Alignment-free detection and seed-based identification of multi-loci V(D)J recombinations in Vidjil-algo

Cyprien Borée, Mathieu Giraud, Mikaël Salson

Alignment-free detection and seed-based identification of multi-loci V(D)J recombinations in Vidjil-algo

Cyprien Borée, Mathieu Giraud, Mikaël Salson

Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL – Centre de Recherche en Informatique Signal et Automatique de Lille – F-59000 Lille, France

Correspondence: contact@vidjil.org

Abstract

The diversity of the immune repertoire is grounded on V(D)J recombinations in several loci. Many algorithms and software detect and designate these recombinations in high-throughput sequencing data. To improve their efficiency, we propose a multi-loci seed identification through an Aho-Corasick like automaton as well as a seed-based gene filtration. These algorithms were implemented into Vidjil-algo, used routinely by several labs for the analysis of hematologic malignancies.
We benchmark the results of Vidjil-algo and of MiXCR on five datasets, evaluating the specificity and sensitivity of the detection, as well as the adequation of the designation to manually curated sequences. Compared to the previous algorithms, the new algorithms implemented in Vidjil-algo bring speedups between 3× and 30×, with a smaller memory footprint and without quality loss in results. They enable to precisely annotate in a few minutes millions of sequences coming from V(D)J recombinations, including incomplete V(D)J-like recombinations, improving our knowledge on immune repertoires.

Availability: https://www.vidjil.org/data#2024-pcicompbio

Preprint. July 3, 2024

Keywords: Spaced seeds; Aho-Corasick automaton; Alignment-free algorithms; Immune repertoire; V(D)J recombinations; Adaptive Immune Receptor Repertoire (AIRR); Repertoire Sequencing (RepSeq)

1 Introduction

V(D)J recombinations are genetic events occurring in immature immunologic cells, the lymphoblasts. These recombinations are an important factor of the diversity of the receptors on B- and T-cells (Tonegawa, 1983). A V(D)J recombination is the result of a random process combining a V gene, possibly a D gene, and a J gene on the genome, building either VDJ or a VJ recombination. The recombination involves enzymes called recombination-activating genes (RAGs). At the junction of these V, D, and J gene segments, nucleotides can be removed and other random ones can be added (see Figure 1), improving again the diversity. Depending on the cell type, these recombinations occur at several loci. Human B-cell receptors have an heavy chain (IGH) and a light chain λ or κ (denoted here IGL and IGK), whereas T-cell receptors either have γ and δ chains (TRG and TRD) or α and β chains (TRA and TRB).

T- and B-cell receptors are crucial for recognizing antigens, which is a key part of the adaptative immune response. Identifying and possibly quantifying V(D)J recombinations helps thus describe the immune response more efficiently. Moreover, a V(D)J recombination can be seen as an identifier – that may be unique – of a *clonal population*, or shortly *clone*, that is a cell population coming from a same lymphoblast. This identifying sequence is called a *clonotype*, but there may be several clonotypes of the same clone (at different loci or alleles) or several different clones with the same clonotype (that may differ elsewhere). In hemato-oncology, V(D)J recombinations are thus used to identify and track clonotypes along time in blood cancers (Cavé et al., 1998). Library preparation for these studies was recently standardized by the EuroClonality-NGS working group (Brüggemann et al., 2019; Langlois de Septenville et al., 2022; Villarese et al., 2022).

On the software side, since the 1980s, after the pioneering work by the Universität zur Köln with DNAPLOT, many tools for the in-depth analysis of V(D)J recombinations were developed by IMGT (Giudicelli, Chaume, et al., 1998; Lefranc, 2011). In the 2010s, new methods and software were proposed to analyze these V(D)J recombinations in high-throughput sequencing data with millions of sequences, such as (Arnaout et al., 2011), IgBlast (Ye et al., 2013), Decombinator (Thomas et al., 2013), miTCR (Bolotin, Shugay, et al., 2013), TCRKlass (Yang et al., 2014), Vidjil (Giraud et al., 2014), MiXCR (Bolotin, Poslavsky, et al., 2015), IMSEQ (Kuchenbecker et al., 2015), Partis (Ralph and Iv, 2016), IgReC (Shlemov et al., 2017), and IGoR (Marcou et al., 2018). Afzal et al. (2019) did a comparison of several of those software tools. These Adaptive Immune Receptor Repertoire (AIRR-Seq) methods and software, also called Repertoire Sequencing (RepSeq) (Benichou et al., 2012), developed and invented text algorithmics methods adapted to the specificity of V(D)J recombinations.

We call here:

- *detection*, the process of identifying the presence of a V(D)J recombination within a given DNA sequence and determining the related chain (the locus);
- *designation*, the process of determining the specific germline V, (D), and J genes that have undergone recombination, as well as identifying nucleotide deletions, or insertions that may have occurred at the junction between those genes, and possibly detecting the mutations in the whole sequence;
- *clusterization*, the process of gathering equal or similar recombinations, according to some criteria, into clonotypes.

Most V(D)J analysis software detect and designate V(D)J recombinations at the same time, for each of the input sequences, and some of them cluster sequences thereafter. Vidjil-algo rather cluster sequences *before* the

Figure 1. A VDJ recombination. After removal of a few nucleotides at the junction of V, D, and J, genes, `AATA` was inserted between gene segments V and D and `GCT` was inserted between gene segments D and J.

locus	regular recombinations		incomplete/irregular recombinations	
14q1.12	**TRA**	Va-Ja		
7q34	**TRB**	Vb-(Db)-Jb	**TRB+**	Db-Jb
14q11.2	**TRD**	Vd-(Dd)-Jd	**TRD+**	Vd-Dd3, Dd2-(Dd)-Jd, Dd2-Dd3
			TRA+D	Vd-(Dd)-Ja, Dd-Ja
7p14	**TRG**	Vg-Jg		
14q32.33	**IGH**	Vh-(Dh)-Jh	**IGH+**	Dh-Jh
22q11.2	**IGL**	Vl-Jl		
2p11.2	**IGK**	Vk-Jk	**IGK+**	Vk-KDE, INTRON-KDE

Table 1. Regular and incomplete/irregular human V(D)J and V(D)J-like recombinations analyzed by `vidjil-algo`. There are seven loci with at least 16 recombination possibilities (called *recombination systems*). For example, on the 14q11.2 TRD locus, apart from the regular recombination, there are at least three known systems of irregular recombinations (TRD+) as well as two systems involving the very close TRA locus (TRA+D). These recombination systems are formally described in the JSON file `homo-sapiens.g`, and the corresponding 5' and 3' genes (V and J genes for the regular recombinations) are indexed in a Aho-Corasick automaton. Note also that, for short genes (J genes, and D genes involved in incomplete recombination systems), both the genes and 60 bp of their downstream/upstream region are indexed (Duez et al., 2016).

full designation (Giraud et al., 2014). Indeed, for many applications, the designation for each sequence is not useful and more efficient alignment-free approaches, including k-mers indexing, can cluster the sequences after detection.

A mature B- or T- cell needs only *two* V(D)J recombinations, on only one allele. However, V(D)J recombinations may occur at seven diffent loci (IGH, IGL, IGK, TRG, TRD, TRA, TRB, see Table 1). One may find unproductive recombinations on the other allele, or even in the TR loci in B-cells. Some of these non-productive recombinations may be *incomplete* or *irregular*, such as D-J recombinations on the IGH locus or V-KDE on the IGK, the RAGs enzymes handling the KDE sequence as a J gene. These bi-allelic, incomplete, or irregular recombination are even more frequent in pathological samples, and are also tracked in hemato-oncology studies (Brüggemann et al., 2019).

Altogether, one frequently studies datasets with V(D)J or V(D)J-like recombinations on different loci. Table 1 lists $\ell = 16$ different known recombination systems. Some AIRR-Seq/RepSeq software allow to analyze several systems. The previous Vidjil-algo algorithm was able to detect recombinations in time $O(\ell n)$, where n is the length of the input sequence and ℓ the number of recombination systems.

We propose here a new algorithm able to process at once all recombinations systems, detect any V(D)J or V(D)J-like recombinations in time $O(\ell' n)$, where $\ell' \leq \ell$ is an average number of gene labels per position, with usually $\ell' \ll \ell$ (Section 2). We also propose a filtering algorithm to improve the V(D)J designation, in time $O(M' n)$ instead of the previous $O(Mn)$, where M is the total size of genes and usually $M' \ll M$ (Section 3). These algorithms were implemented in vidjil-algo: We finally report benchmarks, both on simulated and real datasets, on the quality and the speed of detection and designation (Section 4). The new algorithms bring speedups between $3\times$ and $30\times$, with a smaller memory footprint and without quality loss in results.

2 Linear detection of multi-loci V(D)J recombinations

An efficient strategy to *detect* V(D)J recombinations in a sequence is to look for a position splitting the sequence into two zones the *5' zone* with significant hits from a given type (*e.g.* V genes) followed by a *3' zone*, with significant hits from another type (*e.g.* J genes). In the original Vidjil-algo version, this strategy was used successively for all recombination systems listed in Table 1 (Giraud et al., 2014).

We use here an Aho-Corasick automaton, well-suited to look for a set of patterns inside a sequence. This

search has a time complexity linear in the sequence size and independent from the size of the set of patterns (Aho and Corasick, 1975). Note that Decombinator (Thomas et al., 2013) also relies on an Aho-Corasick automaton by indexing some tags from the V, D and J genes in order to designate the sequences more efficiently. In our case, the automaton will be used for the detection of recombinations and it does not index V, D, and J genes, but rather *spaced seeds* extracted from these genes. Using the automaton enables to store in a single pass several types of seeds and to detect, in linear time, the recombination system (including incomplete/irregular recombinations), together with an estimation of the boundaries of the 5' and 3' zones.

Seeds and seed occurrences. A word of size n is a sequence of symbols $u_1u_2\dots u_n$, and we denote a factor of a word by $u[i,j] = u_iu_{i+1}\dots u_j$. We consider symbols denoting nucleotides, $\Sigma = \{A, C, G, T\}$, as well as *match* (#) and *don't-care* (-) symbols. A gene $g \in \Sigma^*$ is a sequence of nucleotides, and $\mathcal{G}$ is the set of genes. Each gene $g \in \mathcal{G}$ has a label $T(g)$ such as V_H^- or J_B^+. The label encodes both the V/J type, the locus, and the strand information.

A spaced seed u (also called spaced k-mer) is a sequence of # and - symbols. We denote by $seed(w, u) = v$ the *projection* of the seed u on the word w of the same length, that is, for $1 \leq i \leq |w| = |u|$, $v_i = w_i$ if $u_i =$ # and otherwise $v_i =$ -. For example, $seed(\texttt{ATCG}, \texttt{\#\#-\#}) = \texttt{AT-G}$. The *weight* of a seed is its number of # symbols. For example, $weight(\texttt{\#\#-\#}) = 3$. We use there the seeds $12s =$ `######-######` and $10s =$ `#####-#####`, with $weight(12s) = 12$ and $weight(10s) = 10$. Spaced seeds are more effective than consecutive seeds of the same weight to "seed" approximate alignments (Brown, 2008). For example, on a sequence of at least 40 nucleotides, the spaced seed $12s$ can match any alignment with up to 12% mismatches, compared to 4% mismatches for the contiguous seed of size 12. Optimization of such seeds is discussed at the end of the paper.

Indexation. We extract the seed occurrences at each position of a V, (D), or J genes according to a given seed. Let $Fact(s)$ be the set of all the factors of s. We call $P(g, u)$ the set of words that are factor of a gene g relatively to the seed u: $P(g, u) = \{w \in \Sigma^* \mid \exists i, seed(g_{i\dots i+|u|-1}, u) = seed(w, u)\}$. As an example, $P(\texttt{GCCAT}, \texttt{\#\#\#\#}) = \{\texttt{GCCA}, \texttt{CCAT}\}$, whereas $P(\texttt{ACAC}, \texttt{\#-\#}) = \{\texttt{AAA}, \texttt{ACA}, \texttt{AGA}, \texttt{ATA}, \texttt{CAC}, \texttt{CCC}, \texttt{CGC}, \texttt{CTC}\}$,

There are at most $O(|g|4^z)$ of these words, where $z = |w| - weight(w)$ is the number of don't-care symbols in w. Typical seeds have $z = 1$ or 2. All words from $P(g, u)$, for all genes $g \in \mathcal{G}$, are indexed by an Aho-Corasick automaton (see Figure 2). Note that, in practice, failure transitions are removed by replacing them with four transitions, corresponding to the four nucleotides. More specifically a failure transition to a state representing sequence s will be replaced by a transition to state $s \cdot c$ when it exists, with c one of each nucleotide. When the state $s \cdot c$ does not exist, we will recursively follow the failure transition of state s until a state can be reached through a transition c. If no such state exists, the failure transition will be replaced by a transition by c to the initial state. Thus the (nondeterministic) Aho-Corasick automaton is transformed in linear time into a (deterministic) factor automaton while keeping the same number of states.

The *accepting states*, that is here the end of seed occurrences, are labeled with the list of the labels $T(g)$ of the genes g occurring at that point – there can be one or several such genes. Note that, contrarily to a lookup table index, the Aho-Corasick automaton may store words with different lengths, representing different spaced seeds (possibly with different weights) according to the recombination system, the gene V/J type, or even an individual gene.

Querying. Querying a sequence s simply means traversing it while following the transitions in the automaton, in $O(|s|)$ time. All accepting states encountered indicate the end of at least one spaced seed that occurs in the sequence. The output is a label sequence, that is a sequence of list of gene labels such as $Aff(s, \mathcal{G}) =$ - $\texttt{J}_G^+$ - $\texttt{V}_H^+$ $\texttt{V}_H^+$ $\texttt{V}_H^+$ $\texttt{J}_H^+$ - - $\texttt{V}_H^+$ $\texttt{V}_H^+$ - $\texttt{V}_B^-$ - - $\texttt{J}_H^+$ $\texttt{J}_H^+$ $\texttt{J}_H^+$ -. As there may be several gene labels per state, the label sequence can have a maximum size of $O(nL)$, where $L \leq \ell$ is the maximum number of gene labels per state. L is considered constant (on the usual germlines, $L \leq 6$). Calling ℓ' the average number of gene labels per

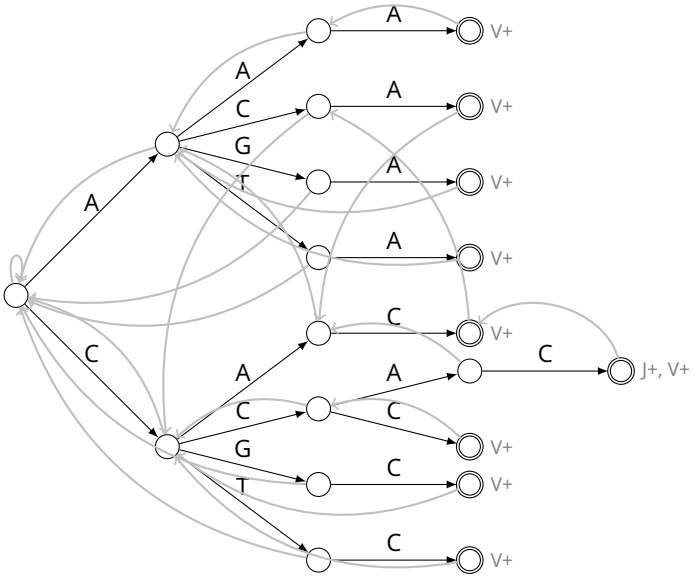

Figure 2. Aho-Corasick automaton for $P(\texttt{ACAC}, \texttt{\#-\#}) \cup P(\texttt{CCAC}, \texttt{\#\#\#\#})$, that is for the projected seeds `A-A`, `C-C`, and CCAC. The gray arrows correspond to the failure transitions in the Aho-Corasick automaton. They can be seen as ε-transitions and are removed in a pre-processing step, without changing the number of states. The accepting states have been labeled with V+ for the first seed and with J+ for the second one. Thus the state CCAC is labeled with J+ but it is also labeled with V+ as its failure function points to an accepting state labeled with V+.

Input: a reduced label sequence $a = a_1 a_2 \dots a_n$
$\delta \leftarrow 0$; $\delta_{\max} \leftarrow 0$
$i \leftarrow 0$; $j \leftarrow 0$

For each q from 1 to n
  Invariant: $\delta = |a[1, q-1]|_{\texttt{V}} - |a[1, q-1]|_{\texttt{J}}$
  if $a_t = \texttt{V}$, then $\delta \leftarrow \delta + 1$
  if $a_t = \texttt{J}$, then $\delta \leftarrow \delta - 1$
  if $\delta > \delta_{\max}$, then $\delta_{\max} \leftarrow \delta$ and $i \leftarrow q$
  if $\delta = \delta_{\max}$, then $j \leftarrow q$
End for
Return i and j

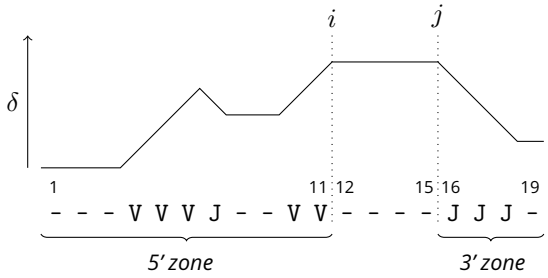

Figure 3. (Left) $O(n)$ time search of the (i, j) plateau reached around the V-J recombination zone. The actual implementation (`affectanalyser.h`) uses bitsets to check the values of a_t, and also computes, in the same linear time, the values $|a[1, i]|_{\texttt{V}}$, $|a[1, i]|_{\texttt{J}}$, $|a[j, n]|_{\texttt{V}}$ et $|a[j, n]|_{\texttt{J}}$ for filters and the statistical evaluation. (Right) On this reduced label sequence, the maximal values of δ are $\delta(11) = \dots = \delta(15) = 4$. Thus here $i = 11$ and $j = 15$.

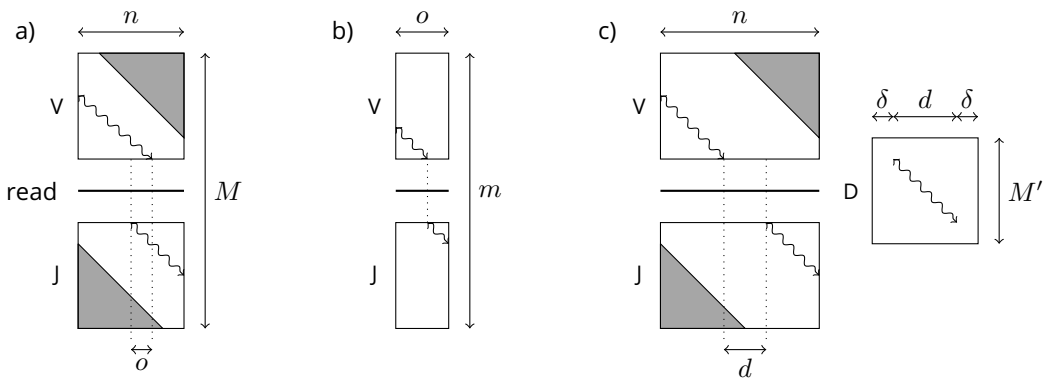

Figure 4. V(D)J designation by dynamic programming, declared in `segment.h`. See Jones and Pevzner (2004, chapter 6) for an introduction on dynamic programming methods to compare sequences. Grayed out triangles show parts excluded from the computation. a) Search of the best alignments between the read with a V gene and a J gene, in $O(Mn)$ time, where M is the total size of indexed genes and n the size of the read. Candidate V and J segments are independently predicted. b) When the best alignements make the candidate V and J segments overlap on $o \leq n$ positions, the best split point is found by another search in time $O(mo)$, where $m \ll M$ is the total size of considered V and J genes. c) In the case of a VDJ-like recombination, the central segment is predicted by a local aligment in time $O(M'(d+2\delta))$, where M' is the total size of indexed D genes and $d+2\delta \leq n$ is the size of the zone where the D segment is searched. Overlaps between V and D or between D and J candidate segmentes are handled as previouly. The algorithm finally runs in time $O((M+M')n)$.

state, the label sequence has an average size of $O(n\ell')$. On the usual germlines, $\ell' = 1.14$ gene labels per accepting state.

One can then focus on a reduced label sequence $Aff(s,\mathcal{G})^{\pm}_{\mathrm{LOC}}$ focusing on V and J of one locus LOC and one strand. For example, focusing on the labels V_H^+ and J_H^+, we consider a reduced label sequence $Aff(s,\mathcal{G})^{+}_{\mathrm{IGH}} =$ `---VVVJ--VV----JJJ-`.

Given a gene label t, we denote by $|s|_t = |\{P(g,u) \cap Fact(s) \text{ such that } T(g) = t\}|$, the number of t in the label sequence $Aff(s,\mathcal{G})$, with the seed u being used for t. On the same example, $|s|_{\mathtt{V}_H^+} = 5$.

Locus estimation. The label sequence is analyzed according to the two most probable gene labels[1] (see *p-value estimation* section below). These two gene labels may represent complete or incomplete V(D)J recombinations (Table 1), but unexpected recombinations can also be detected and are tagged as such.

Recombination detection. Given these two most probable gene labels, the algorithm detects a *5′ zone* with seed occurrences from a given label followed by a *3′ zone* with seed occurrences of another label, but allowing other labels, such as a few (random) J_H^+ or even V_B^- in a significant V_H^+ zone.

Let $a = Aff(s,\mathcal{G})^{\pm}_{\mathrm{LOC}}$ a reduced label sequence. We look for positions t such as $\delta(t) = |a[1,t]|_{\mathtt{V}} - |a[1,t]|_{\mathtt{J}}$ is maximal (many V and few J at the left). This is equivalent to maximize $\delta'(t) = |a[t,n]|_{\mathtt{J}} - |a[t,n]|_{\mathtt{V}}$ (many J and few V at the right), because, for every t, $\delta(t) - \delta'(t) = |a|_{\mathtt{V}} - |a|_{\mathtt{J}}$ is constant. The algorithm described on Figure 3 computes, in linear time, both positions $i \leq j$ that are the first and the last to maximize δ, allowing to detect in the label sequence the 5′ and 3′ zones.

p-value estimation. To estimate the significance of the (i,j) zone split and exclude chimeric sequences such as `VVVV--JJJ--VVV-JJ`, the first check is that the 5′ zone has significantly more 5′ seeds than the 3′

[1] The actual implementation, in $O(\ell^\bullet n)$ time, builds a bitset for each of the $\ell^\bullet \leq 4\ell$ gene labels for at least one seed in the read. The bitsets of the two most probable gene labels are later reused for the recombinaison detection. The step could however be improved in time $O(\ell' n)$, that is the size of the label sequence, by counting at once the gene labels, then by building the bitsets only for the two most probable ones.

zone, that is $|a[1,i]|_{\mathtt{V}} \geq \tau \cdot |a[j,n]|_{\mathtt{V}}$ with $\tau = 2$, as well as the symmetrical check for the 3′ zone. The p-value of a recombination is then estimated as follows. We call $p'_{\mathtt{V}}$ the probability to observe as many labels V in a random label sequence, and estimate $p'_{\mathtt{V}} = B(p_{\mathtt{V}}, |a[1,i]|_{\mathtt{V}}, i)$, where $B(p,k,n) = \sum_{k \leq t \leq n} \binom{n}{t} p^t (1-p)^{(n-t)}$ is the cumulated probability to have an event of probability p at least k times out of n in a Bernoulli schema. This is a very simple model, as the occurrences of seeds are actually not independent.

The probability p to have a label V on fg random seed of weight k (its number of match symbols) is estimated as $p = N_{\mathtt{V}}/4^k$, where $N_{\mathtt{V}}$ is the number of seeds V stored in the index. We similarly define $p'_{\mathtt{J}} = B(p_{\mathtt{J}}, |a[j,n]|_{\mathtt{J}}, n-j+1)$, and roughly estimate the p-value of a V-J recombination as $p'_{\mathtt{V}} + p'_{\mathtt{J}}$. As a multiple testing correction, this p-value is multiplied by the number of processed sequences, giving an E-value.

Altogether, when this E-value is below a given threshold, a V(D)J recombination has been *detected*. The middle of the recombination zone can then be estimated at around $(j+i+k-1)/2$. The next section details how we precisely *designate* such a recombination.

3 Fast V(D)J designation through seed-based heuristics

Designating a V(D)J recombination requires to compare the sequence against all V(D)J germline genes from the detected locus. Precisely aligning a gene against V, (D), and J germline genes can be done with dynamic programming techniques (Figure 4). This alignment is done in time $O(Mn)$, where M is the total length of the considered V and J genes. As V genes are about 300 bp in length, this is time consuming. Banded alignments (Chao et al., 1992) bring some improvements, however due to the deletions that occur at the end of V genes or the start of J genes, the constraint on the alignment is imposed on a single part on the gene to prevent restrictions on the number of deletions (Figure 4a and c). However, aligning *all* the genes of a given locus to a given sequence is still very long, in particular for some locus such as the B-cell heavy chain (IGH) with about 350 genes and alleles.

Selecting candidate genes with seed-based heuristics. Several V(D)J designation methods use seed-based heuristics (*eg.* (Bolotin, Poslavsky, et al., 2015; Thomas et al., 2013; Ye et al., 2013)). We propose here to use the previous filtering phase to speed-up the designation phase. The V(D)J detection heuristic presented in the previous section, giving an information on the gene label in sequences, is extended to identify genes which have seeds occurring in any sequence, in order to determine against which genes the sequence s will be aligned.

For each gene g, the accepting states in the Aho-Corasick automaton are now marked with the gene identifier $I(g)$ along with the gene label $T(g)$. Computing, still in linear time, the labels on the gene identifiers, we consider $C_{\mathcal{G}}(s) = \{g \in \mathcal{G} \,|\, |s|_{I(g)} > 0\}$ the set of genes, having at least one seed in the sequence s, and $n_{max} = \max_{g \in \mathcal{G}} |s|_{I(g)}$ the maximal number of seeds from s that matched on a gene. We want to align s against all the genes whose number of matched seeds is *close* to n_{max}. This proximity is determined by assuming that the number of matched seeds follows a binomial distribution. Thus we compute a confidence interval for n_{max} (with a confidence interval of 99.9% by default) which gives us a range $[n_a, n_b]$. We finally consider the set of candidate genes $C^\star_{\mathcal{G}}(s) = \{g \in \mathcal{G} \,|\, |s|_{I(g)} \geq n_a\}$: All the genes from $\mathcal{G}$ which have at least n_a seeds matching on s are aligned against s using dynamic programming as on the Figure 4. The designation algorithm runs in $O(M'n)$, where $M' \leq M$ is the total size of these candidate genes. For example, on the IGH germlines, a typical value of $n_{max} = 200$ on sequences of length about 300 gives $[n_a, n_b] = [174, 216]$, and there are usually less than 10 genes/alleles (out of 350+) matching at least 174 seeds, thus $M' \approx 10 \times 200 \ll M \approx 350 \times 200$.

4 Evaluation and results

4.1 Datasets

Five datasets were used to benchmark the detection and the designation of V(D)J recombinations.

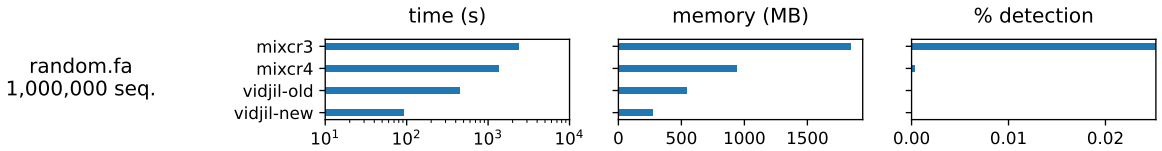

Figure 5. Detection results of MiXCR and Vidjil-algo on random sequences (dataset A)

Evaluation of the specificity of the detection

- A. 10^6 random DNA sequences, generated with %GC ratios and sequence length similar to the V(D)J IGH germline genes, in which no V(D)J recombination should be found.

Evaluation of the sensitivity of the detection

- B. $2.3 \cdot 10^6$ synthetic sequences on all loci, both for regular and "incomplete" recombinations. For each possible combination of V, D, and J genes, 10 sequences were generated, by taking the full gene lengths. Insertions and deletions at the junctions were added according to a normal distribution of 5 ± 5, and substitutions to the whole sequence on 2% of the nucleotides to take into account sequencing artifacts but also individual variations. We also generated datasets with 5% and 10% substitutions (resp. $B_{5\%}$ and $B_{10\%}$).

- C. TRB simulated sequences from the benchmark of Afzal et al. (2019). We focus on their datasets with more than a single clone and with some errors (low, .1%, and medium, 1%). Each dataset is made of 1M sequences of 250bp. For each error rate, we average the results obtained for the datasets with varying level of clonal populations, as they were very similar.

Evaluation of the correctness of the V(D)J designation

- D. 1,351 sequences from LIGM-DB (Giudicelli, Duroux, et al., 2006), focusing on the two most represented loci in LIGM-DB, IGH and IGK.

- E. 301 sequences from patient data with curated VDJ designations (Salson et al., 2016).

4.2 Software

Methods described here were implemented in C++ in a development Vidjil-algo version[2] and compared to Vidjil-algo 2018.02. In the following figures and tables, for short we refer to the former as vidjil-new and to the latter as vidjil-old. We benchmarked against MiXCR [3] (versions 3.0.13 and 4.4.1) (Bolotin, Poslavsky, et al., 2015). MiXCR is widely used, and, although it is not open-source, its code is available. After systematic comparison between several V(D)J analysis tools, MiXCR was assessed by Afzal et al. (2019) as the most balanced generic tools in terms of flexibility and accuracy. Note that MiXCR and Vidjil-algo were launched on the same germline sequences coming from the same IMGT GENE-DB version. The four programs were launched on one thread on a server with 2 Intel Xeon Gold 6130 processors (2.10 GHz, 32 MB cache) and 128 GB RAM. The benchmark is fully reproducible (from the retrieval of the data to the production of the paper's figures) using Snakemake (Köster and Rahmann, 2012), with the instructions provided at https://www.vidjil.org/data#2024-pcicompbio.

[2] This version is available at feature-a/aho-alignment-free-multi-loci.
[3] MiXCR code is available at https://github.com/milaboratory/mixcr, but is not released under an open-source licence.

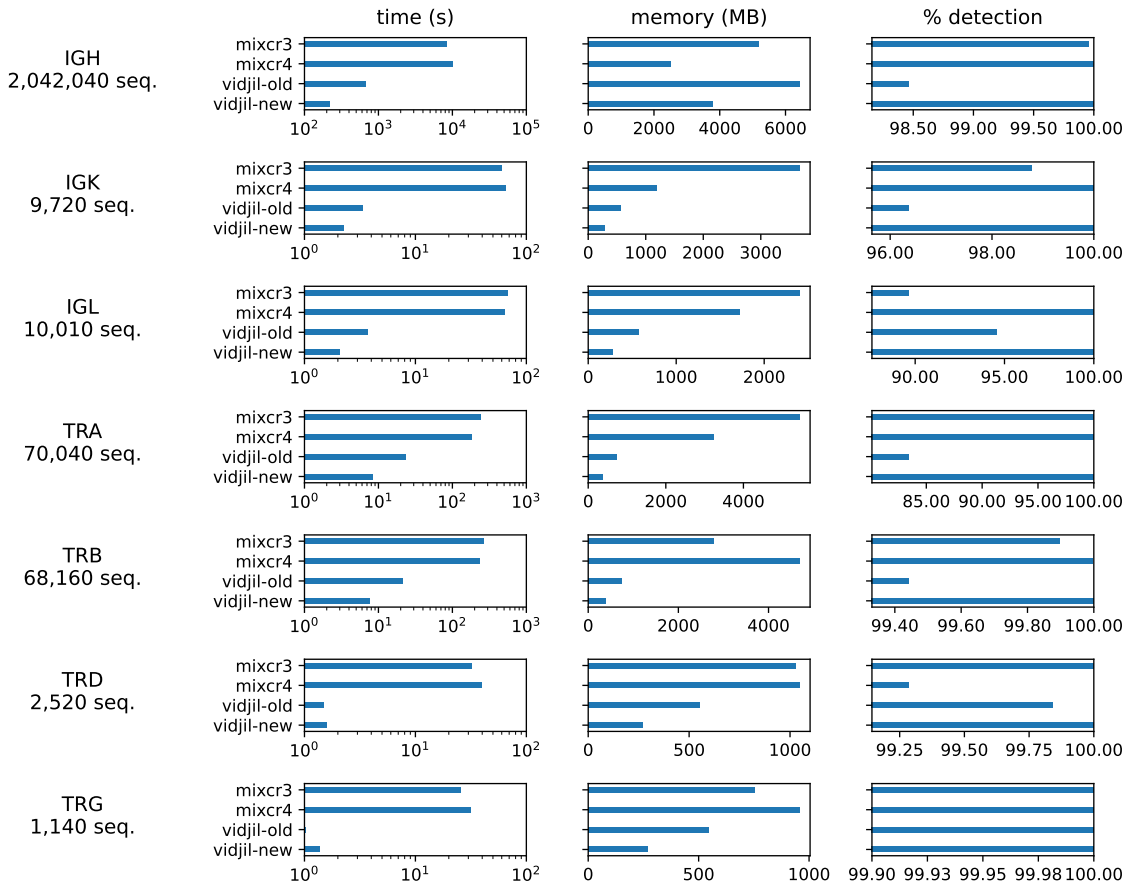

Figure 6. Detection by MiXCR and Vidjil-algo on synthetic V(D)J recombinations on all human loci (dataset B). The X-axis on the time diagrams is logarithmic.

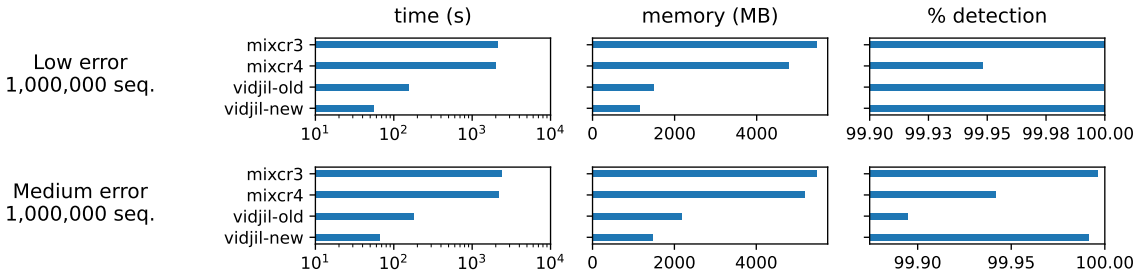

Figure 7. Detection on Afzal et al. (2019) TRB benchmark (dataset C) by MiXCR and Vidjil-algo.

4.3 V(D)J detection

Specificity and sensitivity. We restrict MiXCR to launch the smallest analysis it can do – performing a V(D)J designation on each sequence. Thus the results they produce are much more detailed than what Vidjil-algo provides. On the opposite, Vidjil-algo first tries to identify a V(D)J recombination, then determines an identifier for this V(D)J recombination and clusters all sequences sharing the same identifier into clonotypes. Then, only the 100 most abundant clonotypes are designated (which is not discussed in this section).

On random sequences (dataset A, see Figure 5), only MiXCR detects spurious V(D)J recombinations among the million random sequences. However we note a large improvement in specificity, with a two-fold improvement between MiXCR 3 and MiXCR 4.

Sensitivity is assessed with the dataset B (Figure 6), consisting of randomly generated V(D)J recombinations with 2% substitutions. Our new algorithm is very sensitive to detect those recombinations. It is the most sensitive among all the tested tools. It is the only one to reach 100% sensitivity on all loci. This is a noticeable improvement compared to previous versions of Vidjil-algo. In Supplementary File 1, figure 1, we show that even with 10 % errors the new heuristic shows very good results. On average, on the complete loci, it reaches 99.1 % detection which is better than Vidjil-old (66.7 %) and MiXCR 3 (94.8 %) but less than MiXCR 4 (100.0 %). In Supplementary File 1, figure 3, we show that with 5 % errors (20 % of them being indels), the detection ratios of Vidjil-new and MiXCR4 are mainly unchanged. Vidjil-old is much more affected by indels. This is due to the spaced seeds used in Vidjil-algo. In the Vidjil-old version, we couldn't use seeds of different lengths for V and J, thus the seeds are too long in the J gene to still have a significant number of hits matching in spite of the indels. This show that our new approch is much more robust to noisy data.

On the TRB benchmark from Afzal et al. (2019) (dataset C, Figure 7) our new algorithm is among the most sensitive: it detects all the recombinations in the low error condition (.1%) and more than 99.991% in the medium error condition (1%). In this second condition, this is slightly less than MiXCR3 (99.996%) but more than MiXCR 4 (99.94%). On IGH and IGK recombinations from LIGM-DB (dataset D, Figure 8), MiXCR 3 gives again the best results for the detection. However Vidjil-algo's new heuristic improves the former one, and for IGH, is now much closer to MiXCR 3 results. The improvement is less pronounced on IGK. However in both cases, our new heuristics gives better detection results than MiXCR 4. The reason why IGK results are worse is that IGK recombined sequences in LIGM-DB are very short on the J side, with 60% of the J gene being at most 20bp long and even 20% sequences at most 10bp long, whereas germinal IGKJ genes are 38-39bp long. Among the 68 sequences that Vidjil-algo did not detect and MiXCR 3 did, almost all of them ($\frac{67}{68}$) had an IGKJ gene 12bp long or shorter. With so short sequences, a single mutation may prevent any spaced seed to match the J sequence.

On the dataset E, results are shown in Table 2. Results are very similar between the four programs for complete recombinations (IGH, IGK, IGL, TRA, TRB, TRD, TRG). As expected, only Vidjil-algo detects some incomplete recombinations, as MiXCR does not deal with them.

locus	nb. seq.	mixcr3	mixcr4	vidjil-old	vidjil-new
IGH	95	95	94	93	92
IGK	2	2	2	2	2
IGL	2	2	2	2	2
TRA	1	1	1	1	1
TRB	16	15	15	16	16
TRD	18	18	18	18	18
TRG	31	30	31	30	31
IGH+	23			23	23
IGK+	29			29	29
TRA+D	28			28	26
TRB+	20			19	19
TRD+	31			25	27

Table 2. Detection on MiXCR, and Vidjil-algo on curated V(D)J designations (dataset E)

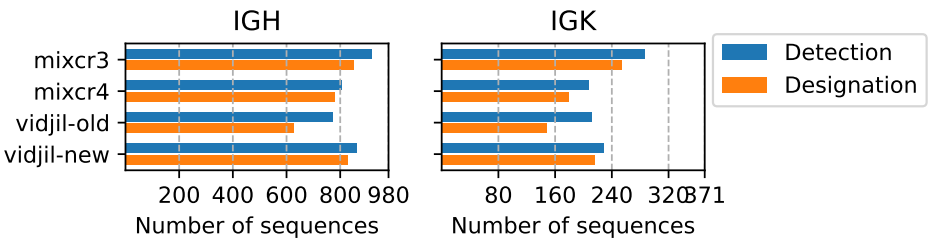

Figure 8. Correct detection and designation of V(D)J recombinations on LIGM-DB version LIGMDB_V12 (dataset D) with MiXCR and Vidjil-algo.

Speed and memory. On top of that, the new algorithm is much quicker than the former version of Vidjil-algo (at least three times quicker on large enough datasets). For datasets A, B, and C, it is between 15 and 40 times quicker than MiXCR. The difference is less striking on small datasets due to the construction of the Aho-Corasick automaton at each startup. In spite of this data structure, the memory consumption is lower than before because it used lookup tables (that were sparse). MiXCR almost systematically has the highest memory consumption, apart from dataset with a large number of distinct recombinations (IGH on Figure 6).

4.4 V(D)J designation

V(D)J designations predicted by the software, taking the same reference genes, are compared to the reference datasets D and E (Table 3 and Figure 8). Only the names of the V and of the J genes are checked (or D and J genes for incomplete recombinations). As some genes are very similar, designating one gene is very dependant to fine tunings in the scoring of the sequence comparisons. The output of both software could thus be considered correct in some cases even when departing from those reference datasets – and note that the dataset E already contains some alternative acceptable designations.

Both Vidjil-algo and MiXCR designate the same V(D)J recombinations than in the reference datasets in most of the sequences. Designations on the complete loci (IGH, TRB, TRD, TRG) are particularly close to the curated dataset E, with more than 90% correct designations, showing that the software do not have any specific difficulty to identify the V and J genes involved. Surprisingly, MiXCR 4 with the default parameters does not designate some sequences in the way that MiXCR 3 did. This may come from stricter parameters to improve specificity, as it was also shown on random sequences in Figure 5. Note also that in some cases (see IGK in Figure 8, dataset D), the detection step of Vidjil-algo can assign borderline sequences to an incorrect locus. The figure for dataset E is shown in Supplementary file 1, figure 2. During the designation step such an error would be fixed.

locus	nb. seq.	mixcr3	mixcr4	vidjil-old	vidjil-new
IGH	95	89	87	88	89
IGK	2	2	2	2	2
IGL	2	2	2	2	2
TRA	1	1	1	1	1
TRB	16	14	15	14	15
TRD	18	15	14	15	13
TRG	31	28	28	28	28
IGH+	23	1		21	20
IGK+	29			19	18
TRA+D	28	2		23	21
TRB+	20	2		19	19
TRD+	31			28	27

Table 3. Correct designation on V(D)J recombinations on manually curated V(D)J sequences (dataset E) with MiXCR and Vidjil-algo. `nb. seq.` is the number of sequences in the dataset with the given locus.

Anyway, while staying very specific, the new Vidjil-algo designation is much closer to the best results that were obtained by MiXCR 3, and even slightly better on the IGH dataset. Moreover, as expected, only Vidjil is capable of handling incomplete recombinations (Table 3), with 79% correct designations on IGH+, IGK+, and TRB+ incomplete recombinations, the TRD+/TRA+D recombinations being more challenging.

On the E dataset, our new heuristic to avoid the alignment against many genes leads to a more than 10 fold improvement in time consumption. Thus, Vidjil time consumption of designation with our new heuristic becomes comparable to MiXCR, while Vidjil-algo didn't optimize the alignment by itself (by using SIMD for instance). Note that in the classic usage, and due to its approach, Vidjil-algo can limit the designation to the 100 most abundant clonotypes.

5 Discussion and perspectives

Studying immune repertoire by high-throughput sequecing for immunological or onco-hematological applications requires adapted methods. We introduced a seed-based alignment-free algorithm, based on an Aho-Corasick automaton, to detect in a single pass, in almost linear time ($O(\ell' n)$), V(D)J recombinations coming from different loci, as well as a filtering algorithm improving the designation of V(D)J gene segments from a recombination. Both algorithms are fast and sensitive, and come with a statistical evaluation of their results, including on irregular or incomplete recombinations.

Our solution is another example where alignment-free approaches, here seed-based heuristics, provide pertinent results to analyze huge datasets, using a fraction of the resources required by alignment-based approaches. With this contribution, we almost have a linear processing of sequences. Other improvements on time or memory consumption are still possible but, as shown in our benchmarks, improvements on the quality of the results can only be marginal.

Our new version of Vidjil-algo is hence one of the fastest available programs for analyzing large datasets with billions of immune recombinations, and is moreover released under an open-source licence. The two algorithms provided an up to 5$\times$ speed-up compared to the previous Vidjil-algo version, still keeping excellent sensibility and specificity and a low memory footprint. Other software provide more information – notably the V(D)J designation of each sequence – but they are not necessarily needed in several applications. We also show that Vidjil-algo is highly effective to identify and filter sequences that do not exhibit V(D)J recombinations. This is a feature of interest to analyze large sequencing datasets, such as RNA-seq data that contain very few V(D)J recombinations.

Vidjil-algo is already used in reference protocols dealing with sequencing and analyzing immunogenetical data (Langlois de Septenville et al., 2022; Villarese et al., 2022). With the rise of large-scale analysis of public

datasets (Edgar et al., 2022), Vidjil-algo could become one of the preferred methods for V(D)J detection on such large-scale analyses. In a second time, MiXCR could be used to obtain detailed informations on the V(D)J recombinations detected by Vidjil-algo.

As the Aho-Corasick automaton stores words projected from different seeds, *seed optimization* could be further studied. The smaller the seed, the more sensitive, but the less specific. Gene repertoires have very different sizes according to the locus, with for example more than 200 kB of sequences on IGH V-J and just a few nucleotides for the TRD+ Dd2/Dd3 (see Supplementary Figure 1). In the assessed version, shorter seed sizes were selected for J genes, enabling a better recognition. Further work could optimize the seed lengths and weights depending on each recombination system.

More generally, research could include efficient detection and designation of recombined sequences with three or more segments, as well as improving again the statistical evaluation of recombinations.

Acknowledgments

We thank the Mésocentre de Lille for their computing resources, and Inria for the support as well as computing resources. We also thank Nika Abdollahi (Abdollahi, 2021) and the Bonsai team for discussions on alignment-free methods. We finally thank the VidjilNet consortium, users of Vidjil as well as the EuroClonality-NGS consortium. Their feedback helped us improve the algorithm.

Conflict of interest disclosure

Mathieu Giraud and Mikaël Salson are members of the Scientific and Technical Committee of the not-for-profit VidjilNet consortium. They do not receive any financial compensation from that consortium. The authors of this preprint declare that they have no financial conflict of interest with the content of this article.

References

Abdollahi N (July 2021). B cell receptor repertoire analysis in clinical context : new approaches for clonal grouping, intra-clonal diversity studies, and repertoire visualization. PhD thesis. Sorbonne Université.

Afzal S, I Gil-Farina, R Gabriel, S Ahmad, C von Kalle, M Schmidt, and R Fronza (2019). Systematic comparative study of computational methods for T-cell receptor sequencing data analysis. *Briefings in Bioinformatics* 20, 222–234. https://doi.org/10.1093/bib/bbx111.

Aho AV and MJ Corasick (1975). Efficient string matching: An aid to bibliographic search. *Communications of the ACM* 18, 333–340.

Arnaout R, W Lee, P Cahill, T Honan, T Sparrow, M Weiand, C Nusbaum, K Rajewsky, and SB Koralov (2011). High-Resolution Description of Antibody Heavy-Chain Repertoires in Humans. *PLoS ONE* 6, e22365.

Benichou J, R Ben-Hamo, Y Louzoun, and S Efroni (2012). Rep-Seq: uncovering the immunological repertoire through next-generation sequencing. *Immunology* 135, 183–91.

Bolotin DA, M Shugay, IZ Mamedov, MAT Ekaterina V Putintseva, IV Zvyagin, OV Britanova, and DM Chudakov (2013). MiTCR: software for T-cell receptor sequencing data analysis. *Nature Methods* 10, 813–814. https://doi.org/10.1038/nmeth.2555.

Bolotin DA, S Poslavsky, I Mitrophanov, M Shugay, IZ Mamedov, EV Putintseva, and DM Chudakov (2015). MiXCR: software for comprehensive adaptive immunity profiling. en. *Nature Methods* 12, 380–381. ISSN: 1548-7091. https://doi.org/10.1038/nmeth.3364.

Brown DG (2008). Bioinformatics Algorithms: Techniques and Applications. In: chap. A survey of seeding for sequence alignment, pp. 126–152.

Brüggemann M, M Kotrová, H Knecht, J Bartram, M Boudjogrha, V Bystry, G Fazio, E Froňková, M Giraud, A Grioni, et al. (2019). Standardized next-generation sequencing of immunoglobulin and T-cell receptor gene recombinations for MRD marker identification in acute lymphoblastic leukaemia; a EuroClonality-NGS validation study. *Leukemia*. https://doi.org/10.1038/s41375-019-0496-7.

Cavé H, J van der Werff Ten Bosch, S Suciu, C Guidal, C Waterkeyn, J Otten, M Bakkus, K Thielemans, B Grandchamp, E Vilmer, B Nelken, M Fournier, P Boutard, E Lebrun, F Méchinaud, R Garand, A Robert, N Dastugue, E Plouvier, E Racadot, A Ferster, J Gyselinck, O Fenneteau, M Duval, G Solbu, and AM Manel (1998). Clinical significance of minimal residual disease in childhood acute lymphoblastic leukemia. *New England Journal of Medicine* 339, 591–598.

Chao KM, WR Pearson, and W Miller (1992). Aligning two sequences within a specified diagonal band. *Bioinformatics* 8, 481–487.

Duez M, M Giraud, R Herbert, T Rocher, M Salson, and F Thonier (2016). Vidjil: A web platform for analysis of high-throughput repertoire sequencing. *PLOS One* 11, e0166126. https://doi.org/10.1371/journal.pone.0166126.

Edgar RC, J Taylor, V Lin, T Altman, P Barbera, D Meleshko, D Lohr, G Novakovsky, B Buchfink, B Al-Shayeb, et al. (2022). Petabase-scale sequence alignment catalyses viral discovery. *Nature* 602, 142–147.

Giraud M, M Salson, M Duez, C Villenet, S Quief, A Caillault, N Grardel, C Roumier, C Preudhomme, and M Figeac (2014). Fast multiclonal clusterization of V(D)J recombinations from high-throughput sequencing. *BMC Genomics* 15, 409. https://doi.org/10.1186/1471-2164-15-409.

Giudicelli V, D Chaume, G Mennessier, HH Althaus, W Müller, J Bodmer, A Malik, and MP Lefranc (1998). IMGT, the international ImMunoGeneTics database: a new design for immunogenetics data access. In: *MEDINFO'98*. IOS Press, pp. 351–355.

Giudicelli V, P Duroux, C Ginestoux, G Folch, J Jabado-Michaloud, D Chaume, and MP Lefranc (2006). IMGT/LIGM-DB, the IMGT® comprehensive database of immunoglobulin and T cell receptor nucleotide sequences. *Nucleic acids research* 34, D781–D784.

Jones NC and PA Pevzner (2004). *An introduction to bioinformatics algorithms*. MIT Press. ISBN: 0-262-10106-8.

Köster J and S Rahmann (2012). Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics* 28, 2520–2522.

Kuchenbecker L, M Nienen, J Hecht, AU Neumann, N Babel, K Reinert, and PN Robinson (2015). IMSEQ – a fast and error aware approach to immunogenetic sequence analysis. en. *Bioinformatics* 31, btv309. ISSN: 1367-4803, 1460-2059. https://doi.org/10.1093/bioinformatics/btv309.

Langlois de Septenville A, M Boudjoghra, C Bravetti, M Armand, M Salson, M Giraud, and F Davi (2022). Immunoglobulin Gene Mutational Status Assessment by Next Generation Sequencing in Chronic Lymphocytic Leukemia. In: *Immunogenetics*. Ed. by Langerak AW. Vol. 2453. Methods in Molecular Biology. Springer, pp. 153–167. https://doi.org/10.1007/978-1-0716-2115-8_10.

Lefranc MP (2011). IMGT, the International ImMunoGeneTics Information System. *Cold Spring Harbor Protocols* 2011, pdb.top115. https://doi.org/10.1101/pdb.top115.

Marcou Q, T Mora, and AM Walczak (2018). High-throughput immune repertoire analysis with IGoR. *Nature communications* 9, 561.

Ralph DK and FAM Iv (Jan. 2016). Consistency of VDJ Rearrangement and Substitution Parameters Enables Accurate B Cell Receptor Sequence Annotation. *PLOS Comput Biol* 12, e1004409. ISSN: 1553-7358. https://doi.org/10.1371/journal.pcbi.1004409.

Salson M, A Caillault, M Duez, Y Ferret, A Fievet, M Kotrova, F Thonier, P Villarese, S Wakeman, G Wright, and M Giraud (2016). A Dataset of Sequences with Manually Curated V(D)J Designations. Workshop on Immune Repertoire Sequencing : Bioinformatics and Applications in Hematology and Immunology (RepSeq 2016).

Shlemov A, S Bankevich, A Bzikadze, MA Turchaninova, Y Safonova, and PA Pevzner (2017). Reconstructing antibody repertoires from error-prone immunosequencing reads. *The Journal of Immunology* 199, 3369–3380.

Thomas N, J Heather, W Ndifon, J Shawe-Taylor, and B Chain (2013). Decombinator: a tool for fast, efficient gene assignment in T-cell receptor sequences using a finite state machine. *Bioinformatics* 29, 542–550.

Tonegawa S (1983). Somatic generation of antibody diversity. *Nature* 302, 575–581.

Villarese P, C Abdo, M Bertrand, F Thonier, M Giraud, M Salson, and E Macintyre (2022). One-Step Next-Generation Sequencing of Immunoglobulin and T-Cell Receptor Gene Recombinations for MRD Marker Identification in Acute Lymphoblastic Leukemia. In: *Immunogenetics. Methods and Protocols*. Ed. by Langerak AW. Vol. 2453. Methods in Molecular Biology. Springer, pp. 43–59. https://doi.org/10.1007/978-1-0716-2115-8_3.

Yang X, D Liu, N Lv, F Zhao, F Liu, J Zou, Y Chen, X Xiao, J Wu, P Liu, J Gao, Y Hu, Y Shi, J Liu, R Zhang, C Chen, J Ma, GF Gao, and B Zhu (2014). TCRklass: A New K-String-Based Algorithm for Human and Mouse TCR Repertoire Characterization. *Journal of Immunology* 194. https://doi.org/10.4049/jimmunol.1400711.

Ye J, N Ma, TL Madden, and JM Ostell (2013). IgBLAST: an immunoglobulin variable domain sequence analysis tool. *Nucleic Acids Research* 41, W34–W40. https://doi.org/10.1093/nar/gkt382.