


REVIEW

Open Access



Technology dictates algorithms: recent developments in read alignment

Mohammed Alser^{1,2,3†}, Jeremy Rotman^{4†}, Dhrithi Deshpande⁵, Kodi Taraszka⁴, Huwenbo Shi^{6,7}, Pelin Icer Baykal⁸, Harry Taegyun Yang^{4,9}, Victor Xue⁴, Sergey Knyazev⁸, Benjamin D. Singer^{10,11,12}, Brunilda Balliu¹³, David Koslicki^{14,15,16}, Pavel Skums⁸, Alex Zelikovsky^{8,17}, Can Alkan^{2,18}, Onur Mutlu^{1,2,3†} and Serghei Mangul^{5*†} 

* Correspondence: serghei.mangul@gmail.com

[†]Mohammed Alser and Jeremy Rotman contributed equally to this work.

[†]Onur Mutlu and Serghei Mangul jointly supervised this work.

⁵Department of Clinical Pharmacy, School of Pharmacy, University of Southern California, Los Angeles, CA 90089, USA

Full list of author information is available at the end of the article

Abstract

Aligning sequencing reads onto a reference is an essential step of the majority of genomic analysis pipelines. Computational algorithms for read alignment have evolved in accordance with technological advances, leading to today's diverse array of alignment methods. We provide a systematic survey of algorithmic foundations and methodologies across 107 alignment methods, for both short and long reads. We provide a rigorous experimental evaluation of 11 read aligners to demonstrate the effect of these underlying algorithms on speed and efficiency of read alignment. We discuss how general alignment algorithms have been tailored to the specific needs of various domains in biology.

Introduction

In April 2003, the high-throughput sequencing era started with the Human Genome Project, which led to the successful sequencing of a nearly complete human genome and establishment of a reference genome that is still in use [1]. The Human Genome Project cost approximately \$3 billion over 13 years to sequence the genome of an individual human. Recent advances in high-throughput sequencing technologies have enabled cost-effective and time-efficient probing of the DNA sequences of living organisms through a process known as DNA sequencing [2]. Modern high-throughput sequencing techniques are capable of producing millions of nucleotide sequences of an individual's DNA [3] and providing multifold coverage of whole genomes or particular genomic regions. The output of high-throughput sequencing consists of sets of relatively short genomic sequences, usually referred to as *reads*. Contemporary sequencing technologies are capable of generating tens of millions to billions of reads per sample, with read lengths ranging from a few hundred to a few million base pairs [4].

The trade-off for decreased cost and increased throughput offered by modern sequencing technologies is a larger margin of noise in sequencing data [5]. The magnitude of error rates in data produced by state-of-the-art sequencing platforms varies from $\sim 10^{-3}$ for short reads to $\sim 15 \times 10^{-2}$ for the relatively new long and ultra-long



© The Author(s). 2021 **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>. The Creative Commons Public Domain Dedication waiver (<http://creativecommons.org/publicdomain/zero/1.0/>) applies to the data made available in this article, unless otherwise stated in a credit line to the data.

reads [6]. The increased error rate of today's emerging long-read technologies may negatively impact biological interpretations. For example, errors in protein-coding regions can bias the accuracy of protein predictions [7]. Sequenced reads lack information about the order and origin (i.e., which part, homolog, and strand of the subject genome) of reads. The main challenge in genome analysis today is to reconstruct the complete genome of an individual. This process, *read alignment* (also known as *read mapping*), typically requires the reference genome which is used to determine the potential location of each read. Accuracy of alignment has a strong effect on many downstream analyses [8]. For example, most trans-eQTL signals were shown to be solely caused by alignment errors [9].

Read alignment can be performed in a brute force manner but is impractical for modern sequencing platforms capable of producing hundreds of millions of reads. Instead, today's efficient bioinformatics algorithms enable fast and accurate read alignment and can be thousands of orders of magnitude faster when compared to the naive brute force approach [10] (Supplementary Note 1). Read alignment enables observation of the differences between the read and the reference genome. These differences can be caused by either real genetic variants in the sequenced genome or errors generated by the sequencing platform. These sequencing errors and read lengths, which are typically short, make the read alignment problem computationally challenging. The continued increase in the throughput of modern sequencing technologies creates additional demand for efficient algorithms for read alignment. Over the past several decades, a plethora of tools were developed to align reads onto reference genomes across various domains of biology. Previous efforts that provide overviews of various algorithms and techniques used by read aligners are presented elsewhere [10–12], including studies that present benchmarks of existing tools [13, 14]. Since the time those efforts were published, many new alignment algorithms have been developed. Additionally, previous efforts lack a historical perspective on algorithm development.

Our review provides a historical perspective on how technological advancements in sequencing are shaping algorithm development across various domains of modern biology, and we systematically assess the underlying algorithms of a large number of aligners ($n = 107$). Algorithmic development and challenges associated with read alignment are to a large degree data- and technology-driven, and emerging highly accurate ultra-long-read sequencing techniques promise to expand the application of read alignment.

Where do reads come from—advantages and limitations of read alignment

One can study an individual genome using sequencing data in two ways: by mapping reads to a reference genome, if it exists, or by de novo assembling the reads. The complexity of the human genome, in combination with the short length of sequenced reads, poses substantial challenges to our ability to accurately assemble personal genomes [15]. Even recently-introduced ultra-long reads [16] (up to 2 Mb) offer the limited capacity to build a de novo assembly of an individual genome with no prior knowledge about the reference genome [16]. The presence of many repetitive regions in the human genome limits our ability to assemble a personal human genome as a single sequence. Emerging long-read sequencing technologies that are capable of producing ultra-long reads [16] promise to deliver more accurate assemblies [17]. However, the

relatively high error rate of data output from recently developed long-read sequencing technologies often results in inaccuracies in the assembled genomes, especially when using low sequencing coverage [18, 19].

The read alignment problem is known to be solvable in polynomial time [20], while a polynomial-time solution for genome assembly is still unknown [20–22]. Genome assembly is typically slower and more computationally intensive than read alignment [17, 23, 24] due to the presence of repeats that are much longer than the typical read length. This makes assembly impractical in studies that involve large-scale clinical cohorts of thousands of individuals. At the same time, when the reference genome is unknown, long reads are a valuable resource for assembling genomes that are far more complex than the human genome, such as the hexaploid bread wheat genome [17, 23, 25].

The availability of a large number of alignment methods that are scalable to both read length and genome size has enabled read alignment to become an essential component of high-throughput sequencing analysis (Table 1) [26]. However, read alignment also has its own fundamental challenges. First, some challenges are caused by the incompleteness of the reference genomes that have multiple assembly gaps [16]. Reads originating from these gaps often remain unmapped or are incorrectly mapped to homologous regions. Second, the presence of repetitive regions of the genome confounds current read alignment techniques, which often map reads originating from one region to match *several* other repetitive regions (such reads are known as multi-mapped reads). In such cases, most read aligners *simply* report one location randomly selected among the possible mapping locations, in turn, significantly reducing the number of detected variants [27]. Third, read alignment techniques should tolerate differences between reads and the reference genome. These differences may correspond to a single nucleotide (including deletion, insertion, and substitution of a nucleotide) or to larger structural variants [28]. Fourth, read alignment algorithms need to align reads to both forward and reverse DNA strands of the same reference genome in order to tackle the strand bias problem, defined as the difference in genotypes identified by reads that map to forward and reverse DNA strands. Strand bias is likely caused by errors introduced during library preparation and not by mapping artifacts [27, 29].

Co-evolution of read alignment algorithms and sequencing technologies

Over the past few decades, we have observed an increase in the number of alignment tools developed to accommodate rapid changes in sequencing technology (Table 1). Published alignment tools use a variety of algorithms to improve the accuracy and speed of read alignment (Table 2). At the same time, the development of read alignment algorithms is impacted by rapid changes in sequencing technologies, such as read length, throughput, and error rates (Supplementary Table 1). For example, some of the first alignment algorithms (e.g., BLAT [38]) were designed to align expressed sequence tag (EST) sequences, which are 200 to 500 bp in length. Another early alignment algorithm, BLASTZ [39], was designed to align 1 Mb human contigs onto the mouse genome. After short reads became available, the majority of the algorithms have focused on the problem of aligning hundreds of millions of short reads to a reference genome. Recent sequencing technologies are capable of producing multi-megabase reads at the cost of high error rates (up to 20%)—a development that poses additional challenges

Table 1 Summary of algorithms and features of the examined read alignment methods. We surveyed 107 alignment tools published from 1988 to 2020 (indicated in column “Year of publication”). The table is sorted by year of publication, and then grouped according to the area(s) of application (indicated in column “Application”) within each year. In column “Indexing,” we document the algorithms used to index the genome (the first step in read alignment). In column “Global Positioning,” we document the algorithms used to determine a global position of the read in the reference genome (the second step). In column “Pairwise alignment,” we document the algorithm used to determine the similarity between the read and the corresponding region of the reference genome (the last step). SW, NW, HD, and DP stand for Smith-Waterman algorithm, Needleman-Wunsch algorithm, Hamming distance, and dynamic programming, respectively. In column “Wrapper,” we document the read alignment algorithms that are built on top of other read alignment tools. Finally, we report the maximum read length tested in the corresponding paper in column “Max. Read Length Tested in the Paper (bp).” The tested read length in each paper is not necessarily the maximum read length that each tool can handle

Aligner	URL	Year of publication	Application	Indexing	Global Positioning			Pairwise alignment	Wrapper	Max. read length tested in the paper (bp)
					Fix length seed	Spaced seed	Seed chaining			
FASTA [30]	https://fasta.bioch.virginia.edu/fasta_www2/fasta_list2.shtml	1988	DNA	Hashing	Y	N	Y	SW and NW	N	1500
BLAST [31]	https://blast.ncbi.nlm.nih.gov/Blast.cgi	1990	DNA	Hashing	Y	N	Y	Non-DP Heuristic	N	73360
Gapped BLAST [32]	https://blast.ncbi.nlm.nih.gov/Blast.cgi	1997	DNA	Hashing	Y	N	Y	SW	N	246
SSAHA [33]	https://www.sanger.ac.uk/science/tools/ssaha	2001	DNA	Hashing	Y	N	N	NW	N	500
PatternHunter [34–37]	https://www.bioinformatics.org/	2002	DNA	Hashing	Y	Y	Y	Non-DP heuristic	N	500
BLAT [38]	https://genome.ucsc.edu/cgi-bin/hgBlat	2002	DNA	Hashing	Y	N	Y	Non-DP heuristic	N	500
BLASTZ [39]	https://www.bx.psu.edu/miller_lab/	2003	DNA	Hashing	Y	N	N	SW	Y	3000
C4 [40]	https://github.com/nathanweeks/exonerate	2005	DNA	Hashing	Y	N	Y	Sparse DP	N	N/A
GMAP [41]	https://github.com/juliangehring/GMAP-GSNAP	2005	DNA	Hashing	N	N	Y	NW	N	N/A
BWT-SW [42]	https://github.com/mruffalo/bwt-sw	2008	DNA	BWT	Y	N	N	SW	N	2000
MAQ [43]	http://maq.sourceforge.net/maq-man.shtml	2008	DNA	Hashing	Y	Y	N	SW	N	63
RMAP [44]	https://github.com/smithlabcode/rmap	2008	DNA	Hashing	Y	N	N	HD	N	36
SOAP [45]	https://github.com/ShujiaHuang/SOAPaligner	2008	DNA	Hashing	Y	N	N	Non-DP heuristic	N	50

Table 1 Summary of algorithms and features of the examined read alignment methods. We surveyed 107 alignment tools published from 1988 to 2020 (indicated in column “Year of publication”). The table is sorted by year of publication, and then grouped according to the area(s) of application (indicated in column “Application”) within each year. In column “Indexing,” we document the algorithms used to index the genome (the first step in read alignment). In column “Global Positioning,” we document the algorithms used to determine a global position of the read in the reference genome (the second step). In column “Pairwise alignment,” we document the algorithm used to determine the similarity between the read and the corresponding region of the reference genome (the last step). SW, NW, HD, and DP stand for Smith-Waterman algorithm, Needleman-Wunsch algorithm, Hamming distance, and dynamic programming, respectively. In column “Wrapper,” we document the read alignment algorithms that are built on top of other read alignment tools. Finally, we report the maximum read length tested in the corresponding paper in column “Max. Read Length Tested in the Paper (bp).” The tested read length in each paper is not necessarily the maximum read length that each tool can handle (*Continued*)

Aligner	URL	Year of publication	Application	Indexing	Global Positioning			Pairwise alignment	Wrapper	Max. read length tested in the paper (bp)
					Fix seed	length seed	Spaced seed	Seed chaining		
SOCS [46]	http://socs.biology.gatech.edu/	2008	DNA	Hashing	Y	N	N	N	Rabin-Karp Algorithm	35
SeqMap [47]	http://www-personal.umich.edu/~jianghui/seqmap/	2008	DNA	Hashing	Y	N	N	N	Non-DP Heuristic	30
ZOOM [48]	http://www.bioinform.com/zoom-1-3-gui-release-next-gen-seq/	2008	DNA	Hashing	Y	Y	Y	N	SW	36
QPALMA [49, 50]	http://www.raetschlab.org/suppl/qpalma	2008	RNA-Seq	Suffix array	Y	N	Y	Y	SW	36
BRAT [51]	http://compbio.cs.ucr.edu/brat/	2009	BS-Seq	Hashing	Y	N	N	N	HD	26
BSMAP [52]	https://github.com/genome-vendor/bsmap	2009	BS-Seq	Hashing	Y	N	N	N	HD	32
BFAST [53]	https://github.com/nh13/BFAST/	2009	DNA	Hashing	N	Y	Y	N	SW	55
BWA [54]	https://github.com/lh3/bwa	2009	DNA	BWT-FM	N	N	N	N	Semi-Global	125
Bowtie [55]	http://bowtie-bio.sourceforge.net/manual.shtml	2009	DNA	BWT-FM	Y	N	N	N	HD	76
CloudBurst [56]	https://sourceforge.net/projects/cloudburst-bio/	2009	DNA	Hashing	Y	N	N	N	Landau-Vishkin	36
GNUMAP [57]	https://github.com/byuucs/gnumap	2009	DNA	Hashing	Y	N	N	Y	NW	36
GenomeMapper [58]	http://1001genomes.org/software/genomemapper_singleref.html	2009	DNA	Hashing	Y	N	N	Y	NW	200
MOM [59]	https://github.com/hugheaves/MOM	2009	DNA	Hashing	Y	N	N	N	HD	40

Table 1 Summary of algorithms and features of the examined read alignment methods. We surveyed 107 alignment tools published from 1988 to 2020 (indicated in column “Year of publication”). The table is sorted by year of publication, and then grouped according to the area(s) of application (indicated in column “Application”) within each year. In column “Indexing,” we document the algorithms used to index the genome (the first step in read alignment). In column “Global Positioning,” we document the algorithms used to determine a global position of the read in the reference genome (the second step). In column “Pairwise alignment,” we document the algorithm used to determine the similarity between the read and the corresponding region of the reference genome (the last step). SW, NW, HD, and DP stand for Smith-Waterman algorithm, Needleman-Wunsch algorithm, Hamming distance, and dynamic programming, respectively. In column “Wrapper,” we document the read alignment algorithms that are built on top of other read alignment tools. Finally, we report the maximum read length tested in the corresponding paper in column “Max. Read Length Tested in the Paper (bp).” The tested read length in each paper is not necessarily the maximum read length that each tool can handle (*Continued*)

Aligner	URL	Year of publication	Application	Indexing	Global Positioning			Pairwise alignment	Wrapper	Max. read length tested in the paper (bp)
					Fix length seed	Spaced seed	Seed chaining			
PASS [60]	http://pass.cribi.unipd.it/cgi-bin/pass.pl	2009	DNA	Hashing	Y	N	Y	NW	N	32
PerM [61]	https://code.google.com/archive/p/perm/downloads	2009	DNA	Hashing	Y	Y	N	HD	N	47
RazerS [62]	https://github.com/seqan/seqan/tree/master/apps/razers	2009	DNA	Hashing	Y	Y	Y	Myers Bit Vector	N	76
SHRIMP [63]	http://compbio.cs.toronto.edu/shrimp/	2009	DNA	Hashing	N	N	N	SW	N	35
SOAP2 [64]	https://github.com/ShujiaHuang/SOAPaligner	2009	DNA	BWT-FM	Y	N	N	SW	N	44
Slider [65]	http://www.bcgsc.ca/platform/bioinfo/software/slider	2009	DNA	Hashing	Y	N	N	HD	N	36
segemehl [66]	https://www.bioinf.uni-leipzig.de/Software/segemehl/	2009	DNA	Suffix array	N	N	Y	SW	N	35
TopHat [67]	https://ccb.jhu.edu/software/tophat/index.shtml	2009	RNA-Seq	BWT-FM	Y	N	N	HD	Y	42
BS-Seeker [68]	http://pellegriini-legacy.mcdm.ucla.edu/bs_seeker/BS_Seeker.html	2010	BS-Seq	BWT-FM	Y	N	N	HD	Y	36
BWA-SW [54]	https://github.com/lh3/bwa	2010	DNA	BWT-FM	N	N	N	SW	N	10000
GASST [35]	http://www.irisa.fr/symbiose/projects/gasst/	2010	DNA	Hashing	Y	Y	Y	Semi-Global	N	500
GSNAP [37]	https://github.com/juliangehring/GMAP-GSNAP	2010	DNA	Hashing	Y	N	Y	Non-DP Heuristic	N	100
SMALT [69]	https://github.com/rcallahan/smalt	2010	DNA	Hashing	Y	N	Y	SW	N	150

Table 1 Summary of algorithms and features of the examined read alignment methods. We surveyed 107 alignment tools published from 1988 to 2020 (indicated in column “Year of publication”). The table is sorted by year of publication, and then grouped according to the area(s) of application (indicated in column “Application”) within each year. In column “Indexing,” we document the algorithms used to index the genome (the first step in read alignment). In column “Global Positioning,” we document the algorithms used to determine a global position of the read in the reference genome (the second step). In column “Pairwise alignment,” we document the algorithm used to determine the similarity between the read and the corresponding region of the reference genome (the last step). SW, NW, HD, and DP stand for Smith-Waterman algorithm, Needleman-Wunsch algorithm, Hamming distance, and dynamic programming, respectively. In column “Wrapper,” we document the read alignment algorithms that are built on top of other read alignment tools. Finally, we report the maximum read length tested in the corresponding paper in column “Max. Read Length Tested in the Paper (bp).” The tested read length in each paper is not necessarily the maximum read length that each tool can handle (*Continued*)

Aligner	URL	Year of publication	Application	Indexing	Global Positioning			Pairwise alignment	Wrapper	Max. read length tested in the paper (bp)
					Fix seed	length seed	Spaced seed	Seed chaining		
Slider II [70]	http://www.bcgsc.ca/platform/bioinfo/software/SliderII	2010	DNA	Hashing	Y		N	N	Y	42
VMATCH [71]	http://www.vmatch.de/	2010	DNA	Suffix array	Y		N	Y	Y	N/A
mrsFAST [72]	https://github.com/sfu-compbio/mrsfast	2010	DNA	Hashing	Y		N	N	N	100
MapSplice [73]	https://github.com/LiuBioinfo/MapSplice	2010	RNA-Seq	BWT-FM	Y		N	N	Y	100
MicroRazerS [74]	https://github.com/seqan/seqan/tree/master/apps/micro_razers	2010	RNA-Seq	Hashing	Y		N	Y	N	36
SpliceMap [75]	http://web.stanford.edu/group/wonglab/SpliceMap/	2010	RNA-Seq	Hashing	Y		N	N	Y	50
Supersplat [76]	http://mocklerlab.org/tools/1/manual	2010	RNA-Seq	Hashing	N		N	N	N	36
Bismark [77]	https://github.com/FelixKrueger/Bismark	2011	BS-Seq	BWT-FM	Y		N	Y	Y	50
LAST [78]	http://last.cbrc.jp/	2011	DNA/BS-Seq/RNA	Suffix array	N		Y	N	N	105
DynMap [79]	https://dl.acm.org/citation.cfm?id=2147845&dl=ACM&coll=DL	2011	DNA	Hashing	Y		N	N	N	52
SHRIMP2 [80]	http://compbio.cs.toronto.edu/shrimp/	2011	DNA	Hashing	Y		Y	Y	N	75
SNAP [81]	http://snap.cs.berkeley.edu/	2011	DNA	Hashing	Y		N	N	N	10000
Stampy [82]	https://www.well.ox.ac.uk/project-stampy	2011	DNA	Hashing	Y		N	N	N	4500

Table 1 Summary of algorithms and features of the examined read alignment methods. We surveyed 107 alignment tools published from 1988 to 2020 (indicated in column “Year of publication”). The table is sorted by year of publication, and then grouped according to the area(s) of application (indicated in column “Application”) within each year. In column “Indexing,” we document the algorithms used to index the genome (the first step in read alignment). In column “Global Positioning,” we document the algorithms used to determine a global position of the read in the reference genome (the second step). In column “Pairwise alignment,” we document the algorithm used to determine the similarity between the read and the corresponding region of the reference genome (the last step). SW, NW, HD, and DP stand for Smith-Waterman algorithm, Needleman-Wunsch algorithm, Hamming distance, and dynamic programming, respectively. In column “Wrapper,” we document the read alignment algorithms that are built on top of other read alignment tools. Finally, we report the maximum read length tested in the corresponding paper in column “Max. Read Length Tested in the Paper (bp).” The tested read length in each paper is not necessarily the maximum read length that each tool can handle (*Continued*)

Aligner	URL	Year of publication	Application	Indexing	Global Positioning			Pairwise alignment	Wrapper	Max. read length tested in the paper (bp)
					Fix length seed	Spaced seed	Seed chaining			
TMAP	https://github.com/ontorrent/TS/tree/master/Analysis/TMAP	2011	DNA	BWT-FM	N	N	Y	SW	N	N/A
X-Mate [83]	http://grimmond.imb.uq.edu.au/X-MATE/	2011	DNA	Hashing	N	N	N	Non-DP Heuristic	N	50
SOAPSplice [84]	http://soap.genomics.org.cn/soapsplice.html	2011	RNA-Seq	BWT-FM	Y	N	N	Non-DP Heuristic	N	150
BRAT-BW [51]	http://compbio.cs.ucr.edu/brat/	2012	BS-Seq	BWT-FM	N	N	N	HD	N	62
BLASR [85]	https://github.com/mchaisso/blasr/	2012	DNA	Suffix array	Y	N	Y	NW	N	8000
Batmis [86]	https://code.google.com/archive/p/batmis/	2012	DNA	BWT-ST	Y	N	N	HD	N	100
Bowtie2 [87]	http://bowtie-bio.sourceforge.net/bowtie2	2012	DNA	BWT-FM	Y	N	Y	SW & NW	N	400
GEM [88]	https://github.com/smarco/gem3-mapper	2012	DNA	BWT-FM	N	N	Y	SW & NW	N	150
RazerS3 [89]	https://github.com/seqan/seqan/tree/master/apps/razers3	2012	DNA	Hashing	Y	Y	Y	Banded Myers Bit Vector	N	800
SeqAlto [90]	https://web.stanford.edu/group/wonglab/seqalto/	2012	DNA	Hashing	Y	N	N	NW	N	200
SplazerS [91]	https://github.com/seqan/seqan/blob/master/apps/splazers/README	2012	DNA	Hashing	Y	N	Y	Banded Myers Bit Vector	N	150
WHAM [92]	http://pages.cs.wisc.edu/~jignesh/wham/	2012	DNA	Hashing	Y	N	N	NW	N	74

Table 1 Summary of algorithms and features of the examined read alignment methods. We surveyed 107 alignment tools published from 1988 to 2020 (indicated in column “Year of publication”). The table is sorted by year of publication, and then grouped according to the area(s) of application (indicated in column “Application”) within each year. In column “Indexing,” we document the algorithms used to index the genome (the first step in read alignment). In column “Global Positioning,” we document the algorithms used to determine a global position of the read in the reference genome (the second step). In column “Pairwise alignment,” we document the algorithm used to determine the similarity between the read and the corresponding region of the reference genome (the last step). SW, NW, HD, and DP stand for Smith-Waterman algorithm, Needleman-Wunsch algorithm, Hamming distance, and dynamic programming, respectively. In column “Wrapper,” we document the read alignment algorithms that are built on top of other read alignment tools. Finally, we report the maximum read length tested in the corresponding paper in column “Max. Read Length Tested in the Paper (bp).” The tested read length in each paper is not necessarily the maximum read length that each tool can handle (*Continued*)

Aligner	URL	Year of publication	Application	Indexing	Global Positioning			Pairwise alignment	Wrapper	Max. read length tested in the paper (bp)
					Fix length seed	Spaced seed	Seed chaining			
YAHA [93]	https://github.com/GregoryFaust/yaha	2012	DNA	Hashing	Y	N	Y	SW	N	10000
OSA [94]	http://www.arrayserver.com/wiki/index.php?title=OSA	2012	RNA-Seq	Hashing	Y	N	N	NA	N	100
Passion [95]	https://trac.nbic.nl/passion/	2012	RNA-Seq	Hashing	Y	N	Y	SW	Y	75
BS-Seeker2 [96]	https://github.com/BSSeeker/BSseeker2	2013	BS-Seq	BWT-FM	Y	N	Y	SW & NW	Y	250
Subread [97]	http://subread.sourceforge.net/	2013	DNA/RNA-Seq	Hashing	Y	Y	Y	SW	N	202
BWA-MEM [98]	https://github.com/lh3/bwa	2013	DNA	BWT-FM	N	N	Y	SW & NW	N	650
Masai [99]	http://www.seqan.de/projects/masai	2013	DNA	Suffix tree	N	N	Y	Banded Myers Bit Vector	N	150
NextGenMap [100]	http://cibiv.github.io/NextGenMap/	2013	DNA	Hashing	Y	N	N	SW & NW	N	250
SFmapper [101]	http://www.umsl.edu/~wongch/software.html	2013	DNA	Hashing	Y	N	N	HD	N	100
mrFAST [102]	https://github.com/BilkentCompGen/mrfast	2013	DNA	Hashing	Y	N	N	Semi-Global	N	180
CRAC [103]	http://crac.gforge.inria.fr/	2013	RNA-Seq	BWT-FM	Y	N	N	Non-DP Heuristic	N	200
STAR [104]	https://github.com/alexdobin/STAR	2013	RNA-Seq	Suffix array	N	N	Y	SW	N	5000

Table 1 Summary of algorithms and features of the examined read alignment methods. We surveyed 107 alignment tools published from 1988 to 2020 (indicated in column “Year of publication”). The table is sorted by year of publication, and then grouped according to the area(s) of application (indicated in column “Application”) within each year. In column “Indexing,” we document the algorithms used to index the genome (the first step in read alignment). In column “Global Positioning,” we document the algorithms used to determine a global position of the read in the reference genome (the second step). In column “Pairwise alignment,” we document the algorithm used to determine the similarity between the read and the corresponding region of the reference genome (the last step). SW, NW, HD, and DP stand for Smith-Waterman algorithm, Needleman-Wunsch algorithm, Hamming distance, and dynamic programming, respectively. In column “Wrapper,” we document the read alignment algorithms that are built on top of other read alignment tools. Finally, we report the maximum read length tested in the corresponding paper in column “Max. Read Length Tested in the Paper (bp).” The tested read length in each paper is not necessarily the maximum read length that each tool can handle (*Continued*)

Aligner	URL	Year of publication	Application	Indexing	Global Positioning			Pairwise alignment	Wrapper	Max. read length tested in the paper (bp)
					Fix length seed	Spaced seed	Seed chaining			
TopHat2 [105]	https://ccb.jhu.edu/software/tophat/index.shtml	2013	RNA-Seq	BWT-FM	Y	N	Y	SW & NW	Y	101
Subjunc [106]	http://subread.sourceforge.net/	2013	RNA-seq	Hashing	Y	Y	Y	NW	N	202
BWA-PSSM [107]	http://bwa-pssm.binf.ku.dk/	2014	DNA	BWT-FM	Y	N	N	SW	Y	100
CUSHAW3 [108]	http://cushaw3.sourceforge.net/homepage.htm#latest	2014	DNA	BWT-FM	Y	N	Y	SW & Semi-Global	N	100
Hobbes2 [109]	https://hobbes.ics.uci.edu/download.shtml	2014	DNA	Hashing	Y	N	Y	Banded Myers Bit Vector	N	100
MOSAik [110]	https://github.com/wanpinglee/MOSAik	2014	DNA	Hashing	Y	N	N	SW	N	100
hpg-Aligner [111]	https://github.com/opench/hpg-aligner	2014	DNA	Suffix array	N	N	Y	SW	N	5000
mrFAST-Ultra [112]	https://github.com/sfu-compbio/mrFAST	2014	DNA	Hashing	Y	N	N	HD	N	100
JAGuar [113]	http://www.bcgsc.ca/platform/bioinfo/software/jaguar	2014	RNA-Seq	BWT-FM	Y	N	N	SW	Y	100
ContextMap 2 [114]	http://www.bio.fifi.lmu.de/ContextMap	2015	RNA-Seq	BWT-FM	Y	N	Y	SW & NW	Y	76
HISAT [115]	http://www.ccb.jhu.edu/software/hisat/index.shtml	2015	RNA-Seq	BWT-FM	Y	N	N	Non-DP Heuristic	N	100

Table 1 Summary of algorithms and features of the examined read alignment methods. We surveyed 107 alignment tools published from 1988 to 2020 (indicated in column “Year of publication”). The table is sorted by year of publication, and then grouped according to the area(s) of application (indicated in column “Application”) within each year. In column “Indexing,” we document the algorithms used to index the genome (the first step in read alignment). In column “Global Positioning,” we document the algorithms used to determine a global position of the read in the reference genome (the second step). In column “Pairwise alignment,” we document the algorithm used to determine the similarity between the read and the corresponding region of the reference genome (the last step). SW, NW, HD, and DP stand for Smith-Waterman algorithm, Needleman-Wunsch algorithm, Hamming distance, and dynamic programming, respectively. In column “Wrapper,” we document the read alignment algorithms that are built on top of other read alignment tools. Finally, we report the maximum read length tested in the corresponding paper in column “Max. Read Length Tested in the Paper (bp).” The tested read length in each paper is not necessarily the maximum read length that each tool can handle (*Continued*)

Aligner	URL	Year of publication	Application	Indexing	Global Positioning			Pairwise alignment	Wrapper	Max. read length tested in the paper (bp)
					Fix length seed	Spaced seed	Seed chaining			
ERNE 2 [116]	http://erne.sourceforge.net/	2016	DNA/BS-Seq	BWT-FM + hashing	Y	N	N	HD	N	100
GraphMap [117]	https://github.com/isovic/graphmap	2016	DNA	Hashing	Y	Y	Y	Semi-global	N	9000
NanoBLASTer [118]	https://github.com/ruhulsbu/NanoBLASTer	2016	DNA	Hashing	Y	N	Y	NW	N	7040
minimap [119]	https://github.com/lh3/minimap	2016	DNA	Hashing	Y	N	N	N/A	N	13000
rHAT [120]	https://github.com/dfguan/rHAT	2016	DNA	Hashing	Y	N	Y	SW	N	8000
KART [121]	https://github.com/hsinnan75/KART	2017	DNA	BWT-FM	N	N	Y	NW	N	7118
LAMSA [122]	https://github.com/hitbc/LAMSA	2017	DNA	BWT-FM + hashing	Y	N	Y	Sparse DP	Y	100000
DART [123]	https://github.com/hsinnan75/DART	2017	RNA-Seq	BWT-FM	N	N	Y	NW	N	251
minimap2 [124]	https://github.com/lh3/minimap2	2018	DNA/RNA-Seq	Hashing	Y	N	Y	NW	N	11628
DREAM-Yara [125]	https://gitlab.com/pirovcdream_yara/	2018	DNA	BWT-FM	Y	N	N	Banded Myers Bit Vector	Y	150
MUMmer4 [126]	https://github.com/mummer4/mummer	2018	DNA	Suffix array	Y	N	Y	SW	Y	7821

Table 1 Summary of algorithms and features of the examined read alignment methods. We surveyed 107 alignment tools published from 1988 to 2020 (indicated in column “Year of publication”). The table is sorted by year of publication, and then grouped according to the area(s) of application (indicated in column “Application”) within each year. In column “Indexing,” we document the algorithms used to index the genome (the first step in read alignment). In column “Global Positioning,” we document the algorithms used to determine a global position of the read in the reference genome (the second step). In column “Pairwise alignment,” we document the algorithm used to determine the similarity between the read and the corresponding region of the reference genome (the last step). SW, NW, HD, and DP stand for Smith-Waterman algorithm, Needleman-Wunsch algorithm, Hamming distance, and dynamic programming, respectively. In column “Wrapper,” we document the read alignment algorithms that are built on top of other read alignment tools. Finally, we report the maximum read length tested in the corresponding paper in column “Max. Read Length Tested in the Paper (bp).” The tested read length in each paper is not necessarily the maximum read length that each tool can handle (*Continued*)

Aligner	URL	Year of publication	Application	Indexing	Global Positioning			Pairwise alignment	Wrapper	Max. read length tested in the paper (bp)
					Fix length seed	Spaced seed	Seed chaining			
NGMLR [127]	https://github.com/philres/ngmlr	2018	DNA	Hashing	Y	N	Y	SW	N	50000
lordFAST [128]	https://github.com/vpc-ccg/lordfast	2018	DNA	BWT-FM + hashing	N	N	Y	SW & NW	N	35489
BatMeth2 [129]	https://github.com/GuoliangLi-HZAU/BatMeth2/	2019	BS-Seq	BWT-FM	Y	N	Y	SW & NW	N	125
GraphMap2 [130]	https://github.com/lbcb-sci/graphmap2	2019	DNA/RNA-Seq	Hashing	Y	Y	Y	Semi-global	N	9000
Magic-BLAST [131]	https://github.com/ncbi/magicblast	2019	DNA/RNA-Seq	Hashing	Y	N	N	Non-DP Heuristic	N	90000
BWA-MEM2 [132]	https://github.com/bwa-mem2/bwa-mem2	2019	DNA	BWT-FM	N	N	Y	SW	N	650
HISAT2 [133]	https://ccb.jhu.edu/software/hisat2/index.shtml	2019	DNA	BWT-FM	Y	N	N	Non-DP Heuristic	N	100
deSALT [134]	https://github.com/hitbc/deSALT	2019	RNA-seq	Hashing	Y	N	Y	SW	N	8000
conLSH [135]	https://www.dropbox.com/s/3jcu4i240kyu2tc/source%20code%20conLSH_bio.tar.gz?dl=0	2020	DNA	Hashing	Y	N	Y	Sparse DP	N	8000

Table 2 Advantages and limitations of read alignment algorithms. We compare the ease of implementing each algorithm (“Easy to implement”). We define the “ease of implementation” as the ability to quickly implement such an algorithm and its indexing technique, flexibly apply some changes to it, and easily understand its working principle. We also record whether the algorithm allows for an exact and/or inexact match (“Search for exact/inexact match”). The use of spaced seeds enables searching for inexact match using a hash table. We also compare the size of the genome index (indicated in column “Index size”), the speed of seed query (indicated in column “Seed query speed”), and the possibility to vary the length of the seed (“Seed length”)

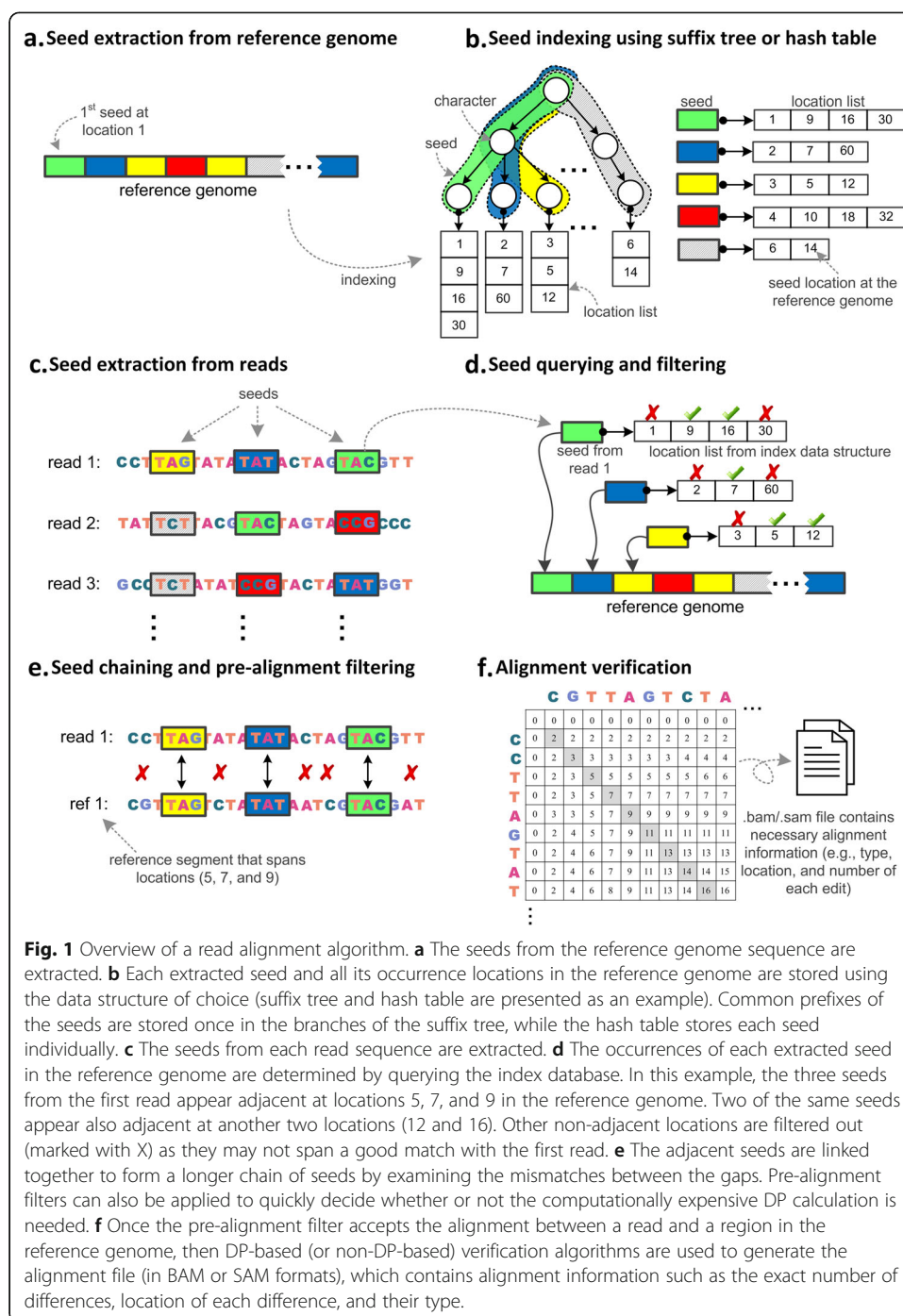
	Hashing	Suffix tree and BWT-FM
Easy to implement	Yes	No
Search for exact/inexact match	Exact	Exact and inexact
Index size	Large	Compressed (small)
Indexing time	Small	Large
Seed query speed	O(1), fast	Slow
Seed length	Fixed length per index	Can be fixed or variable

for modern read alignment methods [17]. A recent improvement in circular consensus sequencing (CCS) allows a substantial reduction in sequencing error rates; for example, the error rate has dropped from 15% down to 0.0001% by sequencing the same molecule at least 30 times and further correcting errors by calculating consensus [136].

We have studied the underlying algorithms of 107 read alignment tools that were designed for the short- and long-read sequencing technologies and were published from 1988 to 2020 (Table 1). We defined read alignment as a three-step procedure (Supplementary Note 2). First, indexing with the aim of quickly locating genomic subsequences in the reference genome is performed. This step includes building a large index database from a reference genome and/or the set of reads (Fig. 1a, b). Second, global positioning is performed to determine the potential positions of each read in the reference genome. In this step, alignment algorithms use the prepared index to determine one or more possible regions of the reference genome that are likely to be similar to each read sequence (Fig. 1c, d). Lastly, pairwise alignment is performed between the read and each of the corresponding regions of the reference genome to determine the exact number, location, and type of differences between the read and corresponding region (Fig. 1e, f).

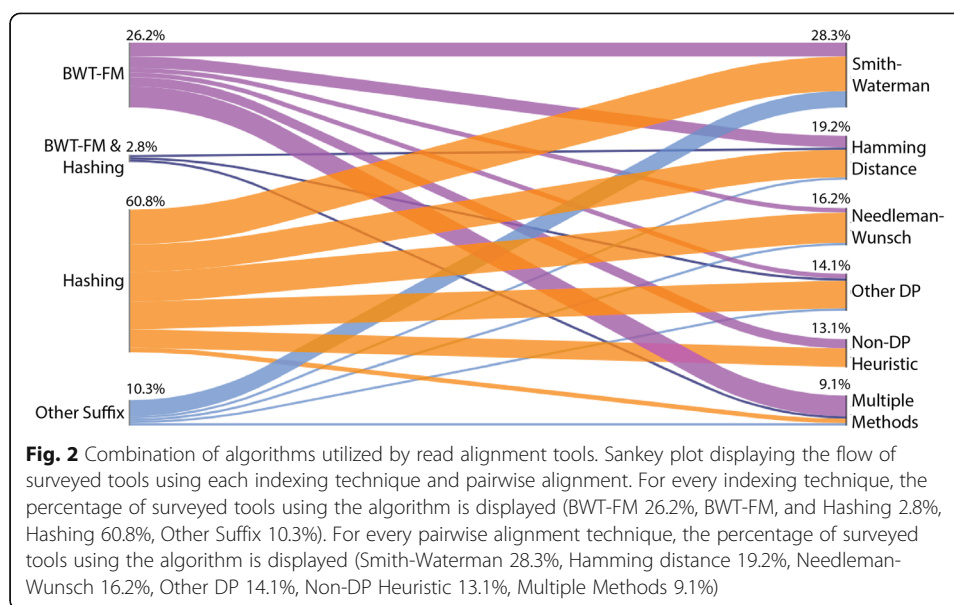
Hashing is the most popular technique for indexing the reference genome

The key goal of the indexing step is to facilitate quick and efficient querying over the whole reference genome sequence, producing a minimal memory footprint by storing the redundant subsequences of the reference genome only once [17, 20, 137]. Rapid advances in sequencing technologies have shaped the development of read alignment algorithms, and major changes in technology have rendered many tools obsolete. For example, some early methods [43, 44, 47, 48, 80] built the index database from the reads. Today’s longer read lengths and increased throughput of sequencing technologies make such an approach infeasible for analyzing modern sequencing data. Modern alignment algorithms typically build the index database from the reference genome and then use the subsequences of the reads (known as seeds or qgrams) to query the index database (Fig. 1a). In general, indexing the reference genome compared to the read set



is a more practical and resource-frugal solution. Additionally, it allows reusing the constructed reference genome index across multiple samples.

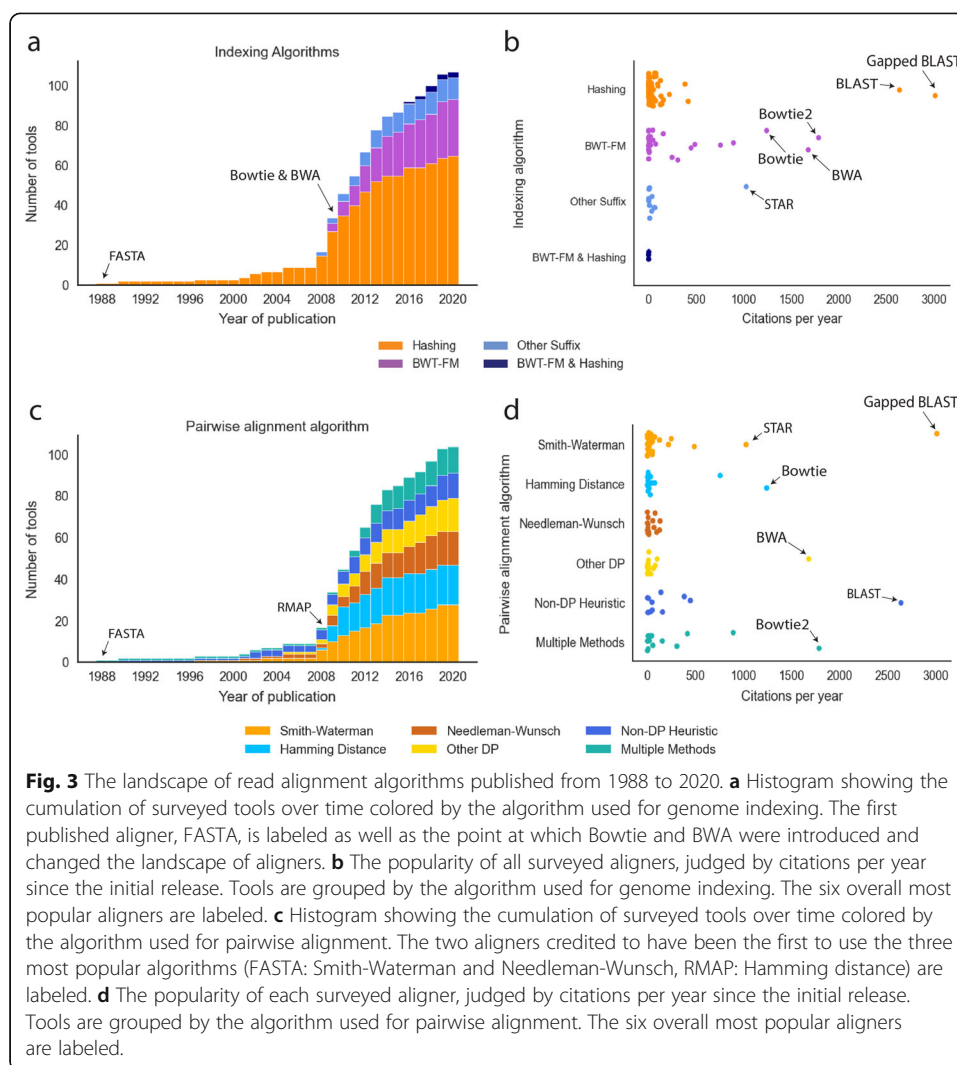
We observe that the most popular indexing technique used by read alignment tools is hashing, which is used exclusively by 60.8% of our surveyed read aligner tools from various domains of biological research (Fig. 2). Hashing is also the most popular individual indexing method for aligners that can handle DNA-Seq data, accounting for 68.3% of the surveyed read aligner tools. Hash table indexing was first used in 1988 by FASTA [30, 138] and has since dominated the landscape of read alignment tools.



Hashing was also the only dominant technique to be used until the BWT-FM index was introduced by Bowtie [55] (Fig. 3a). Its popularity can be explained by the simplicity and ease of implementation when compared to other indexing techniques. Other advantages and limitations of hashing are outlined in Table 2. The hash table is a data structure that stores the content of some short regions of the genome (e.g., seeds) and their corresponding locations in the reference genome (Fig. 1b). Such regions are also known as *k*-mers or qgrams [139]. After the genomic seeds are produced, the alignment algorithm extracts the seeds from each read and uses them as a key to query the hash table index. The hash table returns a location list storing all occurrence locations of the read seed in the reference genome.

Alignment tools utilizing suffix-tree-based indexing are generally faster and more widely used

The second most popular approach to indexing is the suffix-tree-based techniques, used exclusively by 36.5% of the surveyed read aligner tools (Fig. 2) (Table 1). ERNE 2 [116], LAMSA [122], and lordFAST [128] are categorized separately since they combine hashing with a suffix-tree-based technique. A suffix tree is a tree-like data structure where separate branches represent different suffixes of the genome; the shared prefix between two suffixes of the genome is stored only once. Every leaf node of the suffix tree stores all occurrence locations of this unique suffix in the reference genome (Fig. 1b). Unlike a hash table, a suffix tree allows searching for both exact and inexact match seeds [140, 141] by walking through the tree branches from the root to a leaf node, detouring as needed, following the query sequence (Table 2). While some algorithms [142, 143] specifically rely on creating suffix trees, the most frequently chosen tools from this category use the Burrows-Wheeler Transform (BWT) and the FM index (hence called BWT-FM-based tools) to mimic the suffix-tree traversal process while generating a smaller memory footprint [99]. The performance of the read aligners in this category degrades as either the sequencing error rate increases or the genetic

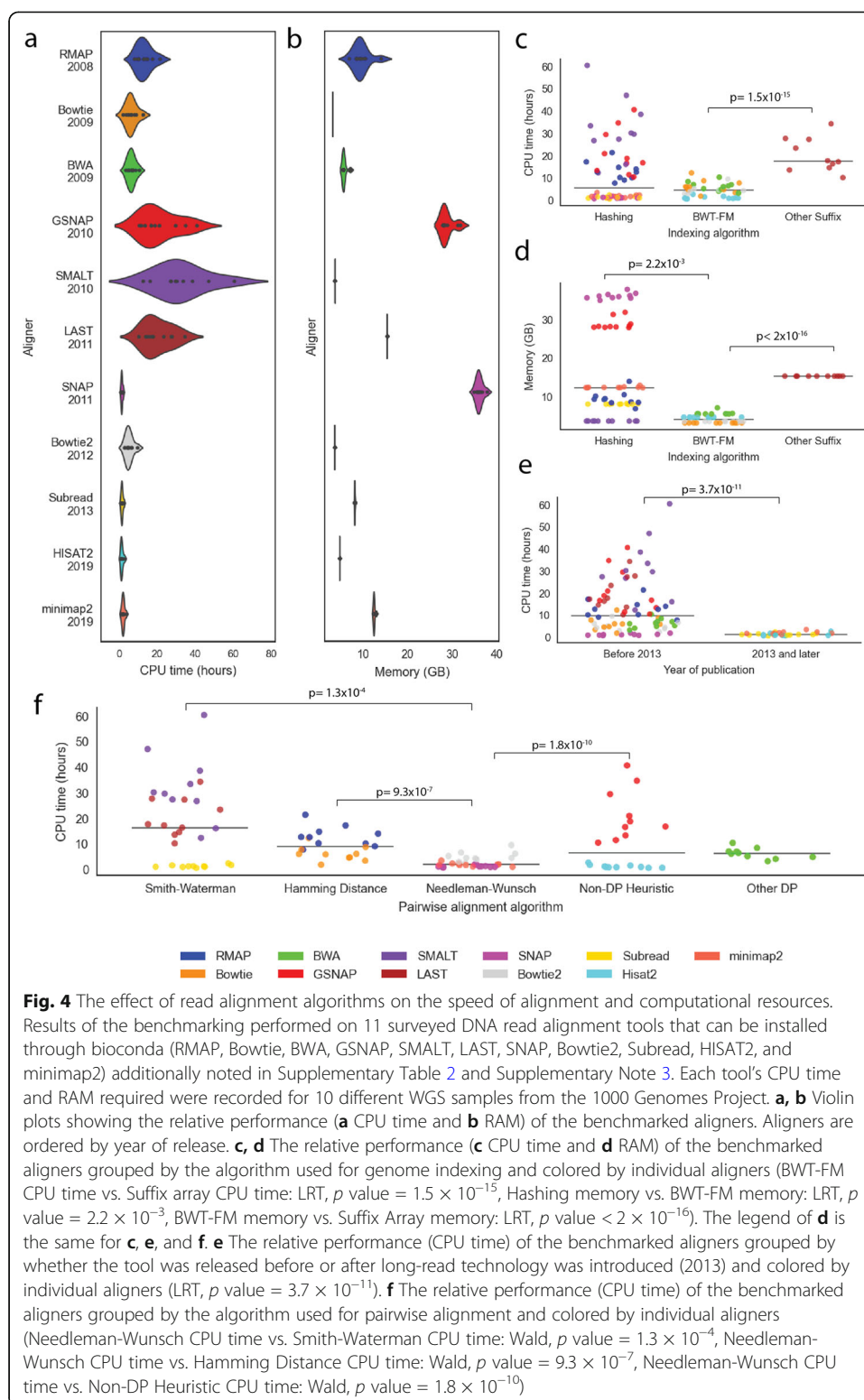


differences between the subject and the reference genome are more likely to occur [144, 145].

The effect of read alignment algorithms on speed of alignment and computational resources

To measure the effect of read alignment algorithms on speed of alignment and computational resources, we have compared the running time and memory (RAM) required of eleven read alignment tools when applied to ten real WGS datasets (Fig. 4a, b). We used tools available via the Bioconda package manager [146]. We ran these tools using their default parameters. We randomly selected ten WGS samples from the 1000 Genomes Project. We excluded tools specifically designed for RNA-Seq or BS-Seq. Details on how the tools were installed and ran are provided in Supplementary Note 3.

We found no significant difference in the runtime for BWT-FM tools and hashing-based tools when adjusting for year of publication, chain of seeds, and type of pairwise alignment (Likelihood ratio test (LRT) p value = 0.5) (Fig. 4c, Supplementary Table 3, 4). SMALT [69] is an outlier to this observation, and it shows the highest execution



time (Fig. 4c) as it uses standard non-accelerated pairwise alignment algorithm (Smith-Waterman algorithm). BWT-FM-based tools did require, on average, 3.8× less computational resources when compared to hashing-based tools, adjusting for year of publication, chain of seeds, and type of pairwise alignment algorithm (LRT p value = $2.2 \times$

10^{-3}) (Fig. 4d, Supplementary Table 5, 6). SNAP [81] shows the highest memory footprint (Fig. 4d) as its index exceptionally uses much longer (> 20 bp) seeds compared to most other tools. The default suffix array implemented by LAST [78] requires, on average, 4.38 \times more running time and 3.58 \times more computational resources when compared to BWT-FM-based tools (LRT test p value = 1.5×10^{-15} and $< 2 \times 10^{-16}$ for runtime and memory, respectively) (Fig. 4c, d, Supplementary Table 3, 4, 5, 6).

Despite the difference in performance driven by algorithms, we observed an overall improvement (9.2 \times reduction) in computation time of read alignment over time (s.e. = 0.09; LRT test p value = 3.7×10^{-11}) (Fig. 4e, Supplementary Table 3, 4) but no significant improvement (only 1.57 \times reduction) of their memory requirements (s.e. = 0.24; LRT p value = 0.41) (Supplementary Figure 1, Supplementary Table 5, 6). Usually, the index is created separately for each genome. Some methods incorporate multiple genomes into a single index graph [58, 76, 115], while other methods use a de Bruijn graph for hashing [58, 116]. Although computing the genome index can take up to four hours, it usually needs to be computed only once and is often already precomputed for various species (Supplementary Figure 2). Updating the genome index can create a bottleneck in the analysis, especially for extremely large genome databases. Bloom1-filter-based algorithms promise to provide an alternative way of indexing while preserving faster search times [125, 147].

We surveyed 28 BWT-FM-based tools to compare the popularity of the read alignment algorithms using the number of times the introductory publication has been cited in other papers. Of those, three aligners have accumulated more than 1000 citations per year since release, and 18% of the BWT-FM-based tools have been cited by at least 500 papers per year. In contrast, only two of the 63 hashing-based tools have more than 1000 citations per year, but those two aligners (BLAST [31] and Gapped BLAST [32]) are, by far, the most popular with 2726 and 3143 citations per year, respectively (Fig. 3b). Notably, tools cited more than 500 times per year were among the most effective both in terms of runtime and required computational resources (Supplementary Figure 3).

Majority of the tools utilize fix length seeding to find the global position of the read in the reference genome

The goal of the second step of read alignment is to find the global position of the read in the reference genome. This step is known as global positioning and uses the generated genome index to retrieve the locations (in the genome) of various seeds extracted from the sequencing reads (Fig. 1c). The read alignment algorithm uses the determined seed locations to reduce the search space from the entire reference genome to only the neighborhood region of each seed location (Supplementary Note 4).

The number of possible locations of a seed in the reference genome is affected by two key factors: the seed length and the seed type. The estimated number of such locations is extremely large for short seeds and can reach tens of thousands for the human genome. The high frequency of short seeds is due to the repetitive nature of most genomes, which creates a high probability of finding the same short seed frequently in a long string of only four DNA letters. A large number of possible locations for short seeds imposes a significant computational burden on read alignment algorithms [148, 149]. Only a few read alignment algorithms examine all the seed locations reported in

the location list [102]. Most of the read alignment algorithms apply heuristic devices to avoid examining all the locations of the seed in the reference genome (Fig. 1d, Supplementary Note 4).

Longer seed lengths can help reduce both the number of possible locations of a seed in the reference genome and the number of chosen seeds from each read. These benefits come at the cost of a possible reduction in alignment sensitivity, especially in cases where the mismatches between the read and the genome are located within the seed sequence. To enable increasing the seed length without reducing the alignment sensitivity, seeds can be generated as spaced seeds (Supplementary Note 4) [34–37, 139].

The majority of the surveyed alignment algorithms use seeds of fixed length at run time. Some algorithms generate seeds of various lengths [83, 108, 150] in order to reduce the hit frequencies while tolerating mismatches. Varying the seed length or using different types of seed during the same run is often referred to as hybrid seeding [108] and was used by 20 of the 107 surveyed alignment algorithms. The first tool to use variable-length seeds was GMAP [41]. Hybrid seeding with a hash-based index would require the creation of multiple hash tables of the same genome and would require extra computational resources. As a result, the vast majority of tools that use variable-length seeds use a suffix tree indexing technique (BWT-FM or other).

Majority of the tools utilize Hamming distance and Smith-Waterman to determine similarity between the read and its global positions in the reference genome

The goal of the last step of a read alignment algorithm is to determine regions of similarity between each read and the global positions of each read in the reference genome, which was determined in the previous step. These regions are potentially highly similar to the reads, but read alignment algorithms still need to determine the minimum number of differences between two genomic sequences, the nature of each difference, and the location of each difference in one of the two given sequences. Such information about the optimal location and the type of each edit is normally calculated using a verification algorithm (Fig. 1f) that first verifies the similarity between the query read and the corresponding region in the reference genome. Verification algorithms can be categorized into algorithms based on dynamic programming (DP) [151] and non-DP-based algorithms. The DP-based verification algorithms can be implemented as local alignment (e.g., Smith-Waterman [152]) or global alignment (e.g., Needleman-Wunsch [153]). DP-based verification algorithms can also be implemented as semi-global alignment, where the entirety of one sequence is aligned to one of the ends of the other sequence [108, 109, 117].

The non-DP verification algorithms include Hamming distance [154] and the Rabin-Karp algorithm [155]. When one is interested in finding genetic substitutions, insertions, and deletions, DP-based algorithms are favored over non-DP algorithms. In general, the local alignment algorithm is preferred over global alignment when only a fraction of the read is expected to match with some regions of the reference genome due to, for example, large structural variations [63]. The Smith-Waterman [152] and Needleman-Wunsch [153] alignment algorithms were both first used by FASTA [30, 138] in 1988, which we categorize as “Multiple Methods” (Fig. 3c). Smith-Waterman remains the most popular algorithm and is used by 28.3% of our surveyed tools (Fig. 2).

Needleman-Wunsch, in contrast, has only been used by 16.2% of our surveyed tools (Fig. 2). However, if we include the tools which allow for multiple methods, Smith-Waterman represents 38.3% and Needleman-Wunsch represents 26.2% of alignment algorithms used. This trend is due to the fact that 12 of the 13 tools classified as “Multiple Methods” use or allow both Smith-Waterman and Needleman-Wunsch. Non-DP verification using Hamming distance [154] has been the second most popular single technique since used for the first time by RMAP [44] in 2008 (Fig. 3c). There is no significant correlation between the indexing technique used and the pairwise alignment algorithm chosen. Most major indexing techniques are used in conjunction with most pairwise alignments. However, BWT-FM-based aligners do comprise the largest percentage of tools that allow multiple pairwise alignment methods (Fig. 2).

As the number of differences between two sequences is not necessarily equivalent to the sum of the number of differences between the subsequences of these sequences, it is necessary to perform verification for the entire read sequence and the corresponding region in the reference sequence [156]. Existing DP-based algorithms can be inefficient as they require quadratic time and space complexity. Despite more than three decades of attempts to improve their algorithmic implementation, the fastest known edit distance computation algorithm is still nearly quadratic [157]. Some of the read alignment algorithms use DP only for seed chaining, which provides suboptimal alignment calculation [38, 40]. This approach is called sparse DP and is used in C4 [40], conLSH [135], and LAMSA [122]. An alternative way to accelerate the alignment algorithms is by reducing the maximum number of differences that can be detected by the verification algorithm, which reduces the search space of the DP algorithm and shortens the computation time [106, 158–164, 167, 168] (Supplementary Note 5).

We found that tools which use the Needleman-Wunsch [153] algorithm are faster than tools which use other algorithms (faster by 3.57×, 4.14×, and 6.7× and Wald test p values 9.3×10^{-7} , 1.8×10^{-10} , and 1.3×10^{-4} for Hamming distance, non-DP heuristics, and SW algorithms, respectively) (Fig. 4f, Supplementary Table 3), adjusting for publication year, seed chaining, and indexing method. Despite the overall longer runtime of Hamming distance-based methods, the latest hashing-based tools (e.g., HISAT2 [133]) provide a comparable running time with the fastest Needleman-Wunsch-based tools. We also found significant differences in the amount of computational resources required by read alignment tools using different pairwise alignment algorithms after adjusting for publication year, type of seed, and indexing method (LRT; p value = 0.04) (Supplementary Figure 4, Supplementary Table 6). Notably, the algorithms with the smallest computational footprints use various types of pairwise alignment algorithms.

Influence of long-read technologies on the development of novel read alignment algorithm

Alignment of the long reads produced by modern long-read technologies [16, 136, 169] provides a unique possibility to discover previously undetectable structural variants [16, 170, 171]. Long reads also improve the construction of an accurate hybrid de novo assembly [16, 172], in cases where long and short reads are suffix-prefix overlapped, or in cases where reads are aligned using pairwise alignment algorithms, to construct an entire assembly graph. This is helpful when a reference genome is either unavailable [173, 174] or is complex and contains large repetitive genomic regions [175].

Existing long-read alignment algorithms still follow the three-step-based approach of short-read alignment. Some long-read alignment tools even divide every long read into short segments (e.g., 250 bp), align each short segment individually, and determine the mapping locations of each long read based on the adjacent mapping locations of these short segments [123, 127]. Some long-read alignment tools use hash-based indexing [110, 120, 176], while others use BWT-FM indexing [54, 98, 177]. The major challenge with the long-read alignment algorithms is dealing with large sequencing errors and a significantly large number of short seeds extracted from each long or ultra-long read [178]. Thus, the most recently developed long-read alignment algorithms require heuristically extracting fewer seeds per read length when compared to those extracted from short reads. Instead of creating a hash table for the full set of seeds, recent long-read alignment algorithms find the minimum representative set of seeds from a group of adjacent seeds within a genomic region. These representative seeds are called minimizers [179, 180] and can also be used to compress genomic data [181] or taxonomically profile metagenomic samples [182]. Long-read alignment algorithms [119, 124, 183] that use hashed minimizers as an indexing technique provide a faster alignment process compared to other algorithms that use conventional seeding or BWT-FM. They also provide a significantly faster ($> 10\times$) indexing time (Supplementary Table 1). However, their accuracy degrades with the use of short reads as they process a fewer number of seeds per short read [124].

Box 1. Advantages and limitations of short- versus long-read alignment algorithms

-
- **Error rate.** The error rate of modern short-read sequencing technologies is smaller than that of modern long-read technologies.
 - **Genome coverage.** Throughput (i.e., the number of reads) of modern short-read sequencing technologies is higher than that of modern long-read technologies.
 - **Global position.** Determine a global position of the read by identifying the starting position or positions of the reads in the reference genome. This step is ambiguous with short reads, as the repetitive structure of the human genome causes such reads to align to multiple locations of the genome. In contrast, long reads are usually longer than the majority of repeat regions and are aligned to a single location in the genome.
 - **Local pairwise alignment.** After determining the global position of each read, the algorithms map all bases of the read to the reference segments, located at these global positions, in order to account for indels. Due to the smaller error rate of short-read technologies, it is usually easier to perform local alignment on short reads than on long ones.
 - **Genomic variants.** Single-nucleotide polymorphisms (SNPs) are easy to detect using short reads when compared to long reads due to the lower error rate and higher coverage of short-read sequencing technologies. Structural variants (SVs) are easy to detect with long reads, which span the entire SV region. Current long-read-based tools [184] are able to detect deletions and insertions with high precision. The sparse coverage of long reads may lower the sensitivity of detection.
-

Read alignment across various domains of biological research

We discuss the challenges and the features of these algorithms that are specific to the various domains of modern biological research. Often the domain-specific alignment problem can be solved by creating a novel tool from scratch or wrapping the existing algorithms into a domain-specific alignment tool (Supplementary Figure 5 and 6). Additionally, longer reads make the read alignment problem similar across areas of biological research. For example, tools recently designed to align long reads can handle both DNA and RNA-Seq reads [131].

RNA-Seq alignment

RNA sequencing is a technique used to investigate transcriptomics by generating millions of reads from a collection of human alternative spliced isoform transcripts, referred to as a transcriptome [185]. RNA-Seq has been widely used for gene expression analysis as well as splicing analysis [14, 185, 186]. However, the alignment of RNA sequencing reads needs to overcome additional challenges when mapping the reads originating from human transcriptome onto the reference genomes. Those challenges arise due to differences between the human transcriptome and the human genome; these differences define a subset of alignment problems known as *spliced alignment*. Spliced alignment requires that the one takes into account reads spanning over large gaps caused by spliced out introns [185]. Reads spanning only a few bases across the junctions can be easily aligned to an adjacent intron or aligned in a wrong location, making the accurate alignment more difficult [14, 185].

Several spliced alignment tools have been developed to address this issue and align RNA-Seq reads in a splicing-aware manner (Table 1 and Fig. 1c). Hashing is the most popular technique among RNA-Seq aligners (Supplementary Figure 7). This is even more evident if we remove the RNA-Seq aligners that are wrappers of existing DNA-Seq alignment methods (Supplementary Figure 5). Over 60% of the RNA-Seq aligners which are wrappers of existing DNA-Seq alignment methods use Bowtie or Bowtie2 (Supplementary Figure 5). When considering only stand-alone RNA-Seq aligners, the number of aligners using hashing more than doubles the number of aligners using an FM index (Supplementary Figure 8).

The most popular tool based on the number of citations was TopHat2 [105] (Table 1). TopHat2 uses Bowtie2 to align reads that may span more than one exon by splitting the reads into smaller segments and stitching the segments together to form a whole read alignment. The stitched read alignment spans a splicing junction on the human genome. This method allows identification of the splicing junction without transcriptome alignment. A more recent tool, HISAT2, uses a hierarchical indexing algorithm that leverages the Burrows-Wheeler Transform and Ferragina-Manzini index to align parts of reads and extend the alignment [115]. Another popular method, RNA-Seq aligner—called STAR—utilizes suffix arrays to identify a maximal mappable prefix, which is used as seeds or anchors, and stitch together the seeds that aligned within the same genomic window [104]. Although those tools can detect splicing junctions within their algorithm, it is possible to supply known gene annotation to increase the accuracy of a spliced alignment. The alignment accuracy, measured by correct read placement, can be increased 5–10% by supplying known gene annotations [14, 185]. HISAT2 and STAR are able to align the reads accurately with or without a splicing junction [14]. Furthermore, the discovery and quantification of novel splicing junctions can be significantly improved using two passes in STAR, which generates a list of possible junctions in the first pass and identifies aligning reads leveraging the junctions in the second pass [187]. While spliced alignment can provide an important splicing junction information, those tools require intensive computational resources [14].

To align RNA-Seq reads onto the transcriptome reference instead of the genome reference, regular DNA aligners are typically used. Mapping to the transcriptome is usually performed to estimate expression levels of genes and alternatively spliced isoforms by assigning reads to genes and alternatively spliced isoforms [104, 188]. Since many

alternatively spliced isoforms share exons, which are usually longer than the short reads, probabilistic models are used as it is impossible to uniquely assign reads to the isoform transcripts [189].

Alternatively, one can avoid computationally expensive alignment and perform pseudo-alignment, such as Kallisto [104] and Salmon [187]. Kallisto [190] uses transcriptome de Bruijn graph as an index where its nodes are seeds. Kallisto determines the locations of each input read by matching seeds extracted from reads with the seeds of the index without performing sequence alignment. Kallisto also exploits the structure of the de Bruijn graph to avoid examining more than a few seeds located at the same graph's path (between two junctions). This reduces the number of seed lookups in the index and hence reduces expensive memory accesses.

In contrast, Salmon [190, 191] can optionally perform either pseudo-alignment or read alignment. Salmon approximates the locations of each input read by building a hashing index in conjunction with a suffix array index. The seeds extracted from each read are looked up in the hash table and then the suffix array provides all suffixes of the reference genome containing the matched seed. Similar to Kallisto, Salmon tries to reduce the number seed lookups by finding the longest subsequence of the read that exactly matches the reference suffixes and excluding these regions from seed lookups.

In contrast to regular alignment algorithms, pseudo-alignment algorithms [190, 191] are unable to provide the precise alignment position of the read in the genome nor alignment profile (e.g., CIGAR string). Instead, pseudo-alignment algorithms assign the reads to a corresponding gene and/or alternatively spliced isoform. Usually, such information can be sufficient to accurately estimate gene expression levels of the sample [192]. A higher sequencing depth is demonstrated to improve the accuracy of Salmon and decreases the accuracy of Kallisto, as only Salmon exploits abundance information of each isoform to assist the seed matching [188].

Metagenomic alignment

Metagenomics is a technique used to investigate the genetic material in human or environmental microbial samples by generating millions of reads from the microbiome—a complex microbial community residing in the sample. Metagenomic data often contains an increased number of reads required to be aligned against more than hundreds of thousands of microbial genomes. For example, as of July 2018, the total number of nucleotides in NCBI's collection of bacterial genomes measures over 204 times the number of nucleotides present in the Genome Reference Consortium Human Build 38 (Supplementary Note 6). The increased number of reads and the size of reference databases pose unique challenges to existing alignment algorithms when applied to metagenomics studies.

In targeted gene sequencing studies, such as those that sequence portions of the 16S ribosomal RNA of prokaryotes or internally transcribed spacers (ITS) of eukaryotes, a number of task-specific aligners are utilized to identify the origin of candidate reads or to perform homology searches. For example, Infernal [193] utilizes profile hidden Markov models to perform alignment based on RNA secondary structure information. Multiple sequence aligners are also utilized in metagenomic analysis pipelines such as QIIME [194], Mothur [195], and Megan [195, 196]. For example, NAST [195–197] and PyNAST [198] use 7-mer seeds and a BLAST alignment that is then further refined

using a bidirectional search to handle indels. Similarly, MUSCLE [198, 199] uses an initial distance estimation based on k -mers and proceeds through a progressively constructed hierarchical guide tree while optimizing a log expectation for multiple sequence alignment [199].

For untargeted whole genome shotgun (WGS) metagenomic studies, the task of identifying the genomic or taxonomic origin of sequencing reads (referred to as “fragment recruitment” or “taxonomic read binning”) is even more difficult, individual reads can originate from multiple organisms due to shared homology or horizontal gene transfer and reads may originate from previously unsequenced organisms. This has sparked the development of a variety of tools [200] which aim to identify the presence and relative abundance of taxa or organisms present in a metagenomic sample via a reference-free and/or alignment-free fashion (referred to as “taxonomic profiling”). Similar in spirit to RNA-Seq alignment, these tools avoid computationally expensive base-level alignment and perform pseudo-alignment or multiple types of k -mer matching to detect the presence of organisms in a metagenomic sample [182, 201, 202], as well as use minimizers to reduce computational time [182].

Other approaches handle growing reference database sizes by aligning reads onto a reduced reference database, sometimes composed of marker microbial genes that are present in specific taxa. Reads mapping to those genes can be used to determine the presence of specific taxa in a sample [203]. Such tools typically use existing DNA alignment algorithms (e.g., MetaPhlAn [203] uses the Bowtie2 aligner).

Even with the development of these new metagenomic tools, existing read alignment tools (e.g., MOSAIK, SOAP, and BWA) are still used for fragment recruitment purposes [204]. However, the use of existing read alignment tools for metagenomics carries a significant computational burden and is identified as the main bottleneck in the analysis of such data. This major limitation suggests the need for the development of alignment tools capable of handling the increased number of reads and reference genomes seen in such studies [205].

Metagenomics studies are also capable of functional annotation of microbiome samples by aligning the reads to genes, gene families, protein families, or metabolic pathways. Protein alignment is beyond the scope of this manuscript, but many of the algorithmic approaches previously discussed are utilized for functional annotation [204, 206]. For example, RAPSearch2 [204, 206] uses a collision-free hash table based on amino acid 6-mers. The protein aligner DIAMOND [207] utilizes a spaced-seed-and-extend approach based on a reduced alphabet and unique indexing of both reference and query sequences. Indexing of both the reference *and* the query reads provides multiple orders of magnitude in speed improvements over older tools (such as BLASTX) at the cost of increased memory usage. Recently, MMseqs2 [205] utilizes consecutive, similar k -mer matches to further improve the speed of protein alignment.

Viral quasispecies alignment

RNA viruses such as human immunodeficiency virus (HIV) are highly mutable, with the mutation rates being as high as 10^{-4} per base per cell [208] allowing such viruses to form highly heterogeneous populations of closely related genomic variants commonly referred to as quasispecies [209]. Rare genomic variants, which are a few mutations away from the major strain, are often responsible for immune escape, drug resistance,

viral transmission, and increase of virulence and infectivity of the viruses [210, 211]. Massively parallel sequencing techniques allow for sampling of intra-host viral populations at high depth and provide the ability to profile the full spectra of viral quasispecies, including rare variants.

Similar to other domains, accurate read alignment is essential for assembling viral genomic variants including the rare ones. Aligning reads that originated from heterogeneous populations of closely related genomic variants to the reference viral genome give rise to unique challenges for existing read alignment algorithms. For example, read alignment methods should be extremely sensitive to small genomic variations while being robust to artificial variations introduced by sequencing technologies. At the same time, the genetic difference between viral quasispecies of different hosts is usually substantial (unless they originated from the same viral outbreak or transmission cluster), which makes the application of predefined libraries of reference sequences for viral read alignment problematic or even impossible.

Currently, viral haplotyping tools [212, 213] and variant calling tools [214, 215] frequently rely on existing independent alignment tools. While viral samples contain several distinct haplotypes, the read alignment tools such as BWA [145] and BowTie [216] can only map reads to a single reference sequence. Since certain haplotypes may be further or closer to the reference sequence, the reads emitted by such haplotypes may have different mapping quality. Some tools re-align reads to the consensus sequence instead of keeping the original alignment to the reference. Nevertheless, even alignment to the perfect reference or consensus sequence can reject perfectly valid short reads because of multiple mismatches. Rejection of such reads may cause loss of rare haplotypes and mutations. Systematic sequencing errors (such as homopolymer errors) frequently cause alignment errors. Although the sequencing error rate, both systematic and random, is comparatively low, such errors can be more frequent than the rarest variants. The alignment errors caused by sequencing errors may cause drastic sensitivity and reduction in specificity of haplotyping and variant calling methods (Supplementary Figure 9).

Aligning bisulfite-converted sequencing reads

Bisulfite-converted sequencing is a technique used to sequence methylated fragments [217, 218]. During sequencing, most of the cytosines (C) in the reads become thymines (T). Since every sequenced T could either be a genuine genomic T or a converted C, special techniques are used to map those reads [219]. Some tools substitute all C in reads with wildcard bases, which can be aligned to C or T in the reference genome [37, 52], while other tools substitute all C by T in all reads and reference and work with a three-letter alphabet aligning to a C-to-T-converted genome [77, 96]. Unlike RNA-Seq aligners, FM index was the most popular technique among BS-Seq aligners (Supplementary Figure 10). One-third of the surveyed BS-Seq aligners were wrappers of existing DNA-Seq alignment methods (Supplementary Figure 6), with all three of those wrapping Bowtie or Bowtie2 (Supplementary Figure 6). As a result, when considering only stand-alone BS-Seq aligners, the numbers of aligners using each indexing algorithm become extremely similar (Supplementary Figure 11).

Other domains

Other domains requiring specialized alignment include B and T cell receptor repertoire analysis. The repertoire data is generated using targeter repertoire sequencing protocols, known as BCR- or TCR-Seq. For example, tools designed to align reads to the V(D)J genes use combinations of fast alignment algorithms and more sensitive modified Smith-Waterman and Needleman-Wunsch algorithms [182, 220, 221].

Discrepancies between the reads and the reference may reveal the historical errors in the reference assembly

Genome sequencing datasets, especially those generated with long reads, provide a unique perspective to reveal errors in the reference assemblies (e.g., human reference genome) based on the discrepancies between the reads and the reference sequence. References and reads (e.g., resequencing data) are often produced using different technologies, and there are usually disagreements between references and reads that produce mapping errors. Similarly, some of these errors also come from the errors in the reads used for assembly, collapsed/merged duplications/repeats, and heterozygosity. For example, a study for structural variation discovery led to the identification of incorrectly inverted segments in the reference genome [222]. Similarly, Dennis et al. [223] characterized a duplicated gene that was not represented accurately because it collapsed in the reference genome. Therefore, using the most recent version of a reference genome is always the best practice, as demonstrated by an analysis of the latest version of the human genome [223, 224].

Structural errors in the reference genomes can be found and corrected by using various orthogonal technologies such as mate-pair and paired-end sequencing [225, 226], optical mapping [227], and linked-read sequencing [228]. Smaller-scale errors (i.e., substitutions and indels) can also be corrected using assembly polishing tools such as Pilon, which employs short-read sequencing data [229]. However, long reads are more powerful in detecting and correcting errors due to the fact that they can span the most common repeat elements. Long-read-based assembly polishers include Quiver [230] that uses Pacific Biosciences data, Nanopolish [231] that uses Nanopore sequencing, and Apollo [232] that can use read sets from any sequencing technology to polish large genomes. Additionally, more modern long-read genome assemblers, such as Canu [233], include built-in assembly polishing tools.

Discussion

Rapid advances in sequencing technologies shaped the landscape of modern read alignment algorithms leading to today's diverse array of alignment methods. Those technological changes rendered some read alignment algorithms irrelevant—yet provide context for the development of new tools better suited for modern next-generation sequencing data. The development of alignment algorithms is shaped not only by the characteristics of sequencing technologies but also by the specific characteristics of the application domain. Often different biological questions can be answered using similar bioinformatics algorithms. For example, BLAT [38, 234], a tool that was originally designed to map EST and Sanger reads, is now used to map the assembled contigs to the reference genome [234]. Specific features of various domains of biological research, including whole transcriptome, adaptive immune repertoire, and human

microbiome studies, confront the developer with a choice of developing a novel algorithm from scratch or adjusting existing algorithms.

In general, the read alignment problem is extremely challenging due to the large size of analyzed datasets and numerous technological limitations of modern sequencing platforms. A modern read aligner should not only be able to maintain a good balance between speed and memory usage but also be able to preserve small and large genetic variations. It should be capable of tackling numerous technological limitations and changes, ultimately inducing rapid evolution of sequencing technologies such as constant growth of read length and changes in error rates. In general, determining an accurate global position of the read in the reference genome provides no guarantee that accurate local pairwise alignment can be found. This is especially challenging for the error-prone long reads, where determining the accurate global position of the read in the reference genome is usually easy, but local pairwise alignment represents a substantial challenge due to a high error rate.

This review not only provides an understanding of the basic concepts of read alignment, its limitations, and how they are mitigated but also helps inform its future directions in read alignment development. We believe the future is bright for read alignment algorithms, and we hope that the many examples of read alignment algorithms presented in this work inspire researchers and developers to enhance the field of computational genomics by accurate and scalable tools.

Supplementary Information

The online version contains supplementary material available at <https://doi.org/10.1186/s13059-021-02443-7>.

Additional file 1. Supplementary tables 1-6; supplementary Figures 1-11; supplementary notes 1-6; supplementary materials.

Additional file 2. Review history.

Acknowledgements

We thank the authors of the tools surveyed in this work for providing helpful feedback and verifying the information related to their tool. We also thank Martin Frith (University of Tokyo), Heng Li (Harvard University), Cenik Sahinalp (National Cancer Institute), and Steven Salzberg (Johns Hopkins University) for their valuable feedback and discussion.

Peer review information

Andrew Cosgrove was the primary editor of this article and managed its editorial process and peer review in collaboration with the rest of the editorial team.

Review history

The peer review history is available as additional file 2.

Authors' contributions

M.A. and S.M. led the project, S.M. conceived of the presented idea, B.B. performed the statistical analysis, J.R., D.D., and M.A. produced the figures. H.S., J.R., K.T., and M.A. compiled Table 1. J.R., P.I.B., and V.X. created scripts for running and evaluating software tools. A.Z., B.B., B.D.S., C.A., D.K., H.S., H.T.Y., J.R., M.A., O.M., P.S., S.K., and S.M. wrote, reviewed, and edited the manuscript. All authors read and approved the final manuscript.

Funding

B.D.S. is supported by NIH/NHLBI K08HL128867, P.S. is supported by NIH 1R01EB025022 and National Science Foundation grants 2047828, P.I.B. and S.K. are supported by the Molecular Basis of Disease (MBD), B.S. is supported by NIH R01HL149883 and NIH R01HL153122, O.M. is supported by Intel, VMware, and NIH HG006004, and S.M. is supported by National Science Foundation grant 2041984. The authors acknowledge the Computational Genomics Summer Institute (CGSI), funded by NIH GM112625, which fostered international collaboration among the groups involved in this project.

Availability of data and materials

All data and code required to produce the figures contained within this text are freely available on GitHub: <https://github.com/Mangul-Lab-USC/review.technology.dictates.algorithms>.

Declarations

Competing interests

The authors declare no competing interests.

Author details

¹Computer Science Department, ETH Zürich, 8092 Zürich, Switzerland. ²Computer Engineering Department, Bilkent University, 06800 Bilkent, Ankara, Turkey. ³Information Technology and Electrical Engineering Department, ETH Zürich, Zürich 8092, Switzerland. ⁴Department of Computer Science, University of California Los Angeles, Los Angeles, CA 90095, USA. ⁵Department of Clinical Pharmacy, School of Pharmacy, University of Southern California, Los Angeles, CA 90089, USA. ⁶Department of Epidemiology, Harvard T.H. Chan School of Public Health, Boston, MA 02115, USA. ⁷Program in Medical and Population Genetics, Broad Institute of MIT and Harvard, Cambridge, MA 02142, USA. ⁸Department of Computer Science, Georgia State University, Atlanta, GA 30302, USA. ⁹Bioinformatics Interdepartmental Ph.D. Program, University of California Los Angeles, Los Angeles, CA 90095, USA. ¹⁰Division of Pulmonary and Critical Care Medicine, Northwestern University Feinberg School of Medicine, Chicago, IL 60611, USA. ¹¹Department of Biochemistry & Molecular Genetics, Northwestern University Feinberg School of Medicine, Chicago, IL 60611, USA. ¹²Simpson Querrey Institute for Epigenetics, Northwestern University Feinberg School of Medicine, Chicago, IL 60611, USA. ¹³Department of Computational Medicine, University of California Los Angeles, Los Angeles, CA 90095, USA. ¹⁴Computer Science and Engineering, Pennsylvania State University, University Park, PA 16801, USA. ¹⁵Biology Department, Pennsylvania State University, University Park, PA 16801, USA. ¹⁶The Huck Institutes of the Life Sciences, Pennsylvania State University, University Park, PA 16801, USA. ¹⁷The Laboratory of Bioinformatics, I.M. Sechenov First Moscow State Medical University, Moscow 119991, Russia. ¹⁸Bilkent-Hacettepe Health Sciences and Technologies Program, Ankara, Turkey.

Received: 15 October 2020 Accepted: 28 July 2021

Published online: 26 August 2021

References

- Weissenbach J. Human Genome Project: Past, Present, Future. In: *The Human Genome*; 2002. p. 1–9.
- Shendure J, Ji H. Next-generation DNA sequencing. *Nat Biotechnol*. 2008;26:1135–45.
- Metzker ML. Sequencing technologies — the next generation. *Nat Rev Genet*. 2009;11:31–46.
- Payne A, Holmes N, Rakan V, Loose M. BulkVis: a graphical viewer for Oxford nanopore bulk FAST5 files. *Bioinformatics*. 2019;35:2193–8.
- Goodwin S, McPherson JD, McCombie WR. Coming of age: ten years of next-generation sequencing technologies. *Nat Rev Genet*. 2016;17:333–51.
- Fox EJ, Reid-Bayliss KS, Emond MJ, Loeb LA. Accuracy of Next Generation Sequencing Platforms, Nextgeneration, sequencing & applications. 2014;1:106–14.
- Watson M, Warr A. Errors in long-read assemblies can critically affect protein prediction. *Nat Biotechnol*. 2019;37:124–6.
- Srivastava A, Malik L, Sarkar H, Zakeri M, Almodaresi F, Sonesson C, Love MI, Kingsford C, Patro R. Alignment and mapping methodology influence transcript abundance estimation. *Genome biology*. 2020;21(1):1–29.
- Saha A, Battle A. False positives in trans-eQTL and co-expression analyses arising from RNA-sequencing alignment errors. *F1000Res*. 2018;7:1860.
- Schbath S, Martin V. Mapping Reads on a Genomic Sequence: An Algorithmic Overview and a Practical Comparative Analysis. *J Comput Biol*. 2012;19(6):796–813. <https://doi.org/10.1089/cmb.2012.0022>.
- Fonseca NA, Rung J, Brazma A, Marioni JC. Tools for mapping high-throughput sequencing data. *Bioinformatics*. 2012;28:3169–77.
- Li H, Homer N. A survey of sequence alignment algorithms for next-generation sequencing. *Brief Bioinform*. 2010;11:473–83.
- Hatem A, Bozdağ D, Toland AE, Çatalyürek ÜV. Benchmarking short sequence mapping tools. *BMC Bioinform*. 2013;14:184.
- Baruzzo G, et al. Simulation-based comprehensive benchmarking of RNA-seq aligners. *Nat Methods*. 2017;14:135–9.
- Nagarajan N, Pop M. Sequence assembly demystified. *Nat Rev Genet*. 2013;14:157–67.
- Jain M, Koren S, Miga KH, Quick J, Rand AC, Sasani TA, Tyson JR, Beggs AD, Dilthey AT, Fiddes IT, Malla S. Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nature biotechnology*. 2018;36(4):338–45.
- Sedlazeck FJ, Lee H, Darby CA, Schatz MC. Piercing the dark matter: bioinformatics of long-range sequencing and mapping. *Nature Reviews Genetics*. 2018;19(6):329–46.
- Wenger AM, Peluso P, Rowell WJ, Chang PC, Hall RJ, Concepcion GT, Ebler J, Fungtammasan A, Kolesnikov A, Olson ND, Töpfer A. Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. *Nature biotechnology*. 2019;37(10):1155–62.
- Wee Y, Bhyan SB, Liu Y, Lu J, Li X, Zhao M. The bioinformatics tools for the genome assembly and analysis based on third-generation sequencing. *Briefings in functional genomics*. 2019;18(1):1–12.
- Canzar S, Salzberg SL. Short Read Mapping: An Algorithmic Tour. *Proc IEEE Inst Electr Electron Eng*. 2017;105:436–58.
- Steinberg KM, Schneider VA, Alkan C, Montague MJ, Warren WC, Church DM, Wilson RK. Building and improving reference genome assemblies. *Proceedings of the IEEE*. 2017;105(3):422–35.
- Baichoo S, Ouzounis CA. Computational complexity of algorithms for sequence comparison, short-read assembly and genome alignment. *Biosystems*. 2017;156–157:72–85.
- Eklom R, Wolf JBW. A field guide to whole-genome sequencing, assembly and annotation. *Evol Appl*. 2014;7:1026–42.
- Bradnam KR, et al. Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. *Gigascience*. 2013;2:10.
- Zimin AV, et al. The first near-complete assembly of the hexaploid bread wheat genome, *Triticum aestivum*. *Gigascience*. 2017;6:1–7.

26. Flicek P, Birney E. Sense from sequence reads: methods for alignment and assembly. *Nat Methods*. 2009;6:S6–S12.
27. Firtina C, Alkan C. On genomic repeats and reproducibility. *Bioinformatics*. 2016;32:2243–7.
28. Weiss LA, et al. Association between microdeletion and microduplication at 16p11.2 and autism. *N Engl J Med*. 2008; 358:667–75.
29. Guo Y, et al. The effect of strand bias in Illumina short-read sequencing data. *BMC Genomics*. 2012;13:666.
30. Pearson WR, Lipman DJ. Improved tools for biological sequence comparison. *Proc Natl Acad Sci U S A*. 1988;85:2444–8.
31. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. *J Mol Biol*. 1990;215:403–10.
32. Altschul SF, et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*. 1997;25:3389–402.
33. Ning Z, Cox AJ, Mullikin JC. SSAHA: a fast search method for large DNA databases. *Genome Res*. 2001;11(10):1725–9. <https://doi.org/10.1101/gr.194201>.
34. Egidi L, Manzini G. Better spaced seeds using Quadratic Residues. *J Comput Syst Sci*. 2013;79:1144–55.
35. Rizk G, Lavenier D. GASST: global alignment short sequence search tool. *Bioinformatics*. 2010;26:2534–40.
36. Ma B, Tromp J, Li M. PatternHunter: faster and more sensitive homology search. *Bioinformatics*. 2002;18:440–5.
37. Wu TD, Nacu S. Fast and SNP-tolerant detection of complex variants and splicing in short reads. *Bioinformatics*. 2010;26: 873–81.
38. Kent WJ. BLAT—the BLAST-like alignment tool. *Genome Res*. 2002;12(4):656–64. <https://doi.org/10.1101/gr.229202>.
39. Schwartz S, et al. Human-mouse alignments with BLASTZ. *Genome Res*. 2003;13:103–7.
40. Slater GSC, Birney E. Automated generation of heuristics for biological sequence comparison. *BMC Bioinform*. 2005;6:31.
41. Wu TD, Watanabe CK. GMAP: a genomic mapping and alignment program for mRNA and EST sequences. *Bioinformatics*. 2005;21:1859–75.
42. Lam TW, Sung WK, Tam SL, Wong CK, Yiu SM. Compressed indexing and local alignment of DNA. *Bioinformatics*. 2008; 24:791–7.
43. Li H, Ruan J, Durbin R. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res*. 2008;18(11):1851–8. <https://doi.org/10.1101/gr.078212.108>.
44. Smith AD, Xuan Z, Zhang MQ. Using quality scores and longer reads improves accuracy of Solexa read mapping. *BMC Bioinform*. 2008;9:128.
45. Li R, Li Y, Kristiansen K, Wang J. SOAP: short oligonucleotide alignment program. *Bioinformatics*. 2008;24:713–4.
46. Ondov BD, Varadarajan A, Passalacqua KD, Bergman NH. Efficient mapping of Applied Biosystems SOLiD sequence data to a reference genome for functional genomic applications. *Bioinformatics*. 2008;24:2776–7.
47. Jiang H, Wong WH. SeqMap: mapping massive amount of oligonucleotides to the genome. *Bioinformatics*. 2008;24: 2395–6.
48. Lin H, Zhang Z, Zhang MQ, Ma B, Li M. ZOOM! Zillions of oligos mapped. *Bioinformatics*. 2008;24(21):2431–7. <https://doi.org/10.1093/bioinformatics/btn416>.
49. De Bona F, Ossowski S, Schneeberger K, Rätsch G. Optimal spliced alignments of short sequence reads. *Bioinformatics*. 2008;24:174–80.
50. Jean G, Kahles A, Sreedharan VT, De Bona F, Rätsch G. RNA-Seq read alignments with PALMapper. *Curr Protoc Bioinform*. 2010;Chapter 11:Unit 11.6.
51. Harris EY, Ponts N, Le Roch KG, Lonardi S. BRAT-BW: efficient and accurate mapping of bisulfite-treated reads. *Bioinformatics*. 2012;28:1795–6.
52. Xi Y, Li W. BSMAP: whole genome bisulfite sequence MAPping program. *BMC Bioinform*. 2009;10:232.
53. Homer N, Merriman B, Nelson SF. BFAST: an alignment tool for large scale genome resequencing. *PLoS One*. 2009;4: e7767.
54. Li H, Durbin R. Fast and accurate long-read alignment with Burrows–Wheeler transform. *Bioinformatics*. 2010;26:589–95.
55. Langmead B, Trapnell C, Pop M, Salzberg SL. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol*. 2009;10:R25.
56. Schatz MC. CloudBurst: highly sensitive read mapping with MapReduce. *Bioinformatics*. 2009;25:1363–9.
57. Clement NL, et al. The GNUMAP algorithm: unbiased probabilistic mapping of oligonucleotides from next-generation sequencing. *Bioinformatics*. 2010;26:38–45.
58. Schneeberger K, et al. Simultaneous alignment of short reads against multiple genomes. *Genome Biol*. 2009;10:R98.
59. Eaves HL, Gao Y. MOM: maximum oligonucleotide mapping. *Bioinformatics*. 2009;25:969–70.
60. Campagna D, et al. PASS: a program to align short sequences. *Bioinformatics*. 2009;25:967–8.
61. Chen Y, Souaiaia T, Chen T. PerM: efficient mapping of short sequencing reads with periodic full sensitive spaced seeds. *Bioinformatics*. 2009;25:2514–21.
62. Weese D, Emde A-K, Rausch T, Döring A, Reinert K. RazerS—fast read mapping with sensitivity control. *Genome Res*. 2009;19:1646–54.
63. Rumble SM, et al. SHRIMP: accurate mapping of short color-space reads. *PLoS Comput Biol*. 2009;5:e1000386.
64. Li R, et al. SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics*. 2009;25:1966–7.
65. Malhis N, Butterfield YSN, Ester M, Jones SJM. Slider—maximum use of probability information for alignment of short sequence reads and SNP detection. *Bioinformatics*. 2009;25:6–13.
66. Hoffmann S, et al. Fast mapping of short sequences with mismatches, insertions and deletions using index structures. *PLoS Comput Biol*. 2009;5:e1000502.
67. Trapnell C, Pachter L, Salzberg SL. TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics*. 2009;25:1105–11.
68. Chen P-Y, Cokus SJ, Pellegrini M. BS Seeker: precise mapping for bisulfite sequencing. *BMC Bioinform*. 2010;11:203.
69. Hannes Ponting ZN. SMALT - A New Mapper for DNA Sequencing Reads; 2010.
70. Malhis N, Jones SJM. High quality SNP calling using Illumina data at shallow coverage. *Bioinformatics*. 2010;26:1029–35.
71. Kurtz S. 2016. <http://www.vmatch.de/virtman.pdf>. Accessed Feb 2020.
72. Hach F, et al. mrsFAST: a cache-oblivious algorithm for short-read mapping. *Nat Methods*. 2010;7:576–7.
73. Wang K, et al. MapSplice: accurate mapping of RNA-seq reads for splice junction discovery. *Nucleic Acids Res*. 2010;38:e178.
74. Emde A-K, Grunert M, Weese D, Reinert K, Sperling SR. MicroRazerS: rapid alignment of small RNA reads. *Bioinformatics*. 2010;26:123–4.

75. Au KF, Jiang H, Lin L, Xing Y, Wong WH. Detection of splice junctions from paired-end RNA-seq data by SpliceMap. *Nucleic Acids Res.* 2010;38:4570–8.
76. Bryant DW Jr, Shen R, Priest HD, Wong W-K, Mockler TC. Supersplat–spliced RNA-seq alignment. *Bioinformatics.* 2010;26:1500–5.
77. Krueger F, Andrews SR. Bismark: a flexible aligner and methylation caller for Bisulfite-Seq applications. *Bioinformatics.* 2011;27:1571–2.
78. Kielbasa SM, Wan R, Sato K, Horton P, Frith MC. Adaptive seeds tame genomic sequence comparison. *Genome Res.* 2011;21:487–93.
79. Flouri T, Iliopoulos CS, Pissis SP. DynMap: mapping short reads to multiple related genomes; 2011.
80. David M, Dzamba M, Lister D, Ilie L, Brudno M. SHRIMP2: sensitive yet practical SHort Read Mapping. *Bioinformatics.* 2011;27:1011–2.
81. Zaharia, M, et al. Faster and More Accurate Sequence Alignment with SNAP. *arXiv [cs.DS].* 2011.
82. Lunter G, Goodson M. Stampy: a statistical algorithm for sensitive and fast mapping of Illumina sequence reads. *Genome Res.* 2011;21:936–9.
83. Wood DLA, Xu Q, Pearson JV, Cloonan N, Grimmond SM. X-MATE: a flexible system for mapping short read data. *Bioinformatics.* 2011;27:580–1.
84. Huang S, et al. SOAPsplice: Genome-Wide ab initio Detection of Splice Junctions from RNA-Seq Data. *Front Genet.* 2011;2:46.
85. Chaisson MJ, Tesler G. Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory. *BMC Bioinform.* 2012;13:238.
86. Tennakoon C, Purbojati RW, Sung W-K. BatMis: a fast algorithm for k-mismatch mapping. *Bioinformatics.* 2012;28:2122–8.
87. Langmead B, Salzberg SL. Fast gapped-read alignment with Bowtie 2. *Nat Methods.* 2012;9:357–9.
88. Marco-Sola S, Sammeth M, Guigó R, Ribeca P. The GEM mapper: fast, accurate and versatile alignment by filtration. *Nat Methods.* 2012;9:1185–8.
89. Weese D, Holtgrewe M, Reinert K. RazerS 3: faster, fully sensitive read mapping. *Bioinformatics.* 2012;28:2592–9.
90. Mu JC, et al. Fast and accurate read alignment for resequencing. *Bioinformatics.* 2012;28:2366–73.
91. Emde A-K, Schulz MH. Detecting genomic indel variants with exact breakpoints in single- and paired-end sequencing data using SplazerS. *Bioinformatics.* 2012;28(5):619–27. <https://doi.org/10.1093/bioinformatics/bts019>.
92. Li Y, Terrell A, Patel JM. WHAM: A High-throughput Sequence Alignment Method; 2011.
93. Faust GG, Hall IM. YAHA: fast and flexible long-read alignment with optimal breakpoint detection. *Bioinformatics.* 2012;28:2417–24.
94. Hu J, Ge H, Newman M, Liu K. OSA: a fast and accurate alignment tool for RNA-Seq. *Bioinformatics.* 2012;28(14):1933–4. <https://doi.org/10.1093/bioinformatics/bts294>.
95. Zhang Y, et al. PASSion: a pattern growth algorithm-based pipeline for splice junction detection in paired-end RNA-Seq data. *Bioinformatics.* 2012;28:479–86.
96. Guo W, et al. BS-Seeker2: a versatile aligning pipeline for bisulfite sequencing data. *BMC Genomics.* 2013;14:774.
97. Liao Y, Smyth GK, Shi W. The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote. *Nucleic Acids Res.* 2013;41:e108.
98. Li, H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv [q-bio.GN].* 2013.
99. Siragusa E, Weese D, Reinert K. Fast and accurate read mapping with approximate seeds and multiple backtracking. *Nucleic Acids Res.* 2013;41:e78.
100. Sedlazeck FJ, Rescheneder P, von Haeseler A. NextGenMap: fast and accurate read mapping in highly polymorphic genomes. *Bioinformatics.* 2013;29:2790–1.
101. Gontarz PM, Berger J, Wong CF. SRmapper: a fast and sensitive genome-hashing alignment tool. *Bioinformatics.* 2013;29:316–21.
102. Alkan C, et al. Personalized copy number and segmental duplication maps using next-generation sequencing. *Nat Genet.* 2009;41:1061–7.
103. Philippe N, Salson M, Commes T, Rivals E. CRAC: an integrated approach to the analysis of RNA-seq reads. *Genome Biol.* 2013;14:R30.
104. Dobin A, et al. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics.* 2013;29:15–21.
105. Kim D, et al. TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biol.* 2013;14:R36.
106. Sahinalp SC, Vishkin U. Efficient approximate and dynamic matching of patterns using a labeling paradigm. In *Proceedings of 37th IEEE Conference on Foundations of Computer Science.* October 1996;320–328.
107. Kerpedjiev P, Frellsen J, Lindgreen S, Krogh A. Adaptable probabilistic mapping of short reads using position specific scoring matrices. *BMC Bioinform.* 2014;15:100.
108. Liu Y, Popp B, Schmidt B. CUSHAW3: sensitive and accurate base-space and color-space short-read alignment with hybrid seeding. *PLoS One.* 2014;9:e86869.
109. Kim J, Li C, Xie X. Improving read mapping using additional prefix grams. *BMC Bioinform.* 2014;15(1):42. <https://doi.org/10.1186/1471-2105-15-42>.
110. Lee W-P, et al. MOSAIK: a hash-based algorithm for accurate next-generation sequencing short-read mapping. *PLoS One.* 2014;9:e90581.
111. Tárraga J, et al. Acceleration of short and long DNA read mapping without loss of accuracy using suffix array. *Bioinformatics.* 2014;30:3396–8.
112. Hach F, et al. mrsFAST-Ultra: a compact, SNP-aware mapper for high performance sequencing applications. *Nucleic Acids Res.* 2014;42:W494–500.
113. Butterfield YS, Kreitzman M. JAGuar: junction alignments to genome for RNA-seq reads. *PLoS One.* 2014;9(7):e102398. <https://doi.org/10.1371/journal.pone.0102398>.
114. Bonfert T, Kirner E, Csaba G, Zimmer R, Friedel CC. ContextMap 2: fast and accurate context-based RNA-seq mapping. *BMC Bioinform.* 2015;16:122.

115. Kim D, Langmead B, Salzberg SL. HISAT: a fast spliced aligner with low memory requirements. *Nat Methods*. 2015;12:357–60.
116. Prezza N, Vezzi F, Kaller M, Policriti A. Fast, accurate, and lightweight analysis of BS-treated reads with ERNE 2. *BMC Bioinform*. 2016;17(Suppl 4):69.
117. Sovic I, et al. Fast and sensitive mapping of nanopore sequencing reads with GraphMap. *Nat Commun*. 2016;7:11307.
118. Amin MR, Skiena S, Schatz MC. NanoBLASTer: Fast alignment and characterization of Oxford Nanopore single molecule sequencing reads, 2016 IEEE 6th International Conference on Computational Advances in Bio and Medical Sciences (ICCBS); 2016. <https://doi.org/10.1109/iccabs.2016.7802776>.
119. Li H. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*. 2016;32:2103–10.
120. Liu B, Guan D, Teng M, Wang Y. rHAT: fast alignment of noisy long reads with regional hashing. *Bioinformatics*. 2016;32:1625–31.
121. Lin H-N, Hsu W-L. Kart: a divide-and-conquer algorithm for NGS read alignment. *Bioinformatics*. 2017;33:2281–7.
122. Liu B, Gao Y, Wang Y. LAMSA: fast split read alignment with long approximate matches. *Bioinformatics*. 2017;33:192–201.
123. Lin H-N, Hsu W-L. DART: a fast and accurate RNA-seq mapper with a partitioning strategy. *Bioinformatics*. 2018;34:190–7.
124. Li H. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*. 2018;34(18):3094–100. <https://doi.org/10.1093/bioinformatics/bty191>.
125. Dadi TH, et al. DREAM-Yara: an exact read mapper for very large databases with short update time. *Bioinformatics*. 2018;34:i766–72.
126. Marais G, et al. MUMmer4: A fast and versatile genome alignment system. *PLoS Comput Biol*. 2018;14:e1005944.
127. Sedlazeck FJ, et al. Accurate detection of complex structural variations using single-molecule sequencing. *Nat Methods*. 2018;15:461–8.
128. Haghsheenas E, Sahinalp SC, Hach F. lordFAST: sensitive and Fast Alignment Search Tool for LOnG noisy Read sequencing Data. *Bioinformatics*. 2019;35:20–7.
129. Zhou Q, Lim J-Q, Sung W-K, Li G. An integrated package for bisulfite DNA methylation data analysis with Indel-sensitive mapping. *BMC Bioinform*. 2019;20:47.
130. Maric J, Sovic I, Krizanovic K, Nagarajan N, Sikic M. Graphmap2-splice-aware RNA-seq mapper for long reads. *bioRxiv*. 2019; p.720458.
131. Boratyn GM, Thierry-Mieg J, Thierry-Mieg D, Busby B, Madden TL, Boratyn GM, Thierry-Mieg J, Thierry-Mieg D, Busby B, Madden TL. Magic-BLAST, an accurate RNA-seq aligner for long and short reads. *BMC bioinformatics*. 2019;20(1):1–19.
132. Vasimuddin M, Misra S, Li H, Aluru S. Efficient Architecture-Aware Acceleration of BWA-MEM for Multicore Systems, 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS); 2019. <https://doi.org/10.1109/ipdps.2019.00041>.
133. Kim D, Paggi JM, Park C, Bennett C, Salzberg SL. Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype. *Nat Biotechnol*. 2019;37:907–15.
134. Liu B, et al. deSALT: fast and accurate long transcriptomic read alignment with de Bruijn graph-based index. <https://doi.org/10.1101/612176>.
135. Chakraborty A, Bandyopadhyay S. conLSH: Context based Locality Sensitive Hashing for mapping of noisy SMRT reads. *Comput Biol Chem*. 2020;85:107206.
136. Wenger AM, et al. Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. *Nat Biotechnol*. 2019. <https://doi.org/10.1038/s41587-019-0217-9>.
137. Yorukoglu D, Yu YW, Peng J, Berger B. Compressive mapping for next-generation sequencing. *Nat Biotechnol*. 2016;34:374–6.
138. Wilbur WJ, Lipman DJ. Rapid similarity searches of nucleic acid and protein data banks. *Proc Natl Acad Sci U S A*. 1983;80:726–30.
139. Burkhardt S, Karkkainen J. Better Filtering with Gapped q-Grams. *Comb Pattern Matching*. 2001:73–85. https://doi.org/10.1007/3-540-48194-x_6.
140. Ukkonen E. Approximate string-matching over suffix trees. In: *Combinatorial Pattern Matching*. Berlin Heidelberg: Springer; 1993. p. 228–42.
141. Ghodsi M, Pop M. Inexact Local Alignment Search over Suffix Arrays. In: 2009 IEEE International Conference on Bioinformatics and Biomedicine; 2009. p. 83–7.
142. Cokus SJ, et al. Shotgun bisulphite sequencing of the Arabidopsis genome reveals DNA methylation patterning. *Nature*. 2008;452:215–9.
143. Kurtz S, et al. Versatile and open software for comparing large genomes. *Genome Biol*. 2004;5:R12.
144. Medina I, et al. Highly sensitive and ultrafast read mapping for RNA-seq analysis. *DNA Res*. 2016;23:93–100.
145. Li H, Durbin R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*. 2009;25:1754–60.
146. Gruning B, et al. Bioconda: sustainable and comprehensive software distribution for the life sciences. *Nat Methods*. 2018;15:475–6.
147. Mohamadi H, Vandervalk BP. DIDA: Distributed Indexing Dispatched Alignment. *PLoS One*. 2015;10(4):e0126409. <https://doi.org/10.1371/journal.pone.0126409>.
148. Xin H, et al. Accelerating read mapping with FastHASH. *BMC Genomics*. 2013;14(Suppl 1):S13.
149. Xin H, Nahar S. Optimal seed solver: optimizing seed selection in read mapping. *Bioinformatics*. 2016;32(11):1632–42. <https://doi.org/10.1093/bioinformatics/btv670>.
150. Zhang H, Chan Y, Fan K, Schmidt B, Liu W. Fast and efficient short read mapping based on a succinct hash index. *BMC Bioinform*. 2018;19:92.
151. Eddy SR. What is dynamic programming? *Nat Biotechnol*. 2004;22:909.
152. Smith TF, Waterman MS. Identification of common molecular subsequences. *J Mol Biol*. 1981;147:195–7.
153. Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*. 1970;48:443–53.
154. Hamming RW. Error detecting and error correcting codes. *Bell Syst Tech J*. 1950;29:147–60.

155. Karp RM, Rabin MO. Efficient randomized pattern-matching algorithms. *IBM J Res Dev.* 1987;31:249–60.
156. Calude C, Salomaa K, Yu S. Additive distances and quasi-distances between words. *J Univ Comput Sci.* 2002;8:141–52.
157. Backurs A, Indyk P. Edit Distance Cannot Be Computed in Strongly Subquadratic Time (unless SETH is false), *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing - STOC '15*; 2015. <https://doi.org/10.1145/2746539.2746612>.
158. Ukkonen E. Algorithms for approximate string matching. *Information and control.* 1985;64(1-3):100-18.
159. Cole R, Hariharan R. Approximate String Matching: A Simpler Faster Algorithm. *SIAM J Comput.* 2002;31:1761–82.
160. Alser M, Hassan H, Kumar A, Mutlu O, Alkan C. Shouji: a fast and efficient pre-alignment filter for sequence alignment. *Bioinformatics.* 2019;35(21):4255-63.
161. Alser M, Hassan H, Xin H, Ergin O, Mutlu O, Alkan C. GateKeeper: a new hardware architecture for accelerating pre-alignment in DNA short read mapping. *Bioinformatics.* 2017;33(21):3355-63.
162. Alser, M., Mutlu, O. & Alkan, C. MAGNET: Understanding and Improving the Accuracy of Genome Pre-Alignment Filtering. *arXiv [q-bio.GN]*. 2017.
163. Kim JS, et al. GRIM-Filter: Fast seed location filtering in DNA read mapping using processing-in-memory technologies. *BMC Genomics.* 2018;19:89.
164. Alser M, Shahroodi T, Gómez-Luna J, Alkan C, Mutlu O. SneakySnake: a fast and accurate universal genome pre-alignment filter for CPUs, GPUs and FPGAs. *Bioinformatics.* 2020;36(22-23):5282-90.
165. Zhang J, et al. BGSA: A Bit-Parallel Global Sequence Alignment Toolkit for Multi-core and Many-core Architectures. *Bioinformatics.* 2018. <https://doi.org/10.1093/bioinformatics/bty930>.
166. Turakhia Y, Goenka SD, Bejerano G, Dally WJ. Darwin-WGA: A Co-processor Provides Increased Sensitivity in Whole Genome Alignments with High Speedup, 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA); 2019. <https://doi.org/10.1109/hpca.2019.00050>.
167. Cali DS, et al. GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis. In: 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO); 2020. p. 951–66.
168. Alser M, et al. Accelerating Genome Analysis: A Primer on an Ongoing Journey. *IEEE Micro.* 2020;40:65–75.
169. Kloosterman WP, et al. Characteristics of de novo structural changes in the human genome. *Genome Res.* 2015;25:792–801.
170. Vollger MR, et al. Long-read sequence and assembly of segmental duplications. *Nat Methods.* 2019;16:88–94.
171. Merker JD, Wenger AM. Long-read genome sequencing identifies causal structural variation in a Mendelian disease. *Genet Med.* 2018;20(1):159–63. <https://doi.org/10.1038/gim.2017.86>.
172. Goodwin S, et al. Oxford Nanopore sequencing, hybrid error correction, and de novo assembly of a eukaryotic genome. *Genome Res.* 2015;25:1750–6.
173. Eccles D, et al. De novo assembly of the complex genome of *Nippostrongylus brasiliensis* using MinION long reads. *BMC Biol.* 2018;16:6.
174. Quick J, et al. Real-time, portable genome sequencing for Ebola surveillance. *Nature.* 2016;530:228–32.
175. Kolmogorov M, Yuan J, Lin Y, and Pevzner, P.A., Assembly of long, error-prone reads using repeat graphs. *Nature biotechnology.* 2019;37(5):540-6.
176. Misra S, Agrawal A, Liao W-K, Choudhary A. Anatomy of a hash-based long read sequence mapping algorithm for next generation DNA sequencing. *Bioinformatics.* 2011;27:189–95.
177. Liu Y, Schmidt B. Long read alignment based on maximal exact match seeds. *Bioinformatics.* 2012;28:1318–24.
178. Firtina C, Bar-Joseph Z, Alkan C, Cicek AE. Hercules: a profile HMM-based hybrid error correction algorithm for long reads. *Nucleic acids research.* 2018;46(21):e125.
179. Roberts M, Hayes W, Hunt BR, Mount SM, Yorke JA. Reducing storage requirements for biological sequence comparison. *Bioinformatics.* 2004;20:3363–9.
180. Schleimer S, Wilkerson DS, Aiken A. Winnowing, *Proceedings of the 2003 ACM SIGMOD international conference on Management of data - SIGMOD '03*; 2003. <https://doi.org/10.1145/872757.872770>.
181. Liu Y, Yu Z, Dinger ME, Li J. Index suffix-prefix overlaps by (w, k)-minimizer to generate long contigs for reads compression. *Bioinformatics.* 2019;35(12):2066-74.
182. Wood DE, Salzberg SL. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol.* 2014;15:R46.
183. Jain C, Dilthey A, Koren S, Aluru S, Phillippy AM. A Fast Approximate Algorithm for Mapping Long Reads to Large Reference Databases. *J Comput Biol.* 2018;25:766–79.
184. Gong L, et al. Picky comprehensively detects high-resolution structural variants in nanopore long reads. *Nat Methods.* 2018;15:455–60.
185. Engström PG, et al. Systematic evaluation of spliced alignment programs for RNA-seq data. *Nat Methods.* 2013;10:1185–91.
186. Goldstein LD, et al. Prediction and Quantification of Splice Events from RNA-Seq Data. *PLoS One.* 2016;11:e0156132.
187. Veeneman BA, Shukla S, Dhanasekaran SM, Chinnaiyan AM, Nesvizhskii AI. Two-pass alignment improves novel splice junction quantification. *Bioinformatics.* 2016;32:43–9.
188. Mangul S, et al. Transcriptome assembly and quantification from Ion Torrent RNA-Seq data. *BMC Genomics.* 2014; 15(Suppl 5):S7.
189. Nicolae M, Mangul S, Măndoiu II, Zelikovsky A. Estimation of alternative splicing isoform frequencies from RNA-Seq data. *Algorithms Mol Biol.* 2011;6:9.
190. Bray NL, Pimentel H, Melsted P, Pachter L. Near-optimal probabilistic RNA-seq quantification. *Nat Biotechnol.* 2016;34: 525–7.
191. Patro R, Duggal G, Love MI, Irizarry RA, Kingsford C. Salmon provides fast and bias-aware quantification of transcript expression. *Nat Methods.* 2017;14:417–9.
192. Zhang C, Zhang B, Lin L-L, Zhao S. Evaluation and comparison of computational tools for RNA-seq isoform quantification. *BMC Genomics.* 2017;18:583.
193. Nawrocki EP, S. R. E. Infernal 1.1: 100-fold faster RNA homology searches. *Bioinformatics.* 2013;29:2933.

194. Kuczynski J, et al. Using QIIME to analyze 16S rRNA gene sequences from Microbial Communities. *Curr Protoc Bioinform.* 2011;CHAPTER:Unit10.7.
195. Schloss PD, et al. Introducing mothur: Open-Source, Platform-Independent, Community-Supported Software for Describing and Comparing Microbial Communities. *Appl Environ Microbiol.* 2009;75:7537–41.
196. Huson DH, Auch AF, Qi J, Schuster SC. MEGAN analysis of metagenomic data. *Genome Res.* 2007;17:377–86.
197. DeSantis TZ, Hugenholtz P. NAST: a multiple sequence alignment server for comparative analysis of 16S rRNA genes. *Nucleic Acids Res.* 2006;34(Web Server):W394–9. <https://doi.org/10.1093/nar/gkl244>.
198. Caporaso JG, Bittinger K, Bushman FD, DeSantis TZ, Andersen GL, Knight R. PyNAST: a flexible tool for aligning sequences to a template alignment. *Bioinformatics.* 2010;26(2):266–7. Access date: February 2020.
199. Edgar RC. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.* 2004; 32(5):1792–7. <https://doi.org/10.1093/nar/gkh340>.
200. Sczyrba A, et al. Critical Assessment of Metagenome Interpretation—a benchmark of metagenomics software. *Nat Methods.* 2017;14:1063–71.
201. Ounit R, Wanamaker S, Close TJ, Lonardi S. CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers. *BMC Genomics.* 2015;16(1):236. <https://doi.org/10.1186/s12864-015-1419-2>.
202. Lee AY, Lee CS, Van Gelder RN. Scalable metagenomics alignment research tool (SMART): a scalable, rapid, and complete search heuristic for the classification of metagenomic sequences from complex sequence populations. *BMC Bioinform.* 2016;17:292.
203. Segata N, et al. Metagenomic microbial community profiling using unique clade-specific marker genes. *Nat Methods.* 2012;9:811–4.
204. Sharpton TJ. An introduction to the analysis of shotgun metagenomic data. *Front Plant Sci.* 2014;5:209.
205. Steinegger M, Söding J. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat Biotechnol.* 2017;35:1026–8.
206. Zhao Y, Tang H, Ye Y. RAPSearch2: a fast and memory-efficient protein similarity search tool for next-generation sequencing data. *Bioinformatics.* 2012;28:125–6.
207. Buchfink B, Xie C, Huson DH. Fast and sensitive protein alignment using DIAMOND. *Nat Methods.* 2014;12:59.
208. Sanjuan R, Nebot MR, Chirico N, Mansky LM, Belshaw R. Viral Mutation Rates. *J Virol.* 2010;84:9733–48.
209. Domingo E, Sheldon J, Perales C. Viral quasispecies evolution. *Microbiol Mol Biol Rev.* 2012;76:159–216.
210. Beerenwinkel N, et al. Computational methods for the design of effective therapies against drug resistant HIV strains. *Bioinformatics.* 2005;21:3943–50.
211. Skums P, Bunimovich L, Khudyakov Y. Antigenic cooperation among intrahost HCV variants organized into a complex network of cross-immunoreactivity. *Proc Natl Acad Sci.* 2015;112:6653–8.
212. Knyazev S, Tsyvin V, Melnyk A, Artyomenko A, Malygina T, Porozov YB, Campbell E, Switzer WM, Skums P, Zelikovsky A. Cliquesnv: Scalable reconstruction of intra-host viral populations from ngs reads. *BioRxiv.* 2018:264242.
213. Zagordi O, Bhattacharya A, Eriksson N, Beerenwinkel N. ShoRAH: estimating the genetic diversity of a mixed sample from next-generation sequencing data. *BMC Bioinform.* 2011;12:119.
214. Yang X, Charlebois P, Macalalad A, Henn MR, Zody MC. V-Phaser 2: variant inference for viral populations. *BMC Genomics.* 2013;14(1):674. <https://doi.org/10.1186/1471-2164-14-674>.
215. Huber M, et al. MinVar: A rapid and versatile tool for HIV-1 drug resistance genotyping by deep sequencing. *J Virol Methods.* 2017;240:7–13.
216. Langmead B. Aligning short sequencing reads with Bowtie. *Curr Protoc Bioinform.* 2010;Chapter 11:Unit 11.7.
217. Harris RA, et al. Comparison of sequencing-based methods to profile DNA methylation and identification of monoallelic epigenetic modifications. *Nat Biotechnol.* 2010;28:1097–105.
218. Singer BD. A Practical Guide to the Measurement and Analysis of DNA Methylation. *Am J Respir Cell Mol Biol.* 2019;61: 417–28.
219. Sun X, Han Y, Zhou L, Chen E, Lu B, Liu Y, Pan X, Cowley Jr AW, Liang M, Wu Q, Lu Y. A comprehensive evaluation of alignment software for reduced representation bisulfite sequencing data. *Bioinformatics.* 2018;34(16):2715–23.
220. knights-lab. knights-lab/BURST. GitHub. <https://github.com/knights-lab/BURST>. Access date: February 2020.
221. Bolotin DA, et al. MiXCR: software for comprehensive adaptive immunity profiling. *Nat Methods.* 2015;12:380–1.
222. Kidd JM, et al. Mapping and sequencing of structural variation from eight human genomes. *Nature.* 2008;453:56–64.
223. Dennis MY, et al. Evolution of human-specific neural SRGAP2 genes by incomplete segmental duplication. *Cell.* 2012; 149:912–22.
224. Schneider VA, et al. Evaluation of GRCh38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly. *Genome Res.* 2017;27:849–64.
225. Phillippy AM, Schatz MC, Pop M. Genome assembly forensics: finding the elusive mis-assembly. *Genome Biol.* 2008;9: R55.
226. Hunt M, et al. REAPR: a universal tool for genome assembly evaluation. *Genome Biol.* 2013;14:R47.
227. Muggli MD, Puglisi SJ, Ronen R, Boucher C. Misassembly detection using paired-end sequence reads and optical mapping data. *Bioinformatics.* 2015;31:i80–8.
228. Jackman SD, et al. Tigmint: correcting assembly errors using linked reads from large molecules. *BMC Bioinform.* 2018;19: 393.
229. Walker BJ, et al. Pilon: an integrated tool for comprehensive microbial variant detection and genome assembly improvement. *PLoS One.* 2014;9:e112963.
230. Chin C-S, et al. Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. *Nat Methods.* 2013;10:563–9.
231. Loman NJ, Quick J, Simpson JT. A complete bacterial genome assembled de novo using only nanopore sequencing data. *Nat Methods.* 2015;12:733–5.
232. Firtina C, Kim JS, Alser M, Cali DS, Cicek AE, Alkan C, et al. Apollo: A Sequencing-Technology-Independent, Scalable, and Accurate Assembly Polishing Algorithm; 2019.
233. Koren S, et al. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome Res.* 2017;27:722–36.

234. Davidson NM, Oshlack A. Necklace: combining reference and assembled transcriptomes for more comprehensive RNASeq analysis. *GigaScience*. 2018;7(5):45–51.
235. Siren J, Valimäki N, Mäkinen V. Indexing Graphs for Path Queries with Applications in Genome Research. *IEEE/ACM Trans Comput Biol Bioinform*. 2014;11:375–88.
236. Artyomenko A, et al. Long Single-Molecule Reads Can Resolve the Complexity of the Influenza Virus Composed of Rare, Closely Related Mutant Variants. *J Comput Biol*. 2017;24:558–70.
237. Brudno M, et al. Glocal alignment: finding rearrangements during alignment. *Bioinformatics*. 2003;19(Suppl 1):i54–62.
238. Kircher M, Heyn P, Kelso J. Addressing challenges in the production and analysis of illumina sequencing data. *BMC Genomics*. 2011;12:382.
239. Delcher AL, Phillippy A, Carlton J, Salzberg SL. Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acids Res*. 2002;30:2478–83.
240. Mäkinen V, Sahlin K. Chaining with overlaps revisited; 2020.
241. Chen S, Wang A, Li LM. SEME: A Fast Mapper of Illumina Sequencing Reads with Statistical Evaluation. *Lect Notes Comput Sci*. 2013:14–29. https://doi.org/10.1007/978-3-642-37195-0_2.
242. Ahmadi A, et al. Hobbes: optimized gram-based methods for efficient read alignment. *Nucleic Acids Res*. 2012;40:e41.
243. Cheng H, Jiang H, Yang J, Xu Y, Shang Y. BitMapper: an efficient all-mapper based on bit-vector computing. *BMC Bioinform*. 2015;16:192.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Ready to submit your research? Choose BMC and benefit from:

- fast, convenient online submission
- thorough peer review by experienced researchers in your field
- rapid publication on acceptance
- support for research data, including large and complex data types
- gold Open Access which fosters wider collaboration and increased citations
- maximum visibility for your research: over 100M website views per year

At BMC, research is always in progress.

Learn more biomedcentral.com/submissions



{Supplementary Information}

Supplementary Table 1. Genome index size across three read alignment tools.

Tool	Version	Index Size	Indexing Time
mrFAST	2.2.5	16.5 GB	1202 seconds
minimap2	0.12.7	7.2 GB	200 seconds
BWA-MEM	0.7.17	4.7 GB	2998 seconds

Supplementary Table 2. Tools available in the bioconda package manager. Here we have included only the tools that are primarily built for DNA read alignment.

Software tool	Version	Publication	Conda command
Bowtie2	2.2.5	¹	conda install -c bioconda bowtie2
Bowtie	0.12.7	²	conda install -c bioconda bowtie
BWA	0.7.17	³	conda install -c bioconda bwa
GSNAP	2018-03-25	⁴	conda install -c compbiocore gsnap
HISAT2	2.1.0	⁵	conda install -c bioconda hisat2
LAST	963	⁶	conda install -c bioconda last
minimap2	2.12-r827	⁷	conda install -c bioconda minimap2
RMAP	2.1	⁸	conda install -c bioconda rmap
SMALT	0.7.6	⁹	conda install -c bioconda smalt
SNAP	1.0beta.23	¹⁰	conda install -c bioconda snap-aligner
Subread	v1.6.2	¹¹	conda install -c bioconda subread

Supplementary Table 3. Fixed effect size estimates, standard errors (se), test statistics, and p-values for the effect of the listed variables on the expected CPU run time. The parameters were estimated using the Gamma generalized linear mixed model in Equation (1). “Variable name: Level 1 vs Level 2” indicates that Level 1 is the reference level and the coefficient quantifies the increase / decrease in expected CPU run time for Level 2 over Level 1.

	Estimate	se	t stat	p-value
Intercept	0.19	0.23	0.81	4.2e-01
Year of publication	-0.7	0.09	-7.96	1.7e-15
Chain of seeds: No vs Yes	1.45	0.19	7.46	8.8e-14
Pairwise alignment: NW vs HD	1.37	0.28	4.91	9.3e-07
Pairwise alignment: NW vs Non-DP Heuristic	1.22	0.19	6.37	1.8e-10
Pairwise alignment: NW vs SW	0.78	0.2	3.83	1.3e-04
Indexing: hashing vs BWT-FM	-0.11	0.16	-0.68	5.0e-01

Supplementary Table 4. Likelihood ratio test p-values for the effect of the listed variables on the expected CPU run time. The parameters (Supplementary Table 3) were estimated using the Gamma generalized linear mixed model in Equation (1). The null Gamma generalized linear mixed model is generated as in Equation (1), but without the variable of interest. Additionally, we ran one LRT between BWT-FM tools and LAST, an aligner that does not use BWT and the FM-index by default.

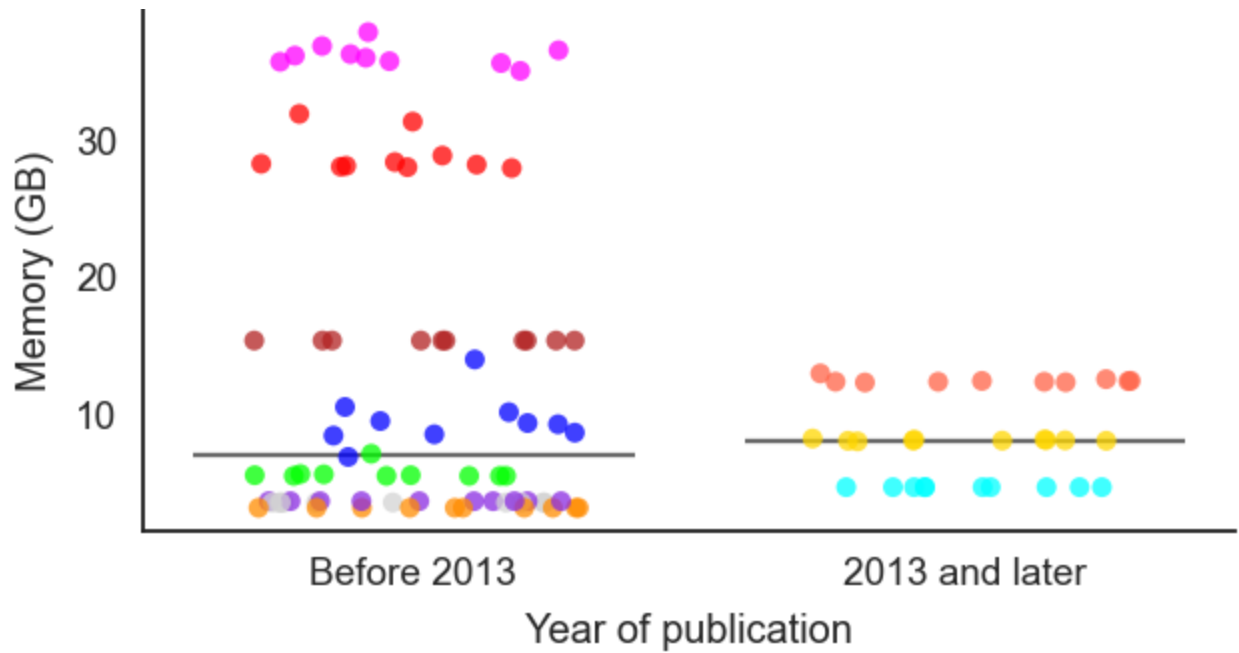
Variable of Interest	LRT p-value
Indexing	5.0e-01
Year of publication	3.7e-11
Pairwise alignment	3.7e-08
BWT-FM vs LAST	1.5e-15

Supplementary Table 5. Fixed effect size estimates, standard errors (se), test statistics, and p-values for the effect of the listed variables on the expected RAM usage. The parameters were estimated using the Gamma generalized linear mixed model in Equation (2). “Variable name: Level 1 vs Level 2” indicates that Level 1 is the reference level and the coefficient quantifies the increase / decrease in expected RAM usage for Level 2 over Level 1.

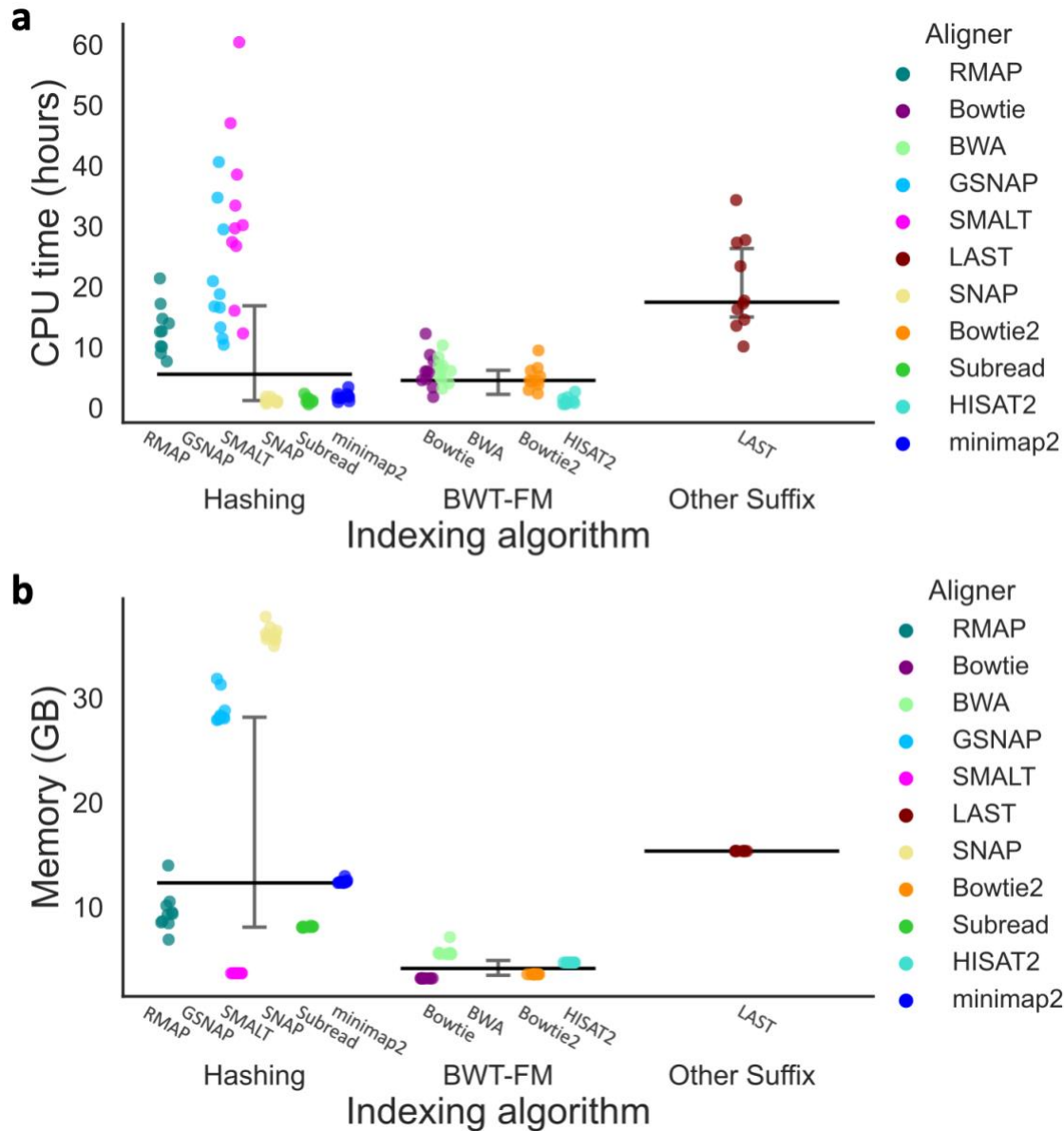
	Estimate	se	t stat	p-value
Intercept	3.41	0.51	6.67	2.2e-02
Year of publication	-0.21	0.24	-0.85	4.8e-01
Chain of seeds: No vs Yes	-0.5	0.51	-0.99	4.3e-01
Pairwise alignment: NW vs HD	-1.12	0.73	-1.53	2.7e-01
Pairwise alignment: NW vs Non-DP Heuristic	0.2	0.5	0.4	7.3e-01
Pairwise alignment: NW vs SW	-1.11	0.54	-2.06	1.8e-01
Indexing: hashing vs BWT-FM	-1.51	0.44	-3.43	7.6e-02

Supplementary Table 6. Likelihood ratio test p-values for the effect of the listed variables on the expected RAM usage. The parameters (Supplementary Table 5) were estimated using the Gamma generalized linear mixed model in Equation (2). The null Gamma generalized linear mixed model is generated as in Equation (2), but without the variable of interest. Additionally, we ran one LRT between BWT-FM tools and LAST, an aligner that does not use BWT and the FM-index by default.

Variable of Interest	LRT p-value
Indexing	2.2e-03
Year of publication	4.1e-01
Pairwise alignment	3.9e-02
BWT-FM vs LAST	3.2e-77

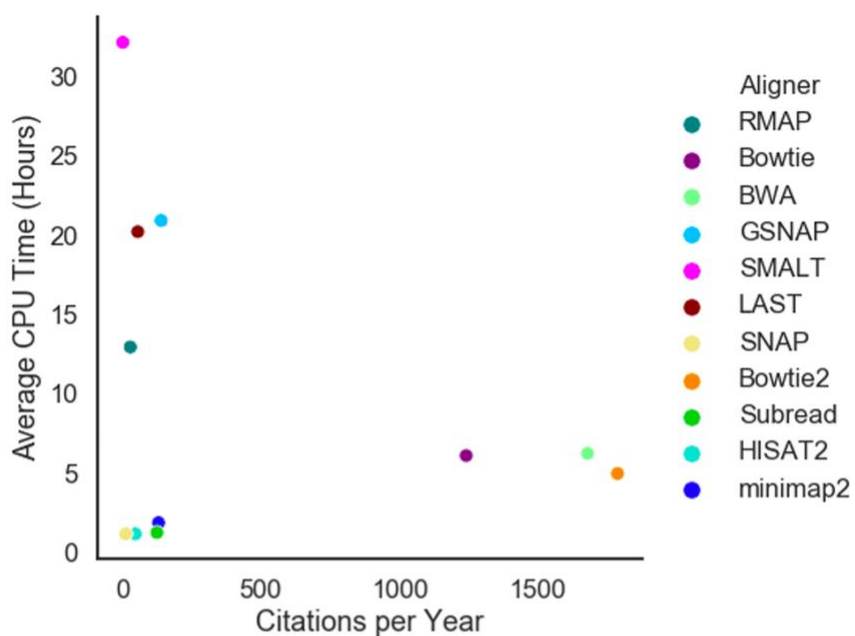


Supplementary Figure 1. The effect of year of publication on computational resources. The relative performance (RAM) of the benchmarked aligners grouped by whether the tool was released before or after long read technology was introduced (2013) and colored by individual aligners.

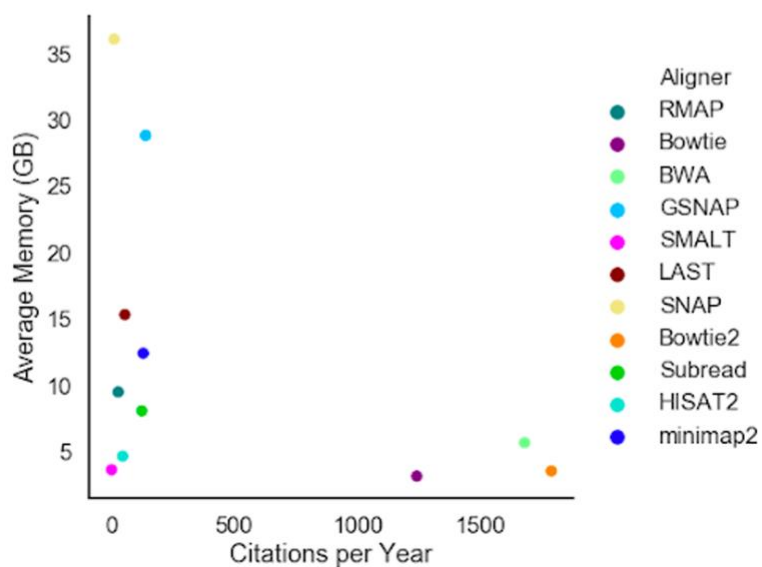


Supplementary Figure 2. Relative performance of human genome indexing performed by various read alignment tools. Tools are grouped based on the type of algorithm used for genome indexing. Tools are ordered from oldest (segemehl, 2009) to newest (HISAT2, 2019). (a) CPU time (b) RAM.

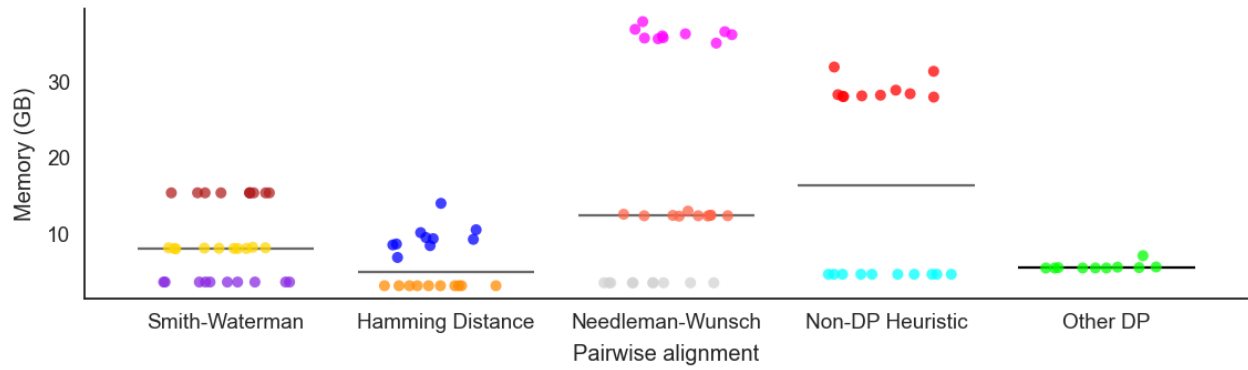
a



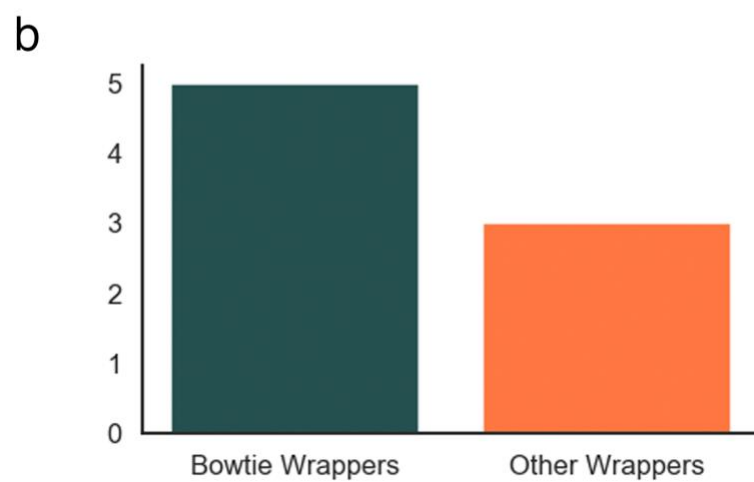
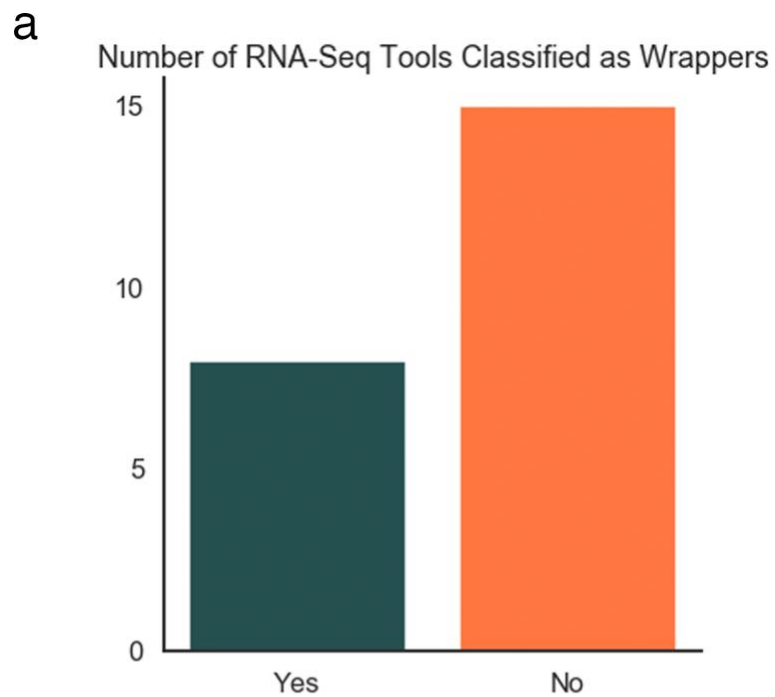
b



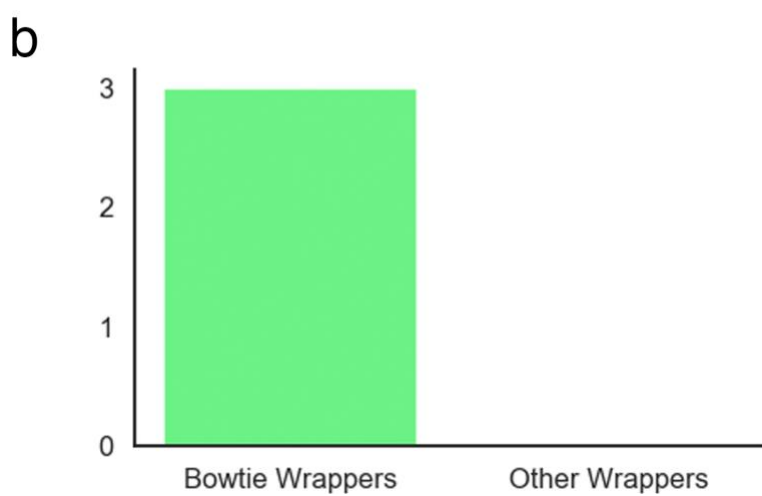
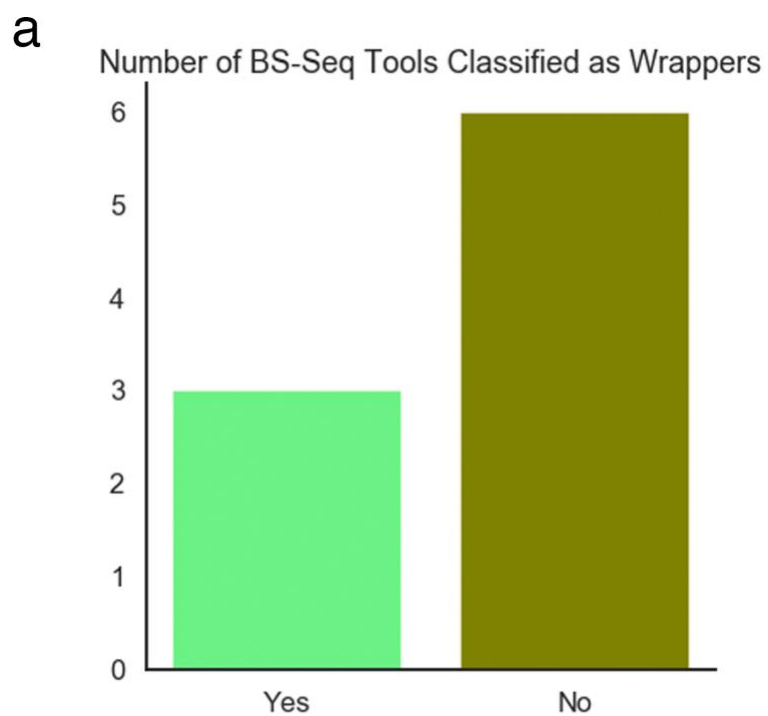
Supplementary Figure 3. Average relative performance of various read alignment tools plotted against the number of citations the tool's corresponding paper has received yearly since being published. Tools are ordered from oldest (RMAP, 2008) to newest (minimap2, 2019). (a) CPU time. (b) RAM.



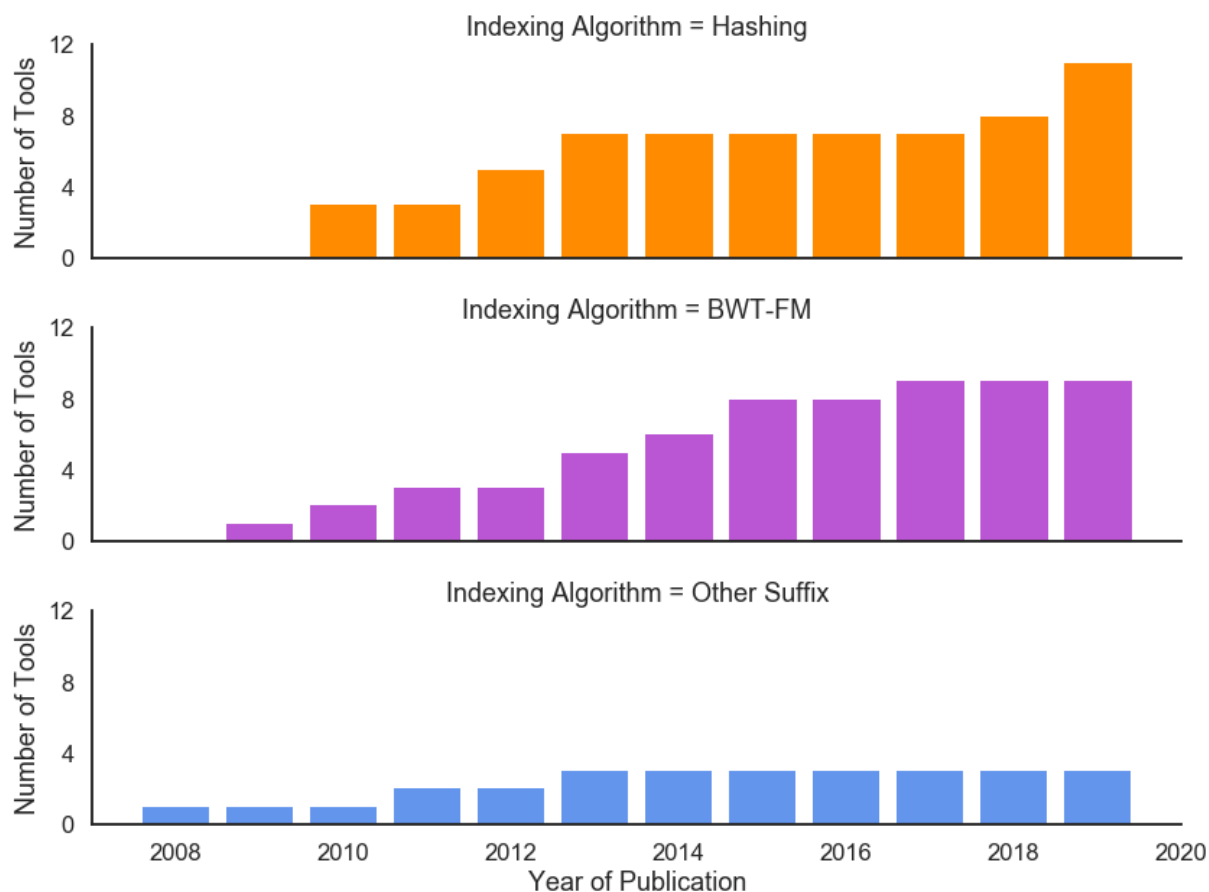
Supplementary Figure 4. The effect of pairwise alignment algorithms on computational resources. The relative performance (RAM) of the benchmarked aligners grouped by the algorithm used for pairwise alignment and colored by individual aligners.



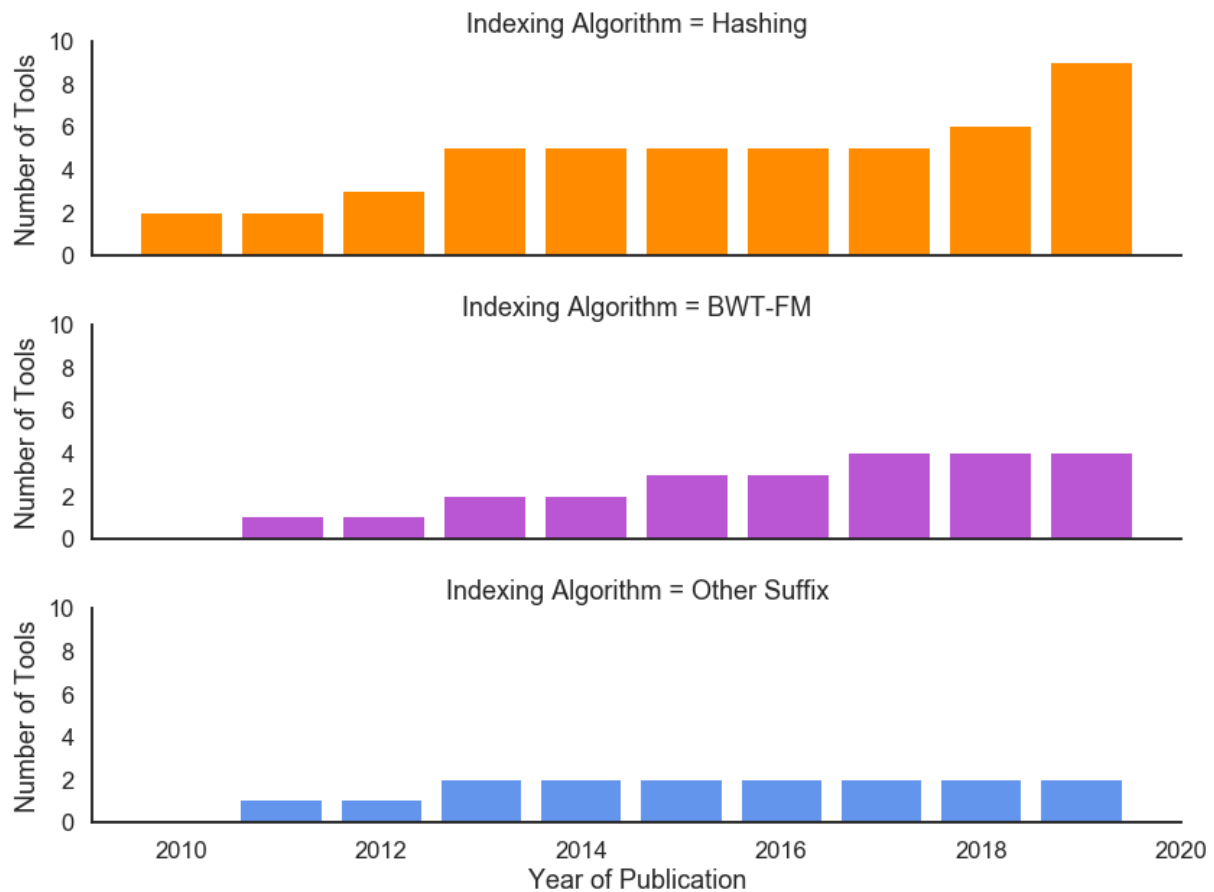
Supplementary Figure 5. (a) Bar chart showing the number of surveyed RNA-Seq tools which are wrappers of existing DNA-Seq aligners tools. (b) Bar chart showing the number of surveyed RNA-Seq tools which are wrappers of Bowtie or Bowtie2.



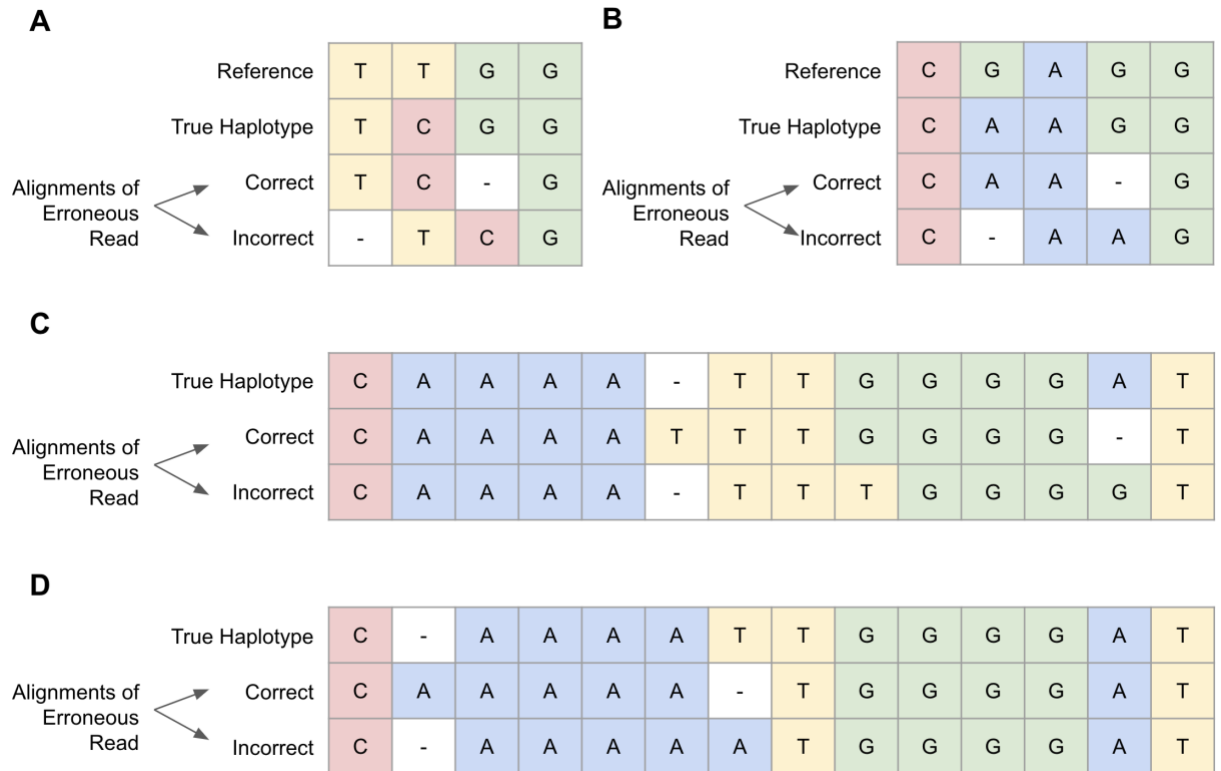
Supplementary Figure 6. (a) Bar chart showing the number of surveyed BS-Seq tools which are wrappers of existing DNA-Seq aligners tools. (b) Bar chart showing the number of surveyed BS-Seq tools which are wrappers of Bowtie or Bowtie2.



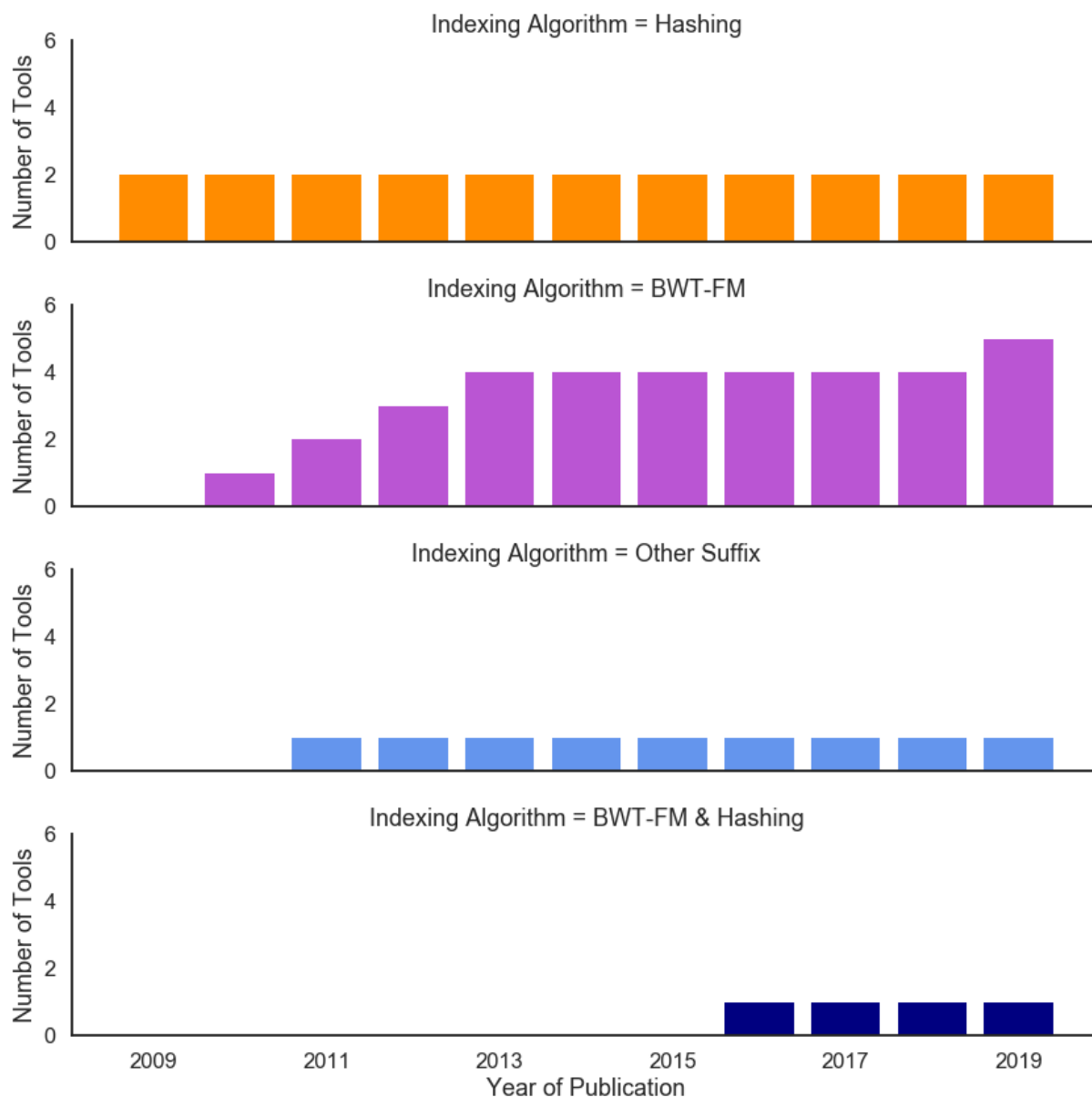
Supplementary Figure 7. Histogram showing the cumulation of surveyed RNA-Seq tools over time separated by the algorithm used for genome indexing. This includes both stand alone RNA-Seq tools and wrappers of existing DNA-Seq alignment tools.



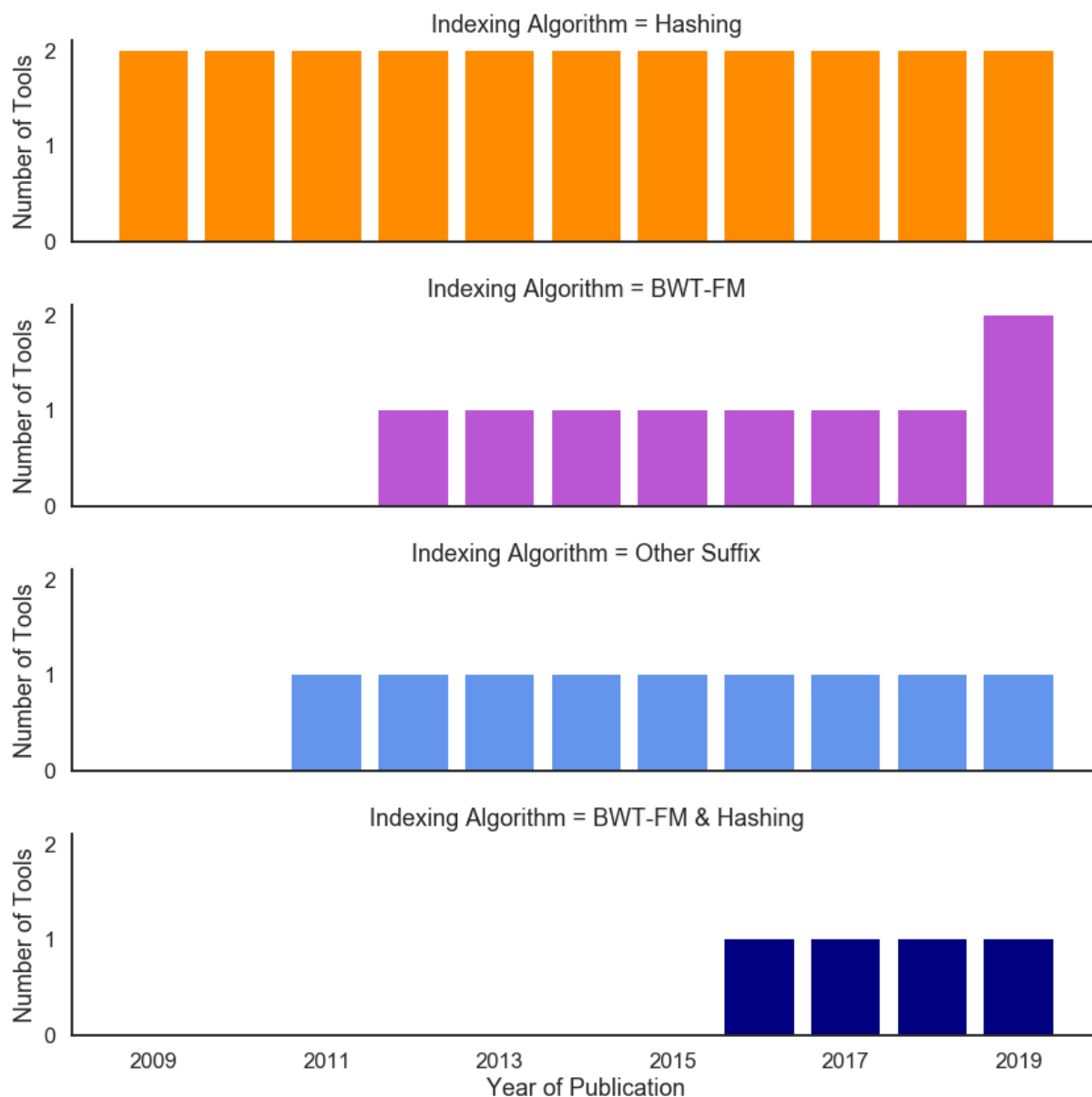
Supplementary Figure 8. Histogram showing the cumulation of surveyed RNA-Seq tools over time separated by the algorithm used for genome indexing. Only stand alone RNA-Seq aligners tools are included (not the wrappers of existing DNA-Seq aligners).



Supplementary Figure 9. Examples of erroneous alignments of Influenza A virus PacBio sequencing dataset¹². True viral haplotypes contain deletions that are inconsistently aligned to the reference. Such inconsistencies cause erroneous single-position shifts in the alignment, which in turn results in discovery of false positive single nucleotide variations. (A) and (B) Reads come from a true haplotype with the deletion with respect to the reference. Using BWA scoring method, the reads have two different alignments with the optimal score, but only the first alignment is correct. (C) and (D) Correct read alignment with the homopolymer errors should introduce an insertion and a deletion instead of “optimal” two mismatches.



Supplementary Figure 10. Histogram showing the cumulation of surveyed BS-Seq tools over time separated by the algorithm used for genome indexing. This includes both stand alone BS-Seq tools and wrappers of existing DNA-Seq alignment tools.



Supplementary Figure 11. Histogram showing the cumulation of surveyed BS-Seq tools over time separated by the algorithm used for genome indexing. Only stand alone BS-Seq aligners tools are included (not the wrappers of existing DNA-Seq aligners).

Supplementary Note 1

We evaluate the effect of indexing on the end-to-end execution time of today's read alignment algorithms. We align a single read of length 100 bp to the human reference genome (hg38) using BWA-MEM (with *-a* parameter selected to report all mapping locations). Building the index for the human reference genome takes 3,476 seconds. The read alignment step using BWA-MEM takes only 3.4 seconds after building the index for the human reference genome. Now we want to perform brute-force read alignment for the same read sequence and the same reference genome that we use for the BWA-MEM experiment. We divide the human reference genome into about 3.3 billion sequences, each of which is 100 bp long. That is, the first sequence is the first 100 bp of the reference genome and the second sequence is the segment that starts from the second bp of the reference genome and ends at the 101th bp and so forth for the other sequences. We then use Edlib's global alignment tool (DP-based pairwise alignment) to check the similarity of the read sequence with each of the 3.3 billion generated sequences. We observe that Edlib takes about 24,200 seconds to complete the brute-force read alignment approach. This means that the indexing technique (and probably other filtering heuristics) used in BWA-MEM saves the execution time of read alignment by at least 7,100X. If we include the time needed to build the index in the total time of read alignment, then BWA-MEM is only 7X faster than the brute-force read alignment approach. Note that indexing the reference genome is performed only once for each reference genome.

Supplementary Note 2

In our review, we define read alignment as a three-step process, which includes indexing, global positioning, and pairwise alignment. In this case, pairwise alignment is considered to be performed between a read and a section of the reference determined by global positioning. Alternatively, the entire process can be viewed as local alignment with respect to the reference, and global alignment with respect to the read. In this formulation, the read is aligned end-to-end to the best substring in the reference and is expressed as semi-global alignment¹³.

We have simplified pairwise alignment into overarching algorithm classifications like Smith-Waterman or Needleman-Wunsch, but tools that use dynamic programming can be classified into subcategories that are beyond the scope of this review. For example, read alignment algorithms can choose to be gapless (ignoring some variants), compute edit distance (the minimum number of edits needed to convert one string into the other), or use an affine gap penalty where variants are weighted differently based on their length. It is also worth noting that BWT-based tools do not use seeding in the traditional sense, and seed classification might be performed differently.

Supplementary Note 3

We first built the index data, then ran the alignment procedure and extracted the data in bam format. Some tools do not provide the output in bam format, so in this case we used samtools toolkit to convert sam output to bam output.

To install samtools from conda: `conda install -c bioconda samtools`

1) Bowtie2

Build index:

`bowtie2-build <reference_in> <index_basename>`

*reference_fasta: fasta file of reference genome

*index_basename: write index data to files with this basename

Mapping WGS data:

`bowtie2 -x <index_basename> -1 <r1_fastq> -2 <r2_fastq> | samtools view -bS - > output.bam`

*r1_fastq, r2_fastq: fastq files of the paired end reads

2) Bowtie

Build index:

`bowtie-build <reference_in> <index_basename>`

*reference_in: fasta file of reference genome

*index_basename: write index data to files with this basename

Mapping WGS data:

`bowtie -S <index_basename> -1 <r1_fastq> -2 <r2_fastq> | samtools view -bS - > output.bam`

3) BWA

Build index:

`bwa index <reference_fasta>`

Mapping WGS data:

`bwa mem <reference_fasta> <r1_fastq> <r2_fastq> | samtools view -bS - > output.bam`

4) GSNAP

Build index:

```
gmap_build -D <destination_directory_path> -d <genome_name> <reference_fasta>
```

Mapping WGS data:

```
gsnap -D <destination_directory_path> -d <genome_name> <r1_fastq> <r2_fastq> -A sam |
```

```
samtools view -bS - > output.bam
```

5) HISAT2

Build index:

```
hisat2-build <reference_fasta> <index_basename>
```

Mapping WGS data:

```
hisat2 -q -x <index_basename> -1 <r1_fastq> -2 <r2_fastq> | samtools view -bS - > output.bam
```

*-q: input as fastq file

6) LAST

Build index:

```
lastdb -uNEAR -R01 <index_basename> <reference_fasta>
```

*-uNEAR and -R01 optional

Mapping WGS data:

```
lastal -Q1 <index_basename> <r1_fastq> <r2_fastq> | last-split > output.maf
```

*Q1: fastq-sanger format

7) minimap2

Build index:

```
Minimap2 -d <index_file> <reference_fasta>
```

* index file with “.mmi” extension

Mapping WGS data:

```
Minimap2 -a <index_file> <r1_fastq> | samtools view -bS - > output.bam
```

8) RMAP

```
rmap <read_fastq> -c <reference_fasta> -o output.sam | samtools view -bS - > output.bam
```

9) SMALT

Build index:

```
smalt index [options] <index_name> <reference_fasta>
```

Mapping WGS data:

```
smalt map <index_name> <r1_fastq> <r2_fastq> | samtools view -bS - > output.bam
```

10) SNAP

Build index:

```
snap-aligner <index_name> <reference_fasta> <index_dir_name>
```

Mapping WGS data:

```
snap-aligner paired <index_dir_name> <r1_fastq> <r2_fastq> -o output.bam
```

11) Subread

Build index:

```
subread-buildindex -o <index_name> <reference_fasta>
```

Mapping WGS data:

```
subread-align -t 1 -i <index_name> -r <r1_fastq> -R <r1_fastq> -o output.bam
```

Supplementary Note 4

While a typical seed is a contiguous subsequence, a spaced seed contains in its sequence characters from a subsequence of the reference genome while ignoring the other characters of the same subsequence. Spaced seeds increase alignment sensitivity and enable hash tables to provide hits for both exact and inexact matches by ignoring certain bases of the seed. This approach was pioneered by PatternHunter^{4,14–16} in 2002 and has been adopted by 14 tools. A majority of the tools using spaced seeds are designed for short read technologies (Table 1). Spaced seeds can also be used in long read alignment to tolerate high error rates¹⁷. Another approach to account for the error rate of sequencing technologies involves generating seeds as prefixes of the read sequence. Generating the prefixes of the reads—as opposed to generating the suffixes—allows the read alignment algorithm to tolerate an increased error rate towards the end of a read¹⁸. Other methods generate both suffix seeds and prefix seeds in order to tolerate large genetic variations¹⁹.

Instead of choosing a large number of seeds from each read, read alignment algorithms can choose only a small number of seeds that are apart from each other. This approach also allows larger genetic variations and sequencing errors that are located between every two adjacent seeds²⁰. Most read alignment algorithms that follow this approach try to limit the number of differences that are located at the gaps in order to avoid aligning a read to highly dissimilar regions in the reference genome. This approach can be performed using seed extension followed by seed chaining. First, after finding a matching seed shared between a read and the reference genome, the read alignment algorithm extends the matching seed in both directions until there

are no more exact matches (such extended seeds are called maximal exact matches (MEMs)²¹). Second, the read alignment algorithm examines the gaps between every two adjacent extended seeds in the reference genome using a pairwise alignment algorithm^{22,23} to construct a longer chain of these adjacent extended seeds²⁴. The pairwise alignment can be performed end-to-end (e.g., global alignment) for two sequences of the same length^{22,23}, or by using a local alignment algorithm^{11,25,26}, where subsequences of the two given sequences are aligned. The two sequences can also be examined using a Hamming distance algorithm in cases where insertions or deletions are not allowed²⁷. This seed chaining approach can also be applied to non-hashing-based read alignment algorithms, such as Bowtie2²⁸ and BWA-MEM²⁹. We observe that 54 read alignment algorithms out of the 107 surveyed alignment algorithms use a seed chaining approach.

Supplementary Note 5

Modern read alignment algorithms (e.g., Hobbes³⁰, Hobbes2³¹, Bitmapper³², mrFAST³³, RazerS³⁴) develop heuristics that quickly decide whether or not the computationally expensive DP calculation is needed—if not, significant time is saved by avoiding DP calculation. Such heuristics are called *pre-alignment filters*^{35–39}, and they approximate the total number of differences between two sequences to determine if this count is greater than a threshold (Figure 1e). If so, these heuristics decide that the verification calculation is not needed due to high dissimilarity between the two sequences. Verification algorithms can also be accelerated using specialized or general-purpose hardware accelerators such as multi-core processors^{40–}

Supplementary Note 6

To obtain the nucleotide count in all bacterial genomes possessed by NCBI, we utilized the tool RepoPhlAn(<https://bitbucket.org/nsegata/repophlan>) to download via <ftp.ncbi.nih.gov> all genomes contained in the `genomes/all` subdirectory. Taxonomic identifiers were used to identify bacterial genomes and subsequently obtain a nucleotide count.

```
# obtain RepoPhlAn
```

```
wget https://bitbucket.org/nsegata/repophlan/get/03f614c13cf0.zip
```

```
unzip 03f614c13cf0.zip
```

```
cd 03f614c13cf0
```

```
# run RepoPhlAn
```

```
./run.sh # this can take upwards of 5 days to complete this step
```

```
cd out/microbes_<time_stamp>/fna
```

```
# count number of bacterial nucleotides
```

```
nohup ls -U | xargs -P 15 -I{} sh -c "bzcat {} | grep -v '>' | wc -m" | awk '{sum+= $1} END {print sum}' > ~/bacteria_bp_count.txt
```

```
# 676153484835
```

The human genome build GRCh38 was obtained from NCBI via <ftp> and nucleotides counted in the following way:

```
# download the human genome
```

```
wget -r https://ftp.ncbi.nih.gov/genomes/Homo\_sapiens/Assembled\_chromosomes/seq/*
```

```
# select just the fasta files
```

```
cd ftp.ncbi.nih.gov/genomes/Homo_sapiens/Assembled_chromosomes/seq/
```

```
ls | grep -v "\.fa\." | xargs -l{} rm {}
```

```
#Uncompress
```

```
ls | xargs -l{} gunzip {}
```

```
# count nucleotides
```

```
ls *.fa | xargs -l{} sh -c " grep -v '>' {} | wc -m" | awk '{sum+=$1}END{print sum}' >
```

```
~/human_bp_count.txt
```

```
#3303852965
```

```
# compare the two
```

```
echo "`cat ~/bacteria_bp_count.txt` / `cat ~/human_bp_count.txt`" | bc -l
```

```
204.65604613702898246260
```

Supplementary Materials

Install the read alignment tools

We have selected tools available on bioconda and have installed them using the following commands (Table S1).

Public Sequence Data

We used 10 WGS datasets for comparing the tools listed in Table S1. The SRA run accession numbers of the 10 datasets are as follows: ERR009309, ERR013127, ERR013138, ERR045708, ERR050158, ERR162843, ERR181410, ERR183377, SRR061640, and SRR360549. To download this data, we used the SRA toolkit which is available as a conda package.

Here are the commands that we used for the downloading process:

- To download sra toolkit: *conda install -c bioconda sra-tools*
- To download fastq files:
 - For single end fastq files: *fastq-dump <SRA_id>*
 - For paired end fastq files: *fastq-dump --split-files <SRA_id>*

Compare the performance of the read alignments

We recorded CPU time and RAM usage to compare the read alignment tools.

Tools were run in the UCLA's Shared Hoffman2 Cluster.

Command that we used to submit our jobs in the cluster:

```
qsub -o <logfiles/> -e <logfiles/> -m bea -cwd -V -N <name_job> -l  
h_data=32G,highp,time=24:00:00 <exe_script>
```

*-m bea: define mailing rules

- b- start time of the job
- e- end time of the job
- a- time when the job is aborted

-cwd: changes the directory to where your executed file is, all log output will be created in this file unless you specify another directory (see command above output logs and error logs are directed to a folder named logfiles)

-V: export environment variables

-N: give a name to the submitted job

-l h_date: resource allocation

-l highp: submission of high priority jobs

-l time: job running time

Statistical analyses

We model expected CPU time c_{ij} across all algorithms i and datasets j using the following gamma generalized linear mixed model regression

$$\log(E(c_{ij})) = \alpha + a_j + \beta_1 \times \text{Chain_of_seeds}_{ij} + \beta_2 \times \text{Indexing}_{ij} + \beta_3 \times \text{Year_of_publication}_{ij} + \beta'_4 \times \text{Pairwise_alignment}_{ij} \quad (1)$$

where α is the intercept and $a_j \sim N(0, \sigma_j)$ is a data-level random intercept modelling the shared noise within each data set. β_1 is the effect of the *Chain_of_seeds* where *Chain_of_seeds* is coded as zero for no and one for yes. β_2 is the effect of *Indexing*, where *Indexing* is coded as 0 for BWT-FM and 1 for hashing or suffix array, depending on the group being compared to BWT-FM. β_3 is the effect of *Year_of_publication*, coded as a continuous variable of year scaled to have mean zero and variance one. β_4 is a vector with the effects of *Pairwise_alignment*, where *Pairwise_alignment* is a matrix of indicator variables for HD, Non-DP Heuristic, and SW algorithms, making NW the reference category. Parameter estimates are provided in (Table S3). We use a likelihood ratio test to test the effect of each variable discussed, e.g. year of publication or indexing, on the CPU time.

We use a similar model for the median across all datasets of the expected RAM usage med_mem_i , i.e.

$$\log(E(\text{med_mem}_i)) = \alpha + \beta_1 \times \text{Chain_of_seeds}_{ij} + \beta_2 \times \text{Indexing}_{ij} + \beta_3 \times \text{Year_of_publication}_{ij} + \beta_4 \times \text{Pairwise_alignment}_{ij} \quad (2)$$

Parameter estimates are provided in (Table S4). Note that, as memory usage does not vary considerably within algorithms across data sets, we use the median expected RAM usage across all datasets for each algorithm.

References

1. Langmead, B. & Salzberg, S. L. Fast gapped-read alignment with Bowtie 2. *Nature Methods* vol. 9 357–359 (2012).
2. Langmead, B., Trapnell, C., Pop, M. & Salzberg, S. L. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.* **10**, R25 (2009).
3. Li, H. & Durbin, R. Fast and accurate long-read alignment with Burrows–Wheeler transform. *Bioinformatics* **26**, 589–595 (2010).
4. Wu, T. D. & Nacu, S. Fast and SNP-tolerant detection of complex variants and splicing in short reads. *Bioinformatics* **26**, 873–881 (2010).
5. Kim, D., Paggi, J. M., Park, C., Bennett, C. & Salzberg, S. L. Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype. *Nature Biotechnology* vol. 37 907–915 (2019).
6. Kiełbasa, S. M., Wan, R., Sato, K., Horton, P. & Frith, M. C. Adaptive seeds tame genomic sequence comparison. *Genome Res.* **21**, 487–493 (2011).
7. Li, H. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* **34**, 3094–3100 (2018).
8. Kim, D. *et al.* TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biol.* **14**, R36 (2013).

9. Hannes Ponsting, Z. N. SMALT - A New Mapper for DNA Sequencing Reads. (2010).
10. Zaharia, M. *et al.* Faster and More Accurate Sequence Alignment with SNAP. *arXiv [cs.DS]* (2011).
11. Liao, Y., Smyth, G. K. & Shi, W. The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote. *Nucleic Acids Res.* **41**, e108 (2013).
12. Artyomenko, A. *et al.* Long Single-Molecule Reads Can Resolve the Complexity of the Influenza Virus Composed of Rare, Closely Related Mutant Variants. *J. Comput. Biol.* **24**, 558–570 (2017).
13. Brudno, M. *et al.* Glocal alignment: finding rearrangements during alignment. *Bioinformatics* **19 Suppl 1**, i54–62 (2003).
14. Egidi, L. & Manzini, G. Better spaced seeds using Quadratic Residues. *Journal of Computer and System Sciences* vol. 79 1144–1155 (2013).
15. Rizk, G. & Lavenier, D. GASSST: global alignment short sequence search tool. *Bioinformatics* **26**, 2534–2540 (2010).
16. Ma, B., Tromp, J. & Li, M. PatternHunter: faster and more sensitive homology search. *Bioinformatics* **18**, 440–445 (2002).
17. Sović, I. *et al.* Fast and sensitive mapping of nanopore sequencing reads with GraphMap. *Nat. Commun.* **7**, 11307 (2016).
18. Kircher, M., Heyn, P. & Kelso, J. Addressing challenges in the production and analysis of illumina sequencing data. *BMC Genomics* **12**, 382 (2011).
19. Emde, A.-K. *et al.* Detecting genomic indel variants with exact breakpoints in single- and paired-end sequencing data using SplazerS. *Bioinformatics* **28**, 619–627 (2012).

20. Kloosterman, W. P. *et al.* Characteristics of de novo structural changes in the human genome. *Genome Res.* **25**, 792–801 (2015).
21. Delcher, A. L., Phillippy, A., Carlton, J. & Salzberg, S. L. Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acids Res.* **30**, 2478–2483 (2002).
22. Slater, G. S. C. & Birney, E. Automated generation of heuristics for biological sequence comparison. *BMC Bioinformatics* **6**, 31 (2005).
23. Siragusa, E., Weese, D. & Reinert, K. Fast and accurate read mapping with approximate seeds and multiple backtracking. *Nucleic Acids Res.* **41**, e78 (2013).
24. Mäkinen, V. & Sahlin, K. Chaining with overlaps revisited. (2020).
25. Chen, S., Wang, A. & Li, L. M. SEME: A Fast Mapper of Illumina Sequencing Reads with Statistical Evaluation. *Lecture Notes in Computer Science* 14–29 (2013) doi:10.1007/978-3-642-37195-0_2.
26. David, M., Dzamba, M., Lister, D., Ilie, L. & Brudno, M. SHRiMP2: sensitive yet practical SHort Read Mapping. *Bioinformatics* **27**, 1011–1012 (2011).
27. Hach, F. *et al.* mrsFAST: a cache-oblivious algorithm for short-read mapping. *Nat. Methods* **7**, 576–577 (2010).
28. Langmead, B. & Salzberg, S. L. Fast gapped-read alignment with Bowtie 2. *Nat. Methods* **9**, 357–359 (2012).
29. Li, H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv [q-bio.GN]* (2013).
30. Ahmadi, A. *et al.* Hobbes: optimized gram-based methods for efficient read alignment. *Nucleic Acids Res.* **40**, e41 (2012).

31. Kim, J., Li, C. & Xie, X. Improving read mapping using additional prefix grams. *BMC Bioinformatics* **15**, 42 (2014).
32. Cheng, H., Jiang, H., Yang, J., Xu, Y. & Shang, Y. BitMapper: an efficient all-mapper based on bit-vector computing. *BMC Bioinformatics* **16**, 192 (2015).
33. Alkan, C. *et al.* Personalized copy number and segmental duplication maps using next-generation sequencing. *Nat. Genet.* **41**, 1061–1067 (2009).
34. Weese, D., Emde, A.-K., Rausch, T., Döring, A. & Reinert, K. RazerS--fast read mapping with sensitivity control. *Genome Res.* **19**, 1646–1654 (2009).
35. Alser, M., Hassan, H., Kumar, A., Mutlu, O. & Alkan, C. Shouji: A Fast and Efficient Pre-Alignment Filter for Sequence Alignment. *Bioinformatics* (2019) doi:10.1093/bioinformatics/btz234.
36. Alser, M. *et al.* GateKeeper: a new hardware architecture for accelerating pre-alignment in DNA short read mapping. *Bioinformatics* **33**, 3355–3363 (2017).
37. Alser, M., Mutlu, O. & Alkan, C. MAGNET: Understanding and Improving the Accuracy of Genome Pre-Alignment Filtering. *arXiv [q-bio.GN]* (2017).
38. Kim, J. S. *et al.* GRIM-Filter: Fast seed location filtering in DNA read mapping using processing-in-memory technologies. *BMC Genomics* **19**, 89 (2018).
39. Alser, M., Shahroodi, T., Gómez-Luna, J., Alkan, C. & Mutlu, O. SneakySnake: A Fast and Accurate Universal Genome Pre-Alignment Filter for CPUs, GPUs, and FPGAs. *Bioinformatics* (2020) doi:10.1093/bioinformatics/btaa1015.
40. Zhang, J. *et al.* BGSA: A Bit-Parallel Global Sequence Alignment Toolkit for Multi-core and Many-core Architectures. *Bioinformatics* (2018) doi:10.1093/bioinformatics/bty930.

41. Turakhia, Y., Goenka, S. D., Bejerano, G. & Dally, W. J. Darwin-WGA: A Co-processor Provides Increased Sensitivity in Whole Genome Alignments with High Speedup. *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)* (2019) doi:10.1109/hpca.2019.00050.
42. Cali, D. S. *et al.* GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis. in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* 951–966 (2020).
43. Alser, M. *et al.* Accelerating Genome Analysis: A Primer on an Ongoing Journey. *IEEE Micro* vol. 40 65–75 (2020).

1st round

Reviewer 1

Is the topic of the article timely and of interest to a wide range of readers?

Yes

Is the article well written and presented in a logical way?

Yes

Do the authors cover the relevant literature in an accurate and balanced way?

Yes

Do the authors provide a useful synthesis of the topic?

Yes

Do the authors provide insightful discussion of the future directions for the field?

Yes: Sequence alignment is the fundamental step for a myriad of downstream applications in genomics. Although this has been a subject of research for over multiple decades, given the rapid growth of new sequencing technologies and applications, a detailed understanding of the use case for a specific aligner should be clear to the community. There are many open source aligners that are available at one's disposal for analyzing a specific dataset. Authors of this manuscript laid out specific building blocks for the existing aligners and performed a rigorous analysis for the time and memory performance. Subsequently they did statistical modeling to assess the contribution of a particular attribute that goes into the final performance of the aligner. As I outlined in my review, there is scope of improvement in the manuscript to make it acceptable. But in my opinion the authors did a great job in setting up a principle way for understanding and building future aligners.

Do any tables, figures or boxes that are present enhance the manuscript?

Yes

Would the manuscript benefit from any additional tables, figures or boxes?

Yes

Comments to author

The manuscript "Technology dictates algorithms: Recent developments in read alignment" by Alser et al. presents a rigorous description of the ever-growing field of "biological sequence alignment" that serves as a core engine for multitudes of downstream applications such as gene expression quantification, differential expression, etc. Alser et al. surveyed 107 different alignment algorithms developed in the span of 32 years from 1988 to 2020. The manuscript rightfully captured various heterogeneity of biological sequences, from protocol type to length of the reads to organism-specific challenges.

Given the astronomic growth in the publicly available sequence datasets and the alignment algorithms, the manuscript is of significant importance to analyze and dissect what algorithm is best suited for a specific purpose. Additionally, the manuscript provides a historical perspective that is crucial in understanding the evolution of the alignment algorithms. I want to particularly commend the authors' efforts to reproduce the plots and tables via the open-source GitHub repository.

While the paper nicely ties together the subject matter, I have the following concerns about the present manuscript,

-Major Concerns-

The manuscript quite rightly outlined the challenges of a modern-day aligner concerning the type of sequences it is designed to tackle. It describes that "Error rate", "Genome Coverage", "Global position" and "Local pairwise alignment" etc., play a crucial role in determining the usefulness of an aligner. It seems the real challenge is to find out the right middle ground between specificity: keeping a set of heuristics to accommodate possible approximate matches, sensitivity: being too

conservative about accepting an alignment. Leaning towards either direction can lead to undesirable outcomes. Therefore the algorithm has to be tailor-made for the specific application.

My primary concern with the manuscript is that it does not provide an in-depth analysis of how these different attributes affect the accuracy of the aligners. While the rigorous time and memory benchmark are crucial for choosing one method over the other, the fundamental element of choice lies in the alignment's actual accuracy. To be specific, a user selects an aligner by judging it on the scale of specificity and sensitivity.

The manuscript bases the results on a range of real data samples, reflecting the performance across a heterogeneous set of datasets. For benchmarking the accuracy of these tools, I would highly recommend using simulated read datasets incorporating different parameters outlined in the manuscript. Varying "Error rate", "Genome Coverage", "Global position", "Local pairwise alignment", "Genomic variants", "Read length" as tunable parameters, one could single out the effect of each of these attributes for a particular alignment algorithm. Available DNA-seq/RNA-seq simulators (e.g., ART or more recent <https://github.com/schmeing/ReSeq>) for single-end, paired-end reads sort reads could be used for benchmarking DNA-seq specific aligners. Additionally, PacBio long-read simulators (such as <https://github.com/rrwick/Badread>) can be used to generate long reads with realistic error profiles. It is not necessary, but the authors can also perform a similar analysis or point to an already existing benchmark for metagenomic analysis. A realistic measure of accuracy in the alignment is quite straightforward, as it depends on the reads that are correctly mapped back to the location of origin, determining true positive/ false positives, etc. Although it is understandable under a high variation rate, it might be infeasible to map a read to a unique location correctly, but those cases will be infrequent.

Some similar classification is done for time and memory usage can also be implemented for the accuracy metric. All the design principles that are discussed in the manuscripts, such as hashing to BWT-FM based indexing or using a mixed alignment scheme to form the final set of alignment, can have a recognizable effect on accuracy.

In my opinion, such a measure will increase the usefulness of the manuscript to a great extent.

-Other concerns-

1. The plots of figure 4 correspond to the performance of 11 different alignment methodologies on various experiments. I guess the multiple points with the same color for each method refer to the experiments. It would be helpful to mention the legend in the caption.
2. The corresponding paragraph referring to Figure 4(c) mentions that there is not much difference between the Hashing based methods and the BWT-FM based methods. It seems SMALT is a hashing based method that stands out from other hashing based methods in terms of run time, while SNAP is the hashing based method that takes the highest memory. It will be interesting to discuss the reason for such a performance difference.
3. The colors used in Figure 4 are not very distinctive to me. I had a hard time identifying the tools back, especially when the colors are too similar. Given that there are multiple experiments for one tool, it would help identify a tool by an arrow whenever possible (specifically for figure 4f). For example, the text in the manuscript mentions that "Despite the overall longer runtime of Hamming-distance-based methods, the latest hashing based tool hashing based too (e.g., HISAT2) provide a comparable.", unfortunately, I could not identify HISAT2 back from the diagram. Using a different legend might be useful.

4. Table 2 classifies the alignment algorithms from a practical standpoint based on metrics such as "ease of implementation" etc. I am curious how these metrics are computed.

5. "; the shared prefix between the read and the genome is stored only once and used by all the reads with that prefix." - I could not comprehend this sentence correctly; the suffix tree should be independent of the reads; it's part of the index made on the reference sequence. An explanation would be beneficial.

6. "Unlike a hash table, a suffix tree allows searching for both the exact and inexact match seeds by walking through.."

The initial seeding strategy can have a variable length of k-mers for selecting prefixes from the suffix array. Still, I could not understand how the suffix tree built on the index can allow inexact search if an additional set of heuristics is not employed.

7. β is used as a coefficient for the generalized linear mixed model regression. The values themselves do not express a lot, in my opinion (except their signs). It would be really helpful if some explanation can be added with positive and negative values for the corresponding coefficient.

8. Page 23 paragraph starting from line 29 describes the standard process for transcriptome quantification using DNA-seq aligners. The following paragraph briefly describes the use of light-weight mappers to do transcript level read assignment. There are some issues with the generalization in the description. The commonly used pseudo-alignment-based methods, such as Kallisto, uses De-Bruijn graphs. In contrast, the other tool, Salmon, uses k-mer hash in conjunction with a Suffix Array index (using mapper such as RapMap) with additional heuristics to speed up the mapping process.

Moreover, Salmon can provide mapping locations and a mapping score. However, as the authors rightly pointed out, none of these tools provide CIGAR strings or any other alignment profiles for further analysis. I would suggest the authors to rewrite the sections with a more detailed treatment of the individual methods given both of the tools are highly cited and used.

9. Referring to page 15 line 4: it will be helpful to elaborate on the kind of heuristics that the aligner algorithms used, such as bidirectional jump, skipping, hamming distance thresholding, etc. As with other aspects of the alignment pipeline, the seed extension heuristic can have a profound effect on the alignment results.

Reviewer 2

Is the topic of the article timely and of interest to a wide range of readers?

Yes

Is the article well written and presented in a logical way?

Yes

Do the authors cover the relevant literature in an accurate and balanced way?

Yes

Do the authors provide a useful synthesis of the topic?

Yes

Do the authors provide insightful discussion of the future directions for the field?

Yes: The authors indeed provided insightful discussion of the future directions for the field. **Do any tables, figures or boxes that are present enhance the manuscript?**

Yes

Would the manuscript benefit from any additional tables, figures or boxes?

Yes

Comments to author

This is a very-well written review paper with broad vision and unique perspective. The algorithmic details of the alignments methods were clearly explained to a proper depth, and all methods are well categorized based on the underlying algorithms used. The advantages and limitations of each category are convincingly described and well illustrated with tables and figures. I also like the discussion about the challenges and opportunities for reads alignments with the recent advancement of long-reads technologies

Two suggestions:

1, 10 WGS samples were used to evaluate the chosen methods. I believe they are all short-reads data. If so, it would be more helpful if some long-reads data can be included to evaluate the mentioned long-reads aligners.

2, In the GitHub repo

(https://github.com/Mangul-Lab-USC/review_technology_dictates_algorithms) users were directed to the repository's wiki for more information. But I didn't find the wiki page. A detailed description on how to use the provided scripts to reproduce the experimental results in the manuscript would be very helpful to the community.

Authors' response

Reviewer 1

The manuscript quite rightly outlined the challenges of a modern-day aligner concerning the type of sequences it is designed to tackle. It describes that "Error rate", "Genome Coverage", "Global position" and "Local pairwise alignment" etc., play a crucial role in determining the usefulness of an aligner. It seems the real challenge is to find out the right middle ground between specificity: keeping a set of heuristics to accommodate possible approximate matches, sensitivity: being too conservative about accepting an alignment. Leaning towards either direction can lead to undesirable outcomes. Therefore the algorithm has to be tailor-made for the specific application.

My primary concern with the manuscript is that it does not provide an in-depth analysis of how these different attributes affect the accuracy of the aligners. While the rigorous time and memory benchmark are crucial for choosing one method over the other, the fundamental element of choice lies in the alignment's actual accuracy. To be specific, a user selects an aligner by judging it on the scale of specificity and sensitivity. The manuscript bases the results on a range of real data samples, reflecting the performance across a heterogeneous set of datasets. For benchmarking the accuracy of these tools, I would highly recommend using simulated read datasets incorporating different parameters outlined in the manuscript. Varying "Error rate", "Genome Coverage", "Global position", "Local pairwise alignment", "Genomic variants", "Read length" as tunable parameters, one could single out the effect of each of these attributes for a particular alignment algorithm.

Available DNA-seq/RNA-seq simulators (e.g., ART or more recent

[https://urldefense.com/v3/__https://github.com/schmeing/ReSeq__;!!Llr3w8kk_Xxm!6t79Yi8uYPnGb6lbOG_SMZ1pLX-jpHO-ikoh20FtEGuRbw39nh20n3A68d_Q5g\\$](https://urldefense.com/v3/__https://github.com/schmeing/ReSeq__;!!Llr3w8kk_Xxm!6t79Yi8uYPnGb6lbOG_SMZ1pLX-jpHO-ikoh20FtEGuRbw39nh20n3A68d_Q5g$)) for single-end, paired-end reads sort reads could be used for benchmarking DNA-seq specific aligners.

Additionally, PacBio long-read simulators (such as

[https://urldefense.com/v3/__https://github.com/rrwick/Badread__;!!Llr3w8kk_Xxm!6t79Yi8uYPnGb6lbOG_SMZ1pLX-jpHO-ikoh20FtEGuRbw39nh20n3Dgt6WhPg\\$](https://urldefense.com/v3/__https://github.com/rrwick/Badread__;!!Llr3w8kk_Xxm!6t79Yi8uYPnGb6lbOG_SMZ1pLX-jpHO-ikoh20FtEGuRbw39nh20n3Dgt6WhPg$)) can be used to generate long reads with realistic error profiles. It is not necessary, but the authors can also perform a similar analysis or point to an already existing benchmark for metagenomic analysis. A realistic measure of accuracy in the alignment is quite straightforward, as it depends on the reads that are correctly mapped back to the location of origin, determining true positive/ false positives, etc. Although it is understandable under a high variation rate, it might be infeasible to map a read to a unique location

correctly, but those cases will be infrequent. Some similar classification is done for time and memory usage can also be implemented for the accuracy metric. All the design principles that are discussed in the manuscripts, such as hashing to BWT-FM based indexing or using a mixed alignment scheme to form the final set of alignment, can have a recognizable effect on accuracy. In my opinion, such a measure will increase the usefulness of the manuscript to a great extent.

- We thank the reviewer for this valuable feedback. We agree on the importance of benchmarking the accuracy of different read alignment algorithms (and metagenomic analysis algorithms). However, as we are aiming to write a comprehensive review paper instead of a benchmarking paper, we believe this feedback is out of the scope of our manuscript. We prefer to keep the paper as a review paper that targets a broader audience and lists the historical perspective of algorithmic foundations and methodologies behind read alignment. We believe evaluating the accuracy using different metrics and datasets is still out of the scope of our review paper. The feedback provided by the reviewer, the suggested read simulator and suggested performance metrics are all very helpful and we will consider them in our another ongoing study that aims to evaluate the robustness and reproducibility of existing read alignment algorithms.

The plots of figure 4 correspond to the performance of 11 different alignment methodologies on various experiments. I guess the multiple points with the same color for each method refer to the experiments. It would be helpful to mention the legend in the caption.

- We mention the legend in the caption of Figure 4, as we show below:

The corresponding paragraph referring to Figure 4(c) mentions that there is not much difference between the Hashing based methods and the BWT-FM based methods. It seems SMALT is a hashing based method that stands out from other hashing based methods in terms of run time, while SNAP is the hashing based method that takes the highest memory. It will be interesting to discuss the reason for such a performance difference.

- We thank the reviewer for the feedback. We address this comment by clarifying two outliers to the observed performance. We added two sentences to explain the reasons for the high execution time of SMALT and the high memory footprint of SNAP, as we show below:

The colors used in Figure 4 are not very distinctive to me. I had a hard time identifying the tools back, especially when the colors are too similar. Given that there are multiple experiments for one tool, it would help identify a tool by an arrow whenever possible (specifically for figure 4f). For example, the text in the manuscript mentions that "Despite the overall longer runtime of Hamming-distance-based methods, the latest hashing based tool hashing based too (e.g., HISAT2) provide a comparable..", unfortunately, I could not identify HISAT2 back from the diagram. Using a different legend might be useful.

- We thank the reviewer for this helpful feedback. To address the reviewer's comment and easily distinguish each aligner performance, we changed four figures, Fig 4, Supplementary Fig 1, Supplementary Fig 2, and Supplementary Fig 4, by 1) adding another label to their x-axes to indicate the corresponding aligner tool and 2) vertically grouping all data points together (by enabling dodge option in Seaborn) that belong to each aligner tool, as we provide below:

Table 2 classifies the alignment algorithms from a practical standpoint based on metrics such as "ease of implementation" etc. I am curious how these metrics are computed.

- We define the "ease of implementation" as the ability to quickly implement such a read alignment algorithm and its indexing technique, flexibly apply some changes to it, and easily understand its working principle.

- To address this feedback, we added a new sentence to the caption of Table 2, as we show below:

"; the shared prefix between the read and the genome is stored only once and used by all the reads with that prefix." - I could not comprehend this sentence correctly; the suffix tree should be independent of the reads; it's part of the index made on the reference sequence. An explanation would be beneficial.

- We agree with the reviewer on the unclarity of this sentence. To address this issue, we change the sentence, as we provide below:

"Unlike a hash table, a suffix tree allows searching for both the exact and inexact match seeds by walking through.."

The initial seeding strategy can have a variable length of k-mers for selecting prefixes from the suffix array. Still, I could not understand how the suffix tree built on the index can allow inexact search if an additional set of heuristics is not employed.

- There are several algorithms that can perform inexact matching directly on suffix arrays, suffix trie, and other structures. We refer the reviewer to the following two papers as examples of such algorithms.

- Ghodsi, Mohammadreza, and Mihai Pop. "Inexact local alignment search over suffix arrays." 2009 IEEE international conference on bioinformatics and biomedicine. IEEE, 2009.

- Ukkonen, Esko. "Approximate string-matching over suffix trees." Annual Symposium on Combinatorial Pattern Matching. Springer, Berlin, Heidelberg, 1993.

It can be briefly done by traversing several branches based on the shared prefix between these branches, where each detour from one branch to another can allow for inexact matching at the corresponding location of the detour.

- To address this issue and make it more clear, we added the two references to the mentioned sentence in the paper, as we provide below:

β is used as a coefficient for the generalized linear mixed model regression. The values themselves do not express a lot, in my opinion (except their signs). It would be really helpful if some explanation can be added with positive and negative values for the corresponding coefficient.

- Thanks for the valuable comments. We agree with the reviewer's comment. To address this issue, we replace all occurrences of β in the paper with the corresponding amount of increase/decrease, as we provide below:

Page 23 paragraph starting from line 29 describes the standard process for transcriptome quantification using DNA-seq aligners. The following paragraph briefly describes the use of light-weight mappers to do transcript level read assignment. There are some issues with the generalization in the description. The commonly used pseudo-alignment-based methods, such as Kallisto, uses De-Bruijn graphs. In contrast, the other tool, Salmon, uses k-mer hash in conjunction with a Suffix Array index (using mapper such as RapMap) with additional heuristics to speed up the mapping process.

Moreover, Salmon can provide mapping locations and a mapping score. However, as the authors rightly pointed out, none of these tools provide CIGAR strings or any other alignment profiles for further analysis. I would suggest the authors to rewrite the sections with a more detailed treatment of the individual methods given both of the tools are highly cited and used.

- We thank the reviewer for the feedback. We agree on the fundamental difference between Kallisto and Salmon. To address this issue and make the description more comprehensive, we added the following paragraphs to the paper, as we provide below:

Referring to page 15 line 4: it will be helpful to elaborate on the kind of heuristics that the aligner algorithms used, such as bidirectional jump, skipping, hamming distance thresholding, etc. As with other aspects of the alignment pipeline, the seed extension heuristic can have a profound effect on the alignment results.

- We thank the reviewer for suggesting to elaborate on the heuristics used in read alignment. To address this issue, we added a new paragraph to the “Supplementary Note 4” section and referenced it in the sentence that the reviewer highlighted, as we provide below. We also mentioned other heuristics used to accelerate pairwise sequence alignment in the “Supplementary Note 5” section in the submitted version.

Reviewer 2

10 WGS samples were used to evaluate the chosen methods. I believe they are all short-reads data. If so, it would be more helpful if some long-reads data can be included to evaluate the mentioned long-reads aligners.

- We thank the reviewer for this feedback. We agree on the importance of using long reads for the evaluation. However, we believe that using long reads will not provide additional insights about the algorithmic developments of read alignment for the following four reasons: 1) Our goal in this paper is to review the historical perspective of the read alignment algorithmic development and foundations and not benchmarking existing tool nor finding “best” read alignment tool. 2) As we mention in the “Influence of long-read technologies on the development of novel read alignment algorithm” section, most of the long read alignment algorithms are still following the three key steps that we observe in short read alignment algorithms. 3) In Table 1, we observe that only a few read alignment algorithms (LAMSA, Magic-BLAST, BLAST, NGMLR, lordFAST, minimap2, YAHA, SNAP, BWA-SW, GraphMap, GraphMap2, conLSH, rHAT, BLASR) support long reads (read length of ≥ 8000). We also observe that only two of these tools (SNAP and minimap2) are available to us on bioconda. Hence, we will not have enough tools on bioconda for our long read evaluation to compare using different performance metrics. 4) We already considered evaluating the performance of the read alignment algorithms that are proposed after 2013 (the first ONT sequencing machine was available in 2014), which can support longer read lengths (> 200 bp), in Figure 4e and Supplementary Figure 1.

In the GitHub repo

([https://urldefense.com/v3/__https://github.com/Mangul-Lab-USC/review_technology_dictates_algorithms__!!Llr3w8kk_Xxm!6t79Yi8uYPnGb6lb0G_SMZ1pLX-jpHO-ikoh20FtEGuRbw39nh20n3BRr8JZ2g\\$](https://urldefense.com/v3/__https://github.com/Mangul-Lab-USC/review_technology_dictates_algorithms__!!Llr3w8kk_Xxm!6t79Yi8uYPnGb6lb0G_SMZ1pLX-jpHO-ikoh20FtEGuRbw39nh20n3BRr8JZ2g$)) users were directed to the repository's wiki for more information.

But I didn't find the wiki page. A detailed description on how to use the provided scripts to reproduce the experimental results in the manuscript would be very helpful to the community.

- We thank the reviewer for this important feedback on reproducibility and user-friendliness. To address this feedback, we improved the GitHub page of our paper to include more details about the project in general and on how to reproduce the same results we have in our paper, as we provide in: <https://github.com/Mangul-Lab-USC/review-technology-dictates-algorithms>