes project using just 31 GB of RAM on a single core in 5 hours. We go on to show how inferring a personalised reference results in better genome inference, looking at a highly challenging region - the MSP3.4 gene of *P. falciparum*. For haploids, our approach has the benefit of allowing people to reuse their existing variant detection approaches.

2 Background: Compressed Text Indexes

Burrows-Wheeler Transform. The Burrows-Wheeler Transform of a string is a reversible permutation of its characters that was originally developed for compression [2]- it tends to cluster together identical symbols, and so can be efficiently stored with techniques such as run-length encoding or move-to-front coding. More recently, it has been applied to full-text indexing because it allows the search of a substring in large texts in linear time with respect to the length

of the substring, with a small memory footprint. The BWT of a string $T = t_1t_2 \dots t_n$ is constructed by sorting its n cyclic shifts $t_1t_2 \dots t_n$, $t_2 \dots t_nt_1$, $\dots$, $t_nt_1 \dots t_{n-1}$ in lexicographic order. The matrix obtained is called the Burrows-Wheeler matrix and the sequence from its last column is the BWT. An example is given below. The last character t_n is always a unique terminating symbol \$ that is lexicographically smaller than all the symbols in T and is essential for decoding. Storing the first and last column of the BWM is sufficient for finding the number of exact matches of a query in T. For locating the position of the matches in T an additional data structure is required, the suffix array.

Suffix Arrays. The suffix array of a string T is an array of integers that provides the starting position of T's suffixes, after they have been ordered lexicographically. Formally, if $T_{i,j}$ is the substring $t_it_{i+1} \dots t_j$ of T and SA is the suffix array of T, then $T_{SA[1],n} < T_{SA[2],n} < \dots < T_{SA[n],n}$. It is related to the BWT, since looking at the substrings preceding the terminating character \$ in the BWM rows gives the suffixes of T in lexicographical order. In fact, the BWT can be derived in linear time from the suffix array by looping through its values and recording the character occurring just before each suffix in the original text, i.e. $BWT[i] = T[SA[i] - 1]$ if $SA[i] \neq 1$ and $BWT[i] = \$$ if $SA[i] = 1$. The suffix array coupled with the BWT and two additional data structures form the FM-index, which enables the backward search of a pattern in text.

Backward search Any occurrence of a pattern P in text is a prefix for some suffix of T, so all occurrences will be adjacent in the suffix array of T, since suffixes starting with P are sorted together in a SA-interval. The backward search starts with the last character of P and successively extends it to longer suffixes, calculating their SA-intervals, until the SA-interval of the entire query is reached. Let $C[a]$ be the total number of occurences in T of characters smaller than a in the alphabet, the C-array essentially representing the first column of the BWM. Then if P' is a suffix of the query P and $[l(P'), r(P')]$ is its corresponding SA-interval, then the search can be extended to aP' by calculating the new SA-interval:

$$l(aP') = C[a] + rank_{BWT}(a, l(P') - 1) + 1 \tag{1}$$

$$r(aP') = C[a] + rank_{BWT}(a, r(P)) \tag{2}$$

The search starts with the SA-interval of the empty string, $[1, n]$ and successively adds one character of P in backward order. When the search is completed, it returns a SA-interval $[l, r]$ for the entire query P. If $r \geq l$, there are $r - l + 1$ matches for P and their locations in T are given by $SA[i]$ for $l \leq i \leq r$. Otherwise, the pattern does not exist in T. If the C-array and the ranks have already been stored, the backward search can be performed in $O(|P|)$ time in strings with DNA alphabet.

Wavelet Trees As the alphabet of a string contains more symbols, rank queries become more computationally expensive since they scale linearly with the alphabet size. The wavelet tree is a data structure designed to store strings with large alphabets efficiently and provide rank calculations in logarithmic time. A wavelet tree converts a string into a balanced binary-tree of bitvectors, whose root is built by taking the sorted alphabet and replacing the lower half of smaller symbols with a 0, and the other half of larger symbols with a 1 in the string. This creates ambiguity initially, but at each tree level, each half of the parent node's alphabet is re-split into 2 and re-encoded, so the ambiguity lessens as the tree is traversed in depth. At the leaves, there is no ambiguity at all. The tree is defined recursively as follows: take the lexicographically ordered alphabet, split it into 2 equal halves; in the string corresponding to the current node (start with original string at root), replace the first half of letters with 0 and the other half with 1; the left child node will contain the 0-encoded symbols and the right child node will contain the 1-encoded symbols, preserving their order from the original string; reapply the first step for each child node recursively until the alphabet left in each node contains only one or two symbols (so a 0 or 1 determines which symbol it is). An example is given in the figure below.

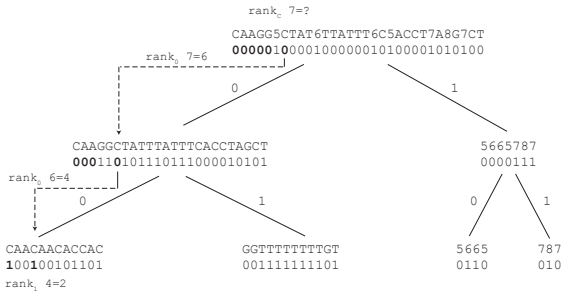

Fig. 1. Wavelet tree encoding of the PRG above. Calculating the rank of the marked C is performed by repeated rank() calls moving down the binary tree until the alphabet remaining is just 2 characters.

In order to answer a rank query over the original string with large alphabet, repeated rank queries over the bitvectors in the wavelet tree nodes are used as a guide to the right subtree that contains the leaf where the queried symbol is non-ambiguously encoded. The rank of the queried symbol in this leaf is equal to its rank in the original string. The number of rank queries needed to reach the leaf is equal to the height of the tree, i.e. $\log_2 |\Sigma|$ if we let Σ be the set of

symbols in the alphabet. Computing ranks over binary vectors can be done in constant time, so a rank query in a wavelet tree-encoded string has complexity $O(\log_2 |\Sigma|)$. Next, we will show how the data structures described in this section can be used to store variation inside a reference genome in a way that supports read mapping.

3 Encoding a variation-aware reference structure

We propose a linear PRG conceptually equivalent to a directed, acyclic, partial order graph, that is generated from a reference sequence and a set of alternative sequences at given variation loci. The graph is linearised into a long string over an alphabet extended with new symbols marking the variants, for which the FM-index can be constructed. Building this data structure requires multiple steps.

1. First, homologous regions of shared sequence between the input reference genomes must be identified. These must be of size k at least (where k is pre-defined), and will act like anchors for the coordinates of the variation sites.
2. Second, for any locus between two anchor regions, the set of possible haplotypes must be determined from the input genomes, but they do not need to be aligned. Indels are naturally supported by haplotypes of different lengths.
3. Each variation locus is assigned two unique numeric identifiers, one even and one odd. The odd identifiers will mark locus boundaries and the even identifiers will mark alternative allele boundaries.
4. For each variation locus, its left anchor is added to the linear PRG, followed by its odd identifier. Then each sequence coming from that locus, starting with the reference sequence, is successively added to the linear PRG, followed by the even locus identifier, except the last sequence, which is followed by the odd identifier.
5. Convert the linear PRG to integer alphabet ($A \to 1$, $C \to 2$, $G \to 3$, $T \to 4$, variation locus identifiers $\to$ 5,6,...)
6. The FM-index (suffix array, BWT, wavelet tree over BWT) of the linear PRG is constructed and we will call this the vBWT.

An illustration of these steps on a toy example is given in Figure.

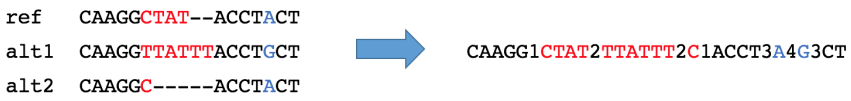

Fig. 2. A simple PRG linearised according to our encoding. The first site has 3 alleles, which do not here look at all similar, and the second is a SNP. This example will be used in subsequent figures.

This vBWT construction allows variation-aware mapping through a modified backward search algorithm. The variation markers that surround homologous alleles prevent wrong alignment across their boundaries and ensure the right path is taken when a read crosses a region with multiple alternative sequences. Since these markers are unique to each variation site, we can locate the beginning and end of a site in the Burrows-Wheeler matrix after the permutation of the linear PRG string. This enables reads to cross site junctions and map to the linear PRG when they span multiple sites of variation, allowing for new recombinations of known haplotypes. It also makes the method potentially adjustable for long reads. More importantly, *the markers force the ends of alternative sequences coming from the same site to be sorted together in a separate block in the Burrows-Wheeler matrix, even if they do not have high sequence similarity.* Therefore, homologous alleles from each site can be queried concurrently instead of looping through every one of them and checking if they match the read, which would happen if the markers were the same for all sites and non-homologous alleles got mixed up.

The linear PRG preserves information about the variant locations, so its coordinates can be projected back onto the primary reference sequence, enabling the integration with functional annotations and standard file formats. A path through the linear PRG is a string obtained by traversing the linear PRG and concatenating the non-variable segments with one sequence from each site that contains variation. If the first sequence is chosen from all variation sites, then the standard reference genome is obtained. For a haploid organism, selecting the best supported allele at each site produces the reference genome that is as close as possible to the sample, which can then be used with usual software for downstream variant calling analysis. For diploids and mixtures, not covered in this paper, our goal is to use the linear PRG mapping to genotype each of the variation sites, then use the link information from reads spanning multiple sites to phase adjacent variants and, infer a set of haplotypes that best explain the data (as in [**?**]), and finally only do full graph alignment to the graph of these haplotypes.

4 Variation aware backward search in vBWT

In this section, we present a modified backward search algorithm for exact matching against the vBWT that is aware of alteternative sequence paths. When reads align to the non-variable part of of the linear PRG or when a variant locus is long enough to enclose the entire read, the usual backward search algorithm can be used. Otherwise, when the read must cross variation site junctions in order to align, site identifiers and some alternative alleles must be ignored by the search. This means a read can align to multiple substrings of the linear PRG that may not be adjacent in the BWM, so the search can return multiple SA-intervals. This is illustrated in figure 2.

At each step in backward search, before extending to the next character, we need to check whether the current matched read substring is preceded by a

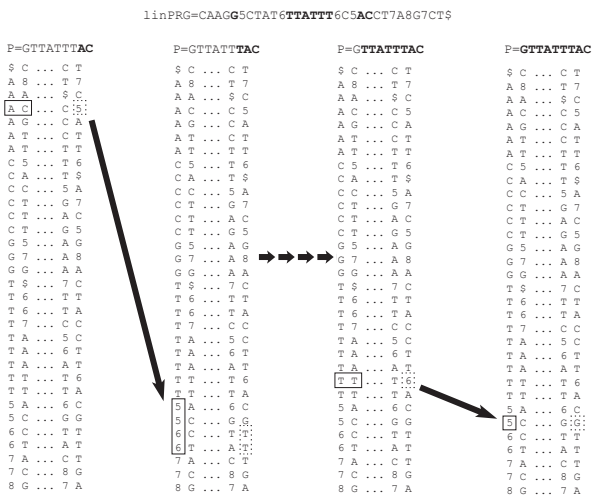

Fig. 3. Backward search across vBWT. We start at the right-hand end of the read GTTATTTAC, with the character C, and as we extend we hit the character 5, signalling the start or end of a variation locus. We check with a mask, and find it is the end. Therefore the read must now continue into one of the alleles, signalled by the number 6. Continuing in this manner we are able to align across the site.

variation marker anywhere in the linear PRG. A scan for symbols larger than 4 must be performed in the BWT within the range given by the current SA-interval (need to explain range search 2d in a wavelet tree). If a variation marker is found and it is an odd number, the read is about cross a site boundary, i.e. is about to walk in or walk out of a site. The suffix array can be queried to find the position of the two odd numbers in the liner PRG: the number occurring at a smaller position will mark the beginning of site and the other one will mark the end of site. If the search cursor is next to the start of the site, it is just the site marker that needs to be skipped so the SA-interval (size 1) of the suffix starting with that marker needs to be added to the set of intervals that will be extended with the next character in the read. If the search cursor is next to the end of a site, all alternative alleles from that site need to be queried. Their ends are sorted together in the BWM because of the markers, so they can be queried concurrently by adding the SA-interval of suffixes starting with all numbers marking that site (even and odd).

If the variation marker found is an even number, the read is about to cross an allele boundary, which means its current suffix matches the beginning of an alternative allele and the read is about to walk out of a site, so the search cursor needs to jump to the start of site. As previously described, the odd markers corresponding to that site can be found in the sorted first column of the BWM, and then querying the suffix array decides which one marks the start of site.

Then the SA-interval (size 1) for the BWM row starting with this odd marker is recorded.

Once the check for variation markers is finished and all candidate SA-intervals have been added, each interval can be extended with the next character in the read by using equations 1 and 2.

5 Performance

5.1 Construction cost : the human genome

One natural test of scalability of this data structure is to see how expensive it is to construct a PRG of the human genome. For comparison, GCSA took over 1Tb of RAM for the reference plus an intermediate release of 1000 genomes SNPs. GCSA2 reduces the memory footprint to below 128Gb RAM, running in 13 hours with 32 cores, and using over 1Tb of I/O to fast Lustre disk.

We constructed four different PRGs from the human reference genome (GRC37 without "alt" contigs) plus the 1000 genomes final VCF [6], as described below. We first excluded structural variants which did not have precisely specified alleles, and variants with allele frequency below a threshold f. If two variants occurred at consecutive bases, they were merged into one, and all possible haplotypes enumerated. If the VCF contained two consecutive records which overlapped, the second was discarded. Using values 0 and 0.05 for f, we produced two PRGs. The results, along with comparative data for bwbble with identical input, are shown in Table 1. Our vBWT construction has a lower memory cost than GCSA, GCSA2 and bwbble, and is faster than GCSA/GCSA2, but slower than bwbble.

Table 1. FM index construction costs for human reference genome plus 1000 genomes variants

Software	Data	Vars (millions)	MemFM(Gb)	Time
vBWT	f>0.05	8.2	15	5
vBWT	f>0	79.2	31	4h35m
Bwbble	f>0.05	8.2	60	1h5m
Bwbble	f>0	81	70	1.75

5.2 Inferring a Closer Reference Genome

P. falciparum is a haploid parasite that undergoes recombination when inside a mosquito. It has an unusually repetitive genome that contains more indels than SNPs [15]. There are several regions that present challenges to mapping because samples often diverge strongly from the reference. For example, the merozoite surface protein gene MSP3.4 is known to have two highly diverged lineages at a 500bp region (the DBL domain), which differ by around 1 SNP every 3 bases.

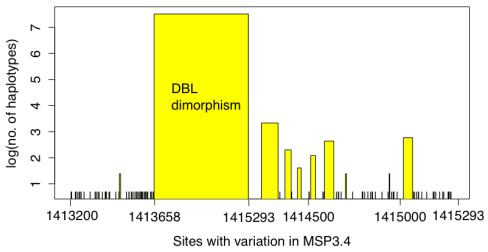

Fig. 4. Histogram of number of alleles at each site in MSP3.4 plotted above the chromosome coordinate

Thus for many samples a "pileup" of mapped reads simply shows a hole (see Figure ?).

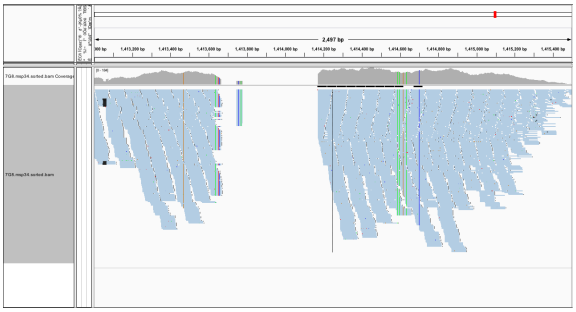

Fig. 5. Mapping reads from sample 7G8 to *P. falciparum* 3D7 reference genome results in a gap

We constructed a catalog of MSP3.4 variation from Cortex [16] and GATK [17] variant calls from 700 *P. falciparum* samples from the Ghana, Laos and Cambodia, and built a PRG just for that chromosome. We aligned Illumina (how long?) reads from a well-studied sample that was not used in graph construction (named 7G8) to the PRG using backward search (exact matching), and collected counts on the number of reads supporting each allele. We used a simple "heaviest path" method similar to that outlined by Valenzuela et al [1] to choose the path through the graph which best supported the data. This was our graph-inferred personalised reference for this sample. We then mapped the reads (using bwa_mem [18]) to the inferred genome. As can be seen in Figure 6, this gives dramatically better results.

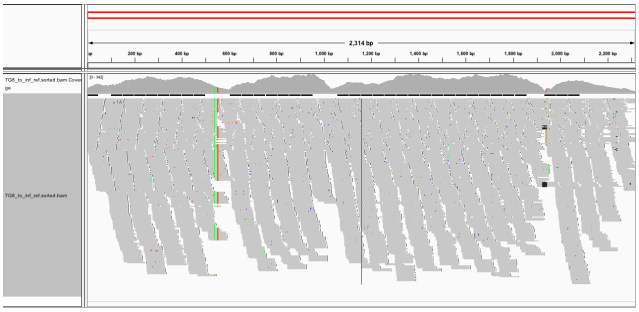

Fig. 6. Mapping reads from sample 7G8 to our vBWT-inferred genome removes the gap and leaves isolated variation which is easy to identify by standard approaches

5.3 Performance cost of large alphabet

We chose this encoding because it provided a natural way to distinguish mapping uncertainty due to distant repeats, and that due to similar alleles, as well as naturally allowing mapping across multiple variants. However in doing so we need to work with an alphabet of size $4 + 2n$ where n is the number of variant sites. The intention is that we only need to incorporate enough variation to ensure that, with high probability, a read will differ from the graph by at most one isolated SNP - in fact most of the value will come from incorporating the more divergent and structural changes. Thus for the human genome we expect only to need to incorporate variants with frequency $> 5\%$, resulting in an alphabet of size 16,397,638. Since rank operations over a wavelet tree scale with log(alphabet size), this is $log_2(16397638/4) = 22$ times slower than for a standard FM index over the DNA alphabet. To get an estimate of how this impacts performance in practise, we measure speed of exact matching of simulated perfect reads from Pf chromosome 10 with the MSP3.4 variation inserted. To avoid unnecessarily repeatedly shrinking the same BWT intervals, we store in a hash the BWT intervals corresponding to all 9mers in the PRG. This results in read-matching at a rate of 277/second.

One other consequence of our approach, shared with bwbble, is that a single pattern can match multiple intervals in the BWT, because of the positional markers. When we store a hash of 9-mers to BWT interval, we actually have to store a list of intervals, along with which sites and alleles are crossed. Our current implementation of this is naive (using C++ std::unordered_map). We also store integer arrays (masks) allowing us to determine if a position in the PRG is in a site, and if so, which allele - again naively encoded (in std::vector). For the human genome example these cost us around 30Gb of RAM. The biggest cost is in these sparse mask arrays, which could be stored much more compactly.

Finally, there is one performance improvement which we have yet to implement - precalculating and storing an array of ranks at marker positions across the BWT - just as in a standard FM-index. The alphabet size does not prohibit this - we can ignore the numeric characters, and store only for A,C,G,T.

6 Discussion

New methods for representing and querying "pan-genomes" are of great interest to a variety of biological communities [19]. In human genetics the Global Alliance for Genomics and Health is trying to develop a new approach that would obviate the need for changing coordinates with every new reference genome version, and would allow better access to medically important regions such as MHC and KIR. In pathogen genomics, levels of diversity are much higher than in humans, and artefacts caused by choice of reference lead to analysis challenges.

By extending the alphabet and placing positional markers, we are able to ensure that alternate alleles sort together in the BWT matrix, allowing mapping across sites and recombination. In fact, we could encode quite general graph structures in this manner, but for simplicity we imposed a constraint on graph construction, and did not allow nesting of sites. There are two great benefits to our approach. First, for haploids, the idea of systematically choosing a closer reference, and then using the standard tools, is something biologists will understand - it is simply a systematisation of what people already try to do where possible, incorporating recombination. Secondly, for other ploidies, our implementation readily lends itself to collecting phase-informative read-data, running an HMM, and finally MEM-based graph alignment. We have made a choice to encode genetic variation as a Directed Acyclic Graph - something close to a multiple sequence alignment, at least in terms of communicating what we do - which will not allow they very general genetic rearrangements that other developing methods (vg) choose to support. Our choice was deliberate - we are aiming for a more limited model, but avoiding heuristics in the details of the graph construction, which should aid in interpretation.

Software We have implemented the vBWT twice. First as a simple prototype, which provided the results on *P. falciparum* above. Secondly, a more careful implementation that provided all other results, and which is available here: http://github.com/iqbal-lab/gramtools. We make the prototype available here purely for sake of reproducibility URL, but all future development (and maintenance) is on the second implementation.

Acknowledgments. We would like to thank Jerome Kelleher, Heng Li, Rayan Chikhi and Jouni Siren for discussions, and Lin Huang and Victoria Popic for help running bwbble. Above all we would like to thank Simon Gog and the SDSL developers for providing a valuable and well-maintained resource. ¡FUNDING¿

References

1. Valenzuela, D., Valimaki, N., Pitkanen, E., Makinen, V. On enhancing variation detection through pan-genome indexing. Biorxiv. http://dx.doi.org/10.1101/021444

2. Burrows, M., Wheeler, D.J. :A block sorting lossless data compression algorithm. Digital Equipment Corporation, Tech. Rep. 124, 1994. Available: http://www.hpl.hp.com/ techreports/Compaq-DEC/SRC-RR-124.html
3. Li, H., Durbin, R.: Fast and accurate short read alignment with Burrows?Wheeler transform. Bioinformatics 25 (14): 1754-1760 (2009)
4. Langmead, B., Salzberg, S.: Fast gapped-read alignment with Bowtie 2. Nature Methods. Mar 4;9(4):357-9 (2012)
5. Reinert, K., Langmead, B., Weese, D., et al: Alignment of Next-Generation Sequencing Reads. Annu Rev Genomics Hum Genet. 2015;16:133-51
6. The 1000 Genomes Project Consortium: A global reference for human genetic variation. Nature 526, 68?74
7. Ossowski, S., Schneeberger, K., Clark, R.M., et al. Sequencing of natural strains of Arabidopsis thaliana with short reads. Genome Research 18, 2024-2033 (2008)
8. Jeffares, D.C., Rallis, C., Rieux, A., et al. The genomic and phenotypic diversity of Schizosaccharomyces pombe. Nature Genetics 47, 235?241 (2015)
9. Dilthey, A., Cox, C., Iqbal, Z., et al: Improved genome inference in the MHC using a population reference graph. Nature Genetics 47, 682?688 (2015)
10. Schneeberger, K.,Hagmann, J., Ossowski, S., et al. Simultaneous alignment of short reads against multiple genomes. Genome Biol. 10, R98 (2009).
11. Siren, J., Valimaki, N., Makinen, V. Indexing Graphs for Path Queries with Applications in Genome Research. IEEE/ACM Trans Comput Biol Bioinform. 2014 Mar-Apr;11(2):375-88
12. Huang, L., Popic, V., Batzoglou, S. Short read alignment with populations of genomes. Bioinformatics. Jul 1;29(13):i361-70 (2013)
13. Siren, J. Indexing Variation Graphs. arXiv:1604.06605
14. Na, J.C., Crochemore, M., Park, H., et al: Suffix tree of alignment: an efficient index for similar data. Combinatorial Algorithms ? Revised Selected Papers of the 24th International Workshop, IWOCA 2013, Rouen, France, July 10?12, 2013 (2013), pp. 337?348
15. Miles, A., Iqbal, Z., Vauterin, P., et al .: Genome variation and meiotic recombination in Plasmodium falciparum: insights from deep sequencing of genetic crosses Biorxiv. http://dx.doi.org/10.1101/024182 (2015)
16. Iqbal, Z., Caccamo, M. Turner, I., et al: De novo assembly and genotyping of variants using colored de Bruijn graphs. Nature Genetics 44, 226?232 (2012)
17. DePristo, M., Banks, E., Poplin, R.E., et al: A framework for variation discovery and genotyping using next-generation DNA sequencing data. Nature Genetics 43(5): 491?498 (2011)
18. Li, H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv:1303.3997
19. Marschall, T., Marz, M., Abeel, T., et al: Computational Pan-Genomics: Status, Promises and Challenges. Biorxiv. http://dx.doi.org/10.1101/043430

7 Checklist of Items to be Sent to Volume Editors

Here is a checklist of everything the volume editor requires from you:

☐ The final LaTeX source files

☐ A final PDF file

☐ A copyright form, signed by one author on behalf of all of the authors of the paper.

☐ A readme giving the name and email address of the corresponding author.