Disclosing the nature of computational tools for the analysis of Next Generation Sequencing data.

Francesca Cordero[1,2], Marco Beccuti[1], Susanna Donatelli[1] and Raffaele A Calogero[2]

(1) Department of Computer Science, University of Torino, Torino, Italy;

(2) Department of Clinical and Biological Sciences, University of Torino, Torino, Italy;

Abstract

Next-generation sequencing (NGS) technologies are rapidly changing the approach to complex genomic studies, opening the way to personalized drugs development and personalized medicine. NGS technologies are characterized by a massive throughput for relatively short-sequences (30-100), and they are currently the most reliable and accurate method for grouping individuals on the basis of their genetic profiles. The first and crucial step in sequence analysis is the conversion of millions of short sequences (reads) into valuable genetic information by their mapping to a known (reference) genome. New computational methods, specifically designed for the type and the amount of data generated by NGS technologies, are replacing earlier widespread genome alignment algorithms which are unable to cope with such massive amount of data.

This review provides an overview of the bioinformatics techniques that have been developed for the mapping of NGS data onto a reference genome, with a special focus on polymorphism rate and sequence error detection. The different techniques have been experimented on an appropriately defined dataset, to investigate their relative computational costs and usability, as seen from an user perspective.

Since NGS platforms interrogate the genome using either the conventional nucleotide space or the more recent color space, this review does consider techniques both in nucleotide and color space, emphasizing similarities and diversities.

1 Introduction

The successful sequencing of a wide range of genomes has opened the way to a new approach in drug design and selection. The Sanger enzymatic dideoxy technique, firstly described in 1977 [10], has been improved continuously over the years, leading to a significant reduction in the overall cost of genome sequencing. It is in 2006 that new sequencing technologies, collectively referred to as Next-Generation Sequencing (NGS), appear on the market, leading to a revolution in the sequencing techniques.

This revolution is currently driven by three commercial platforms: 454 (Roche) [28], Genome Analyzer (Illumina/Solexa) [14] and AB-SOLiD (Applied Biosystems) [4]. A technical review of template preparation, sequencing and an overview of NGS technologies, instrument performance and cost it is reported in [21]. while a review of the sequencing techniques implemented in these platforms have been already published in [6, 22, 23, 27]. The use of NGS inside appropriate workflow, targeting a specific application, have been reviewed in [36] for RNAseq, in [15] for ChiPseq and in [9] for metagenomics.

All NGS technologies have in common the generation of sequences on an unprecedented scale, without the requirement for DNA cloning, and at a fraction of the costs required for traditional sequencing. These features are expected to lead in the near future to personalized genome sequencing

at a cost less than 1000$ per sample, a cost that will allow to address biological questions that were not economically or logistically practical before [29].

In particular in the field of molecular and pharmaceutical biotechnology NGS can enable i) the discovery of new drugs or derivatives from natural products, and ii) the development of personalized medicine.

As explained in [19], drug discovery from natural products has suffered a significant slowdown due to the complicate and long framework of high throughput screening and to the increasing government resections on drug approvals. But we can expect the first cause to be temporary, since the rapid access to inexpensive genome sequencing technologies will simplify the extraction of the genetic information of uncultivated organisms that could potential lead to new drug discovery. For example, the remarkable study [20], conducted by an international consortium, used Roche-454 sequencing of Mycobacterium tuberculosis to identify targets for diarylquinoline-based drug that potently inhibits the growth of drug sensitive/resistant strains of the pathogen. Based on these early reports, it is likely that our understanding of the spectrum of genome variation within clinical isolates will be greatly enhanced in the near future. This knowledge will lead to improved diagnostics, monitoring and treatments [7] as the availability of personal genome sequencing will provide information on patient predisposition to drug response and clearance [31]. New gene-mapping techniques will facilitate the design of diagnostic tests to efficiently highlight the causes of illness, including infections.

A source of potential weakness in NGS data analysis is the definition of specific bioinformatics workflow [17] and the correct choice of mapping methods for the application target. Basic mapping and variant detection tools are provided by the next-generation sequencing platform vendors, sequencing service providers and academic community. Whichever software is used, it is of paramount importance that its strength and weakness is clearly understood. Indeed the choice of the best algorithm may require a deep understanding of the peculiarity of each of them. This review goes along the line of providing information usable for the correct choice of a mapping method.

In this paper, we describe and discuss the main characteristics of alignment approaches. With respect to the complete reviews on the algorithms available in literature [25, 5], here we shall highlight algorithm peculiarities, as seen from an user perspective, by means of their comparison over a synthetic dataset encompassing mutations in 35-mers reads both in nucleotide and color space.

The paper is organized as follows. Section 2 introduces the alignment and mapping problems, thus defining the context of this review. Section 3 provides an overview of the distinctiveness of color space versus nucleotide space. Section 4 analyzes the peculiar features of the most used mapping algorithms for NGS data: Short Oligonucleotide Color Space (SOCS) [3], Mapping and Assembly with Qualities (MAQ) [12], Periodic Seed Mapping (PerM) [34], SHort Read Mapping Package (SHRiMP) [30], and Bowtie [2]. Section 5 shows the description of the synthetic dataset used in the experiments and the results of the comparison, in terms of computational costs and sensibility, of the selected algorithms. We conclude with a discussion on the obtained results in Section 6.

2 Background

NGS technologies allow to sequence more than one billion base pairs (gigabases) in a few days. Roche [28] and Illumina [14] platforms both implement the Sanger methodology, which is based on the principle of *sequencing by extension*: the DNA is used as a template to generate a complementary fragment considering a single base at a time. The identity of the bases is determined by chemical means, the results are a set of probes formed by 35-100 positions, expressed in the classical nucleotide space. AB SOLiD sequencing

technology introduces a new dibase sequencing technique. The system encodes each dimer with one of four available colors using a degenerate coding system: four fluorescent dyes are used to encode the sixteen possible nucleotide pairs, and each nucleotide is interrogated twice in independent reactions. The resulting output is a set of 35-mers probes expressed in color space.

NGS platforms associate with each element of the read a quality score, which is a measure of the likelihood of the nucleotide or dinucleotide (for AB SOLiD) call. In AB SOLiD technology the maximum value of quality is 33 (high probability of a correct dimer). Illumina Genome Analyser and Roche 454 platforms use instead an encoding into an ASCII character of a Phred scoring [18].

The first step in converting the huge volume of sequence data into biologically-valuable objects is their alignment on a reference genome. This step requires specifically devised computational tools.

The alignment between two or more biological sequences is a process initially developed to detect their homology rate. In biological terms, alignment has the objective to align homologous residues. DNA, RNA and proteins change their structure and evolve mainly under the action of three types of modifications: mutation, insertion or deletion. *Mutations* correspond to substitutions of a portion of sequence positions (nucleotides or amino acids). If the mutation involves a single position the term *Single Nucleotide Polymorphism (SNP)* is used, when it involves multiple contiguous positions it is called SNP locus (of a given size k). Otherwise, sequences undergoing insertion and/or deletion mutations will differ in length as part of their sequence will be removed (*deletion*) or new stretches of sequence will be added (*insertion*).

Given two sequences S_1 and S_2, an aligning algorithm must identify the list of operations, expressed by the function f, such as $f(S_1) = S_2$. The choice of f must optimize an objective function that represents the best alignment.

Assuming that evolution is parsimonious, when performing an alignment the aim is to minimize the number of evolutionary changes (substitutions or insertions/deletions) that the alignment implies.

We shall call *full* mapping an alignment algorithm that takes in input two sequences S_1 and S_2 and returns all the positions i of S_2 where S_1 is found *without* mutations (function f is the identity). We call instead *up to* mapping an alignment algorithm in which a similarity level is given to allow a certain number of mutations or insertions/deletions. For example, *up to* 1 SNP mapping searches all positions in S_2 where sequence S_1 is found with up to a SNP mutation.

The mapping algorithms whose input data are the NGS reads (millions of n-mer) and a genome (stored as up to gigabyte of information), are expensive in time and memory, and a brute force approach is usually unfeasible. To minimize these costs different optimization and/or heuristics techniques have been developed. As any heuristics, optimality of the results is not guaranteed. In particular in mapping based on heuristics we can have both *true* and *false* positive. A false positive in the algorithm output is a position j not satisfying the given similarity constraints. For example, considering in a *up to* 2 SNP algorithm, a sequence S_1 is a false positive if strictly more than 2 mutations are needed to obtain from S_1 a full matching onto (a part of) S_2. *False negative* are also possible: these are all the positions that do not belong to the output set but that are indeed proper alignments.

3 Color space features

Mapping algorithms handle input reads given in color and/or nucleotide space. Dibase sequencing techniques (color space) provide some benefits in terms of read accuracy since each base in the template is interrogated twice in independent primer rounds. This feature of the color space facilitates the discrimination between erroneous base calling and true genetic variant. Technology that gives

in output results in nucleotide space, uses instead other techniques to remove likely error earlier in the processing pipeline.

To avoid confusion, in this paper we call *mismatch* single mutation on a sequence in color space, *Single Nucleotide Polymorphism (SNP)* single mutation of nucleotide space and *SNP locus* multiple contiguous mutations in nucleotide or color space.

Hereafter, we deeply describe the color schema characteristics.

3.1 Color scheme

Given a nucleotide sequence of *k* bases the color encoding produces a string of length k: the first positions is a nucleotide base and the remaining *(k-1)* positions are colors. Each color represents four potential combination of two bases. Fig. 1 reports, for each pair of nucleotides, the corresponding color. The translation from nucleotide sequence to color sequence requires only the straight application of the table *ColorTab* shown in Fig 1.

	A	C	G	T
A	0	1	2	3
C	1	0	3	2
G	2	3	0	1
T	3	2	1	0

Figure 1: *ColorTab*: coding scheme of AB SOLiD system

The decoding of the color sequence into nucleotide sequence requires the application of color transformation one by one as follows. Let $f_c(L)$ be the function for the decoding. Then $f_c(L) = L'$ if $ColorTab[c, L'] = L$ (f returns the column of the row N corresponding to color c, for instance $f_3(A) = T$). If we consider a color sequence as a sequence of transformation defined by function f, then to decode a color sequence into a nucleotide sequence it is enough to recursively apply f: at each recursive call a new base is decoded. For example, to decode the sequence A320 we apply $f_3(A)$ to obtain T, to which f_2 is applied, getting $f_2(T) = C$ and so on until all bases are decoded.

Let t be the sequence of $f_c()$ applications, then the decoded sequence heavily depends on the first base. Consider the decoding of A32130 and C32130, then

$$t(A32130) \quad \text{produces} \quad ATCATT$$
$$t(C32130) \quad \text{produces} \quad CGACGG$$

The first nucleotide base should be retained in the color sequence: indeed the same color sequence leads to two completely different nucleotide sequences depending on the initial letter.

As reported in [30] it is mandatory to compare each read with respect to the reference genome before translating color reads in nucleotide space. This is due to two aspects: as first color-coding of every dibase pairing is not unique, a string of colors can represent one of several DNA strings depending on the preceding base. The second aspect regards the possibility that a sequence of matches and mismatches in color-space does not map uniquely into letter space.

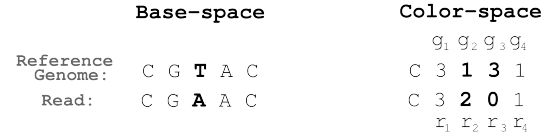

Figure 2: Example of isolate single base variant in color and nucleotide space

For example, as depicted in Fig. 2 a single SNP results in two adjacent color space mismatches. Even though there are 9 possible color pairs only three of them correspond to a SNP, while the rest lead to DNA sequences that is completely different with respect to the reference genome.

In the following we describe the main features of the color space and how it is possible to transform

a color sequence in a nucleotide sequence we explain how and why *up to k* similarity in color space can be correctly interpreted into nucleotide space.

3.2 Detecting mismatches and SNP

Any color can also be considered as a specific nucleotide transformation. Color 0 maps a base into itself, color 3 maps a nucleotide into its complement, color 1 exchange A in C and G in T and vice versa, finally color 2 completes all possible transformations by exchanging A in G and C in T and vice versa. Colors can be considered as transformations listed in Fig. 3.

$\oplus$	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

Figure 3: Color as transformations

Let T be the matrix reported in Fig. 3, then $T(i,j) = k$ means that the application of i followed by j on a base b is equivalent to applying k to b, and we write $i \oplus j = k$. In the Fig. 3 we have highlighted the case $i = 2, j = 1$ where results in $k = 3$.

Considering the example reported in Fig. 2 it is possible to observe how a *single* polymorphism is amplified in color-space on *two* adjacent color positions. When is it the case that k adjacent mismatches correspond to a SNP-locus? And what is the size of the locus? For this we take in account the operator $\oplus$, declared above, from pairs of color to color, through the matrix reported in Fig. 3. Indeed the coding scheme of Fig. 1 has been build so that the following proprieties are met:

- A dibase and its reverse get the same color;
- Two different dibases that have the same first base get different colors;
- Two different dibases that have the same second base get different colors;
- Monodibases get the same color;
- A dibase and its complement get the same color.

The coding scheme reported in Fig. 1 is the only one that satisfies the above proprieties, by two color permutations. Indeed once a row is fixed there is only one possible choice for the other three.

Considering the SNP reported in Fig. 2 the colors g_2, g_3, r_2, r_3 are consistent with this variant, if they satisfy the following proprieties:

$g_2 \neq r_2$ colors are different;

$g_2 \oplus g_3 = r_2 \oplus r_3$ color transformation is the same.

Note that the previous proprieties imply:
$g_3 \neq r_3$

Examining the sequences reported in Fig. 4 if only one of the previous proprieties is not satisfied, then there is only one color change instead of two.

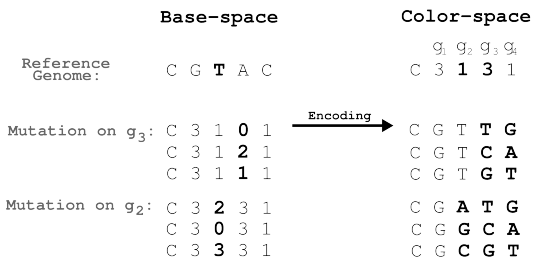

Figure 4: Effect of mismatch into nucleotide sequence

In the previous example both the color positions are relevant to correctly detect a single mismatch in base-space. In the following examples two and three adjacent mismatches are showed. Figure 5 reports an example of two and three contiguous nucleotide variants.

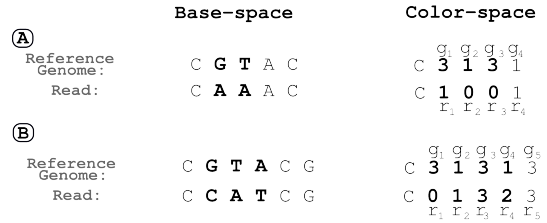

Figure 5: Example of SNP locus composed by two (A) and three (B) mutations in color and nucleotide space

In the first case, Fig 5(A), three necessary proprieties to obtain two adjacent mismatches in the respective nucleotide sequence are:

$g_1 \neq r_1$

$g_1 \oplus g_2 \neq r_1 \oplus r_2$

$g_1 \oplus g_2 \oplus g_3 = r_1 \oplus r_2 \oplus r_3$

Other two proprieties follow from the above:

$g_2 \oplus g_3 \neq r_2 \oplus r_3$

$g_3 \neq r_3$

The colors g_2 and r_2 could be equal or they could be different as well. Even though the center color positions are not mismatches, the analysis still apply since the example satisfy the above proprieties.

In the second case, Fig 5(B), the proprieties are:

$g_1 \neq r_1$

$g_1 \oplus g_2 \neq r_1 \oplus r_2$

$g_1 \oplus g_2 \oplus g_3 \neq r_1 \oplus r_2 \oplus r_3$

$g_1 \oplus g_2 \oplus g_3 \oplus g_4 = r_1 \oplus r_2 \oplus r_3 \oplus r_4$

It is worthwhile to enunciate a general theorem to specify when a color changing correspond to a SNP mutations. As reported in the AB SOLiD white paper, it is necessary to clarify two common misunderstanding that declare an isolate mismatch must always correspond to a sequencing error and that an isolated two-color changes always correspond an error. Indeed only when the following theorem is satisfied it is possible to consider a mutation true in color space and in base space.

Let $G = \langle b_0, g_1, \ldots, g_n \rangle$ and $R = \langle b_0, r_1, \ldots, r_n \rangle$ be two color sequences where $b_0 \in \{ACGT\}$ and $g_i, r_i \in \{0, 1, 2, 3\}$ then the k-color sub-string $R_k = \langle r_j \ldots, r_{k+j} \rangle$ encodes an isolated (k-1) base changing with respect to k-color sub-string $G_k = \langle g_j, \ldots, g_{k+j} \rangle$ iff

- full matching in nucleotide and color before j :$g_1, g_2, ..., g_{j-1} = r_1, r_2, ..., r_{j-1}$
- $\forall_{l<k} \bigoplus_{i=j}^{j+l} r_i \neq \bigoplus_{i=j}^{j+l} g_i$
- $\bigoplus_{i=j}^{j+k} r_i = \bigoplus_{i=j}^{j+k} g_i$

In addition to the mismatches, AB SOLiD is able to detect insertions or deletions. It should be noted that a base deleted/inserted from/in the read is depicted as a "-"; AB SOLiD has not a call corresponding to "-". This symbol would be generated only by aligning the read to the reference.

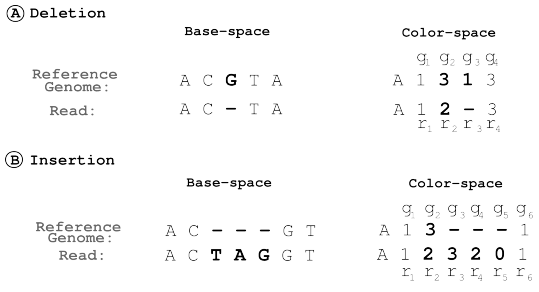

Figure 6: Example of deletion (A) and insertion (B) in color and nucleotide space

Fig. 6(A) reports a single base deletion in read sequence. In this case, the main propriety is $g_3 = r_2 \oplus r_3$, a single base deletion resulted in two elementary colors r_2 and r_3 being replaced by one

g_3. Another evidence is that all colors involved in the deletion are different, but if one of those is the identity color (0) the other two colors must be identical. This peculiar propriety can also be applied for multiple bases. The alignment reported in Fig 6(B) shows the insertion of three nucleotide in the read. The only propriety evinced in this case is that $g_2 = r_2 \oplus r_3 \oplus r_4 \oplus r_5$, the reference color g_2 in reference sequence must be replaced by the four color in the read.

Let $G = \langle b_0, g_1, \ldots, g_n \rangle$ and $R = \langle b_0, r_1, \ldots, r_m \rangle$ be two color sequences where $b_0, b'_0 \in \{ACGT\}$ and $g_i, r_i \in \{0, 1, 2, 3\}$ with $m >= n + k$ then R is obtained by a $k-$insertions from G in positions j iff

- full matching in nucleotide and color before j :$g_1, g_2, ..., g_j = r_1, r_2, ..., r_j$
- $\bigoplus_{i=j}^{j+k} g_i = r_{j+1}$

A deletion of k from G as the complementary operations: if R is obtained by a k insertion into G, the G can also be considered as the result of a k deletion from R.

4 Algorithms comparison

In the last two years the number of available mapping tools designed for NGS output has increased steadily, while the earlier alignment tool, like BLAST, are progressively abandoned due to their computational time and memory costs requirements. Indeed, BLAST was designed for finding the homologous sequences given a protein sequence. The adoption of BLAST as short-read alignment program has several problems in the consideration of both technology error rate and the elevate number of mismatches.

This new generation of tool must take into consideration two new aspects: the large amount of data produced, and the number of mismatches to be considered, a value that is driven by the species polymorphism rate and by the technology error rate [25].

All alignment tools considered follow a similar organization into macro steps:

- *Pre-processing* to pre-process the reference genome (or reads set) into one or more index tables;
- *Mapping* to find matches in the index tables for all queried subsequences and to locate the regions of potential homology;
- *Result Refinement* to examine all potential matches produced in the previous step using the full read-genome substring alignment.

We devote the rest of the section to discuss the main data structures and algorithms used in the first two steps and to present the 5 NGS alignment tools considered.

4.1 Data structures and algorithms

All mapping tool have a pre-processing phase in which either the reference genome or the set of reads are organized into data structures that allow a quick access to the information. Since this data structures are used for a very large number of alignments, then the execution time for their generation is not an issue (since it is easily amortized), while important factors that drive the choice of the data structure are the memory cost, the execution time of search, and the amount of search results that require further processing.

Seed and hash functions An hash table is basically an array which uses a hash function to efficiently map an input string, called key to an associated value. Each input string, typically large, is encoded by a hash function in a smaller datum, called hash, (i.e. a single integer); that is used as

the index of the array element where the associated value is stored. If the hash table stores the position of each substring of 35 nucleotides of a reference genome, and if the same substring appears more than once in the reference, then the hash table is not an array of integers, but an array of sets of integers (usually implemented using the bin data structure). The hash function applied to a read r will return the index of the array that contains the positions in the reference in which the read is found.

Ideally, the hash function should map each possible key to a different index; but this is not ensured in practice: two or more different keys may have the same hash (*hash collision*). For the mapping of genome this has significant consequences, since it may indeed be the case that the set associated with an entry in the hash table contains positions that match different substrings, so that a further step is required to ensure the mapping correctness (the *Result Refinement* step introduced before). This affects the efficiency of each search/lookup, even if the choice of an appropriate hash function can reduce the number of hash collisions.

The hash-based alignment algorithms can encode in hash table either the information of reference genome or of the input reads, so that a full mapping can be performed by the lookup of a read (or a set of reads) in the hash table of the reference genome, or vice versa. Each approach has its advantages and disadvantages: for instance the former requires a smaller and variable memory requirements due to the number and diversity of the input reads, but it may use more computation time to scan the whole genome when input reads are few. In both cases, due to hash collisions, all the found matches have to be verified.

This approach works well only in the full matching case: indeed to find a read with 1 mutation would require to search all reads obtained by introducing a single mutation (and there are 3 of them) in each possible position (as many position as the read size). To deal with mutations the concept of seed [11] has been introduced. A seed is a set of selected positions within a window which generates fixed length sub-sequences along a string. Seeds are used to discover a potential match between a read and a genomic portion with up to n mismatches; indeed it is always possible to divide a read in a set of consecutive seeds so that at least one has to match exactly in a genome position. In this case a potential match is declared and further examination has to be carried out to determine the whole similarity.

According to this idea the hash based techniques can be enriched with **consecutive seeds**: each sequence is divided in $k + m$ fragments. To ensure full sensitivity to k mismatches $k + m - 1$ hash tables are built and $\binom{k+m}{m}$ hashing steps are used to check for exact matches in the different combinations of m fragments. After that, the detected potential matches have to be still verified considering the whole sequence through specialized and accurate alignment algorithms. Increasing the value of m leads to a smaller number of potential matches, but to more hash tables and hash steps.

More recently **spaced seeds** have been introduced in [1]. A spaced seed is a set of "care" and "do not care" positions, annotated as "1" and "*" respectively. Usually the length of the seed is the total length of the string, and the weight of the seed is the number of 1 in the string (total number of care positions). It has been proved [16, 35] that spaced seeds are better than consecutive seeds in finding local similarities between two strings. However the speed of execution depends largely on the spaced seed employed, the number of desired matches and its length. By decreasing the seed weight the execution time increases: usually space seed with a weight around 16 became impractical due to memory limitations.

Burrows Wheeler Transformation and suffix array Other tools for sequence alignment are based on Burrows Wheeler Transformation (BWT) paired with a Compressed Suffix Array (CSA) [33]. The BWT is a well known algorithm used in

data compression (it is used for example in bzip2). It performs a transformation of an input sequence consisting of a reversible permutation of the sequence characters which gives a new string that is "easier to compress": if the original string has several substrings that occur often, then the transformed string has several places where a single character is repeated multiple time.

A suffix array is a data structure designed for efficient searching of a large text. It is an array of integers giving the starting position of the suffixes of a string in lexicographical order. It can be used as index to quickly locate every occurrence of a substring within the string. Finding every occurrence of the substring is equivalent to finding every suffix that begins with the substring. Thanks to the lexicographical ordering these suffixes will be grouped together in the suffix array, and can be found efficiently using a binary search.

Ferraggina and Manzini show in [24] that a suffix array is much more efficient (in both memory and searching time) if it is created from a BWT compressed sequences, rather than from the original sequence. A suffix array that works on compressed data is called Compressed Suffix Array (CSA).

The alignment algorithms based on BWT use a CSA to encode the reference genome and to search all the matches. Creating the CSA requires two steps: first the sequence order of the reference genome is modified using BWT. Next, the translated sequences are compressed and used to create the CSA.

This technique has been extended in [2] to allow alignments up to n mismatches: the binary search has been modified and, each time that a suffix does not occur, the algorithm selects an already-matched position, it replaces with another base and start the search again.

The choice of positions that have to be substituted is performed according to the quality score, so that all the positions with a poor quality score are selected before, a characteristics that may have an impact if the algorithm are instructed to find a single alignment for each read.

4.2 Tools

Hereafter, we describe the peculiar features of the five algortimhs which are compared in the next section.

SOCS is an tool that maps AB SOLiD data onto a reference genome. It works in color space and it uses consecutive seeds and hash tables built on the genome. SOCS takes $m = 1$, therefore for k mismatches it splits the read in $k + 1$ pieces, so that increasing the mismatches tolerance, the fragments used for each partial hash get smaller, and thus their hashes are less unique leading to higher number of "spurious hits".

Whenever a read matches (with one or more mismatches) in multiple positions the quality score is used to rank them taking into account also an approximate distribution of the probability of a given number of sequencing errors in a read.

The results obtained by SOCS are strictly related to color space. A run for k mismatches finds up to k mutations in the color string. As explained in Section 3 a mismatch found in color space does not match one to one with SNP or SNP locus in nucleotide space. We shall come back on this observation when we shall examine the experimental results in Section 5.

MAQ aligns short reads to the reference genome and infers variants, including SNP and short deletion (indel). MAQ works both in color and nucleotide space. The mapping algorithm is based on consecutive seeds dividing each read into $k + m$ fragments to provide full sensitivity to k mismatches. The value of m is fixed to 2. As a threshold between sensibility and efficiency, MAQ does not consider any mapping that has more than two mismatches in the first 28 positions. As we shall see in Section 5, this choice leads to very low sensibility for the case of three or more mutations.

If a read aligns to multiple positions, MAQ scores the various possibilities through quality scores and compatibility on the complementary strand. MAQ finds SNP and indel using mate-pair information. All the hits found on the forward strand of the reference sequence are stored in a queue. While examining the reverse strand, if a hit for a read is found, MAQ looks up the queue to check if there is a partial overlapping with one of the hits found on the forward strand. A pair of reads is correctly mapped if both the end of the hits are consistent, i.e. correct orientation within the proper distance. If only one can be mapped with confidence, a possible scenario is that an indel in one of the two reads occurred. The classical Smith Waterman algorithm [32] is then applied on the two reads to check validity of the alignment with/without indel or SNP. Even though MAQ is not any longer under active development, it has been considered in this review since in the past two years it has been one of the most popular mapping algorithms, used for many interesting biological studies. For instance, MAQ was used for the annotation and mapping task in one of the first papers that characterize single-nucleotide polymorphisms [8]: the authors found four millions SNPs and four hundred thousand structural variants, many of which were previously unknown. MAQ research team is now working on Burrows-Wheeler Aligner (BWA) [13], descibe below.

PerM was designed to provide full sensitivity to a high number of mismatches on genome-scale mapping. PerM can work in both color and base space. PerM uses periodic seeds to reduce execution time, and it offers a set of maximum-weight periodic seeds which are full sensitive to k mutations; PerM performs the post processing required to correctly classify color mismatches into SNP locus, as explained in Section 3, by checking each read, which represents a potential mapping, against the target positions of the genome. This check is efficiently implemented trough a bit-wise procedure.

The choice of the spaced seed is crucial to ensure a good mapping speed and the sensitivity of the algorithm. A user can choose from a set of predefined seeds, or can specify its own. Users should be aware that the design of seeds is all but a trivial task, since the seed does greatly influences the performance of the mapping algorithm.

SHRiMP is a mapping tool that works in color and nucleotide spaces. SHRiMP uses hash and spaced seeds inside the three steps procedure but with the peculiarity of doing hashing and mapping only over k-mer substrings, with k usually much smaller with respect to the read length, and the final refinement step only takes into account reads that have more than a given threshold of hits in the genome portion being examined. An high threshold may lead to low sensibility (some mapping may be lost), while a low threshold leads to an high computation cost of the refinement. The refinement step uses Smith-Waterman algorithm: a peculiarity of SHRiMP is that it implements the Smith-Waterman algorithms to work directly in color space.

SHRiMP also provides a confidence of the possible mappings of each read by using two statistics: *pchance*, the probability that the hit occurred by chance, and *pgenome*, the probability that the hit was generated by the genome, given the observed rates of the various evolutionary and error events, A good alignment should be characterized by a low *pchance* and a high *pgenome*.

Bowtie was specifically designed for the alignment of short sequences. It uses CSA with BWT-based compression to encode the reference genome. This operation is particularly efficient in memory (the human genome is encoded in only 1.3GB), without incurring in a significant time execution penalty. Bowtie implements a backtracking algorithm, that extends the one proposed in [24], to allow mismatches and to privilege high-

quality alignments. Searching for inexact alignments in Bowtie is performed using a quality-aware, greedy depth-first search through the space of possible alignments. If a valid alignment exists, then Bowtie will find it, but the greedy search can lead to "local solutions". Indeed Bowtie outputs stops at the first matching founded for a read, which may not be the *best* in terms of number of mismatches or/and quality. The user can set Bowtie to find *all* possible alignments and to return the best in terms of mismatches; but this choice has an impact on computation time (up to two/three times slower). Bowtie is one of the most used tool based on BWT. Other short-read alignment programs based on BWT and CSA includes BWA [13] and SOAP2 [26]. In particular, BWA is a new tool which implements two different algorithms to perform sequence alignment. These algorithms are both based on BWT transformation: the former algorithm is designed for short reads up to 200bp with low error rate while the latter one for long reads with more errors.

5 Dataset generation and tool comparison

In this section we compare the behavior of the five tools considered. Table 1 lists the websites from which the tools were downloaded, the version used in the experiments, and whether color space, base space, or both are available. Table 2 reports, for each tool, the specific run command used in the experiments.

The comparison is based on three "synthetic" datasets built from the Chromosome 22 genome, and only takes into account mismatches, while insertions and indel are not considered. The use of a synthetic dataset allows, for each choice of allowed mismatches, a known number of mappings, thus simplifying the computation of the sensitivity of the runs.

Dataset generation. To generate the three datasets we have realized a new tool that takes in input a Specific Set (SS) of sequences that are surely present in the reference genome, and a Background Set (BS) of sequences that are surely *not* present in it. If SS is used as input for a mapping algorithm against the reference genome, then the result should be that each read is mapped in at least one position. If BS is used instead, the result should be that no mapping has been found. To allow to test algorithms also in presence of mismatches and SNP, the tool produces in output a set of sequences obtained by introducing in the sequences of BS and SS $0, 1, \ldots, n$ mutations, where n is an input parameter. Introducing mutations is also a way to generate large sets of sequences that can be used to test the efficiencies of the alignment tools in terms of time and space.

The dataset generation tool consists of two separate programs. The first one generates from an input set of sequences a new set, introducing at most n mutations (n is an input parameter defined by the user); the second one takes two sets of sequences and merges them in a unique set according to an user specified policy (e.g. random order, first all the sequences in the first input file, first all the sequences in the second input file ...).

For each newly created sequence the tool maintains in the sequence identifier all the required information to evaluate the mapping algorithms, leading to the following naming scheme:

$$\langle Id\_source\_seq \rangle \text{`-'} \langle n\_m \rangle (\text{`*'} \langle pos \rangle)^{n\_m} \quad ,$$

where n_m is the number of mutations inserted in the sequence identified by Id_source_seq. For example *SEQ_ID$_i$-2*3*17* is the identifier of a sequence that has been derived from the source sequence of identifier *SEQ_ID$_i$*, introducing 2 mutations: one in position 3 and one in position 17.

Name	Download site	color - base space	Version
PerM	http://code.google.com/p/perm/	Yes/Yes	0.2.3
SOCS	http://socs.biology.gatech.edu/	Yes/No	1.2.1
SHRiMP	http://compbio.cs.toronto.edu/shrimp	Yes/Yes	1.3.2
MAQ	http://maq.sourceforge.net	Yes/Yes	0.7.1
Bowtie	http://bowtie-bio.sourceforge.net/index.shtml	Yes/Yes	0.12.3

Table 1: Mapping tools

Tool Name	Parameters
PerM	-B v 4 seed F4
SOCS	3 100000 16 false true
SHRiMP	-o 1 -n 1 -M 35bp -M fast
MAQ	-C 1 n 3
Bowtie	-t -p 16 -k 1 n 3

Table 2: Mapping parameters

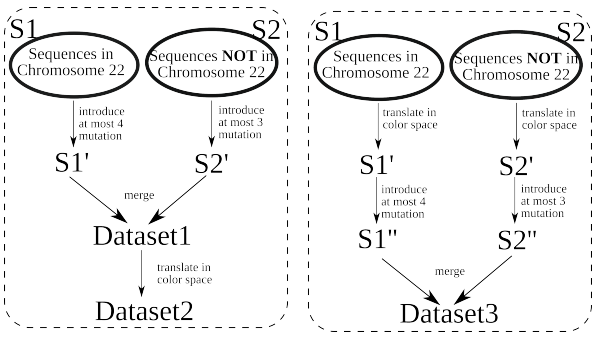

Figure 7: Derivation of the three datasets

Figure 7 illustrates the generation steps of the three datasets used for the experiments. Dataset1 has been generated starting from a SS and BS sets of respectively 54 and 2788 nucleotide sequences of length 35. These cardinalities have been chosen so as to produce from SS about $4,000,000$ sequences (using $n_m = 4$, at most 4 mutations per sequence, and from BS about $20,000,000$ sequences (using $n_m = 3$). The two generated sets have been merged using a random order, resulting in a dataset containing about $24,000,000$ sequences in nucleotide space, a number comparable with the reads produced by an AB SOLiD experiment.

Dataset2 is derived from Dataset1 by translating all its output sequences into color space; the resulting dataset contains about $24,000,000$ sequences in color space. Observe that the translation leads to color sequences that may contain more than 4 mutations, as discussed in Sec. 3.

Dataset3 also consists of about $24,000,000$ sequences and it is built as Dataset1, but translating first SS and BS into color space. In this dataset no sequence has more than 4 color mutations, that can nevertheless correspond to more than 4 mutations in the nucleotide space.

Of the three datasets, Dataset1 and Dataset2 share the same input data from a biological point of view, while Dataset3 can be used to check color space mapping algorithms per se, independently from their ability to correctly detect, using color coding, also the SNP locus.

The dataset generation tool will be public available soon as a web service accessible at: http://www6.unito.it/dataset/seqmdd/program.php

Comparison. The three datasets produced have been used to exercise the five considered tools on mapping, without considering indel and insertion. Although these algorithms share a certain similarity in the mapping process, the presence of specific main data structures, of different algorithms to assess similarity, and of different (and not trivial) optimizations techniques to speed-up comparison, may have a strong impact on computational

time and sensibility, especially when the number of mismatches increases.

The setup of the tool parameters was done using software documentation publicly available and trying to keep as homogeneous as possible the parameters setting over the five tools (see Table 2).

Readers should be aware that, since each of the inspected tool is characterized by many tuning parameters, it is possible that some of the results we have obtained can be further improved by particular parameters tuning. The comparison is far from being complete: more than trying to establish which tool is "the best" (a question whose answer is in most cases strongly dependent on the application at hand), we have tried to reproduce a situation in which an "informed" user runs the tools *to map reads allowing SNP* in base space (and mismatches in color space).

A peculiarity of certain tools is that, depending on an input parameter, they may or may not output reads that match in more than one position of the genome. Therefore, letting the tools tolerate mismatches, might results in a significant number of reads that are discarded. To simplify the mapping results evaluation, we have instructed all tools *not* to discard reads with possible multiple alignments, nevertheless all tools were instructed to produce only one alignment per read, usually the best one, to simplify the comparative analysis of sensitivity. Since the quality scores of the dataset are all set to the same value (the maximum allowed), there are cases in which the "best" option still produces more than one matching, which may induce an overestimation of the sensitivity of the tools.

Table 3 reports the results of the runs on Dataset1 (for base space) and Dataset2 (for color space). For each tool the table reports (column "Space") whether base space (*nt*) or color space (*cs*)was used. Column "Detection rate" reports the sensitivity of the algorithm, as the fraction of the sequences generated from SS that have been correctly detected with 0, 1, 2, and 3 mutations. The computation of the detection rate has required a post-processing of the output of each tool, to check the results against the known characteristics of the input dataset. Column "Time" is the execution time, while "Allowed mismatches/seed" are the number of allowed mismatches given in input for the runs, whenever this was allowed, and/or the seed used. We would have liked to perform our runs with 4 allowed mutations, but this was not possible or meaningful for the tools: SOCS, MAQ and Bowtie have a very low sensitivity already with 3 mutations.

A number of observations are indeed possible even on this limited comparison. Indeed all tools that work in both base and color, do perform better in base space, moreover execution times are quite comparable, but for SHRiMP. Indeed SHRiMP is significantly slower, but, as we shall discuss next, it has the highest sensitivity.

For a correct interpretation of the results we must emphasize that we are dealing with sequences in which up to 4 mutations have been introduced in base space, and readers should be aware that 4 SNPs in a sequence can lead to 8 mismatches when the sequence is translated into colors.

To clarify the difference between the sensitivity results of each tool, we should go a bit deeper into the computation of the results included in the detection rate columns, for $k = 0, 1, 2, 3$.

The column for $k = 0$ reports the percentage of sequences with 0 mutation found. Since 0 SNP results into 0 color mismatches, we expect no difference in sensitivity between base and color, which is indeed the case. As expected, all tools do a perfect job in finding full matching alignments (sensitivity of 1).

The column for $k = 1$ reports the percentage of sequences with 1 single nucleotide mutation that have been found while working in base space (row *nt*) on Dataset1 or in color space on Dataset2 (which is a mere color translation of Dataset1). Since a single SNP results in two adjacent mutations on the corresponding color sequence, then any tool that can find up to 2 mismatches in color, it is able to detect all sequences that contain 1 SNP. Indeed for $k = 1$ all the five tools reach a sensitiv-

Tool	Space	Detection rate				Time	Allowed mismatches-Seed
		0	1	2	3		
PerM	cs	1	1	0.99	0.17	4m 30s	F_4
	nt	1	1	1	0.81	3m	
SOCS	cs	1	1	0.18	0.02	50m	3
SHRiMP	cs	1	1	1	0.69	125m	–
	nt	1	1	1	1	119m	
MAQ	cs	1	1	0.04	0	6m	3
	nt	1	1	1	0.01	6m	
Bowtie	cs	1	1	0.09	0.01	6m	3
	nt	1	1	0.84	0.0	5m 45s	

Table 3: Mapping tools' results on Dataset1 and 2, run on 16 CPUs (4 x Quad-Core Intel Xeon E7320 processor 2.13GHz) with a total 100 Gb RAM.

ity of 100%.

Things gets a bit more tricky when $k = 2$. If the two SNPs are adequately spaced, then the corresponding color sequence contains exactly 4 color mismatches, adjacent two by two, and only the tools that correctly detect 4 mismatches can find them. If the two SNPs are adjacent, then the corresponding color sequence contains at most 3 consecutive mismatches (the first and third are surely different, while the second color may either be a mismatch or not). Note that the case of two SNPs separated by a matching position also leads to at most 4 mismatches. The five tools have rather diverse detection rate. In base space PerM, SHRiMP, and MAQ have full sensitivity, while Bowtie does not correctly aligns a 16% of the input read. In color, SHRiMP is the only one with full sensitivity, PerM looses a 1%, which should not be the case since PerM is run with an F_4 seed which should ensure full sensitivity to 4 mutations. We could not find a good explanation for this behavior. MAQ and Bowtie exhibit a very low sensibility, which is not surprising, considering the low sensibility that they have for base space with $k = 3$, which implies that all pairs of isolated SNPs are almost never detected. Also SOCS does not do a good job for $k = 2$, again, 3 color mismatches only allow to find a 18% of the reads that could be aligned through mutations of 2 base positions (which are presumably the ones in which the two SNPs are adjacent). SOCS does nevertheless shows an improvement over MAQ and Bowtie, nevertheless quite limited considering that SOCS has been developed for color space.

If we consider instead $k = 3$, in base space only SHRiMP reaches full sensitivity, while PerM looses 19% of the reads. Again, it is unclear while this is happening in PerM, considering that we are using the F_4 seed. MAQ and Bowtie have very low sensitivity, presumably because they both follow the policy of considering only sequences with up to 2 mutations in the first 28 positions of the read. For what concerns color space, if the 3 SNPs are well spaced then the corresponding color sequence contains exactly 6 color mismatches, and only the tools that correctly detect 6 mismatches can detect them (and, accordingly to the obtained results, this is never the case). If the 3 SNPs are instead adjacent, then the corresponding color sequence contains at most 4 consecutive mismatches (the first and fourth are surely different, while the second and third positions may either be a mismatch or not). If a tool is limited by 3 mutations, it surely cannot detect all the cases that result in a string of 4 different adjacent colour, while it may depend on the specific tool how the other cases are considered. Again SHRiMP has the best sensitivity, which could be a consequence of having full sen-

sitivity to the base space case for $k = 3$, although only 69% of the read are correctly aligned if the alignment requires 3 SNPs. PerM gets a low 17% of sequences, while the other three tools only detects very few percents.

Dataset3 has been used only for PERM, which results in a sensitivity of 1 for $k = 0, 1, 2$ and a sensitivity of 0.99 for the case of 3 mismatches. Again, it is unclear while there is such a discrepancy between the sensitivity for $k = 3$ in base for Dataset1 with respect to color in Dataset3. A reason could be that PerM is "color-aware" and further investigation is needed to understand how single isolated mismatches are treated.

6 Conclusions

Next Generation Sequencing (NGS) is becoming more used in genomic fields. In the last 5 years we have seen an enormous increase in NGS data throughput and in reads length, also associated with an incredible drop of cost per gigabase. NGS is opening unprecedented opportunities for generating new knowledge and translating it into applications that enhance human health. We are now in the early phase of a new big jump in biological knowledge acquisition which is likely to be bigger than the one produced by the parallelization of transcription analysis. This could lead in an expansion of expression studies of enzyme involved in novel biosynthetic pathways which are a priority in drug discovery from natural products.

In NGS a big effort has been made to develop computational tools to discover the maximum number of correct alignments of reads over a reference genome. New tools have been optimized for this task and in this review we have described their algorithmic structure and we have given a basic view of their strength and weakness when applied to NGS data encoded in nucleotide or color space. The results have been obtained and presented taking an user perspective. Indeed a large discrepancy in the quality of the results between color and nucleotide mapping, in presence of mutations, to explain why it is so this review has devoted special attention to the relationships between mutations in color space and mutations in base space, a topic for which little coverage in the literature exists.

7 Acknowledgments

This study was funded under the auspices of EUCAAD 200755. The project EUCAAD has received research funding from the European Community's Seventh Framework Programme. This work was also supported by grants from Italian Association for Cancer Research; the Italian Ministero dellUniversit e della Ricerca; the University of Torino.

References

[1] Califano A and Rigoutsos I. Flash: a fast look-up algorithm for string homology. computer vision and pattern recognition. In *Proceedings CVPR, IEEE Computer Society Conference*, pages 353–359, 1993.

[2] Langmead B, Trapnell C, Pop M, and Salzberg SL. Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome Biol.*, 10, 2009.

[3] Ondov BD, Varadarajan A, Passalacqua KD, and Bergman NH. Efficient mapping of applied biosystems solid sequence data to a reference genome for functional genomic applications. *Bioinformatics.*, 24:2776–2777, 2008.

[4] ABI-SOLiD (Applied Biosystems). http://www..appliedbiosystems.com.

[5] Horner DS, Pavesi G, Castrignan T, De Meo PD, Liuni S, Sammeth M, Picardi E, and

Pesole G. Bioinformatics approaches for genomics and post genomics applications of next-generation sequencing. *Brief Bioinform.*, 11:181–197, 2010.

[6] Pettersson E, Lundeberg J, and Ahmadian A. Generations of sequencing technologies. *Genomics*, 93:105–111, 2009.

[7] Mardis ER. The impact of next-generation sequencing technology on genetics. *Trends Genet.*, 24:133–141, 2008.

[8] David R. B. et al. Accurate whole human genome sequencing using reversible terminator chemistry. *Nature*, 456:53–59, 2008.

[9] Petrosino J. F., Highlander S., Luna R. A., Gibbs R. A., and Versalovic J. Metagenomic pyrosequencing and microbial identification. *Clin. Chem.*, 55:856866, 2009.

[10] Sanger F, Nicklen S, and Coulson AR. Dna sequencing with chain-terminating inhibitors. *Proc. Natl. Acad. Sci.*, 74:5463–5467, 1977.

[11] Myers G. A fast bit-vector algorithm for approximate string matching based on dynamic programming. *J. ACM*, 46:395–415, 1999.

[12] Li H, Ruan J, and Durbin R. Mapping short dna sequencing reads and calling variants using mapping quality scores. *Genome Res.*, 18:1851–1858, 2008.

[13] Li H. and Durbin R. Fast and accurate short read alignment with burrows-wheeler transform. *Bioinformatics*, 25:17541760, 2009.

[14] Genome Analyzer (Illumina/Solexa). http://www.illumina.com.

[15] Park P. J. Chipseq: advantages and challenges of a maturing technology. *Nature Reviews Genetics*, 10:669680, 2009.

[16] Xu J, Brown D, Li M, and Ma B. Optimizing multiple spaced seeds for homology search. *Journal of Computational Biolology*, 13:1355–1368, 2006.

[17] McPherson JD. Next-generation gap. *Nat Methods.*, 6, 2009.

[18] Bonfield JK and Staden R. The application of numerical estimates of base calling accuracy to dna sequencing projects. *Nucleic Acids Research*, 23:1406–1410, 1995.

[19] Li JW and Vederas JC. Drug discovery and natural products: end of an era or an endless frontier? *Science*, 325:161–165, 2009.

[20] Andries K, Verhasselt P, Guillemont J, Ghlmann HWH, Neefs JM, Winkler H, Van Gestel J, Timmerman P, Zhu M, Lee E, Williams P, De Chaffoy D, Huitric E, Hoffner S, Cambau E, Truffot-Pernot C, Lounis N, and Jarlier V. A diarylquinoline drug active on the atp synthase of mycobacterium tuberculosis. *Science*, 307:223–227, 2005.

[21] Metzker M.L. Sequencing technologies the next generation. *Nature Reviews Genetics*, 11:31–46, 2010.

[22] Rusk N and Kiermer V. Primer: sequencing: the next generation. *Nat. Methods.*, 5, 2008.

[23] Morozova O and Marra MA. Applications of next-generation sequencing technologies in functional genomics. *Genomics*, 92:255–264, 2008.

[24] Ferragina P and Manzini G. An experimental study of an opportunistic index. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete algorithms Washington, DC: Society for Industrial and Applied Mathematics*, pages 269–278, 2001.

[25] Flicek P and Birney E. Sense from sequence reads: methods for alignment and assembly. *Nat Methods.*, 6:S6–S12, 2009.

[26] Li R., Yu C, Li Y., Lam T. W., Yiu S.M., Kristiansen K, and Wang J. Soap2: an improved ultrafast tool for short read alignment. *Bioinformatics*, 25:19661967, 2009.

[27] Strausberg RL, Levy S, and Rogers YH. Emerging dna sequencing technologies for human genomic medicine. *Drug Discov Today*, 13:569–577, 2008.

[28] 454 (Roche). http://www.454.com.

[29] Marguerat S, Wilhelm BT, and Bhler J. Next-generation sequencing: applications beyond genomes. *Biochem Soc Trans.*, 36:1091–1096, 2008.

[30] Rumble SM, Lacroute P, Dalca AV, Fiume M, Sidow A, and Brudno M. Shrimp: accurate mapping of short color-space reads. *PLoS Comput Biol.*, 5, 2009.

[31] Lee SS and Mudaliar A. Racing forward: the genomics and personalized medicine act. *Science*, 323:342, 2009.

[32] Smith TF and Waterman MS. Identification of common molecular subsequences. *J Mol Biol*, 147:195–197, 1981.

[33] Manber U and Myers G. Suffix arrays: a new method for on-line string searches. *SIAM J. Comput.*, 22(5):935–948, 1993.

[34] Chen Y, Souaiaia T, and Chen T. Perm: efficient mapping of short sequencing reads with periodic full sensitive spaced seeds. *Bioinformatics*, 25:2514–2521, 2009.

[35] Sun Y and Buhler J. Designing seeds for similarity search in genomic dna. *Journal of Computer and System Sciences*, pages 67–75, 2003.

[36] Wang Z., Gerstein M., and Snyder M. Rna-seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, 10:57–63, 2009.