The SeqAn C++ template library for efficient sequence analysis: A resource for programmers

Journal of Biotechnology 261 (2017) 157–168

Contents lists available at ScienceDirect

Journal of Biotechnology

journal homepage: www.elsevier.com/locate/jbiotec

The SeqAn C+ + template library for efficient sequence analysis: A resource for programmers☆

Knut Reinert[a,*], Temesgen Hailemariam Dadi[a], Marcel Ehrhardt[a], Hannes Hauswedell[a], Svenja Mehringer[a], René Rahn[a], Jongkyu Kim[b], Christopher Pockrandt[b], Jörg Winkler[b], Enrico Siragusa[c], Gianvito Urgese[d], David Weese[e]

[a] *Algorithmic Bioinformatics, Institute for Bioinformatics, FU Berlin, Takustrasse 9, 14195 Berlin, Germany*
[b] *Efficient Algorithms for -Omics Data, Max Planck Institute for Molecular Genetics, Ihnestrasse 62-73, 14195 Berlin, Germany*
[c] *IBM Watson Research, Yorktown Heights, NY, USA*
[d] *Department of Control and Computer Engineering, Politecnico di Torino, Italy*
[e] *SAP Innovation Center, Potsdam, Germany*

ARTICLE INFO

Keywords:
NGS analysis
Software libraries
C+ +
Data structures

ABSTRACT

Background: The use of novel algorithmic techniques is pivotal to many important problems in life science. For example the sequencing of the human genome (Venter et al., 2001) would not have been possible without advanced assembly algorithms and the development of practical BWT based read mappers have been instrumental for NGS analysis. However, owing to the high speed of technological progress and the urgent need for bioinformatics tools, there was a widening gap between state-of-the-art algorithmic techniques and the actual algorithmic components of tools that are in widespread use. We previously addressed this by introducing the SeqAn library of efficient data types and algorithms in 2008 (Döring et al., 2008).
Results: The SeqAn library has matured considerably since its first publication 9 years ago. In this article we review its status as an established resource for programmers in the field of sequence analysis and its contributions to many analysis tools.
Conclusions: We anticipate that SeqAn will continue to be a valuable resource, especially since it started to actively support various hardware acceleration techniques in a systematic manner.

1. Introduction

The analysis of biological sequences is at the core of computational biology. Advances in biotechnology have driven the development. of many successful algorithms (e.g., Myers' bit-vector search algorithm Myers, 1999, BLAST Altschul et al., 1990) and data structures (e.g., suffix arrays Abouelhoda et al., 2002, *q*-gram based string indices, FM-indices Ferragina and Manzini, 2000) over the last 20 years. The assemblies of large eukaryotic genomes like *Drosophila melanogaster* (Adams et al., 2000), human (Venter et al., 2001), and mouse (Mural et al., 2002) are prime examples where algorithm research was successfully applied to advance biological knowledge. However, with entire genomes at hand, large scale analysis algorithms that require considerable computing resources are becoming increasingly important and so do their implementations as efficient tools. Although these tools use slightly different algorithms, nearly all of them require some basic algorithmic components, like string indices, string searches, or alignments.

It is non-trivial to program efficient implementations of these components, especially if vectorization and multi-threading becomes more and more mandatory. This leads in practice often to the use of suboptimal data types and *ad-hoc* algorithms or the analysis is conducted by stringing together standalone tools. Both approaches may be suitable at times, but it would clearly be much more desirable to use an integrated library of state-of-the-art components that use modern hardware. Those components can be combined in various ways, either to develop new applications or to compare alternative implementations. Also, relying on a software library cuts down development time and ensures correctness and compatibility.

We previously addressed this by introducing the SeqAn library of efficient data types and algorithms in 2008 (Döring et al., 2008). Since then, SeqAn has matured considerably and is currently being supported

☆ SeqAn is supported in the CIBI as part of the de.NBI network.
* Corresponding author.
E-mail address: knut.reinert@fu-berlin.de (K. Reinert).

http://dx.doi.org/10.1016/j.jbiotec.2017.07.017
Received 26 February 2017; Received in revised form 17 July 2017; Accepted 19 July 2017
Available online 06 September 2017

by the de.NBI network for bioinformatics infrastructure as part of the CIBI (Center for Integrative BIoinformatics). In this article we review its status as an established resource for programmers in the field of sequence analysis and its contributions to many analysis tools.

2. Material and methods

The SeqAn library was first published in 2008 and designed with certain goals in mind, namely to promote (1) *high performance* of the provided algorithmic components, (2) *simplicity* and usability, (3) *generality* of data types and core algorithms, (4) the definition of special *refinements* of generic classes of algorithms, (5) the *extensibility* of the library and easy *integration* with other libraries.

2.1. Design and content of SeqAn

The SeqAn library currently consists of around 175,000 lines of code which are split into 47 modules of varying sizes. SeqAn was implemented in C++, a multi-paradigm high level programming language, and is at the time of writing this article available in version 2.3. The key concepts that are used in SeqAn are algorithm-oriented programming (Musser and Stepanov, 1994), *generic* programming using *templates* and *template meta-programming*, and *partial template specialization* as a form of polymorphism. We also make use of *tag dispatching* to select the most efficient algorithm for a certain task at compile time. The reasoning and implementation details are covered in Döring et al. (2008), and have remained largely unchanged for SeqAn 2.x.

The core of the SeqAn project is the SeqAn library, but under the umbrella of the project there are also support utilities including a custom documentation system modeled on doxygen (van Heesch, 2008), a custom test system and a selection of applications that rely heavily on the library (more on these later). For practical reasons, we will only focus on a subset of the modules and give the reader an overview of the extensive content of SeqAn.

align, align_*, score. The align modules offer all functionality to perform the many different types of pairwise alignments that are useful in sequence analysis. For example they provide two data structures to efficiently incorporate gaps into the underlying sequence: `ArrayGaps` and `AnchorGaps`. While `ArrayGaps` are efficient for linear access patterns, `AnchorGaps` have a better runtime complexity for random access patterns, because they can employ binary search for the lookup. In Listing 1 we show how to compute a standard pairwise global alignment using `ArrayGaps` as the gap structure of choice.

Our sequence library also implements many adaptions of the original dynamic programming algorithms among the default pairwise alignments. For example, we have implementations for split break point calculation (Emde et al., 2012a,b; Holtgrewe et al., 2015a,b), seed extensions using x-drop with affine gap costs (Hauswedell et al., 2014a,b), banded chain alignment, a new gap model called Dynamic Gap Selector (Urgese et al., 2014) and many more.

In addition to the sequential one-to-one interface we also implemented a vectorized many-to-many pairwise alignment interface, which uses extended instruction sets of modern CPU architectures to speed-up the computation of many pairwise sequence alignments (see Section 3.1).

arg_parse. SeqAn offers a generic argument parser and option handler that supports convenient access to the command line parameters. The *arg_parse* module makes it easy to define, restrict and retrieve any command line input while remaining flexible for individual needs. A well formatted help page is automatically generated, providing a clear and common interface to all SeqAn based applications.

Since SeqAn version 2.0.0, the argument parser is able to export interface descriptor files in an XML based format, that allow seamless integration of all applications into workflow systems like KNIME (Berthold et al., 2007) or Galaxy (Afgan et al., 2016) (see Section 2.2). Recently, we also introduced a mechanism to inform our users and developers whenever the library or one of our registered applications can be updated to a newer version (see Section 2.2).

blast. The newly introduced blast module offers e-value statistics based on official NCBI source code (Camacho et al., 2009), as well as support for the most commonly used blast output formats, including tab-separated output and full report.

index. SeqAn also provides an indexing module offering numerous string indices for arbitrary alphabets such as (enhanced) suffix arrays, FM indices, lazy suffix trees or q-gram indices. This includes fast and practical implementations of unidirectional and bidirectional FM indices, also the first and currently only openly available implementation of a constant-time bidirectional FM index (Pockrandt et al., 2017). For more details, see the results section. Listing 2 demonstrates an offline-search with the FM-Index in SeqAn.

journaled_string_tree. The journaled string tree is a data structure suitable for streaming over a set of referentially compressed sequences and is applied in the field of compressive genomics (Marschall et al., 2016). It works generically for all algorithms, whose state depends on a sequence context, i.e. a finite number of contiguous characters (Rahn et al., 2014).

modifier. In addition to the many container types that SeqAn provides, it also offers so called modifiers which are what is now commonly referred to as a view in C++ terminology – a light-weight data structure that behaves like a container, but does not store the data. Instead it performs a transformation "on-the-fly" during access. An intuitive biological example is the reverse complement modifier that appears like the reverse complement of a DNA sequence without copying the actual string data.

stream. The stream module is the basis for all other I/O modules. It provides a stream abstraction on top of STL stream buffers with built-in support for transparent (de-)compression with GZIP and BZIP2 (Gailly and Adler, 2003; Seward, 1996) if the corresponding libraries are present on the system. Consequently, all I/O done through SeqAn interfaces has automatic support for reading/writing compressed versions of the corresponding files. This module has been completely rewritten for SeqAn 2 and now profits from simpler interfaces, better performance and improved error handling via exceptions.

seq_io, gff_io, bam_io, rna_io, vcf_io. Sequence I/O is part of most bioinformatics applications and SeqAn supports many popular formats like fasta, fastq (Cock et al., 2010) and genbank, GFF, VCF SAM, BAM and certain RNA formats. The bam_io module contains a full implementation of the SAM and BAM formats, independent of SAMTOOLS and HTSLIB (Li et al., 2009). Listing 3 demonstrates a simple bam-to-sam converter implemented in SeqAn. The listing reads in a BAM file and displays its content on the standard output. A prominent change in SeqAn 2 is the addition of a highly parallel decompressor for BGZF so that our performance in reading and parsing BAM files is significantly higher than in any other implementation. Comparisons with other implementations are available in the results Section 3.3. In SeqAn 2 SAM and BAM files can also be treated as regular sequence input files. This is useful in case the original sequence files are no longer available or if BAM is preferred as storage for its compression. The VCF module contains functionality to read and write files in VCF format (Danecek et al., 2011). We also plan to add BCF format in the near future.

Files for RNA may contain structural information of a sequence or alignment. This module supports I/O for common RNA structure formats, as there are Connect, Dot-bracket, Vienna, Bpseq, and extended Bpseq for sequences and Stockholm for alignments. We created the extension of Bpseq to offer a format that contains multiple base pair probabilities of sequence interactions, which can be gained from experiments or statistical methods.

sequence, sequence_journaled. This module contains the wide range of generic containers, including *std::vector*-like strings, fixed-size arrays, external strings and bit-compressed strings. External strings behave just like regular containers but use the hard disk as storage and keep only a small fraction of the sequence in memory. This is e.g. useful if the sequence is larger than the available memory. A memory-mapped string backend is also available which enables faster I/O, on-demand loading, and inter-process memory sharing on UNIX systems. Bit-compressed strings on the other hand make use of the small biological alphabets like DNA, packing four characters into one byte, thereby reducing the overall memory consumption of the data structure by a factor of four. Of course, it is also possible to combine these features in one container. In addition to these custom SeqAn containers we have made it possible to use the generic STL containers in almost all modules in SeqAn 2. The journaled sequence data structure stores differences to an underlying reference sequence in a dictionary. It supports random-access operations very efficiently with just a logarithmic penalty over the number of differences stored in the dictionary.

2.2. SeqAn resources

Platform support. We take great care and quite some effort to make SeqAn run on most standard operating systems, like various Linux distributions, FreeBSD, Microsoft Windows and macOS. In addition, we support the following C+ + compilers: gcc ≥4.9, clang+ + ≥3.5, MSVC ≥2015 and ICPC ≥16.0.3, which we test for 64 bit as well as 32 bit Intel/AMD architectures. Recently we have also begun testing on other CPU architectures including PowerPC64 and Sparc64.

Online tutorials. Taking advantage of the clear layout from *Read the Docs*,[1] SeqAn offers a variety of detailed online tutorials. For users which are new to SeqAn, the tutorials provide a smooth and easy introduction to the main concepts and data structures of SeqAn. Additionally, they present a systematic way to quickly look up problem-solving strategies, refresh one's memory about names or syntax, and equip the user with helpful sample code in order to facilitate the usage of our library. With individual workflows and *how-to* instructions, we hope to encourage the consideration of SeqAn based solutions to frequent problems in the area of bioinformatics.

User Group Meeting. Once a year we organize a *User Group Meeting* that aims to arrange a compact one-week schedule of hands-on tutorials, application sessions and talks, inviting people of diverse working environments. Topics include the SeqAn library and SeqAn based applications but also KNIME workflow integration, external applications as well as current challenges and issues in the bioinformatics field.

CTD workflow integration. The Common Tool Descriptor format (CTD, 2017) is a XML-based description of the command line arguments of an application. This standardized descriptor file can be used to build plugins for different workflows systems, in order to make the tool available in the respective system.

The GenericWorkflowNodes (GenericWorkflowNodes project, 2017) project provides an implementation of a generic Node for the KNIME workflow system (Berthold et al., 2007)(see Fig. 1). We added a CTD-writer into SeqAn's argument parser, which can write a CTD file from the tool's help page. Thus, any program using SeqAn's argument parser, including all the in-house applications developed in SeqAn, can write a CTD file and hence be integrated into KNIME. Fig. 1 depicts an example workflow for species level identification of microorganisms from metagenomes (Dadi et al., 2016) using a combination of SeqAn nodes and general purpose KNIME nodes.

Version updates and user statistics. As it is very common in major software packages, the SeqAn library (since version 2.3.0) now includes a feature inside the command line parser to inform the user or developer whenever a new version of the application or library is available. At the same time this enables us to roughly assess the number of users of our library and different applications, delivering very valuable feedback. As a side note, external developers that use the SeqAn library are able to register their application with us and benefit from easy version updates and notifications. If not desired, the feature can easily be turned off.

Benchmark tool. The steady increase of large data sets due to benefits from new technologies such as NGS makes it necessary to apply efficient data structures and algorithms. This implies a need for suitable standardized benchmark tools, which must include a clear input and output format as well as individual quality controls besides the usual performance measures. As far as we know, there is still no collective effort towards this aim. SeqAn therefore developed a novel stand-alone benchmark tool, tackling the issues mentioned above.

The tool offers a clean graphical user interface, where the user can either run all predefined benchmarks, testing the system performance, or supply custom binary paths for specific benchmarks, in order to compare his own algorithms and applications to SeqAn and others. It will be used for regular internal performance and sanity checks and also aims to sustain a solution for other developers to compare their work to us and others. The benchmark tool is available under https://github.com/seqan/bench/.

3. Results

In the following we will describe the key algorithmic components of SeqAn that are competitive and used in various applications. As such, we will describe the generic pairwise alignment module, which encompasses many variations of dynamic programming based alignment, the segment based multiple alignment module and our fast SAM and BAM I/O module.

3.1. The generic pairwise alignment module

Pairwise sequence alignments play a significant role in many bioinformatic pipelines. Despite it is quadratic running time, the original Needleman–Wunsch algorithm was modified and used in many applications in the past. Likewise, our sequence library provides among the default pairwise alignments many variants of the original dynamic programming (DP) algorithm to perform special algorithmic tasks, e.g., to calculate split break points (Emde et al., 2012a,b; Holtgrewe et al., 2015a,b), x-drop with affine gap costs (Hauswedell et al., 2014a,b), banded chain alignment, and many more.

To avoid code redundancy we developed a unified DP interface for standard and banded pairwise alignment algorithms which can be configured at compile time.

Our implementation model is based on the following observation. Every pairwise alignment follows the same algorithmic phases, i.e., initializing the matrix, iterating recursively over every cell in the matrix and computing its score, and finally tracing the best alignment if necessary. Yet, different DP approaches require unique policies for certain computational steps, e.g., when using linear gaps instead of affine gaps the recursion method per cell differs, or when computing the best global vs. local alignment, where for the latter approach all cells need to be checked for the best score while for the former only the last cell matters.

Thus, by specifying the algorithm type (e.g., global alignment, local alignment, split alignment, etc.), the gap type (one of linear, affine or dynamic gaps) and the trace type (e.g., no trace, one best trace, all best traces with left or right gaps placement) the new DP interface will use meta-functions and tag dispatching mechanisms (see Section 2.1 for more details) to tell the compiler which code paths should be generated for the selected DP traits.

For example, when choosing the algorithm type to be a global alignment with free end gaps for the top and the bottom of the DP matrix, then the compiler will generate code that initializes the first row with zeros and enables the tracking of the best score in the last row, leading to the computation of a semi-global alignment. At the same time, the algorithm can either be compiled with a recursion policy for linear, dynamic (Urgese et al., 2014), or affine gaps, and can avoid the complete traceback computation, when only the score is needed.

[1] https://readthedocs.org/.

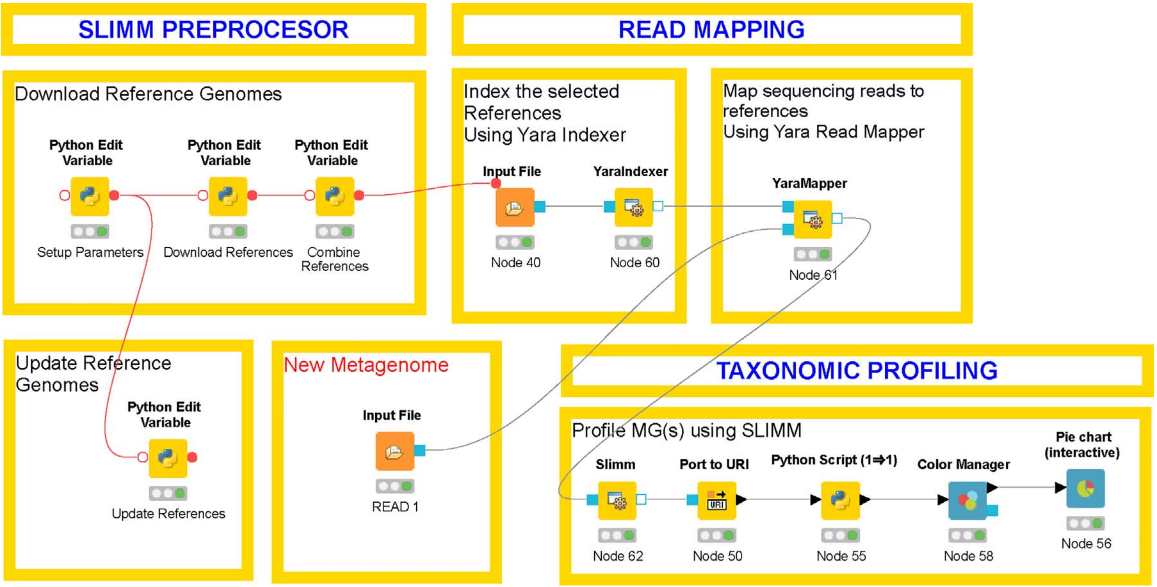

Fig. 1. SLIMM: identification and quantification of microorganisms within and between microbial communities.

The combination of the three DP traits together with the underlying meta-programming model creates a highly flexible and extendible DP module. The code basis can be maintained much easier, since optimizations and improvements are visible to all alignment variants. Furthermore, the generated alignment code is still very competitive since modern compilers can optimize tremendously and remove unused code paths, such that no unnecessary branches in the inner loop remain.

One of the first major improvements to the unified DP code was the parallelization of the recursion policies using a single-instruction multiple-data (SIMD) vectorization scheme. This allows us to generically use extended instruction sets of modern processors to compute several alignments in parallel on a single CPU. In Fig. 2 we compare the runtimes for computing 500,000 global sequence alignments between reads of length 152 base pairs (bp) with affine gap costs. The experiment was repeated using SSE4 (Intel, 2017) and AVX2 (Lomont, 2011) instruction sets, allowing to run 8 and 16 packed alignments in parallel, respectively. At the moment we also develop a generalized parallel execution model to optimally utilize concurrent compute resources for the pairwise alignment algorithms.

3.2. Segment based sequence alignment

For multiple sequence alignments SeqAn::T-Coffee (Rausch et al., 2008) uses a consistency-based, progressive alignment heuristic. As an extension of T-Coffee (Notredame et al., 2000) the algorithm operates on *segments* as opposed to single sequence characters, which improves the run time automatically if the sequences are related. For example, when aligning 6 adenovirus sequences that are quite similar, our implementation is faster than T-Coffee, produces an alignment of higher column identity, and consumes significantly less memory. For the comparison in Table 1 we obtained the sequences from NCBI (NC_001460, NC_004001, NC_001405, NC_002067, NC_003266, NC_001454) and used the current versions of T-Coffee, MAFFT, Clustal Omega, PASTA, and SeqAn::T-Coffee. The average identity column shows the ratio of the characters identical with their column-wise consensus. SeqAn::T-Coffee outperforms the other tools regarding the average identity and memory consumption.

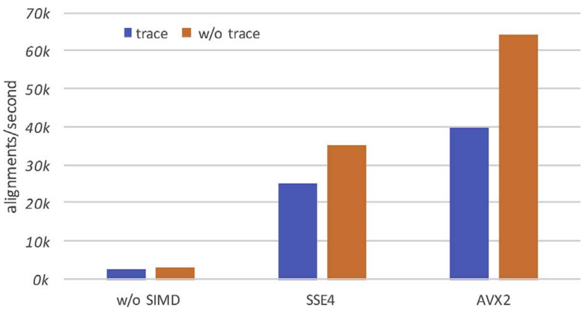

Fig. 2. 500k global pairwise sequence alignments of 152 bp long reads using affine gap costs. The experiments were conducted on an Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30 GHz 2-socket processor with SSE4 and AVX2 support.

Table 1
Multiple alignment of six adenovirus sequences. The number of columns with at least 6, 5, and 4 characters are reported together with the average identity w.r.t. the consensus.

Aligner	Time	Aver. id.	=6	≥5	≥4	Memory
SeqAn::T-Coffee	874 s	77.28%	13050	27536	36055	1901 Mb
SeqAn::T-Coffee-bw 500	89 s	76.58%	13023	25778	34399	150 Mb
T-Coffee	1087 s	74.78%	12943	20911	29782	47507 Mb
MAFFT	156 s	74.68%	12646	21362	30136	59 Mb
Clustal Omega	592 s	74.13%	12346	21346	29574	8323 Mb
PASTA	1972 s	74.66%	12515	21248	29175	14738 Mb

Since then, we improved the SeqAn::T-Coffee implementation for deep alignments, i.e. multiple sequence alignments of hundreds or thousands of sequences. The implemented heuristics aim to reduce the run time of SeqAn::T-Coffee significantly, while keeping its high quality. Firstly, we use a banded alignment to reduce the asymptotic run time of a pairwise alignment from quadratic to linear. The drawback is a limited number of consecutive insertions or deletions, but as the sequences are expected to be similar, this does not cause a significant drop of quality for a band of sufficient width. For a band of width 500 SeqAn::T-Coffee accelerated by almost factor 10 (see Table 1). Secondly, we only perform the triplet extension among sequences of a subtree in the guide tree. Most corrections of the progressive alignment occur in the first steps, where the number of aligned sequences is low. In subsequent steps the columns are almost fixed, so the computational efforts can be reduced by not performing an exhaustive search for support sequences. Thirdly, in the triplet extension step we only increase the weight of existing edges and we do not create edges any more. The new edges increase the computational effort disproportionately by being rarely involved in the final alignment.

This resulted in code that was about 30 times faster in aligning 200 protein sequences (Yasnev, 2015) while maintaining a quality similar to the previous implementation and on par with state-of-the art aligners like Clustal-Omega (Sievers et al., 2011).

3.3. I/O module

The I/O module of SeqAn supports most of the standard sequence and alignment file formats. Among others, FASTA, FASTQ, BLAST, SAM and BAM file formats can be read and written easily and efficiently using the SeqAn library. We organized the I/O module into several submodules namely, sequence_io, bam_io, blast_io, vcf_io, bed_io, gff_io and rna_io submodule.

The sequence_io submodule can handle FASTA, FASTQ, genebank and embl file formats. One can read and write records sequentially one at a time or in batch mode. There is also support for indexing FASTA files with multiple entries through `fai_index`. This indexing enables a faster reading of a particular FASTA entry without having to read through the file starting from the beginning. Starting from version 2.3.X SeqAn also supports using BAM to store unaligned sequences is a convenient alternative to FASTQ as it helps to save storage space and allows parallel decompression.

The SeqAn BAM I/O submodule is very convenient for reading SAM or BAM alignment files or writing alignment results to disk in SAM or BAM format. To demonstrate the performance of SeqAn's BAM I/O module, we carried out a benchmarking using the HTS Compression Benchmarking Suite (Numanagić et al., 2016) adapted to include a subset of the applications, i.e., samtools (Li and Durbin, 2009), sambamba (Tarasov et al., 2015), picard and cramtools (Numanagić et al., 2016) together with a small BAM I/O implementation in SeqAn. The results show that SeqAn's BAM I/O has the fastest compression of all the other tools in both single and multi-threaded modes. SeqAn's BAM I/O decompression is faster than samtools and slower than sambamba (see Fig. 3). We used the *E. coli* dataset sequenced using Illumina MiSeq technology from the benchmarking tool paper (Numanagić et al., 2016). The uncompressed size of the file was 5.2 GB and the coverage was 420 ×.

3.4. Index module

SeqAn's index module consists of a wide range of string indices with various applications and interfaces for different iterators as well as backtracking. Indices include the (enhanced) suffix array with optional tables (`lcp`, `childtab`, `bwt`) depending on the application and anticipated running time, lazy suffix trees, q-gram indices (also known as k-mer indices) and FM indices (Ferragina and Manzini, 2000). For the latter one we do not only offer state-of-the-art unidirectional but also bidirectional implementations. The FM index is implemented using the well known wavelet tree (Grossi et al., 2003) as well as the enhanced prefixsum dictionary (EPR dictionary) for constant-time unidirectional and bidirectional search steps. This makes SeqAn the first and currently only library with a constant-time implementation of a bidirectional FM index, independent from the alphabet size. It is also currently the fastest openly available unidirectional and bidirectional FM index. To show this we give below some benchmark results which can be found in more detail also in Pockrandt et al. (2017).

In the benchmark we will compare FM and 2FM indices in SeqAn with other available 2FM implementations, namely the bidirectional wavelet tree by Schnattinger et al. (2012) which we will call 2SCH and the balanced wavelet tree implementation with plain bit vectors and constant-time rank support in the SDSL (Gog et al., 2014) which we will refer to as 2SDSL. SeqAn has four different implementations of FM indices, a wavelet tree based and an EPR dictionary based one, both working with the unidirectional and bidirectional one, referred to as WT and EPR, respectively 2EPR and 2WT. All wavelet tree based implementations, include 2SDSL and 2SCH have a running time of $\mathcal{O}(\log \sigma)$ per search step where σ is the alphabet size.

The benchmark was performed for different alphabets sizes to test the predicted independence from σ for the EPR implementation. The alphabet sizes are inspired by bioinformatics applications and are of size 4 (DNA), 10 (reduced amino acid alphabet *Murphy10*), 16 (IUPAC alphabet) and 27 (protein alphabet).

We first generated a text of length 10^8 with a uniform distribution and sampled 1 million queries of length 200 from this text. The search in the FM and 2FM indices will determine the number of occurrences of the sampled strings. Our sampling will ensure that the text occurs at least once and the stepwise search is never prematurely stopped. This ensures that we have 200 million single steps in searches. The unidirectional FM indices perform backward searches while for 2FM indices we search the right half of the query first (using forward searches) and then extend the other half of the pattern to the left by backward searches.

Table 2 gives an overview of the running times of all FM and 2FM index implementations. It shows the absolute runtimes as well as the speedup factor relative to the unidirectional resp. bidirectional wavelet tree implementation. Our bidirectional wavelet tree implementation 2WT has a similar runtime compared to 2SDSL. It is slightly faster especially for small alphabets.

When we compare the runtimes of the EPR and 2EPR, they behave as expected, i.e., the unidirectional index is slightly faster, since in each step of the bidirectional index we have to synchronize two indices.

All indices implemented in SeqAn (WT, EPR, 2WT, 2EPR) support up to 3 levels for rank dictionary support: blocks, superblocks and ultrablocks. The tests presented here were performed with a 2-level rank

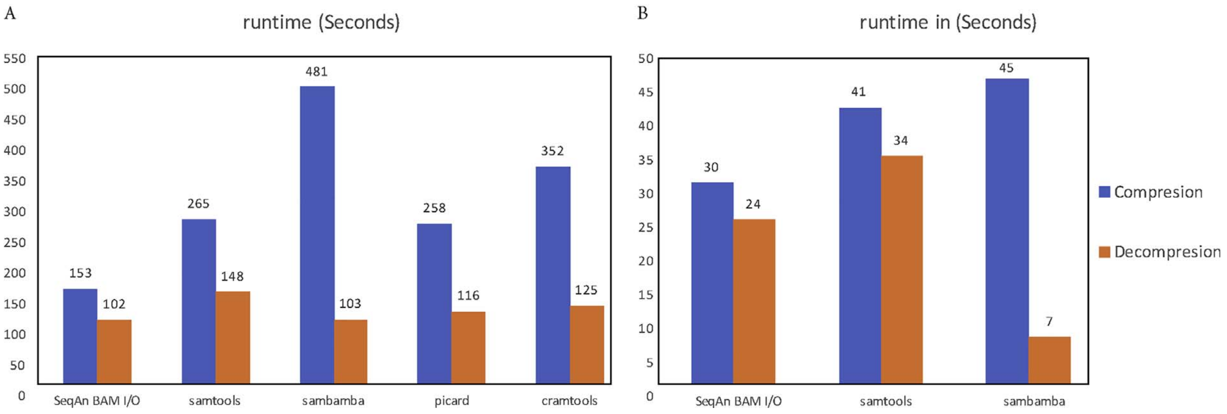

Fig. 3. Time taken by different tools to do compression (reading a file in SAM format and write it in BAM format) and decompression (reading a file in BAM format and write it in SAM format) using single thread (A) and multiple threads (B).

Table 2
Runtimes of various implementations in seconds and their speedup factors with respect to the unidirectional respectively bidirectional wavelet tree.

Index	DNA		Murphy10		IUPAC		Protein	
	Time	Factor	Time	Factor	Time	Factor	Time	Factor
WT	20.73 s	1.00	52.35 s	1.00	66.45 s	1.00	85.55 s	1.00
EPR	**15.09 s**	**1.37**	**22.27 s**	**2.35**	**23.39 s**	**2.84**	**25.31 s**	**3.38**
2WT	41.21 s	1.00	66.59 s	1.00	98.74 s	1.00	120.96 s	1.00
2EPR	**20.09 s**	**2.05**	**23.80 s**	**2.80**	**24.39 s**	**4.05**	**26.08 s**	**4.64**
2SDSL	43.54 s	0.95	74.69 s	0.89	89.07 s	1.11	109.44 s	1.11
2SCH	59.59 s	0.69	91.30 s	0.73	107.04 s	0.92	129.98 s	0.93

Bold is the best performing.

Table 3
Space consumption of the rank data structure in Megabyte of various implementations.

Index	DNA	Murphy10	IUPAC	Protein
EPR	42	156	227	478
2EPR	84	311	454	955
WT	30	51	60	72
2WT	60	102	120	144
2SDSL	68	105	122	145
2SCH	75	108	123	146

dictionary. Table 3 illustrates the practical space consumption for all previously discussed indices. The number for the FM indices given in Table 3 do neither account for storing the text itself nor for storing a compressed suffix array necessary to locate the matches in the text since the libraries use different implementations offering various space-time trade-offs. The running time of the backward and forward searches does not depend on it and the compressed suffix array implementation is independent from the used rank dictionary and thus interchangeable. A typical compressed suffix array implementation as used in the 2SDSL takes $\frac{n}{\delta}\log n$ (when sampling on the text instead of the suffix array). For a sampling rate of 10% ($\delta = 10$) the space consumption for our experiments would be 253 MB and thus still much smaller than the affix array.

Iterators can be used on these indices to easily search the underlying text. Besides the TopDown iterator, which allows traversing the corresponding suffix respectively prefix tree/trie, there are also iterator interfaces for bottom up traversal and more specialized iterators for enumerating maximal repeats, supermaximal repeats or maximal unique matches. Furthermore, SeqAn includes a backtracking interface which allows performing approximate string searching. It is generic with respect to the text's alphabet, the index as well as the error metric (Hamming or Levenshtein distance). It cannot only search for a single string with a given number of errors, it also supports searching an entire string set at once as well as an indexed string set (i.e. double indexing). It also can be executed either in parallel or sequential mode.

3.5. The use of SeqAn in applications

SeqAn is used in many applications. Obviously it is a main resource for the groups of the main developers, but also for many external groups. We will give here a list of applications that use SeqAn in their code. Apart from the SeqAn book (Andreas and Reinert, 2009) we published several applications over the years. Among those are several read mapping applications, namely Masai (Siragusa et al., 2013), RazerS (Weese et al., 2009), RazerS3 (Weese et al., 2012), MicroRazerS (Emde et al., 2010), SplazerS (Emde et al., 2012a,b), and Yara (Siragusa, 2015). All those were complemented by a read mapping benchmark called Rabema (Holtgrewe et al., 2011). We published an exact local alignment program named Stellar (Kehr et al., 2011) and an efficient heuristic method called Lambda (Hauswedell et al., 2014a,b). The Cidane (Canzar et al., 2016) tool for RNA-Seq isoform quantitation makes use of SeqAn and the Fiona (Schulz et al., 2014) tool offers error correction capabilities. The Imseq (Kuchenbecker et al., 2015) tool uses SeqAn's fast alignment for immunogenetic sequence analysis. For variation analysis SeqAn was used in the tools Anise and Basil (Holtgrewe et al., 2015a,b), Gustaf (Trappe et al., 2014) and Vaquita (submitted) who all address the analysis of structural variants. In the field of multiple sequence alignment, the SeqAn::TCoffee engine described above was used in SeqAn Consensus (Rausch et al., 2009) and SeqAn TCoffee (Rausch et al., 2008). Recently we started to employ SeqAn tools in the analysis of metagenomics data sets and published SLIMM (Dadi et al., 2016), a tool for identifying species on microbiomes. Finally we used Seqan for the analysis of virus integration sites in Hüser et al. (2010).

Many other groups build and published tools using SeqAn. First of all, it has been used for various sequence alignment tools. Bowtie2 (Langmead and Salzberg, 2012) and Hobbes (Ahmadi et al., 2012) are short-read aligners and Mugsy (Angiuoli and Salzberg, 2011) is a tool for whole-genome alignment. In addition, Tophat (Trapnell et al., 2009) and Q (Hansen et al., 2015) are designed for RNA sequencing and ChIP-seq data analysis, respectively.

There are many tools regarding structural variation as well. Delly (Rausch et al., 2012) and Wham (Kronenberg et al., 2015) are examples. Furthermore, PopSTR (Kristmundsdóttir et al., 2016) for microsatellite genotyping, PopIns (Kehr et al., 2016) for population scale study of non-reference insertions, PopAlu (Qian et al., 2015) for Alu elements research, MixTaR (Fertin et al., 2015) for tandem repeats detection, and InFusion (Okonechnikov et al., 2016) for gene fusion identification, and isomiR-SEA (Urgese et al., 2016) for the computation of miRNAs and isomiRs expression levels in the RNA-Seq experiments are tools that use SeqAn.

SeqAn also has been used for motif finding (Reid et al., 2011), sequence compression (Wandelt et al., 2015),.bam file processing utility (Óskarsdóttir et al., 2015), string matching library (Ayad et al., 2016), sequence assembly (Klein et al., 2011) and proteomics (Röst et al., 2016). Moreover, SeqAn also has contributed to various researches about microRNA (Rhee et al., 2015), alternative splicing (Hatje and Kollmar, 2013, 2014), enhancers (Comin and Antonello, 2016), triplex formation (Buske et al., 2012), fungal translation (Mühlhausen and Kollmar, 2014), the khmer k-mer counting library (Crusoe et al., 2015), and HIV (Zanini et al., 2015; Seifert et al., 2016).

4. Conclusion

We presented in this paper the state of the software library SeqAn as a resource for quickly developing efficient and robust tools for sequence analysis. We presented the SeqAn eco-system and the current content, we highlighted its performance on some important data structures and finally gave an overview of publications that make use of SeqAn.

Upcoming versions of SeqAn will focus on incorporating vectorized and multi-threaded code and continue to explore the capabilities of modern C+ + standards.

Appendix A

See Listing 1–3.

Acknowledgements

We would like to thank many people and funding agencies that supported SeqAn related projects.

This work was supported by the Intel Parallel Compute Center at FU Berlin; by the German Science Foundation (DFG) [grant numbers RE 1712/4-1 (Data Parallelism), RE 1712/3-1, RE 1712/3-2 (Algorithm Engineering)]; the German Federal Ministry of Education and Research (BMBF) [grant number 031A535B]; and the Max Planck society.

Many people contributed over time to the SeqAn code base. We want to acknowledge Chenxu Pan, Birte Kehr, Manuel Holtgrewe, Andreas Döring, Tobias Rausch, Anne-Katrin Emde as well as external contributors Oleg Yasnev, Jonathan Göke, Stefan Budach and Johannes Röhr.

We also thank the anonymous reviewer who diligently pointed out small mistakes in the code examples.

Listing 1. Global alignment computation.

```cpp
#include <iostream>
#include <seqan/stream.h>
#include <seqan/align.h>

using namespace seqan;

int main()
{
    StringSet<DnaString> stringSet;
    appendValue(stringSet, "AGTTTAATCA");
    appendValue(stringSet, "AGTATACGA");
    // Initialize the Align object using a StringSet.
    Align<DnaString, AnchorGaps<> > align(stringSet);

    int score = globalAlignment(align, EditDistanceScore());
    std::cout << "Score: " << score << std::endl;
    std::cout << align << std::endl;

    return 0;
}
```

Listing 2. Searching with FM-indices.

```
#include <iostream>
#include <seqan/find.h>
#include <seqan/index.h>

using namespace seqan;

int main()
{
    CharString hstck = "I spy with my little eye something "
                       "that is yellow";
    Index<CharString, FMIndex<> > index(hstck);
    Finder<Index<CharString, FMIndex<> > > finder(hstck);

    while (find(finder, "y"))
        std::cout << "Hit at: " << position(finder) << ’\n’;

    // reset Finder
    clear(finder);

    while (find(finder, "t"))
        std::cout << "Hit at: " << position(finder) << ’\n’;
}
```

Listing 3. Reading and writing BAM files.

```
#include <iostream>
#include <seqan/stream.h>
#include <seqan/bam_io.h>

using namespace seqan;

int main()
{
    BamFileIn bamFileIn;
    if (!open(bamFileIn, "example.bam"))
    {
        std::cerr << "Can't open the file." << std::endl;
        return 1;
    }

    // Open output stream to stdcout.
    BamFileOut bamFileOut(bamFileIn);
    open(bamFileOut, std::cout, Sam());

    // Copy header. The header is automatically written.
    BamHeader header;
    readHeader(header, bamFileIn);
    writeHeader(bamFileOut, header);

    // BamAlignmentRecord stores one record at a time.
    BamAlignmentRecord record;
    while (!atEnd(bamFileIn))
    {
        readRecord(record, bamFileIn);
        writeRecord(bamFileOut, record);
    }
    return 0;
}
```

References

Abouelhoda, M.I.I., Ohlebusch, E., Kurtz, S., 2002. Optimal Exact String Matching Based on Suffix Arrays. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 31–43.

Adams, M.D., Brandon, R., Celniker, S., Holt, R.A., Evans, C., Gocayne, J., Amanatides, P., Scherer, S., Li, P., Hoskins, R., Galle, R., George, R., Lewis, S., Richards, S., Ashburner, M., Henderson, S., Sutton, G.G., Wortman, J.R., Yandell, M.D., Zhang, Q., Chen, L., Rogers, Y., Blazej, R., Champe, M., Pfeiffer, B., Wan, K., Doyle, C., Baxter, E., Helt, G., Nelson, C., Miklos, G.G., Abril, J.F., Agbayani, A., An, H., Baldwin, D., Ballew, R., Basu, A., Baxendale, J., Bayraktaroglu, L., Beasley, E., Beeson, K., Benos, P., Berman, B., Bhandari, D., Bolshakov, S., Borkova, D., Botchan, M., Bouck, J., Brokstein, P., Brottier, P., Burtis, K., Busam, D., Butler, H., Cadieu, E., Center, A., Chandra, I., Cherry, J., Cawley, S., Dahlke, C.E., Davenport, L.B., Davies, P., de Pablos, B., Delcher, A., Deng, Z., Mays, A., Dew, I., Dietz, S., Dodson, K., Doup, L., Downes, M., Dugan-Rocha, S., Dunkov, B., Dunn, P., Durbin, K., Evangelista, C., Ferraz, C., Ferriera, S., Fleischmann, W., Fosler, C., Gabrielian, A., Garg, N., Gelbart, W., Glasser, K., Glodek, A., Gong, F., Gorrell, J., Gu, Z., Guan, P., Harris, M., Harris, N., Harvey, D., Heiman, T., Hernandez, J., Houck, J., Hostin, D., Houston, K., Howland, T., Wei, M., Ibegwam, C., Jalali, M., Kalush, F., Karpen, G., Ke, Z., Kennison, J., Ketchum, K., Kimmel, B., Kodira, C., Kraft, C., Kravitz, S., Kulp, D., Lai, Z., Lasko, P., Lei, Y., Levitsky, A., Li, J., Li, Z., Liang, Y., Lin, X., Liu, X., Mattei, B., McIntosh, T.C., McLeod, M.P., McPherson, D., Merkulov, G., Milshina, N.V., Mobarry, C., Morris, J., Moshrefi, A., Mount, S., Moy, M., Murphy, B., Murphy, L., Muzny, D., Nelson, D., Nelson, D., Nelson, K., Nixon, K., Nusskern, D., Pacleb, J., Palazzolo, M., Pittman, G., Pan, S., Pollard, J., Puri, V., Reese, M., Reinert, K., Remington, K., Saunders, R., Scheeler, F., Shen, H., Shue, B., Sidén-Kiamos, S., Simpson, M., Skupski, M., Smith, T., Spier, E., Spradling, A., Stapleton, M., Strong, R., Sun, E., Svirskas, R., Tector, C., Turner, R., Venter, E., Wang, A., Wang, X., Wang, Z., Wassarman, D., Weinstock, G., Weissenbach, J., Williams, S., Woodage, T., Worley, K., Wu, D., Yang, S., Yao, Q., Ye, J., Yeh, R., Zaveri, J.S., Zhan, M., Zhang, G., Zhao, Q., Zheng, L., Zheng, X., Zhong, F., Zhong, W., Zhou, X., Zhu, S., Zhu, X., Smith, H.O., Gibbs, R., Myers, E.W., Rubin, G., Venter, C.J., 2000. The genome sequence of *Drosophila melanogaster*. Science (New York NY) 287 (5461), 2185–2195.

Afgan, E., Baker, D., van den Beek, M., Blankenberg, D., Bouvier, D., Čech, M., Chilton, J., Clements, D., Coraor, N., Eberhard, C., Grüning, B., Guerler, A., Hillman-Jackson, J., Von Kuster, G., Rasche, E., Soranzo, N., Turaga, N., Taylor, J., Nekrutenko, A., Goecks, J., 2016. The galaxy platform for accessible, reproducible and collaborative biomedical analyses: update. Nucleic Acids Res.

Ahmadi, A., Behm, A., Honnalli, N., Li, C., Weng, L., Xie, X., 2012. Hobbes: optimized gram-based methods for efficient read alignment. Nucleic Acids Res. 40 (6), e41.

Altschul, S.F., Gish, W., Myers, G.W., Lipman, D.J., 1990. Basic local alignment search tool. J. Mol. Biol. 215 (3), 403–410.

Andreas, G., Reinert, K., 2009. Biological Sequence Analysis Using the SeqAn C++ Library. CRC.

Angiuoli, S.V., Salzberg, S.L., 2011. Mugsy: fast multiple alignment of closely related whole genomes. Bioinformatics 27 (3), 334–342.

Ayad, L.A., Pissis, S.P.P., Retha, A., 2016. libFLASM: a software library for fixed-length approximate string matching. BMC Bioinform. 17 (1), 454.

Berthold, M.R., Cebron, N., Dill, F., Gabriel, T.R., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., Wiswedel, B., 2007. KNIME: the Konstanz information miner. Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007). Springer.

Buske, F.A., Bauer, D.C., Mattick, J.S., Bailey, T.L., 2012. Triplexator: detecting nucleic acid triple helices in genomic and transcriptomic data. Genome Res. 22 (7), 1372–1381.

Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., Madden, T.L., 2009. Blast+: architecture and applications. BMC Bioinform. 10 (1), 421.

Canzar, S., Andreotti, S., Weese, D., Reinert, K., Klau, G.W., 2016. CIDANE: comprehensive isoform discovery and abundance estimation. Genome Biol. 17 (1), 16.

Cock, P.J., Fields, C.J., Goto, N., Heuer, M.L., Rice, P.M., 2010. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. Nucleic Acids Res. 38 (6), 1767–1771.

Comin, M., Antonello, M., 2016. On the comparison of regulatory sequences with multiple resolution entropic profiles. BMC Bioinform. 1–12.

Crusoe, M.R., Alameldin, H.F., Awad, S., Boucher, E., Caldwell, A., Cartwright, R., Charbonneau, A., Constantinides, B., Edvenson, G., Fay, S., Fenton, J., Fenzl, T., Fish, J., Garcia-Gutierrez, L., Garland, P., Gluck, J., González, I., Guermond, S., Guo, J., Gupta, A., Herr, J.R., Howe, A., Hyer, A., Härpfer, A., Irber, L., Kidd, R., Lin, D., Lippi, J., Mansour, T., McA'Nulty, P., McDonald, E., Mizzi, J., Murray, K.D., Nahum, J.R., Nanlohy, K., Nederbragt, A.J., Ortiz-Zuazaga, H., Ory, J., Pell, J., Pepe-Ranney, C., Russ, Z.N., Schwarz, E., Scott, C., Seaman, J., Sievert, S., Simpson, J., Skennerton, C.T., Spencer, J., Srinivasan, R., Standage, D., Stapleton, J.A., Steinman, S.R., Stein, J., Taylor, B., Trimble, W., Wiencko, H.L., Wright, M., Wyss, B., Zhang, Q., Zyme, E., Brown, C.T., 2015. The khmer software package: enabling efficient nucleotide sequence analysis. F1000Research 4, 900.

Common Tool Descriptor project. (accessed: 15.02.17). https://github.com/WorkflowConversion/CTDSchema.

Döring, A., Weese, D., Rausch, T., Reinert, K., 2008. SeqAn an efficient, generic C++ library for sequence analysis. BMC Bioinform. 9 (1), 11. http://dx.doi.org/10.1186/1471-2105-9-11.

Dadi, T.H., Renard, B.Y., Wieler, L.H., Semmler, T., Reinert, K., 2016. SLIMM: Species Level Identification of Microorganisms from Metagenomes.

Danecek, P., Auton, A., Abecasis, G., Albers, C.A., Banks, E., DePristo, M.A., Handsaker, R.E., Lunter, G., Marth, G.T., Sherry, S.T., et al., 2011. The variant call format and VCFtools. Bioinformatics 27 (15), 2156–2158.

Emde, A., Grunert, M., Weese, D., Reinert, K., Sperling, S.R., 2010. MicroRazerS: rapid alignment of small RNA reads. Bioinformatics (Oxford, England) 26 (1), 123–124.

Emde, A., Schulz, M.H., Weese, D., Sun, R., Vingron, M., Kalscheuer, V.M., Haas, S.A., Reinert, K., 2012a. Detecting genomic indel variants with exact breakpoints in single- and paired-end sequencing data using SplazerS. Bioinformatics (Oxford, England) 28 (5), 619–627.

Emde, A.K., Schulz, M.H., Weese, D., Sun, R., Vingron, M., Kalscheuer, V.M., Haas, S.A., Reinert, K., 2012b. Detecting genomic indel variants with exact breakpoints in single- and paired-end sequencing data using SplazerS. Bioinformatics 28 (5), 619–627.

Ferragina, P., Manzini, G., 2000. Opportunistic data structures with applications. Proceedings of the 41st Annual Symposium on Foundations of Computer Science, FOCS '00. IEEE Computer Society, Washington, DC, USA, pp. 390.

Fertin, G., Jean, G., Radulescu, A., Rusu, I., 2015. Hybrid de novo tandem repeat detection using short and long reads. BMC Med. Genomics 8 (Suppl. 3), S5.

Gailly, J.-l., Adler, M., 2003. Gzip.

GenericWorkflowNodes project. (accessed: 15.02.17). https://github.com/genericworkflownodes.

Gog, S., Beller, T., Moffat, A., Petri, M., 2014. From theory to practice: plug and play with succinct data structures. 13th International Symposium on Experimental Algorithms (SEA 2014). pp. 326–337.

Grossi, R., Gupta, A., Vitter, J.S., 2003. High-order entropy-compressed text indexes. Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, pp. 841–850.

Hüser, D., Andreas, G., Lutter, T., Weger, S., Winter, K., Hammer, E., Cathomen, T., Reinert, K., Heilbronn, R., 2010. Integration preferences of wildtype AAV-2 for consensus rep-binding sites at numerous loci in the human genome. PLoS Pathogens 6 (7) e1000985.

Hansen, P., Hecht, J., Ibrahim, D.M., Krannich, A., Truss, M., Robinson, P.N., 2015. Saturation analysis of ChIP-seq data for reproducible identification of binding peaks. Genome Res. 25 (9), 1391–1400.

Hatje, K., Kollmar, M., 2013. Expansion of the mutually exclusive spliced exome in Drosophila. Nat. Commun. 4 (2460).

Hatje, K., Kollmar, M., 2014. Kassiopeia: a database and web application for the analysis of mutually exclusive exomes of eukaryotes. BMC Genomics 15 (1), 115.

Hauswedell, H., Singer, J., Reinert, K., 2014a. Lambda: the local aligner for massive biological data. Bioinformatics (Oxford, England) 30 (17), i349–i355.

Hauswedell, H., Singer, J., Reinert, K., 2014b. Lambda: the local aligner for massive biological data. Bioinformatics 30 (17), i349–i355.

Holtgrewe, M., Emde, A., Weese, D., Reinert, K., 2011. A novel and well-defined benchmarking method for second generation read mapping. BMC Bioinform. 12 (1), 210.

Holtgrewe, M., Kuchenbecker, L., Reinert, K., 2015a. Methods for the detection and assembly of novel sequence in high-throughput sequencing data. Bioinformatics (Oxford, England) 31 (12), 1904–1912.

Holtgrewe, M., Kuchenbecker, L., Reinert, K., 2015b. Methods for the detection and assembly of novel sequence in high-throughput sequencing data. Bioinformatics 31 (12), 1904–1912.

Intel(R) SSE4 Programming Reference. (accessed: 15.02.17). https://software.intel.com/sites/default/files/m/8/b/8/D9156103.pdf.

Kehr, B., Weese, D., Reinert, K., 2011. STELLAR: fast and exact local alignments. BMC Bioinform. 12 (Suppl. 9), S15.

Kehr, B., Melsted, P., Halldórsson, B.V., 2016. PopIns: population-scale detection of novel sequence insertions. Bioinformatics 32 (7), 961–967. arXiv:1504.01813v1.

Klein, J.D., Ossowski, S., Schneeberger, K., Weigel, D., Huson, D.H., 2011. LOCAS – a low coverage assembly tool for resequencing projects. PLoS ONE 6 (8), e23455.

Kristmundsdóttir, S., Sigurpálsdóttir, B.D., Kehr, B., Halldórsson, B.V., 2016. popSTR: population-scale detection of STR variants. Bioinformatics btw568.

Kronenberg, Z.N., Osborne, E.J., Cone, K.R., Kennedy, B.J., Domyan, E.T., Shapiro, M.D., Elde, N.C., Yandell, M., 2015. Wham: identifying structural variants of biological consequence. PLoS Comput. Biol. 11 (12), e1004572.

Kuchenbecker, L., Nienen, M., Hecht, J., Neumann, A.U., Babel, N., Reinert, K., Robinson, P.N., 2015. IMSEQ – a fast and error aware approach to immunogenetic sequence analysis. Bioinformatics (Oxford, England) 31 (18) btv309-2971.

Langmead, B., Salzberg, S., 2012. Fast gapped-read alignment with Bowtie 2. Nat. Publ. Gr. 9 (4), 357–359.

Li, H., Durbin, R., 2009. Fast and accurate short read alignment with Burrows–Wheeler transform. Bioinformatics 25 (14), 1754–1760.

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R., et al., 2009. The sequence alignment/map format and SAMtools. Bioinformatics 25 (16), 2078–2079.

Lomont, C., 2011. Introduction to intel advanced vector extensions. Intel White Pap. 1–21.

Mühlhausen, S., Kollmar, M., 2014. Predicting the fungal CUG codon translation with Bagheera. BMC Genomics 15, 411.

Marschall, T., Marz, M., Abeel, T., Dijkstra, L., Dutilh, B.E., Ghaffaari, A., Kersey, P., Kloost-erman, W., Makinen, V., Novak, A., et al., 2016. Computational pan-genomics: status, promises and challenges. BioRxiv 043430.

Mural, R.J., Adams, M.D., Myers, E.W., Smith, H.O., Miklos, G.L.G., Wides, R., Halpern, A., Li, P.W., Sutton, G.G., Nadeau, J., Salzberg, S.L., Holt, R.A., Kodira, C.D., Lu, F., Chen, L., Deng, Z., Evangelista, C.C., Gan, W., Heiman, T.J., Li, J., Li, Z., Merkulov, G.V., Milshina, N.V., Naik, A.K., Qi, R., Shue, B.C., Wang, A., Wang, J., Wang, X., Yan, X., Ye, J., Yooseph, S., Zhao, Q., Zheng, L., Zhu, S.C., Biddick, K., Bolanos, R., Delcher, A.L., Dew, I.M., Fasulo, D., Flanigan, M.J., Huson, D.H., Kravitz, S.A., Miller, J.R., Mobarry, C.M., Reinert, K., Remington, K.A., Zhang, Q., Zheng, X.H., Nusskern, D.R., Lai, Z., Lei, Y., Zhong, W., Yao, A., Guan, P., Ji, R.-R., Gu, Z., Wang, Z.-Y., Zhong, F., Xiao, C., Chiang, C.-C., Yandell, M.D., Wortman, J.R., Amanatides, P.G., Hladun, S.L., Pratts, E.C., Johnson, J.E., Dodson, K.L., Woodford, K.J., Evans, C.A., Gropman, B., Rusch, D.B., Venter, E., Wang, M., Smith, T.J., Houck, J.T., Tompkins, D.E., Haynes, C., Jacob, D., Chin, S.H., Allen, D.R., Dahlke, C.E., Sanders, R., Li, K., Liu, X., Levitsky, A.A., Majoros, W.H., Chen, Q., Xia, A.C., Lopez, J.R., Donnelly, M.T., Newman, M.H., Glodek, A., Kraft, C.L., Nodell, M., Ali, F., An, H.-J., Baldwin-Pitts, D., Beeson, K.Y., Cai, S., Carnes, M., Carver, A., Caulk, P.M., Center, A., Chen, Y.-H., Cheng, M.-L., Coyne, M.D., Crowder, M., Danaher, S., Davenport, L.B., Desilets, R., Dietz, S.M., Doup, L., Dullaghan, P., Ferriera, S., Fosler, C.R., Gire, H.C., Gluecksmann, A., Gocayne, J.D., Gray, J., Hart, B., Haynes, J., Hoover, J., Howland, T., Ibegwam, C., Jalali, M., Johns, D., Kline, L., Ma, D.S., MacCawley, S., Magoon, A., Mann, F., May, D., McIntosh, T.C., Mehta, S., Moy, L., Moy, M.C., Murphy, B.J., Murphy, S.D., Nelson, K.A., Nuri, Z., Parker, K.A., Prudhomme, A.C., Puri, V.N., Qureshi, H., Raley, J.C., Reardon, M.S., Regier, M.A., Rogers, Y.-H.C., Romblad, D.L., Schutz, J., Scott, J.L., Scott, R., Sitter, C.D., Smallwood, M., Sprague, A.C., Stewart, E., Strong, R.V., Suh, E., Sylvester, K., Thomas, R., Tint, N.N., Tsonis, C., Wang, G.G., Wang, G.G., Williams, M.S., Williams, S.M., Windsor, S.M., Wolfe, K., Wu, M.M., Zaveri, J.S., Chaturvedi, K., Gabrielian, A.E., Ke, Z., Sun, J., Subramanian, G.M., Venter, J.C., Myers, G.W., Smith, H.O., Miklos, G.L.G., Wides, R., Halpern, A., Li, P.W., Sutton, G.G., Nadeau, J., Salzberg, S.L., Holt, R.A., Kodira, C.D., Lu, F., Chen, L., Deng, Z., Evangelista, C.C., Gan, W., Heiman, T.J., Li, J., Li, Z., Merkulov, G.V., Milshina, N.V., Naik, A.K., Qi, R., Shue, B.C., Wang, A., Wang, J., Wang, X., Yan, X., Ye, J., Yooseph, S., Zhao, Q., Zheng, L., Zhu, S.C., Biddick, K., Bolanos, R., Delcher, A.L., Dew, I.M., Fasulo, D., Flanigan, M.J., Huson, D.H., Kravitz, S.A., Miller, J.R., Mobarry, C.M., Reinert, K., Remington, K.A., Zhang, Q., Zheng, X.H., Nusskern, D.R., Lai, Z., Lei, Y., Zhong, W., Yao, A., Guan, P., Ji, R.-R., Gu, Z., Wang, Z.-Y., Zhong, F., Xiao, C., Chiang, C.-C., Yandell, M.D., Wortman, J.R., Amanatides, P.G., Hladun, S.L., Pratts, E.C., Johnson, J.E., Dodson, K.L., Woodford, K.J., Evans, C.A., Gropman, B., Rusch, D.B., Venter, E., Wang, M., Smith, T.J., Houck, J.T., Tompkins, D.E., Haynes, C., Jacob, D., Chin, S.H., Allen, D.R., Dahlke, C.E., Sanders, R., Li, K., Liu, X., Levitsky, A.A., Majoros, W.H., Chen, Q., Xia, A.C., Lopez, J.R., Donnelly, M.T., Newman, M.H., Glodek, A., Kraft, C.L., Nodell, M., Ali, F., An, H.-J., Baldwin-Pitts, D., Beeson, K.Y., Cai, S., Carnes, M., Carver, A., Caulk, P.M., Center, A., Chen, Y.-H., Cheng, M.-L., Coyne, M.D., Crowder, M., Danaher, S., Davenport, L.B., Desilets, R., Dietz, S.M., Doup, L., Dullaghan, P., Ferriera, S., Fosler, C.R., Gire, H.C., Gluecksmann, A., Gocayne, J.D., Gray, J., Hart, B., Haynes, J., Hoover, J., Howland, T., Ibegwam, C., Jalali, M., Johns, D., Kline, L., Ma, D.S., MacCawley, S., Magoon, A., Mann, F., May, D., McIntosh, T.C., Mehta, S., Moy, L., Moy, M.C., Murphy, B.J., Murphy, S.D., Nelson, K.A., Nuri, Z., Parker, K.A., Prudhomme, A.C., Puri, V.N., Qureshi, H., Raley, J.C., Reardon, M.S., Regier, M.A., Rogers, Y.-H.C., Romblad, D.L., Schutz, J., Scott, J.L., Scott, R., Sitter, C.D., Smallwood, M., Sprague, A.C., Stewart, E., Strong, R.V., Suh, E., Sylvester, K., Thomas, R., Tint, N.N., Tsonis, C., Wang, G.G., Wang, G.G., Williams, M.S., Williams, S.M., Windsor, S.M., Wolfe, K., Wu, M.M., Zaveri, J.S., Chaturvedi, K., Gabrielian, A.E., Ke, Z., Sun, J., Subramanian, G.M., Venter, J.C., Pfannkoch, C.M., Barnstead, M., Stephenson, L.D., 2002. A comparison of whole-genome shotgun-derived mouse chromosome 16 and the human genome. Science (New York, NY) 296 (5573), 1661–1671.

Musser, D.R., Stepanov, A.A., 1994. Algorithm-oriented generic libraries. Softw. Pract. Exp. 24, 623–642.

Myers, G.W., 1999. A fast bit-vector algorithm for approximate string matching based on dynamic programming. J. ACM 46 (3), 395–415.

Notredame, C., Higgins, D.G., Heringa, J., 2000. T-Coffee: a novel method for fast and accurate multiple sequence alignment. J. Mol. Biol. 302 (1), 205–217.

Numanagić, I., Bonfield, J.K., Hach, F., Voges, J., Ostermann, J., Alberti, C., Mattavelli, M., Sahinalp, S.C., 2016. Comparison of high-throughput sequencing data compression tools. Nat. Methods 13 (12), 1005–1008.

Okonechnikov, K., Imai-Matsushima, A., Paul, L., Seitz, A., Meyer, T.F., Garcia-Alcalde, F., 2016. InFusion: advancing discovery of fusion genes and chimeric transcripts from deep RNA-sequencing data. PLOS ONE 11 (12), e0167417.

Óskarsdóttir, A., Másson, G., Melsted, P., 2015. BamHash: a checksum program for verifying the integrity of sequence data. Bioinformatics 32 (1), 140–141.

Pockrandt, C., Ehrhardt, M., Reinert, K., 2017. EPR-Dictionaries: A Practical and Fast Data Structure for Constant Time Searches in Unidirectional and Bidirectional FM Indices. Springer International Publishing, Cham, pp. 190–206. http://dx.doi.org/10.1007/978-3-319-56970-3_12.

Qian, Y., Kehr, B., Halldorsson, B.V., 2015. PopAlu: population-scale discovery of Alu polymorphisms. PeerJ 3, e1269.

Röst, H.L., Sachsenberg, T., Aiche, S., Bielow, C., Weisser, H., Aicheler, F., Andreotti, S., Ehrlich, H.-C., Gutenbrunner, P., Kenar, E., Liang, X., Nahnsen, S., Nilse, L., Pfeuffer, J., Rosenberger, G., Rurik, M., Schmitt, U., Veit, J., Walzer, M., Wojnar, D., Wolski, W.E., Schilling, O., Choudhary, J.S., Malmström, L., Aebersold, R., Reinert, K., Kohlbacher, O., 2016. OpenMS: a flexible open-source software platform for mass spectrometry data analysis. Nat. Methods 13 (9), 741–748.

Rahn, R., Weese, D., Reinert, K., 2014. Journaled string tree – a scalable data structure for analyzing thousands of similar genomes on your laptop. Bioinformatics 30 (24) btu438.

Rausch, T., Emde, A., Weese, D., Döring, A., Notredame, C., Reinert, K., 2008. Segment-based multiple sequence alignment. Bioinformatics (Oxford, England) 24 (16), i187–i192.

Rausch, T., Koren, S., Denisov, G., Weese, D., Emde, A., Doring, A., Reinert, K., 2009. A consistency-based consensus algorithm for de novo and reference-guided sequence assembly of short reads. Method Biochem. Anal. 25 (9), 1118–1124.

Rausch, T., Zichner, T., Schlattl, A., Stütz, A.M., Benes, V., Korbel, J.O., 2012. DELLY: structural variant discovery by integrated paired-end and split-read analysis. Bioinformatics 28 (18), i333–i339.

Reid, J.E., Wernisch, L., 2011. STEME: efficient EM to find motifs in large data sets. Nucleic Acids Res. 39 (18), e126.

Rhee, S., Chae, H., Kim, S., 2015. PlantMirnaT: miRNA and mRNA integrated analysis fully utilizing characteristics of plant sequencing data. Methods 83, 80–87.

Schnattinger, T., Ohlebusch, E., Gog, S., 2012. Bidirectional search in a string with wavelet trees and bidirectional matching statistics. Inf. Comput. 213, 13–22.

Schulz, M.H., Weese, D., Holtgrewe, M., Dimitrova, V., Niu, S., Reinert, K., Richard, H., 2014. Fiona: a parallel and automatic strategy for read error correction. Bioinformatics (Oxford, England) 30 (17), i356–i363.

Seifert, D., Di Giallonardo, F., Töpfer, A., Singer, J., Schmutz, S., Günthard, H.F., Beerenwinkel, N., Metzner, K.J., 2016. A comprehensive analysis of primer IDs to study heterogeneous HIV-1 populations. J. Mol. Biol. 428 (1), 238–250.

Seward, J., 1996. bzip2 and libbzip2. http://www.bzip.org.

Sievers, F., Wilm, A., Dineen, D., Gibson, T.J., Karplus, K., Li, W., Lopez, R., McWilliam, H., Remmert, M., Söding, J., Thompson, J.D., Higginsv D.G, 2011. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. Mol. Syst. Biol. 7 (1), 1–6.

Siragusa, E., Weese, D., Reinert, K., 2013. Fast and accurate read mapping with approximate seeds and multiple backtracking. Nucleic Acids Res. 41 (7) e78–e78.

Siragusa, E., 2015. Approximate String Matching for High-Throughput Sequencing (Ph.D. thesis). Freie Universitt Berlin.

Tarasov, A., Vilella, A.J., Cuppen, E., Nijman, I.J., Prins, P., 2015. Sambamba: fast processing of NGS alignment formats. Bioinformatics 31 (12), 2032–2034.

Trapnell, C., Pachter, L., Salzberg, S.L., 2009. TopHat: discovering splice junctions with RNA-Seq. Bioinformatics 25 (9), 1105–1111.

Trappe, K., Emde, A., Ehrlich, H., Reinert, K., 2014. Gustaf: detecting and correctly classifying SVs in the NGS twilight zone. Bioinformatics (Oxford, England) btu431.

Urgese, G., Paciello, G., Acquaviva, A., Ficarra, E., Graziano, M., Zamboni, M., 2014. Dynamic gap selector: a Smith Waterman sequence alignment algorithm with affine gap model optimisation. In: Proc. 2nd Int. Work-Conf. Bioinform. Biomed. Eng. (IWBBIO): 7–9 April 2014. 640 Granada, Copicentro Granada SL. pp. 1347–1358.

Urgese, G., Paciello, G., Acquaviva, A., Ficarra, E., 2016. isomiR-SEA: an RNA-Seq analysis tool for miRNAs/isomiRs expression level profiling and miRNA–mRNA interaction sites evaluation. BMC Bioinform. 17 (1), 148. http://dx.doi.org/10.1186/s12859-016-0958-0. http://eda.polito.it/isomir-sea.

van Heesch, D., 2008. Doxygen: Source Code Documentation Generator Tool. http://www.doxygen.org.

Venter, J.C., Adams, M.D., Myers, E.W., Li, P.W., Mural, R.J., Sutton, G.G., Smith, H.O., Yandell, M.D., Evans, C.A., Holt, R.A., Gocayne, J.D., Amanatides, P.G., Ballew, R.M., Huson, D.H., Wortman, J.R., Zhang, Q., Kodira, C.D., Zheng, X.H., Chen, L., Skupski, M.P., Subramanian, G.M., Thomas, P.D., Zhang, J., Gabor Miklos, G.L., Nelson, C., Broder, S., Clark, A.G., Nadeau, J., McKusick, V.A., Zinder, N., Levine, A.J., Roberts, R.J., Simon, M., Slayman, C., Hunkapiller, M., Bolanos, R., Delcher, A., Dew, I., Fasulo, D., Flanigan, M., Florea, L., Halpern, A., Hannenhalli, S., Kravitz, S., Levy, S., Mobarry, C., Reinert, K., Remington, K.A., Abu-Threideh, J., Beasley, E.M., Biddick, K., Bonazzi, V., Brandon, R., Cargill, M., Chandramouliswaran, I., Charlab, R., Chaturvedi, K., Deng, Z., Francesco, V.D., Dunn, P., Eilbeck, K., Evangelista, C., Gabrielian, A.E., Gan, W., Ge, W., Gong, F., Gu, Z., Guan, P., Heiman, T.J., Higgins, M.E., Ji, R.-R.R., Ke, Z., Ketchum, K.A., Lai, Z., Lei, Y., Li, Z., Li, J., Liang, Y., Lin, X., Lu, F., Merkulov, G.V., Milshina, N., Moore, H.M., Naik, A.K., Narayan, V.A., Neelam, B., Nusskern, D., Rusch, D.B., Salzberg, S., Shao, W., Shue, B.C., Sun, J., Wang, Z.Y., Wang, A.H.H., Wang, X., Wang, J., Wei, M.-H.H., Wides, R., Xiao, C., Yan, C., Yao, A., Ye, J., Zhan, M., Zhang, W., Zhang, H., Zhao, Q., Zheng, L., Zhong, F., Zhong, W., Zhu, S.C., Zhao, S., Gilbert, D., Baumhueter, S., Spier, G., Carter, C., Cravchik, A., Woodage, T., Ali, F., An, H.H.-J., Awe, A., Baldwin, D., Baden, H., Barnstead, M., Barrow, I., Beeson, K., Busam, D., Carver, A., Center, A., Cheng, M.L., Curry, L., Danaher, S., Davenport, L., Desilets, R., Dietz, S., Dodson, K., Doup, L., Ferriera, S., Garg, N., Gluecksmann, A., Hart, B., Haynes, J., Haynes, C., Heiner, C., Hladun, S., Hostin, D., Houck, J., Howland, T., Ibegwam, C., Johnson, J., Kalush, F., Kline, L., Koduru, S., Love, A., Mann, F., May, D., McCawley, S., McIntosh, T., McMullen, I., Moy, M., Moy, L., Murphy, B., Nelson, K., Pfannkoch, C., Pratts, E., Puri, V., Qureshi, H., Reardon, M., Rodriguez, R., Rogers, Y.-H.H., Romblad, D.L., Ruhfel, B., Scott, R., Sitter, C., Smallwood, M.M., Stewart, E., Strong, R., Suh, E., Thomas, R., Tint, N.N., Tse, S., Vech, C., Wang, G., Wetter, J., Williams, S., Williams, M.S., Windsor, S., Winn-Deen, E., Wolfe, K., Zaveri, J.S., Zaveri, K., Abril, J.F., Guigó, R., Campbell, M.J., Sjolander, K.V., Karlak, B., Kejariwal, A., Mi, H., Lazareva, B., Hatton, T., Narechania, A., Diemer, K., Muruganujan, A., Guo, N., Sato, S., Bafna, V., Istrail, S., Lippert, R., Schwartz, R., Walenz, B.P., Yooseph, S., Allen, D.R., Basu, A., Baxendale, J., Blick, L., Caminha, M., Carnes-Stine, J., Caulk, P., Chiang, Y.-H.H., Coyne, M., Dahlke, C., Mays, A.D., Dombroski, M., Donnelly, M., Ely, D., Esparham, S., Fosler, C., Gire, H., Glanowski, S., Glasser, K., Glodek, A., Gorokhov, M., Graham, K., Gropman, B., Harris, M., Heil, J., Henderson, S., Hoover, J., Jennings, D., Jordan, C., Jordan, J., Kasha, J., Kagan, L., Kraft, C., Levitsky, A., Lewis, M., Liu, X., Lopez, J., Ma, D., Majoros, W., McDaniel, J., Murphy, S., Newman, M., Nguyen, T., Nguyen, N., Nodell, M., Pan, S., Peck, J., Peterson, M., Rowe, W., Sanders, R., Scott, J., Simpson, M., Smith, T., Sprague, A.C., Stockwell, T.B., Turner, R., Venter, E., Wang, M., Wen, M., Wu, D., Wu, M., Xia, A., Zandieh, A., Zhu, X., Myers, G.W., Li, P.W., Mural, R.J., Sutton, G.G., Smith, H.O., Yandell, M.D., Evans, C.A., Holt, R.A., Gocayne, J.D., Amanatides, P.G., Ballew, R.M., Huson, D.H., Wortman, J.R., Zhang, Q., Kodira, C.D., Zheng, X.H., Chen, L., Skupski, M.P., Subramanian, G.M., Thomas, P.D., Zhang, J., Gabor Miklos, G.L., Nelson, C., Broder, S., Clark, A.G., Nadeau, J., McKusick, V.A., Zinder, N., Levine, A.J., Roberts, R.J., Simon, M., Slayman, C., Hunkapiller, M., Bolanos, R., Delcher, A., Dew, I., Fasulo, D., Flanigan, M., Florea, L., Halpern, A., Hannenhalli, S., Kravitz, S., Levy, S., Mobarry, C., Reinert, K., Remington, K.A., Abu-Threideh, J., Beasley, E.M., Biddick, K., Bonazzi, V., Brandon, R., Cargill, M., Chandramouliswaran, I., Charlab, R., Chaturvedi, K., Deng, Z., Di Francesco, V., Dunn, P., Eilbeck, K., Evangelista, C., Gabrielian, A.E., Gan, W., Ge, W., Gong, F., Gu, Z., Guan, P., Heiman, T.J., Higgins, M.E., Ji, R.-R.R., Ke, Z., Ketchum, K.A., Lai, Z., Lei, Y., Li, Z., Li, J., Liang, Y., Lin, X., Lu, F., Merkulov, G.V., Milshina, N., Moore, H.M., Naik, A.K., Narayan, V.A., Neelam, B., Nusskern, D., Rusch, D.B., Salzberg, S., Shao, W., Shue, B.C., Sun, J., Wang, Z.Y., Wang, A.H.H., Wang, X., Wang, J., Wei, M.-H.H., Wides, R., Xiao, C., Yan, C., Yao, A., Ye, J., Zhan, M., Zhang, W., Zhang, H., Zhao, Q., Zheng, L., Zhong, F., Zhong, W., Zhu, S.C., Zhao, S., Gilbert, D., Baumhueter, S., Spier, G., Carter, C., Cravchik, A., Woodage, T., Ali, F., An, H.H.-J., Awe, A., Baldwin, D., Baden, H., Barnstead, M., Barrow, I., Beeson, K., Busam, D., Carver, A., Center, A., Cheng, M.L., Curry, L., Danaher, S., Davenport, L., Desilets, R., Dietz, S., Dodson, K., Doup, L., Ferriera, S., Garg, N., Gluecksmann, A., Hart, B., Haynes, J., Haynes, C., Heiner, C., Hladun, S., Hostin, D., Houck, J., Howland, T., Ibegwam, C., Johnson, J., Kalush, F., Kline, L., Koduru, S., Love, A., Mann, F., May, D., McCawley, S., McIntosh, T., McMullen, I., Moy, M., Moy, L., Murphy, B., Nelson, K., Pfannkoch, C., Pratts, E., Puri, V., Qureshi, H., Reardon, M., Rodriguez, R., Rogers, Y.-H.H., Romblad, D.L., Ruhfel, B., Scott, R., Sitter, C., Smallwood, M.M., Stewart, E., Strong, R., Suh, E., Thomas, R., Tint, N.N., Tse, S., Vech, C., Wang, G., Wetter, J., Williams, S., Williams, M.S., Windsor, S., Winn-Deen, E., Wolfe, K., Zaveri, J.S., Zaveri, K., Abril, J.F., Guig, R., Campbell, M.J., Sjolander, K.V., Karlak, B., Kejariwal, A., Mi, H., Lazareva, B., Hatton, T., Narechania, A., Diemer, K., Muruganujan, A., Guo, N., Sato, S., Bafna, V., Istrail, S., Lippert, R., Schwartz, R., Walenz, B.P., Yooseph, S., Allen, D.R., Basu, A., Baxendale, J., Blick, L., Caminha, M., Carnes-Stine, J., Caulk, P., Chiang, Y.-H.H., Coyne, M., Dahlke, C., Mays, A.D., Dombroski, M., Donnelly, M., Ely, D., Esparham, S., Fosler, C., Gire, H., Glanowski, S., Glasser, K., Glodek, A., Gorokhov, M., Graham, K., Gropman, B., Harris, M., Heil, J., Henderson, S., Hoover, J., Jennings, D., Jordan, C., Jordan, J., Kasha, J., Kagan, L., Kraft, C., Levitsky, A., Lewis, M., Liu, X., Lopez, J., Ma, D., Majoros, W., McDaniel, J., Murphy, S., Newman, M., Nguyen, T., Nguyen, N., Nodell, M., Pan, S., Peck, J., Peterson, M., Rowe, W., Sanders, R., Scott, J., Simpson, M., Smith, T., Sprague, A.C., Stockwell, T.B., Turner, R., Venter, E., Wang, M., Wen, M., Wu, D., Wu, M., Xia, A., Zandieh, A., Zhu, X., 2001. The sequence of the human genome. Science (New York, NY) 291 (5507), 1304–1351.

Wandelt, S., Leser, U., Rahman, M.S., 2015. Sequence factorization with multiple references. PLOS ONE 10 (9), e0139000.

Weese, D., Emde, A., Rausch, T., Döring, A., Reinert, K., 2009. RazerS-fast read mapping with sensitivity control. Genome Res. 19 (9), 1646–1654.

Weese, D., Holtgrewe, M., Reinert, K., 2012. RazerS 3: faster, fully sensitive read mapping. Bioinformatics (Oxford, England) 28 (20), 2592–2599.

Yasnev, O., June 2015. Adaptation of Multiple Sequence Alignment Algorithms from the SeqAn Library for Processing Deep Alignments (Ph.D. thesis).

Zanini, F., Brodin, J., Thebo, L., Lanz, C., Bratt, G., Albert, J., Neher, R.A., 2015. Population genomics of intrapatient HIV-1 evolution. eLife 4 (December). arXiv:1509.02483.