A new encoding of genetic variation in an FM index 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. The abstract should summarize the contents of the paper and should contain at least 70 and at most 150 words. It should be written using the *abstract* environment.

Keywords: We would like to encourage you to list your keywords within the abstract section

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. Since we do not currently have the means to directly "read" a molecule from end to end, this process involves breaking the DNA into smaller fragments, and then via one of several technologies, identifying substrings (called "reads"). 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, 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 and there are now extremely efficient tools to do it [ref bwa, bowtie, Langmead review]. The use of the Burrows Wheeler Transform [REF], a well known algorithm from computer science originally designed for compression, sometimes augmented with auxiliary structures to become an FM-index [REF] underlies the two dominant mappers - bwa and bowtie.

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. However 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 can make use of a single reference highly problematic. To make matters worse, 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.

This paper addresses the question of where the mapping paradigm should go, in a world where sequencing is cheap, we are ambitious about understanding in great detail how individuals within a species differ [ref 1000 genomes 3, 1001 arabidopsis, pombe paper] including challenging regions, and we are interested in studying mixtures of genomes. 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 then infer the sequence of the underlying genomes.

Our method 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. We seek to build a data structure which represents a set of genomes we have seen before, such that given sequence data from a new genome, we can find the closest mosaic we can to this new genome. In the case of "diploid" species, such as humans, who have two different chromosomes, this involves inferring a pair of mosaics. Having done this, we have found a "personalised reference genome", and the mapping approach should now work much better, as reads are more likely to exact-match the mosaic. This precise approach was first introduced in [Dilthey1], applied 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 (DAG); reads were compared against the DAG and counts were collected at informative positions, after which a hidden markov model (HMM) was used to infer a pair of paths across the DAG which together best explained the data. Reads were then mapped using standard technology [ref bwa] against this pair of genomes to discover genetic variation that was not in the DAG already. Although the method worked, the implementation was quite specific to the region and would not scale to the whole genome. [Makinen et al] have recently also espoused a find-the-closest reference approach, and highlighted the lack of suitable software to do this.

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: [Scheeberger] produced an aligner which locally chose the best reference to align to. Siren developed an elegant method (GCSA) 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 [ref Siren et al "Indexing Graphs for Path Queries with Applications in Genome Research"]. [Huang] 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 indel variants to the end of the reference genome. Recent unpublished work includes: GCSA2, which RAM usage of index construction to 100Gb at the cost of ¿1Tb of disk I/O, and HISAT2, which combines many local GCSA graphs with the HISAT aligner to support alignment to reference plus primarily single-character differences.

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 [REF]. We show that index construction is relatively cheap, encoding the human genome and 80 million genetic variants from 2535 individuals from the 1000 gebomes project using just ? 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. Our approach is intended to retain the benefits of incorporating known variation, while allowing people to reuse their existing variant detection approaches [ref samtools, GATK, platypus]. We finish with a discussion of future steps in this program of research.

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 because it tends to cluster together identical symbols, so it 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 $t_1t_2 \dots t_n$, $t_2 \dots t_nt_1$, ..., $t_nt_1 \dots t_{n-1}$ 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 $t_i t_{i+1} \dots t_j$ 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.

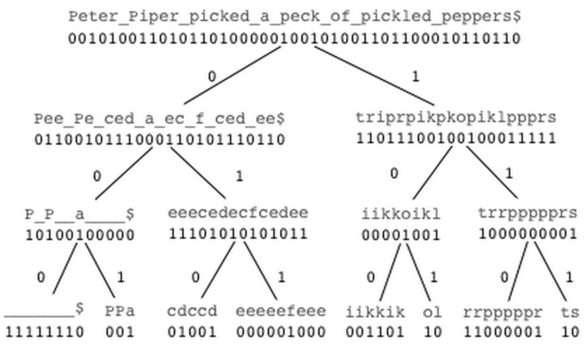

Fig. 1. Bla

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 with 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-¿1,C-¿2,G-¿3,T-¿4, variation locus identifiers-¿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.

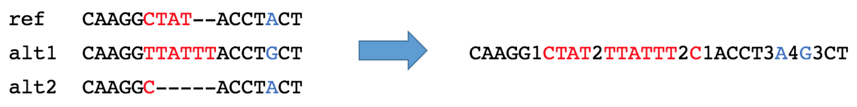

Fig. 2. Bla

This vBWT construction allows reads to be mapped to it 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. Otherwise, a new alternative reference is obtained that can be used with usual software for downstream variant calling analysis. 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 hence infer a personalised reference that corresponds to the path that is closest to the sample analysed.

4 Exact mapping

In this section, we present a modified backward search algorithm for exact matching against the vBWT that is aware of alternative 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.

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 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 96Gb RAM at the cost of over 1Tb of I/O.

We therefore constructed four different PRGs from the human reference genome (GRC37 without "alt" contigs) plus the 1000 genomes final VCF [Auton], 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, 0.001, 0.01 and 0.05 for f, we produced four PRGs. As can be seen from this table, the construction costs are reasonable, especially compared with GCSA.

Table 1. Human genome index construction costs. MemFM refers to peak memory use by vBWT and associated structures, and MemNaive refers to our simple C++ std::vectors used to make two integer "masks", giving the site/allele at each position in the PRG

Software	Data	NumVars(mill)	MemFM(Gb)	MemNaive(Gb)	Time/hrs
vBWT	f>0.05	8.2	15	30	5
vBWT	f>0.01	13.8	?	?	?
vBWT	f>0.001	28.8	?	?	?
vBWT	f>0l	79.2	11	?	1.3
Bwbble	f>0.05	8.2	60	-	1.1

5.2 Inferring a Closer Reference Genome

P. falciparum is a haploid parasite that causes malaria. The genome is repetitive, and unlike most studied species, the genome contains more indels than SNPs. There are several regions that present challenges to mapping, not due to repeats, but because samples often diverge strongly from the reference, For example, for many samples, no reads map to the central region of the merozoite surface protein gene MSP3.4, and a "pileup" of mapped reads simply shows a hole (see Figure ?). The architecture of the gene is as follows: around ? bases where there are ? clustered SNPS, around 500 bases where recombination is not known to occur, and there are only two highly diverged lineages which differ by about 3 SNPs every 10 bases, followed finally by reasonably spaced SNPs and indels. We chose this as a good candidate to display the value of PRGs. We therefore constructed a catalog of variation from Cortex [] and GATK [] variant calls from 700 P. falciparum samples from the Ghana, Laos and Cambodia. 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, and collected counts on the number of reads supporting each allele. We used a simple "heaviest path" method as outlined by [Valenzuela] to choose the path through the graph which best supported the data. We then mapped the reads (using bwa_mem) to the inferred genome. As can be seen in Figure ?, this gives dramatically better results. For haploid species, this is the most natural use of a PRG - find the closest reference and then map to it.

6 Discussion

New methods for representing and querying "pan-genomes" are of great interest to a variety of biological communities. In human genetics the Global Alliance for Genomics and Health is trying to develop a new approach that would obviate the need forchanging 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 graph complexity is therefore different to that in humans,

Dilthey et al outlined a general approach for diploids and provided a proof-of-concept implementation for the MHC only. First "pre-map" reads to the graph, storing enough information to run an HMM and infer the most likely genotypes (at the PRG sites) to have generated the data (i.e.. this set of reads). Second, if possible, phase the variants. Third, full mapping and alignment to the graph to discover novel variants. There are a number of challenges in scaling this up - the size of the human genome, regions of structural divergence or of clustered variation. By extending the alphabet, we are able to ensure that alternate alleles sort together in the BWT matrix, allowing mapping across sites and recombination. Our memory footprint is very modest even for the human genome and 80 million variants (11Gb RAM for the vBWT and 30Gb RAM for naive arrays for site/allele masks), and could drop further by replacing the mask arrays with succinct ones. The major concern with extending to an alphabet of millions of characters is the impact on performance, even with the benefit of the wavelet tree, which reduces rank operation complexity from O(n) to O(log(n)). Although we have not yet implemented inexact matching, we can get a ballpark estimate from measuring the speed of exact-matching.

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 ¡...¿, but recommend users use the new implementation.

6.1 Citations

For citations in the text please use square brackets and consecutive numbers: [1], [2], [4] – provided automatically by LaTeX's `\cite` ... `\bibitem` mechanism.

6.2 Page Numbering and Running Heads

There is no need to include page numbers. If your paper title is too long to serve as a running head, it will be shortened. Your suggestion as to how to shorten it would be most welcome.

Please note that, if your email address is given in your paper, it will also be included in the meta data of the online version.

7 BibTeX Entries

The correct BibTeX entries for the Lecture Notes in Computer Science volumes can be found at the following Website shortly after the publication of the book: `http://www.informatik.uni-trier.de/~ley/db/journals/lncs.html`

Acknowledgments. We would like to thank Jerome Kelleher, Heng Li, Jouni Siren for discussions. Above all we would like to thank Simon Gog and the SDSL developers for providing a valuable and well-maintained resource. ¡FUNDING¿

The following section shows a sample reference list with entries for journal articles [1], an LNCS chapter [2], a book [3], proceedings without editors [4] and [5], as well as a URL [6]. Please note that proceedings published in LNCS are not cited with their full titles, but with their acronyms!

References

1. Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. J. Mol. Biol. 147, 195–197 (1981)
2. May, P., Ehrlich, H.C., Steinke, T.: ZIB Structure Prediction Pipeline: Composing a Complex Biological Workflow through Web Services. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 1148–1158. Springer, Heidelberg (2006)
3. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1999)

4. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid Information Services for Distributed Resource Sharing. In: 10th IEEE International Symposium on High Performance Distributed Computing, pp. 181–184. IEEE Press, New York (2001)
5. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration. Technical report, Global Grid Forum (2002)
6. National Center for Biotechnology Information, `http://www.ncbi.nlm.nih.gov`

8 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.