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 $T = t_1 t_2 \dots t_n$ is constructed by sorting its n cyclic shifts $t_1 t_2 \dots t_n$, $t_2 \dots t_n t_1$, $\dots$, $t_n t_1 \dots t_{n-1}$ 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.

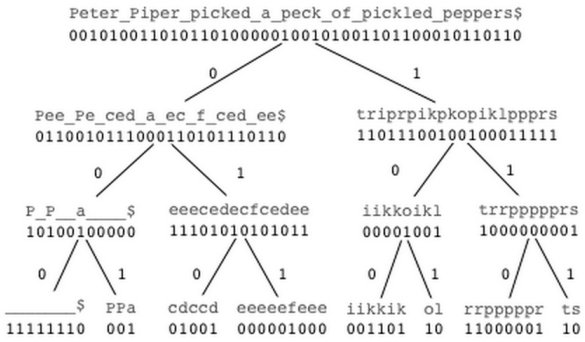

Fig. 1. Bla

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.

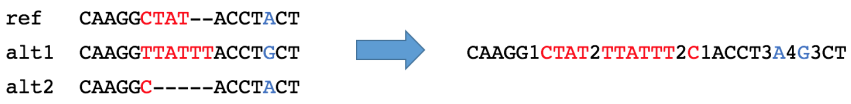

Fig. 2. Bla

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.

Headings. Headings should be capitalized (i.e., nouns, verbs, and all other words except articles, prepositions, and conjunctions should be set with an initial capital) and should, with the exception of the title, be aligned to the left. Words joined by a hyphen are subject to a special rule. If the first word can stand alone, the second word should be capitalized.

Here are some examples of headings: "Criteria to Disprove Context-Freeness of Collage Language", "On Correcting the Intrusion of Tracing Non-deterministic Programs by Software", "A User-Friendly and Extendable Data Distribution System", "Multi-flip Networks: Parallelizing GenSAT", "Self-determinations of Man".

4.1 Figures

For LaTeX users, we recommend using the *graphics* or *graphicx* package and the `\includegraphics` command.

Please check that the lines in line drawings are not interrupted and are of a constant width. Grids and details within the figures must be clearly legible and may not be written one on top of the other. Line drawings should have a resolution of at least 800 dpi (preferably 1200 dpi). The lettering in figures should have a height of 2 mm (10-point type). Figures should be numbered and should have a caption which should always be positioned *under* the figures, in contrast to the caption belonging to a table, which should always appear *above* the table; this is simply achieved as matter of sequence in your source.

Please center the figures or your tabular material by using the `\centering` declaration. Short captions are centered by default between the margins and typeset in 9-point type (Fig. 3 shows an example). The distance between text and figure is preset to be about 8 mm, the distance between figure and caption about 6 mm.

To ensure that the reproduction of your illustrations is of a reasonable quality, we advise against the use of shading. The contrast should be as pronounced as possible.

If screenshots are necessary, please make sure that you are happy with the print quality before you send the files.

Please define figures (and tables) as floating objects. Please avoid using optional location parameters like "`[h]`" for "here".

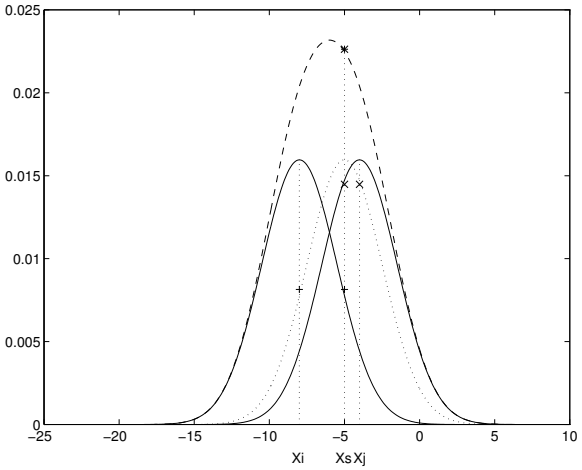

Fig. 3. One kernel at x_s (*dotted kernel*) or two kernels at x_i and x_j (*left and right*) lead to the same summed estimate at x_s. This shows a figure consisting of different types of lines. Elements of the figure described in the caption should be set in italics, in parentheses, as shown in this sample caption.

4.2 Formulas

Displayed equations or formulas are centered and set on a separate line (with an extra line or halfline space above and below). Displayed expressions should be numbered for reference. The numbers should be consecutive within each section or within the contribution, with numbers enclosed in parentheses and set on the right margin – which is the default if you use the *equation* environment, e.g.,

$$\psi(u) = \int_o^T \left[\frac{1}{2} \left(\Lambda_o^{-1} u, u \right) + N^*(-u) \right] dt \, . \tag{3}$$

Equations should be punctuated in the same way as ordinary text but with a small space before the end punctuation mark.

4.3 Footnotes

The superscript numeral used to refer to a footnote appears in the text either directly after the word to be discussed or – in relation to a phrase or a sentence – following the punctuation sign (comma, semicolon, or period). Footnotes should appear at the bottom of the normal text area, with a line of about 2 cm set immediately above them.[1]

[1] The footnote numeral is set flush left and the text follows with the usual word spacing.

4.4 Program Code

Program listings or program commands in the text are normally set in typewriter font, e.g., CMTT10 or Courier.

Example of a Computer Program

```
program Inflation (Output)
  {Assuming annual inflation rates of 7%, 8%, and 10%,...
   years};
   const
     MaxYears = 10;
   var
     Year: 0..MaxYears;
     Factor1, Factor2, Factor3: Real;
   begin
     Year := 0;
     Factor1 := 1.0; Factor2 := 1.0; Factor3 := 1.0;
     WriteLn('Year  7% 8% 10%'); WriteLn;
     repeat
       Year := Year + 1;
       Factor1 := Factor1 * 1.07;
       Factor2 := Factor2 * 1.08;
       Factor3 := Factor3 * 1.10;
       WriteLn(Year:5,Factor1:7:3,Factor2:7:3,Factor3:7:3)
     until Year = MaxYears
end.
```

(Example from Jensen K., Wirth N. (1991) Pascal user manual and report. Springer, New York)

4.5 Citations

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

4.6 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.

5 LNCS Online

The online version of the volume will be available in LNCS Online. Members of institutes subscribing to the Lecture Notes in Computer Science series have

access to all the pdfs of all the online publications. Non-subscribers can only read as far as the abstracts. If they try to go beyond this point, they are automatically asked, whether they would like to order the pdf, and are given instructions as to how to do so.

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.

6 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. The heading should be treated as a subsubsection heading and should not be assigned a number.

7 The References Section

In order to permit cross referencing within LNCS-Online, and eventually between different publishers and their online databases, LNCS will, from now on, be standardizing the format of the references. This new feature will increase the visibility of publications and facilitate academic research considerably. Please base your references on the examples below. References that don't adhere to this style will be reformatted by Springer. You should therefore check your references thoroughly when you receive the final pdf of your paper. The reference section must be complete. You may not omit references. Instructions as to where to find a fuller version of the references are not permissible.

We only accept references written using the latin alphabet. If the title of the book you are referring to is in Russian or Chinese, then please write (in Russian) or (in Chinese) at the end of the transcript or translation of the title.

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`

Appendix: Springer-Author Discount

LNCS authors are entitled to a 33.3% discount off all Springer publications. Before placing an order, the author should send an email, giving full details of his or her Springer publication, to `orders-HD-individuals@springer.com` to obtain a so-called token. This token is a number, which must be entered when placing an order via the Internet, in order to obtain the discount.

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.